Red Hat Enterprise Linux 6 Virtualization Tuning and Optimization Guide

Optimizing your virtual environment
Edition 0.3

Scott Radvan Dayle Parker

Red Hat Enterprise Linux 6 Virtualization Tuning and Optimization Guide

Optimizing your virtual environment
Edition 0.3

Scott Radvan
Red Hat Engineering Content Services
sradvan@redhat.com

Dayle Parker
Red Hat Engineering Content Services
dayleparker@redhat.com

Abstract

The Red Hat Enterprise Linux Virtualization Tuning and Optimization Guide covers KVM and virtualization performance. Within this guide you can find tips and suggestions for making full use of KVM performance features and options for your host systems and guest virtual machines.

Table of Contents

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog-box text; labeled buttons; check-box and radio-button labels; menu titles and submenu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **`Buttons`** tab, select the **`Left-handed mouse`** check box and click **`Close`** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** →

Character Map from the main menu bar. Next, choose **Search** → **Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above: username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                  struct kvm_assigned_pci_dev *assigned_dev)
{
          int r = 0;
          struct kvm_assigned_dev_kernel *match;

          mutex_lock(&kvm->lock);

          match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                        assigned_dev->assigned_dev_id);
          if (!match) {
                  printk(KERN_INFO "%s: device hasn't been assigned before, "
                    "so cannot be deassigned\n", __func__);
                  r = -EINVAL;
                  goto out;
          }

          kvm_deassign_device(kvm, match);

          kvm_free_assigned_device(kvm, match);

out:
          mutex_unlock(&kvm->lock);
          return r;
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled “Important” will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Getting Help and Giving Feedback

2.1. Do You Need Help?

If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer

Portal at http://access.redhat.com. Through the customer portal, you can:

- search or browse through a knowledgebase of technical support articles about Red Hat products.
- submit a support case to Red Hat Global Support Services (GSS).
- access other product documentation.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software and technology. You can find a list of publicly available mailing lists at https://www.redhat.com/mailman/listinfo. Click on the name of any mailing list to subscribe to that list or to access the list archives.

2.2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: http://bugzilla.redhat.com/ against the product **Red Hat Enterprise Linux 6.**

When submitting a bug report, be sure to mention the manual's identifier: *doc-Virtualization_Tuning_and_Optimization_Guide*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Chapter 1. Introduction

1.1. About This Guide

The Red Hat Enterprise Linux Virtualization Tuning and Optimization Guide contains details of configurable options and settings, and other suggestions that will help you achieve optimal performance of your Red Hat Enterprise Linux hosts and guest virtual machines.

Following this introduction, the guide consists of the following sections:

- Virt-manager
- tuned
- Networking
- Memory
- Block I/O
- NUMA
- Performance Monitoring Tools

1.2. Further Resources

Red Hat offers a wealth of documentation solutions across its various virtualization products. Coverage of Red Hat Enterprise Linux and its inbuilt virtualization products includes:

- *Red Hat Enterprise Linux — Virtualization Getting Started Guide*: This guide provides an introduction to virtualization concepts, advantages, and tools, and an overview of Red Hat virtualization documentation and products.
- *Red Hat Enterprise Linux — Virtualization Host Configuration and Guest Installation Guide*: This guide covers the installation of virtualization software and configuration of guest machines on a virtualization host.
- *Red Hat Enterprise Linux — Virtualization Administration Guide*: This guide covers administration of hosts, networking, storage, device and guest management using either virt-manager or virsh, a libvirt and QEMU reference, and troubleshooting information.
- *Red Hat Enterprise Linux — Virtualization Security Guide*: This guide provides an overview of virtualization security technologies provided by Red Hat. Also included are recommendations for securing hosts, guests, and shared infrastructure and resources in virtualized environments.
- *Red Hat Enterprise Linux — Virtualization Tuning and Optimization Guide*: This guide provides tips, tricks and suggestions for making full use of virtualization performance features and options for your systems and guest virtual machines.
- *Red Hat Enterprise Linux — V2V Guide*: This guide describes importing virtual machines from KVM, Xen and VMware ESX/ESX(i) hypervisors to Red Hat Enterprise Virtualization and KVM managed by libvirt.

The Red Hat Enterprise Virtualization documentation suite provides information on installation, development of applications, configuration and usage of the Red Hat Enterprise Virtualization platform and its related products.

- *Red Hat Enterprise Virtualization — Administration Guide* describes how to set up, configure and manage Red Hat Enterprise Virtualization. It assumes that you have successfully installed the Red Hat Enterprise Virtualization Manager and hosts.
- *Red Hat Enterprise Virtualization — Command Line Shell Guide* contains information for installing and using the Red Hat Enterprise Virtualization Manager command line shell.

- *Red Hat Enterprise Virtualization — Developer Guide* explains how to use the REST API. It covers the fundamentals of the REST architectural concepts in the context of a virtualization environment and provides examples of the API in operation. It also documents the installation and use of the Python Software Development Kit.
- *Red Hat Enterprise Virtualization — Evaluation Guide* enables prospective customers to evaluate the features of Red Hat Enterprise Virtualization. Use this guide if you have an evaluation license.
- *Red Hat Enterprise Virtualization — Installation Guide* describes the installation prerequisites and procedures. Read this if you need to install Red Hat Enterprise Virtualization. The installation of hosts, Manager and storage are covered in this guide. You will need to refer to the *Red Hat Enterprise Virtualization Administration Guide* to configure the system before you can start using the platform.
- *Red Hat Enterprise Virtualization — Manager Release Notes* contain release specific information for Red Hat Enterprise Virtualization Managers.
- *Red Hat Enterprise Virtualization — Power User Portal Guide* describes how power users can create and manage virtual machines from the Red Hat Enterprise Virtualization User Portal.
- *Red Hat Enterprise Virtualization — Quick Start Guide* provides quick and simple instructions for first time users to set up a basic Red Hat Enterprise Virtualization environment.
- *Red Hat Enterprise Virtualization — Technical Notes* describe the changes made between the current release and the previous one.
- *Red Hat Enterprise Virtualization — Technical Reference Guide* describes the technical architecture of Red Hat Enterprise Virtualization and its interactions with existing infrastructure.
- *Red Hat Enterprise Virtualization — User Portal Guide* describes how users of the Red Hat Enterprise Virtualization system can access and use virtual desktops from the User Portal.

Note

All of the guides for these products are available at the Red Hat Customer Portal: https://access.redhat.com/site/documentation/

1.3. KVM Overview

The following diagram represents the architecture of KVM:

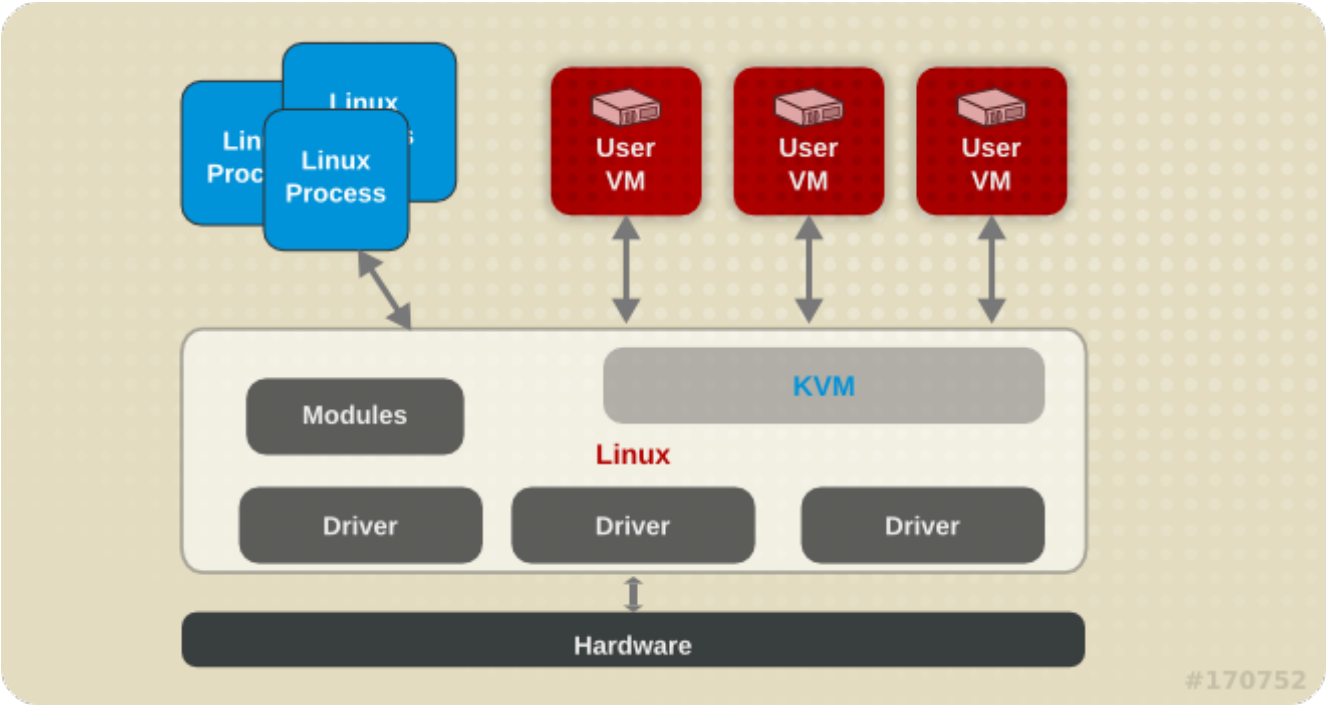

Figure 1.1. KVM architecture

1.4. KVM Performance Architecture Overview

The following points provide a brief overview of KVM as it pertains to system performance and process/thread management:

- When using KVM, guests run as a Linux process on the host.
- Virtual CPUs (vCPUs) are implemented as normal threads, handled by the Linux scheduler.
- Guests inherit features such as NUMA and Huge Pages from the kernel.
- Disk and network I/O settings in the host have a significant performance impact.
- Network traffic typically travels through a software-based bridge.

1.5. Performance Features and Improvements

- CPU/Kernel
  - NUMA - Non-Uniform Memory Access. See Chapter 7, *NUMA* for details on NUMA.
  - CFS - Completely Fair Scheduler. A modern class-focused scheduler.
  - RCU - Read Copy Update. Better handling of shared thread data.
  - Up to 160 virtual CPUs (vCPUs).
- Memory
  - Huge Pages and other optimizations for memory-intensive environments. See Chapter 5, *Memory* for details.
- Networking
  - vhost-net - a fast, kernel-based VirtIO solution.
  - SR-IOV - for near-native networking performance levels.
- Block I/O

- AIO - Support for a thread to overlap other I/O operations.
- MSI - PCI bus device interrupt generation.
- Scatter Gather - An improved I/O mode for data buffer handling.

Note

For more details on virtualization support, limits, and features, refer to the *Red Hat Enterprise Linux 6 Virtualization Getting Started Guide* and the following URLs:
http://www.redhat.com/resourcelibrary/articles/enterprise-linux-virtualization-support
http://www.redhat.com/resourcelibrary/articles/virtualization-limits-rhel-hypervisors

Chapter 2. Virt-manager

2.1. Introduction

This chapter covers performance options for virt-manager, a desktop tool for managing guest virtual machines.

2.2. Operating System Details and Devices

2.2.1. Specifying Guest Virtual Machine Details

The **virt-manager** tool provides different profiles depending on what operating system type and version are selected for a new guest virtual machine. When creating a guest, you should provide as many details as possible; this can improve performance by enabling features available for your specific type of guest.

Refer to the following example screen capture of the **virt-manager** tool. When creating a new guest virtual machine, always specify your intended OS type and Version:

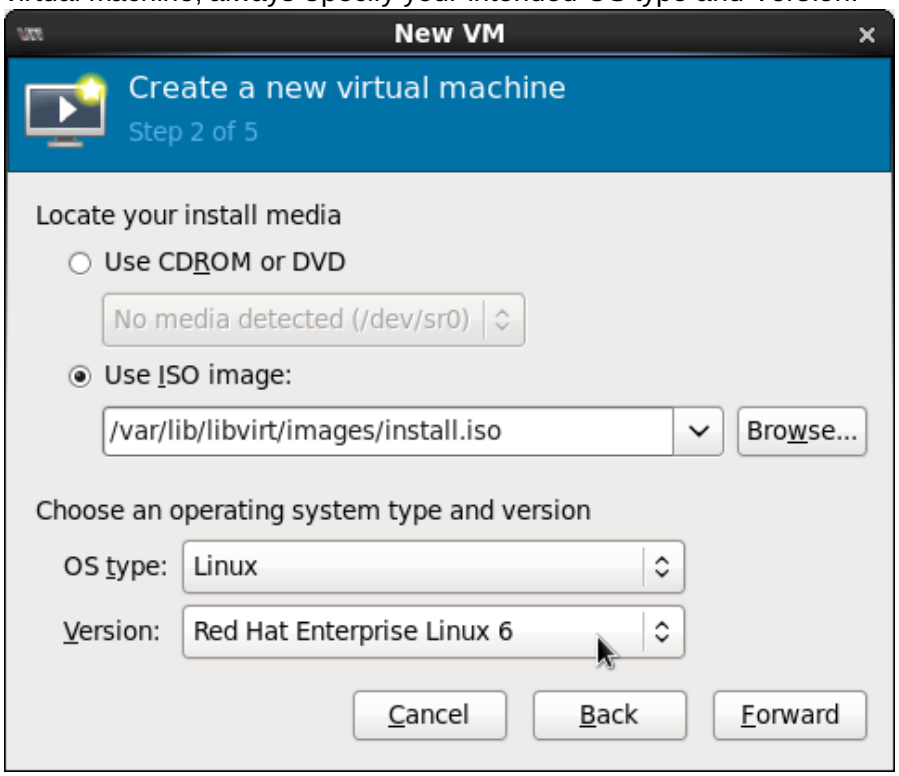

Figure 2.1. Provide the OS type and Version

2.2.2. Remove Unused Devices

Removing unused or unnecessary devices can improve performance. For instance, a guest tasked as a web server is unlikely to require audio features or an attached tablet.

Refer to the following example screen capture of the **virt-manager** tool. Click the **Remove** button to remove unnecessary devices:

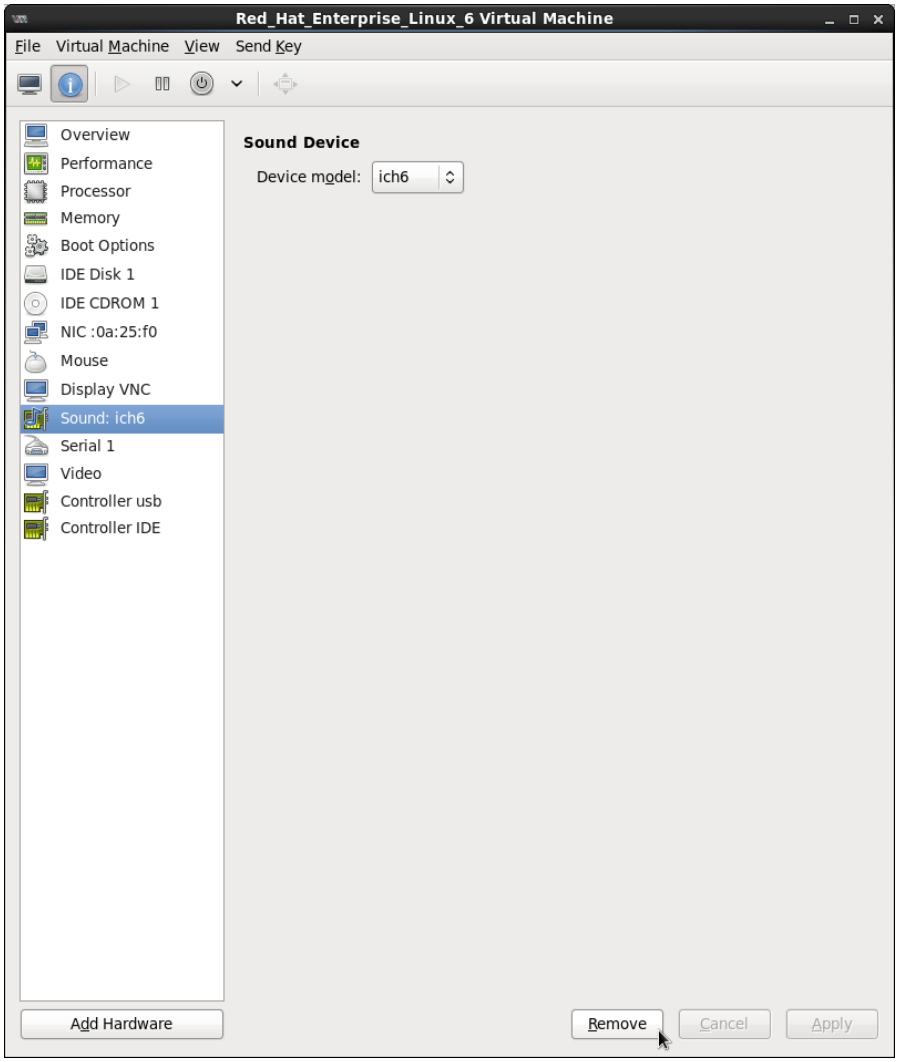

Figure 2.2. Remove unused devices

2.3. CPU Performance Options

Several CPU related options are available to your guest virtual machines. Configured correctly, these options can have a large impact on performance. The following image shows the CPU options available to your guests. The remainder of this section shows and explains the impact of these options.

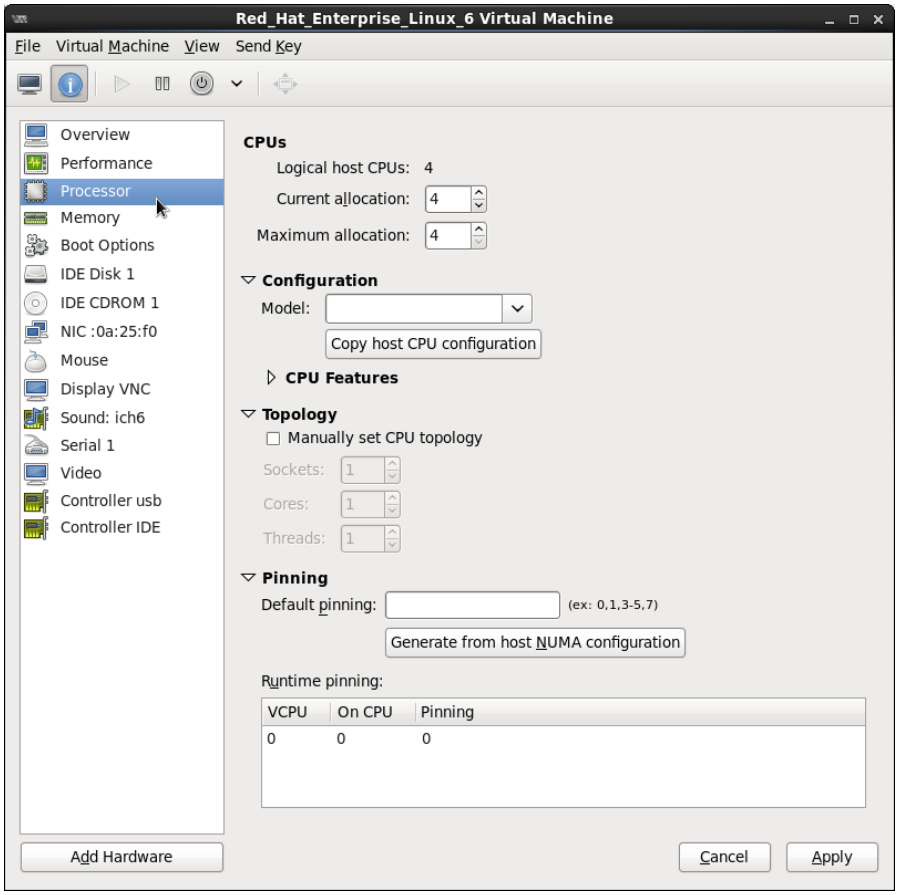

Figure 2.3. CPU Performance Options

2.3.1. Option: Available CPUs

Use this option to adjust the amount of virtual CPUs available to the guest. If you allocate more than is available on the host (known as *overcommitting*), a warning is displayed, as shown in the following image:

Figure 2.4. CPU overcommit

Warning

CPU overcommitting can have a negative impact on performance. Please refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide*, *Overcommitting with KVM* for more details on overcommitting.

2.3.2. Option: CPU Configuration

Use this option to select the CPU configuration type, based on the desired CPU model. Expand the list to

see available options, or click the *Copy host CPU configuration* button to detect and apply the physical host's CPU model and configuration. Once you select a CPU configuration, its available CPU features/instructions are displayed and can be individually enabled/disabled in the *CPU Features* list. Refer to the following diagram which shows these options:

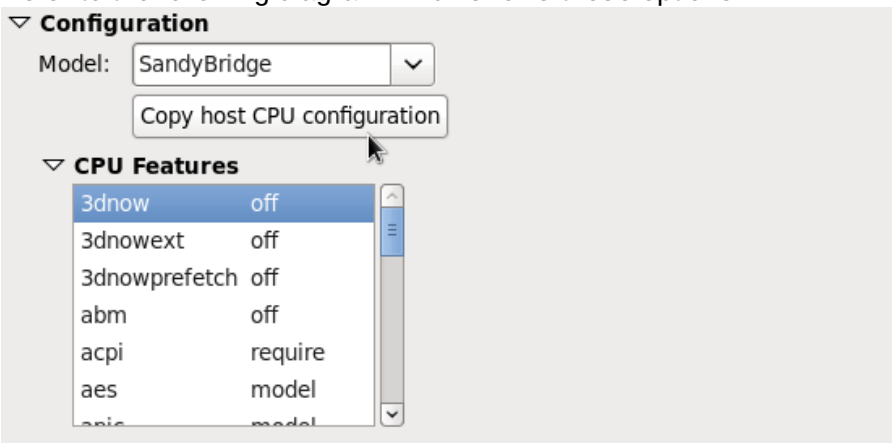

Figure 2.5. CPU Configuration Options

Note

Copying the host CPU configuration is recommended over manual configuration.

2.3.3. Option: CPU Topology

Use this option to apply a particular CPU topology (Sockets, Cores, Threads) to the virtual CPUs for your guest virtual machine. Refer to the following diagram which shows an example of this option:

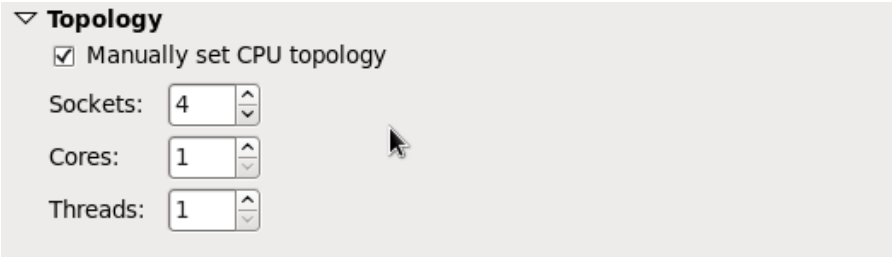

Figure 2.6. CPU Topology Options

Note

Although your environment may dictate other requirements, selecting any desired number of sockets, but with only a single core and a single thread usually gives the best performance results.

2.3.4. Option: CPU Pinning

Large performance improvements can be obtained by adhering to the system's specific NUMA topology. Use this option to automatically generate a pinning configuration that is valid for the host.

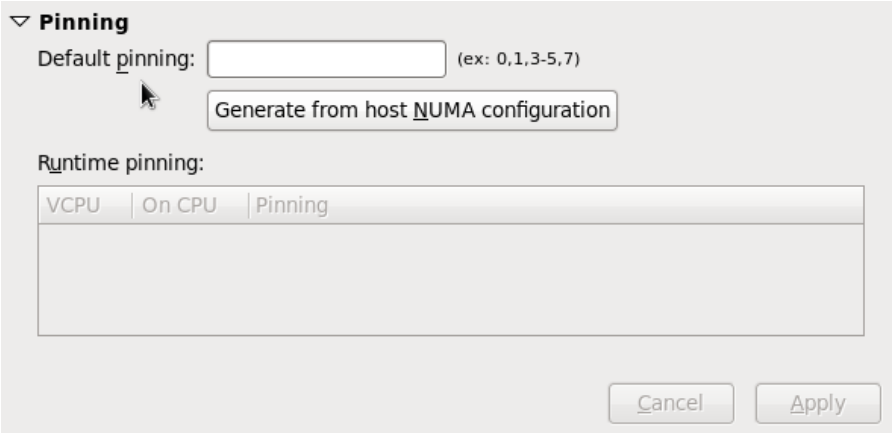

Figure 2.7. CPU Pinning

Warning

Do not use this option if the guest has more vCPUs than a single NUMA node.

Using the Pinning option will constrain the guest's vCPU threads to a single NUMA node; however, threads will be able to move around within that NUMA node. For tighter binding capabilities, use the output from the **`lscpu`** command to establish a 1:1 physical CPU to vCPU binding using **`virsh cpupin`**. Refer to Chapter 7, *NUMA* for more information on NUMA and CPU pinning.

Chapter 3. tuned

3.1. Introduction

This chapter covers using **tuned** daemon for dynamically tuning system settings in virtualized environments.

3.2. tuned and tuned-adm

Tuned is a daemon that monitors and collects data on the usage of various system components, and uses that information to dynamically tune system settings as required. It can react to changes in CPU and network use, and adjust settings to improve performance in active devices or reduce power consumption in inactive devices.

The accompanying **ktune** partners with the **tuned-adm** tool to provide a number of tuning profiles that are pre-configured to enhance performance and reduce power consumption in a number of specific use cases. Edit these profiles or create new profiles to create performance solutions tailored to your environment.

The virtualization-related profiles provided as part of **tuned-adm** include:

virtual-guest

Based on the ***enterprise-storage*** profile, ***virtual-guest*** also decreases the swappiness of virtual memory. This profile is available in Red Hat Enterprise Linux 6.3 and later, and is the recommended profile for guest machines.

virtual-host

Based on the ***enterprise-storage*** profile, ***virtual-host*** also decreases the swappiness of virtual memory and enables more aggressive writeback of dirty pages. This profile is available in Red Hat Enterprise Linux 6.3 and later, and is the recommended profile for virtualization hosts, including both KVM and Red Hat Enterprise Virtualization hosts.

Install the *tuned* package and its associated **systemtap** scripts with the command:

```
yum install tuned
```

Installing the *tuned* package also sets up a sample configuration file at **/etc/tuned.conf** and activates the default profile.

Start **tuned** by running:

```
service tuned start
```

To start **tuned** every time the machine boots, run:

```
chkconfig tuned on
```

To list all available profiles and identify the current active profile, run:

```
tuned-adm list
```

To only display the currently active profile, run:

```
tuned-adm active
```

To switch to one of the available profiles, run:

```
tuned-adm profile profile_name
```

For example:

```
tuned-adm profile virtual-host
```

To disable all tuning:

```
tuned-adm off
```

Note

Refer to the Red Hat Enterprise Linux 6 *Power Management Guide*, available from http://access.redhat.com/site/documentation/, for further information about **tuned**, **tuned-adm** and **ktune**.

Chapter 4. Networking

4.1. Introduction

This chapter covers network optimization topics for virtualized environments.

4.2. Network Tuning Tips

- Use multiple networks to avoid congestion on a single network. For example, have dedicated networks for management, backups and/or live migration.
- Usually, matching the default MTU (1500 bytes) in all components is sufficient. If you require larger messages, increasing the MTU value can reduce fragmentation. If you change the MTU, all devices in the path should have a matching MTU value.
- Use ***`arp_filter`*** to prevent ARP Flux, an undesirable condition that can occur in both hosts and guests and is caused by the machine responding to ARP requests from more than one network interface: **`echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter`** or edit **`/etc/sysctl.conf`** to make this setting persistent.

Note

Refer to the following URL for more information on ARP Flux: http://linux-ip.net/html/ether-arp.html#ether-arp-flux

4.3. Virtio and vhost_net

The following diagram demonstrates the involvement of the kernel in the Virtio and vhost_net architectures.

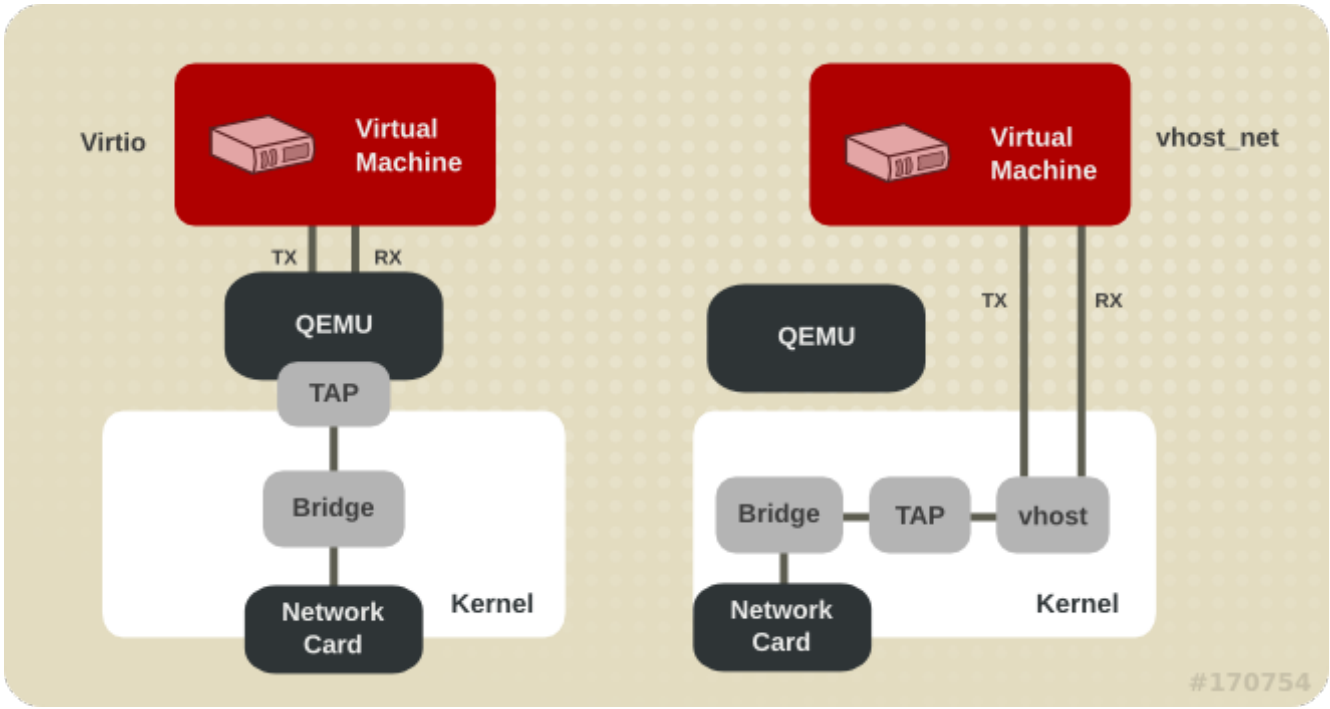

Figure 4.1. Virtio and vhost_net architectures

vhost_net moves part of the Virtio driver from the userspace into the kernel. This reduces copy operations, lowers latency and CPU usage.

4.4. Device Assignment and SR-IOV

The following diagram demonstrates the involvement of the kernel in the Device Assignment and SR-IOV architectures.

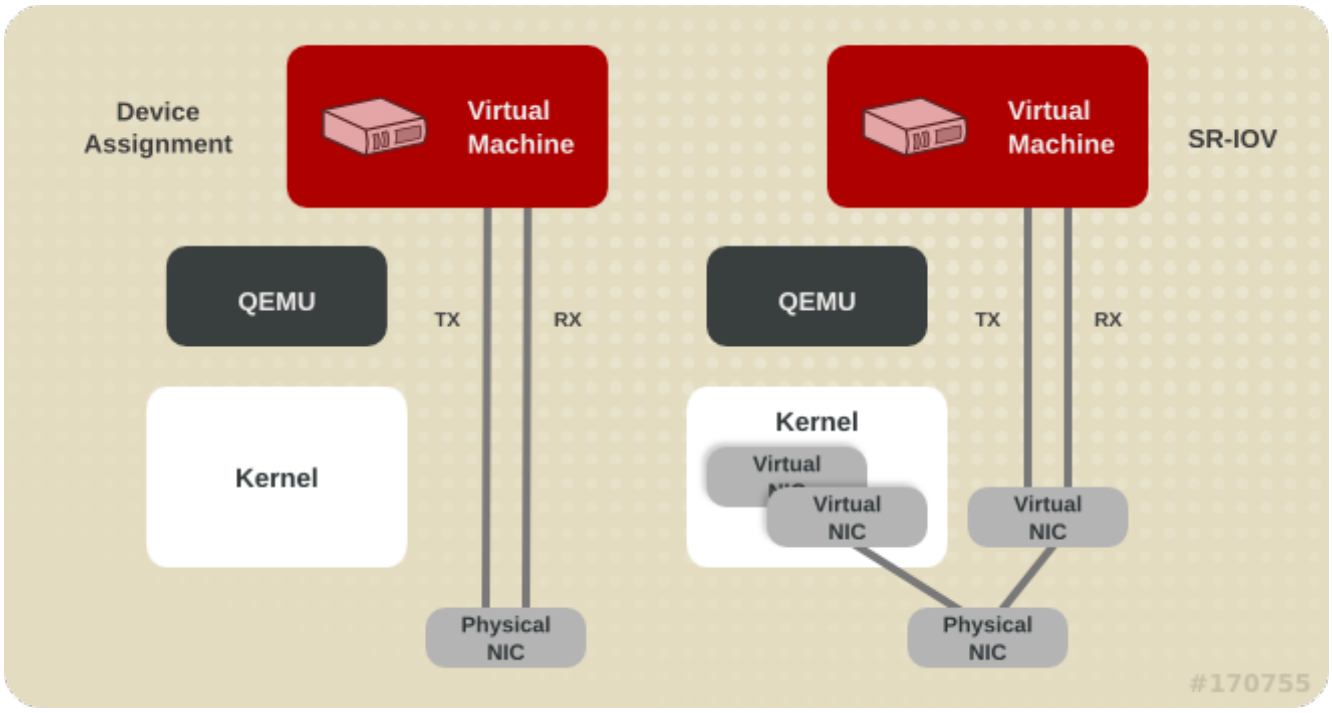

Figure 4.2. Device assignment and SR-IOV

Device assignment presents the entire device to the guest. SR-IOV needs support in drivers and hardware, including the NIC and the system board and allows multiple virtual devices to be created and passed into different guests. A vendor-specific driver is required in the guest, however, SR-IOV offers the lowest latency of any network option.

Chapter 5. Memory

5.1. Introduction

This chapter covers memory optimization options for virtualized environments.

5.2. Huge Pages and Transparent Huge Pages

x86 CPUs usually address memory in 4kB pages, but they are capable of using larger pages known as *huge pages*. KVM guests can be deployed with huge page memory support in order to improve performance by increasing CPU cache hits against the Transaction Lookaside Buffer (TLB).

A kernel feature enabled by default in Red Hat Enterprise Linux 6, huge pages can significantly increase performance, particularly for large memory and memory-intensive workloads. Red Hat Enterprise Linux 6 is able to more effectively manage large amounts of memory by increasing the page size through the use of huge pages.

Add to XML configuration for guests:

```
<memoryBacking>
 <hugepages/>
</memoryBacking>
```

View the current huge pages value:

```
cat /proc/sys/vm/nr_hugepages
```

```
cat /proc/meminfo | grep Huge
```

To set the number of huge pages:

```
echo xyz > /proc/sys/vm/nr_hugepages
```

Note

Alternatively, to make the setting persistent, modify the **vm.nr_hugepages** value in **/etc/sysctl.conf**.

Huge pages can benefit not only the host but also guests, however, their total huge pages value must be less than is available in the host.

By allowing all free memory to be used as cache, performance is increased. Transparent Hugepages are used by default if **/sys/kernel/mm/redhat_transparent_hugepage/enabled** is set to always.

Transparent Hugepage Support does not prevent the use of hugetlbfs. However, when hugetlbfs is not used, KVM will use Transparent Hugepages instead of the regular 4kB page size.

Chapter 6. Block I/O

6.1. Introduction

This chapter covers optimizing I/O settings in virtualized environments.

6.2. Caching

Table 6.1. Caching options

Caching Option	Description
Cache=none	I/O from the guest is not cached on the host, but may be kept in a writeback disk cache. Use this option for guests with large I/O requirements. This option is generally the best choice, and is the only option to support migration.
Cache=writethrough	I/O from the guest is cached on the host but written through to the physical medium. This mode is slower and prone to scaling problems. Best used for small number of guests with lower I/O requirements. Suggested for guests that do not support a writeback cache (such as Red Hat Enterprise Linux 5.5 and earlier), where migration is not needed.
Cache=writeback	I/O from the guest is cached on the host.

The caching mode can be selected in the **`Virtual Disk`** section in **`virt-manager`**. Select the cache mode under **`Performance options`**, as shown in the following image:

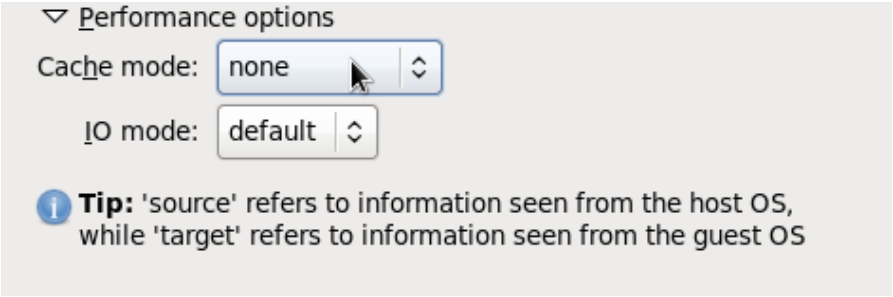

Figure 6.1. Caching mode options in virt-manager

6.3. Block I/O related commands

Use the **`blkiotune`** and **`blkdeviotune`** commands to set, display and query block disk parameters.

Note

Refer to the *Red Hat Enterprise Linux 6 Virtualization Administration Guide* for more details on these commands.

Chapter 7. NUMA

7.1. Introduction

Historically, all memory on x86 systems is equally accessible by all CPUs. Known as Uniform Memory Access (UMA), access times are the same no matter which CPU performs the operation.

This behavior is no longer the case with recent x86 processors. In Non-Uniform Memory Access (NUMA), system memory is divided into zones (called *nodes*), which are allocated to particular CPUs or sockets. Access to memory that is local to a CPU is faster than memory connected to remote CPUs on that system.

This chapter describes memory allocation and NUMA tuning configurations in virtualized environments.

7.2. Memory Allocation Policies

Three policy types define how memory is allocated from the nodes in a system:

Strict

The default operation is for allocation to fall back to other nodes if the memory can not be allocated on the target node. Strict policy means that the allocation will fail if the memory can not be allocated on the target node.

Interleave

Memory pages are allocated across nodes specified by a nodemask, but are allocated in a round-robin fashion.

Preferred

Memory is allocated from a single preferred memory node. If sufficient memory is not available, memory can be allocated from other nodes.

XML configuration enables the desired policy:

```
<numatune>
 <memory mode='preferred' nodeset='0'>
</numatune>
```

7.3. libvirt NUMA Tuning

7.3.1. NUMA vCPU Pinning

The following example XML configuration has a domain process pinned to physical CPUs 0-7. The vCPU thread is pinned to its own cpuset. For example, vCPU0 is pinned to physical CPU 0, vCPU1 is pinned to physical CPU 1, and so on:

```
<vcpu cpuset='0-7'>8</vcpu>
 <cputune>
  <vcpupin vcpu='0' cpuset='0'/>
  <vcpupin vcpu='1' cpuset='1'/>
  <vcpupin vcpu='2' cpuset='2'/>
  <vcpupin vcpu='3' cpuset='3'/>
  <vcpupin vcpu='4' cpuset='4'/>
  <vcpupin vcpu='5' cpuset='5'/>
  <vcpupin vcpu='6' cpuset='6'/>
  <vcpupin vcpu='7' cpuset='7'/>
 </cputune>
```

There is a direct relationship between the vcpu and vcpupin tags. If a vcpupin option is not specified, the value will be automatically determined and inherited from the parent vcpu tag option. The following configuration shows <vcpupin > for **vcpu 5** missing. Hence, **vCPU5** would be pinned to physical CPUs 0-7, as specified in the parent tag <vcpu>:

```
<vcpu cpuset='0-7'>8</vcpu>
 <cputune>
  <vcpupin vcpu='0' cpuset='0'/>
  <vcpupin vcpu='1' cpuset='1'/>
  <vcpupin vcpu='2' cpuset='2'/>
  <vcpupin vcpu='3' cpuset='3'/>
  <vcpupin vcpu='4' cpuset='4'/>
  <vcpupin vcpu='6' cpuset='6'/>
  <vcpupin vcpu='7' cpuset='7'/>
 </cputune>
```

7.3.2. Domain Processes

As provided in Red Hat Enterprise Linux, libvirt uses libnuma to set memory binding policies for domain processes. The nodeset for these policies can be configured either as *static* (specified in the domain XML) or *auto* (configured by querying numad). Refer to the following XML configuration for examples on how to configure these inside the <numatune> tag:

```
<numatune>
 <memory mode='strict' placement='auto'/>
</numatune>
```

```
<numatune>
 <memory mode='strict' nodeset='0,2-3'/>
</numatune>
```

libvirt uses **sched_setaffinity(2)** to set CPU binding policies for domain processes. The cpuset option can either be *static* (specified in the domain XML) or *auto* (configured by querying numad). Refer to the following XML configuration for examples on how to configure these inside the <vcpu> tag:

```
<vcpu placement='auto' current='8'>32</vcpu>
```

```
<vcpu placement='static' cpuset='0-10,^5'>8</vcpu>
```

There are implicit inheritance rules between the placement mode you use for <vcpu> and <numatune>:

- The placement mode for <numatune> defaults to the same placement mode of <vcpu>, or to ***static*** if a <nodeset> is specified.

- Similarly, the placement mode for <vcpu> defaults to the same placement mode of <numatune>, or to ***static*** if <cpuset> is specified.

This means that CPU tuning and memory tuning for domain processes can be specified and defined separately, but they can also be configured to be dependent on the other's placement mode.

Note

Refer to the following URLs for more information on vcpu and numatune: http://libvirt.org/formatdomain.html#elementsCPUAllocation and http://libvirt.org/formatdomain.html#elementsNUMATuning

7.3.3. Domain vcpu Threads

In addition to tuning domain processes, libvirt also permits the setting of the pinning policy for each vcpu thread in XML configuration. This is done inside the <cputune> tags:

```
<cputune>
 <vcpupin vcpu="0" cpuset="1-4,^2"/>
 <vcpupin vcpu="1" cpuset="0,1"/>
 <vcpupin vcpu="2" cpuset="2,3"/>
 <vcpupin vcpu="3" cpuset="0,4"/>
</cputune>
```

In this tag, libvirt uses either cgroup or **sched_setaffinity(2)** to pin the vcpu thread to the specified cpuset.

Note

For more details on cputune, refer to the following URL: http://libvirt.org/formatdomain.html#elementsCPUTuning

7.3.4. Using emulatorpin

Another way of tuning the domain process pinning policy is to use the <emulatorpin> tag inside of <cputune>. For example:

```
<cputune>
 <emulatorpin cpuset="1-3"/>
</cputune>
```

7.3.5. Tuning vcpu CPU Pinning with virsh

Important

These are example commands only. You will need to substitute values according to your environment.

The following example **virsh** command will pin the vcpu thread (rhel6u4) which has an ID of 1 to the physical CPU 2:

```
% virsh vcpupin rhel6u4 1 2
```

You can also obtain the current vcpu pinning configuration with the **virsh** command. For example:

```
% virsh vcpupin rhel6u4
```

7.3.6. Tuning Domain Process CPU Pinning with virsh

Important

These are example commands only. You will need to substitute values according to your environment.

The emulatorpin option applies CPU affinity settings to threads that are associated with each domain process. For complete pinning, you must use both **virsh vcpupin** (as shown previously) and **virsh emulatorpin** for each guest. For example:

```
% virsh emulatorpin rhel6u4 3-4
```

7.3.7. Tuning Domain Process Memory Policy with virsh

Domain process memory can be dynamically tuned. Refer to the following example command:

```
% virsh numatune rhel6u4 --nodeset 0-10
```

More examples of these commands can be found in the **virsh** man page.

Chapter 8. Performance Monitoring Tools

8.1. Introduction

This chapter describes tools used to monitor guest virtual machine environmnents.

8.2. perf kvm

You can use the **perf** command with the **kvm** option to collect guest operating system statistics from the host.

In Red Hat Enterprise Linux, the *perf* package provides the **perf** command. Run **rpm -q perf** to see if the *perf* package is installed. If it is not installed, and you want to install it to collect and analyze guest operating system statistics, run the following command as the root user:

```
yum install perf
```

In order to use **perf kvm** in the host, you must have access to the **/proc/modules** and **/proc/kallsyms** files from the guest. There are two methods to achieve this. Refer to the following procedure, Procedure 8.1, “Copying /proc files from guest to host” to transfer the files into the host and run reports on the files. Alternatively, refer to Procedure 8.2, “Alternative: using sshfs to directly access files” to directly mount the guest and access the files.

Procedure 8.1. Copying /proc files from guest to host

Important

If you directly copy the required files (for instance, via **scp**) you will only copy files of zero length. This procedure describes how to first save the files in the guest to a temporary location (with the **cat** command), and then copy them to the host for use by **perf kvm**.

1. **Log in to the guest and save files**

   Log in to the guest and save **/proc/modules** and **/proc/kallsyms** to a temporary location, **/tmp**:

   ```
   # cat /proc/modules > /tmp/modules
   # cat /proc/kallsyms > /tmp/kallsyms
   ```

2. **Copy the temporary files to the host**

   Once you have logged off from the guest, run the following example **scp** commands to copy the saved files to the host. You should substitute your host name and TCP port if they are different:

   ```
   # scp root@GuestMachine:/tmp/kallsyms guest-kallsyms
   # scp root@GuestMachine:/tmp/modules guest-modules
   ```

   You now have two files from the guest (**guest-kallsyms** and **guest-modules**) on the host, ready for use by **perf kvm**.

3. **Recording and reporting events with perf kvm**

   Using the files obtained in the previous steps, recording and reporting of events in the guest, the host, or both is now possible.

Run the following example command:

```
# perf kvm --host --guest --guestkallsyms=guest-kallsyms \
--guestmodules=guest-modules record -a -o perf.data
```

Note

If both **--host** and **--guest** are used in the command, output will be stored in **perf.data.kvm**. If only **--host** is used, the file will be named **perf.data.host**. Similarly, if only **--guest** is used, the file will be named **perf.data.guest**.

Pressing Ctrl-C stops recording.

4. **Reporting events**

   The following example command uses the file obtained by the recording process, and redirects the output into a new file, **analyze**.

```
perf kvm --host --guest --guestmodules=guest-modules report -i perf.data.kvm
\
--force > analyze
```

   View the contents of the **analyze** file to examine the recorded events:

```
# cat analyze


# Events: 7K cycles
#
# Overhead       Command  Shared Object      Symbol
# ........  ............  ..................  ........................
#
    95.06%            vi  vi                   [.] 0x48287
     0.61%          init  [kernel.kallsyms]  [k] intel_idle
     0.36%            vi  libc-2.12.so       [.] _wordcopy_fwd_aligned
     0.32%            vi  libc-2.12.so       [.] __strlen_sse42
     0.14%       swapper  [kernel.kallsyms]  [k] intel_idle
     0.13%          init  [kernel.kallsyms]  [k] uhci_irq
     0.11%          perf  [kernel.kallsyms]  [k] generic_exec_single
     0.11%          init  [kernel.kallsyms]  [k] tg_shares_up
     0.10%      qemu-kvm  [kernel.kallsyms]  [k] tg_shares_up

[output truncated...]
```

Procedure 8.2. Alternative: using sshfs to directly access files

Important

This is provided as an example only. You will need to substitute values according to your environment.

```
# Get the PID of the qemu process for the guest:
PID=`ps -eo pid,cmd | grep "qemu.*-name GuestMachine" \
| grep -v grep | awk '{print $1}'`

# Create mount point and mount guest
mkdir -p /tmp/guestmount/$PID
sshfs -o allow_other,direct_io GuestMachine:/ /tmp/guestmount/$PID

# Begin recording
perf kvm --host --guest --guestmount=/tmp/guestmount \
record -a -o perf.data

# Ctrl-C interrupts recording. Run report:
perf kvm --host --guest --guestmount=/tmp/guestmount report \
-i perf.data

# Unmount sshfs to the guest once finished:
fusermount -u /tmp/guestmount
```

Revision History

Revision 0.3-53.404 **Mon Nov 25 2013** **Rüdiger Landmann**
Rebuild with Publican 4.0.0

Revision 0.3-53 **Fri Nov 15 2013** **Dayle Parker**
Version for 6.5 GA release.

Revision 0.3-51 **Thurs Oct 31 2013** **Dayle Parker**
Adjusted image scaling parameters throughout guide.

Revision 0.3-50 **Thurs Sept 26 2013** **Dayle Parker**
Minor edits, formatting, and XML markup for beta.

Revision 0.3-49 **Wed Sept 18 2013** **Dayle Parker**
Added reference to Virtualization Administration Guide for BZ#1006880.

Revision 0.3-47 **Sun Feb 17 2013** **Scott Radvan**
Version for 6.4 GA release.

Revision 0.3-46 **Sun Feb 17 2013** **Scott Radvan**
Apply SME feedback.

Revision 0.3-45 **Wed Feb 13 2013** **Scott Radvan**
Minor wording improvements.

Revision 0.3-44 **Wed Feb 13 2013** **Scott Radvan**
Apply SME feedback for Caching mode descriptions.

Revision 0.3-43 **Tue Feb 12 2013** **Scott Radvan**
Apply SME feedback. Word usage changes throughout.

Revision 0.3-42 **Mon Feb 11 2013** **Scott Radvan**
Remove draft status.

Revision 0.3-41 **Mon Feb 11 2013** **Scott Radvan**
Changes from SME feedback. Add admonitions and reword CPU pinning options in virt-manager. Remove reference to non-existent virsh commands.

Revision 0.3-40 **Fri Feb 8 2013** **Scott Radvan**
Minor wording issues.

Revision 0.3-39 **Fri Feb 8 2013** **Scott Radvan**
Indentation fixes for <screen> tags.

Revision 0.3-38 **Fri Feb 8 2013** **Scott Radvan**
Fix build errors (BZ#908666).

Revision 0.3-37 **Fri Feb 8 2013** **Scott Radvan**

libvirt NUMA tuning section

Revision 0.3-36 **Thu Feb 7 2013** **Scott Radvan**
Rebuild with new version of publishing toolchain to fix admonition CSS errors.

Revision 0.3-35 **Thu Feb 7 2013** **Scott Radvan**
wording and formatting review. better placement of x.1 Introductions. Expand vcpu example to include missing parameters.

Revision 0.3-34 **Mon Feb 4 2013** **Scott Radvan**
scalefit and add width parameter for all PNGs.

Revision 0.3-33 **Mon Feb 4 2013** **Scott Radvan**
virt-manager screenshots and options explained

Revision 0.3-32 **Thu Jan 31 2013** **Scott Radvan**
Add SME feedback for NUMA cpusets. Add developer remarks.

Revision 0.3-31 **Thu Jan 31 2013** **Scott Radvan**
Changed to SR-IOV throughout guide, not SR/IOV.

Revision 0.3-30 **Wed Jan 30 2013** **Scott Radvan**
s/mode/policy in NUMA strict

Revision 0.3-29 **Wed Jan 30 2013** **Scott Radvan**
Correct the NUMA memory modes: BZ#854099.

Revision 0.3-28 **Tue Jan 29 2013** **Scott Radvan**
Fix QE feedback. #754935.

Revision 0.3-27 **Tue Jan 22 2013** **Scott Radvan**
Fix wording of huge pages introduction in Memory.xml.

Revision 0.3-26 **Mon Jan 21 2013** **Scott Radvan**
Remove CPU section. NUMA section covers CPU pinning.

Revision 0.3-25 **Mon Jan 14 2013** **Scott Radvan**
Remove Kernel chapter. Bump year to 2013.

Revision 0.3-24 **Mon Jan 14 2013** **Scott Radvan**
Further SME feedback added. Network options and SR-IOV.

Revision 0.3-23 **Thu Jan 3 2013** **Scott Radvan**
Bump to work around publishing issues.

Revision 0.3-22 **Thu Jan 3 2013** **Scott Radvan**
Add SME feedback: numactl and numatune nodesets.

Revision 0.3-21 **Fri Dec 14 2012** **Scott Radvan**

Add SME feedback: hugetlbfs mount, numatune memory modes.

Revision 0.3-20 **Thu Dec 06 2012** **Scott Radvan**

Add SME feedback: vcpu pinning

Revision 0.3-19 **Mon Nov 12 2012** **Scott Radvan**

Show that caching options relate to I/O requirements/number of guests.

Revision 0.3-18 **Mon Oct 29 2012** **Scott Radvan**

Fix validation errors.

Revision 0.3-17 **Mon Oct 29 2012** **Scott Radvan**

Kernel options in nested lists.

Revision 0.3-16 **Tue Oct 16 2012** **Scott Radvan**

Minor typos.

Revision 0.3-15 **Mon Oct 15 2012** **Scott Radvan**

Capitalize headings throughout.

Revision 0.3-14 **Sun Oct 14 2012** **Scott Radvan**

Add tuned section, show tuned-adm commands.

Revision 0.3-13 **Tue Oct 2 2012** **Scott Radvan**

Infrastructure changes, minor typos.

Revision 0.3-12 **Tue Oct 2 2012** **Scott Radvan**

Add NUMA intro and memory policies.

Revision 0.3-11 **Tue Oct 2 2012** **Scott Radvan**

Expand tuned-adm profiles.

Revision 0.3-10 **Tue Oct 2 2012** **Scott Radvan**

Add tuned-adm table.

Revision 0.3-9 **Tue Oct 2 2012** **Scott Radvan**

Add caching table, general network tips.

Revision 0.3-8 **Thu Sep 27 2012** **Scott Radvan**

Add KVM overview and networking images as placeholders. Add 'Further resources' section in Overview.

Revision 0.3-7 **Mon Sep 24 2012** **Scott Radvan**

Minor typos.

Revision 0.3-6 **Wed Sep 19 2012** **Scott Radvan**

Add line breaks so lengthy commands wrap properly.

Revision 0.3-5 **Tue Sep 18 2012** **Scott Radvan**

Add perf kvm chapter and procedures.

Revision 0.3-4 **Wed Sep 12 2012** **Scott Radvan**

Flesh out chapters. Add 'Performance Monitoring Tools' chapter.

Revision 0.3-3 **Wed Sep 12 2012** **Scott Radvan**

Start virt-manager chapter. Add screen captures.

Revision 0.3-2 **Wed Sep 12 2012** **Scott Radvan**

Draft introduction.

Revision 0.3-1 **Wed Sep 12 2012** **Scott Radvan**

Layout guide, provide basic infrastructure settings and ids.