Red Hat Enterprise Linux 7 Virtualization Tuning and Optimization Guide

Optimizing your virtual environment

Dayle Parker           Scott Radvan
Red Hat Subject Matter Experts
Optimizing your virtual environment

Dayle Parker
Red Hat Engineering Content Services
dayle.parker@redhat.com

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

Red Hat Subject Matter Experts
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 virtualized guests.
# Table of Contents

Chapter 1. Introduction .................................................. 2
  1.1. KVM Overview ................................. 2
  1.2. KVM Performance Architecture Overview 2
  1.3. Virtualization Performance Features and Improvements 2

Chapter 2. Virt-manager .................................................. 5
  2.1. Introduction .................................. 5
  2.2. Operating System Details and Devices 5
  2.3. CPU Performance Options ......... 6
  2.4. Virtual Disk Performance Options 10

Chapter 3. tuned .......................................................... 11
  3.1. Introduction .................................. 11
  3.2. tuned and tuned-adm .............. 11

Chapter 4. Networking .................................................. 13
  4.1. Introduction .................................. 13
  4.2. Networking Tuning Tips ............. 13
  4.3. Virtio and vhost_net ............. 13
  4.4. Device Assignment and SR-IOV ....... 14
  4.5. Network Tuning Techniques 14

Chapter 5. Memory ...................................................... 17
  5.1. Introduction .................................. 17
  5.2. Memory Tuning Tips ............... 17
  5.3. Memory Tuning on Virtual Machines 17

Chapter 6. Block I/O .................................................... 21
  6.1. Introduction .................................. 21
  6.2. Block I/O Tuning ...................... 21
  6.3. Caching ......................... 22
  6.4. I/O Mode .......................... 22
  6.5. Block I/O Tuning Techniques 23

Chapter 7. NUMA .......................................................... 25
  7.1. Introduction .................................. 25
  7.2. Memory Allocation Policies ........... 25
  7.3. Automatic NUMA Balancing ....... 25
  7.4. libvirt NUMA Tuning ............... 26
  7.5. NUMA-Aware Kernel SamePage Merging (KSM) 31

Chapter 8. Performance Monitoring Tools .......................... 32
  8.1. Introduction .................................. 32
  8.2. perf kvm ............... 32
  8.3. Virtual Performance Monitoring Unit (vPMU) 34

Revision History .......................................................... 35
Chapter 1. Introduction

1.1. KVM Overview

The following diagram represents the architecture of KVM:

![KVM Architecture Diagram]

Figure 1.1. KVM architecture

1.2. 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.

1.3. Virtualization Performance Features and Improvements

Virtualization Performance Improvements in Red Hat Enterprise Linux 7

The following features improve virtualization performance in Red Hat Enterprise Linux 7:

**Automatic NUMA Balancing**

Automatic NUMA balancing improves the performance of applications running on NUMA hardware systems, without any manual tuning required for Red Hat Enterprise Linux 7 guests. Automatic
Systems, without any manual tuning required for Red Hat Enterprise Linux 7 guests. Automatic NUMA balancing moves tasks, which can be threads or processes, closer to the memory they are accessing.

For more information on automatic NUMA balancing, see Section 7.3, “Automatic NUMA Balancing”.

**Multi-queue virtio-net**

A networking approach that enables packet sending/receiving processing to scale with the number of available vCPUs of the guest.

For more information on multi-queue virtio-net, see Section 4.5.2, “Multi-Queue virtio-net”.

**Bridge Zero Copy Transmit**

Zero copy transmit mode reduces the host CPU overhead in transmitting large packets between a guest network and an external network by up to 15%, without affecting throughput. Bridge zero copy transmit is fully supported on Red Hat Enterprise Linux 7 virtual machines, but disabled by default.

For more information on zero copy transmit, see Section 4.5.1, “Bridge Zero Copy Transmit”.

**APIC Virtualization (APICv)**

Newer Intel processors offer hardware virtualization of the Advanced Programmable Interrupt Controller (APICv). APICv improves virtualized x86_64 guest performance by allowing the guest to directly access the APIC, dramatically cutting down interrupt latencies and the number of virtual machine exits caused by the APIC. This feature is used by default in newer Intel processors and improves I/O performance.

**EOI Acceleration**

End-of-interrupt acceleration for high bandwidth I/O on older chipsets without virtual APIC capabilities.

**Multi-queue virtio-scsi**

Improved storage performance and scalability provided by multi-queue support in the virtio-scsi driver. This enables each virtual CPU to have a separate queue and interrupt to use without affecting other vCPUs.

For more information on multi-queue virtio-scsi, see Section 6.5.2, “Multi-Queue virtio-scsi”.

**Para-virtualized Ticketlocks**

Para-virtualized ticketlocks (pvticketlocks) improve the performance of Red Hat Enterprise Linux 7 guest virtual machines running on Red Hat Enterprise Linux 7 hosts with oversubscribed CPUs.

**Para-virtualized Page Faults**

Para-virtualized page faults are injected into a guest when it attempts to access a page swapped out by the host. This improves KVM guest performance when host memory is overcommitted and guest memory is swapped out.

**Para-virtualized Time vsyscall Optimization**

The `gettimeofday` and `clock_gettime` system calls execute in the userspace through the vsyscall mechanism. Previously, issuing these system calls required the system to switch into
kernel mode, and then back into the userspace. This greatly improves performance for some applications.

Virtualization Performance Features in Red Hat Enterprise Linux

» 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.
  • Disk I/O throttling - Controls on guest disk I/O requests to prevent over-utilizing host resources. See Section 6.5.1, “Disk I/O Throttling” for details.

Note

For more details on virtualization support, limits, and features, refer to the Red Hat Enterprise Linux 7 Virtualization Getting Started Guide and the following URLs:

https://access.redhat.com/site/supported-hypervisors

https://access.redhat.com/site/articles/rhel-kvm-limits
Chapter 2. Virt-manager

2.1. Introduction

This chapter covers performance tuning options available in 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](image)

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.
2.3.1. Option: Available CPUs

Use this option to adjust the amount of virtual CPUs (vCPUs) 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:

![CPU Performance Options](image)

**Figure 2.3. CPU Performance Options**

**2.3.1. Option: Available CPUs**

Use this option to adjust the amount of virtual CPUs (vCPUs) 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:

<table>
<thead>
<tr>
<th>CPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical host CPUs: 4</td>
</tr>
<tr>
<td>Current allocation: 5</td>
</tr>
<tr>
<td>Maximum allocation: 5</td>
</tr>
</tbody>
</table>

![CPU overcommit](image)

**Figure 2.4. CPU overcommit**

CPUs are overcommitted when the sum of vCPUs for all guests on the system is greater than the number of host CPUs on the system. You can overcommit CPUs with one or multiple guests if the total number of vCPUs is greater than the number of host CPUs.
Important

As with memory overcommitting, CPU overcommitting can have a negative impact on performance, for example, in situations with a heavy or unpredictable guest workload. Refer to the Red Hat Enterprise Linux Virtualization Deployment and 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:

![Configuration Diagram]

**Figure 2.5. CPU Configuration Options**

Note

Copying the host CPU configuration is recommended over manual configuration.

Note

Alternately, run the `virsh capabilities` command on your host machine to view the virtualization capabilities of your system, including CPU types and NUMA capabilities.

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:
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.
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.

2.4. Virtual Disk Performance Options

Several virtual disk related options are available to your guest virtual machines during installation that can impact performance. The following image shows the virtual disk options available to your guests.

The cache mode, IO mode, and IO tuning can be selected in the Virtual Disk section in virt-manager. Set these parameters in the fields under Performance options, as shown in the following image:

![Virtual Disk Performance Options](image)

When setting the virtual disk performance options in virt-manager, the virtual machine must be restarted for the settings to take effect.

See Section 6.3, "Caching" and Section 6.4, "I/O Mode" for descriptions of these settings and instructions for editing these settings in the guest XML configuration.
Chapter 3. tuned

3.1. Introduction

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

3.2. tuned and tuned-adm

**tuned** is a tuning profile delivery mechanism that adapts Red Hat Enterprise Linux for certain workload characteristics, such as requirements for CPU-intensive tasks, or storage/network throughput responsiveness.

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 throughput-performance profile, virtual-guest also decreases the swappiness of virtual memory.
  - The virtual-guest profile is automatically selected when creating a Red Hat Enterprise Linux 7 guest virtual machine. It is the recommended profile for virtual machines.
  - This profile is available in Red Hat Enterprise Linux 6.3 and later, but must be manually selected when installing a virtual machine.

- **virtual-host**
  - Based on the throughput-performance profile, virtual-host also decreases the swappiness of virtual memory and enables more aggressive writeback of dirty pages. This profile 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:

```
systemctl start tuned
```

To start tuned every time the machine boots, run:

```
systemctl enable tuned
```

To list all available profiles and identify the current active profile, run:
Available profiles:
- balanced
- desktop
- latency-performance
- network-latency
- network-throughput
- powersave
- sap
- throughput-performance
- virtual-guest
- virtual-host

Current active profile: throughput-performance

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, to switch to the `virtual-host` profile, run:

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

**Note**

It is possible to create custom `tuned` profiles to encapsulate a set of tuning parameters. For instructions on creating custom `tuned` profiles, refer to the `tuned.conf` man page.

In some cases, it is preferable to disable `tuned` to use parameters set manually. To disable all tuning, run:

```
tuned-adm off
```

**Note**

Chapter 4. Networking

4.1. Introduction

This chapter covers network optimization topics for virtualized environments.

4.2. Networking 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](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.

4.5. Network Tuning Techniques

This section describes techniques for tuning network performance in virtualized environments.

4.5.1. Bridge Zero Copy Transmit

Zero copy transmit mode is effective on large packet sizes. It typically reduces the host CPU overhead by up to 15% when transmitting large packets between a guest network and an external network, without affecting throughput.

It does not affect performance for guest-to-guest, guest-to-host, or small packet workloads.

Bridge zero copy transmit is fully supported on Red Hat Enterprise Linux 7 virtual machines, but disabled by default. To enable zero copy transmit mode, set the experimental_zc抄pytx kernel module parameter for the vhost_net module to 1.
Note

An additional data copy is normally created during transmit as a threat mitigation technique against denial of service and information leak attacks. Enabling zero copy transmit disables this threat mitigation technique.

If performance regression is observed, or if host CPU utilization is not a concern, zero copy transmit mode can be disabled by setting `experimental_zcopytx` to 0.

### 4.5.2. Multi-Queue virtio-net

Multi-queue virtio-net provides an approach that scales the network performance as the number of vCPUs increases, by allowing them to transfer packets through more than one virtqueue pair at a time.

Today's high-end servers have more processors, and guests running on them often have an increasing number of vCPUs. In single queue virtio-net, the scale of the protocol stack in a guest is restricted, as the network performance does not scale as the number of vCPUs increases. Guests cannot transmit or retrieve packets in parallel, as virtio-net has only one TX and RX queue.

Multi-queue support removes these bottlenecks by allowing paralleled packet processing.

Multi-queue virtio-net provides the greatest performance benefit when:

- Traffic packets are relatively large.
- The guest is active on many connections at the same time, with traffic running between guests, guest to host, or guest to an external system.
- The number of queues is equal to the number of vCPUs. This is because multi-queue support optimizes RX interrupt affinity and TX queue selection in order to make a specific queue private to a specific vCPU.

#### Note

Multi-queue virtio-net works well for incoming traffic, but can occasionally hurt performance for outgoing traffic. Enabling multi-queue virtio-net increases the total throughput, and in parallel increases CPU consumption.

### 4.5.2.1. Configuring Multi-Queue virtio-net

To use multi-queue virtio-net, enable support in the guest by adding the following to the guest XML configuration (where the value of \( N \) is from 1 to 8, as the kernel supports up to 8 queues for a multi-queue tap device):

```xml
<interface type='network'>
    <source network='default'/>
    <model type='virtio'/>
    <driver name='vhost' queues='N'/>
</interface>
```

When running a virtual machine with \( N \) virtio-net queues in the guest, enable the multi-queue support with the following command (where the value of \( M \) is from 1 to \( N \)):

```bash
# ethtool -L eth0 combined M
```
Chapter 5. Memory

5.1. Introduction

This chapter covers memory optimization options for virtualized environments.

5.2. Memory Tuning Tips

To optimize memory performance in a virtualized environment, consider the following:

- Do not allocate more resources to guest than it will use.
- If possible, assign a guest to a single NUMA node, providing that resources are sufficient on that NUMA node.

5.3. Memory Tuning on Virtual Machines

5.3.1. Memory Monitoring Tools

Memory usage can be monitored in virtual machines using tools used in bare metal environments. Tools useful for monitoring memory usage and diagnosing memory-related problems include:

- `top`
- `vmstat`
- `numastat`
- `/proc/`

**Note**

For details on using these performance tools, refer to the *Red Hat Enterprise Linux 7 Performance Tuning Guide* and the man pages for these commands.

5.3.2. Memory Tuning with virsh

The optional `<memtune>` element in the guest XML configuration allows administrators to configure guest virtual machine memory settings manually. If `<memtune>` is omitted, default memory settings apply.

Display or set memory parameters in the `<memtune>` element in a virtual machine with the `virsh memtune` command, replacing values according to your environment:

```
# virsh memtune virtual_machine --parameter size
```

Optional parameters include:

- **hard_limit**

  The maximum memory the virtual machine can use, in kibibytes (blocks of 1024 bytes).
**Warning**

Setting this limit too low can result in the virtual machine being killed by the kernel.

**soft_limit**

The memory limit to enforce during memory contention, in kibibytes (blocks of 1024 bytes).

**swap_hard_limit**

The maximum memory plus swap the virtual machine can use, in kibibytes (blocks of 1024 bytes). The **swap_hard_limit** value must be more than the **hard_limit** value.

**min_guarantee**

The guaranteed minimum memory allocation for the virtual machine, in kibibytes (blocks of 1024 bytes).

---

**Note**

See # virsh help memtune for more information on using the virsh memtune command.

---

The optional `<memoryBacking>` element may contain several elements that influence how virtual memory pages are backed by host pages.

Setting **locked** prevents the host from swapping out memory pages belonging to the guest. Add the following to the guest XML to lock the virtual memory pages in the host's memory:

```xml
<memoryBacking>
  <locked/>
</memoryBacking>
```

---

**Important**

When setting **locked**, a **hard_limit** must be set in the `<memtune>` element to the maximum memory configured for the guest, plus any memory consumed by the process itself.

Setting **nosharepages** prevents the host from merging the same memory used among guests. To instruct the hypervisor to disable share pages for a guest, add the following to the guest's XML:

```xml
<memoryBacking>
  <nosharepages/>
</memoryBacking>
```

---

### 5.3.3. Huge Pages and Transparent Huge Pages (THP)

x86 CPUs usually address memory in 4kB pages, but they are capable of using larger 2MB 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 7, huge pages can significantly increase performance, particularly for large memory and memory-intensive workloads. Red Hat Enterprise Linux 7 is able to more effectively manage large amounts of memory by increasing the page size through the use of huge pages.

5.3.3.1. Configuring Transparent Huge Pages

Transparent huge pages (THP) automatically optimize system settings for performance. By allowing all free memory to be used as cache, performance is increased.

Transparent huge pages are used by default if `/sys/kernel/mm/transparent_hugepage/enabled` is set to `always`. To disable transparent huge pages:

```
# echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

Transparent Huge Page support does not prevent the use of hugetlbfs. However, when hugetlbfs is not used, KVM will use transparent huge pages instead of the regular 4kB page size.

5.3.3.2. Configuring Static Huge Pages

In some cases, more control of huge pages is preferable. To use static huge pages on guests, add the following to the guest XML configuration:

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

This instructs the host to allocate memory to the guest using huge pages, instead of the default page size.

View the current huge pages value, run the following command:

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

Procedure 5.1. Setting huge pages

The following example procedure shows the commands to set huge pages.

1. View the current huge pages value:

```
# cat /proc/meminfo | grep Huge
AnonHugePages: 2048 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
HugePages_Surp: 0
Hugepagesize: 2048 kB
```

2. Huge pages are set in increments of 2MB. To set the number of huge pages to 25000, use the following command:

```
echo 25000 > /proc/sys/vm/nr_hugepages
```
Note

Alternatively, to make the setting persistent, use the `# sysctl vm.nr_hugepages=N` command with \( N \) being the number of huge pages.

3. Mount the huge pages:

```
# mount -t hugetlbfs hugetlbfs /dev/hugepages
```

4. Restart `libvirtd`, then restart the virtual machine with the following commands:

```
# systemctl start libvirtd
# virsh start virtual_machine
```

5. Verify the changes in `/proc/meminfo`:

```
# cat /proc/meminfo | grep Huge
AnonHugePages:       0 kB
HugePages_Total:    25000
HugePages_Free:      23425
HugePages_Rsvd:      0
HugePages_Surp:      0
Hugepagesize:       2048 kB
```

Huge pages can benefit not only the host but also guests, however, their total huge pages value must be less than what is available in the host.
Chapter 6. Block I/O

6.1. Introduction

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

6.2. Block I/O Tuning

The `virsh blkio_tune` command allows administrators to set or display a guest virtual machine's block I/O parameters manually in the `<blkio>` element in the guest XML configuration.

To display current `<blkio>` parameters for a virtual machine:

```
# virsh blkio_tune virtual_machine
```

To set a virtual machine's `<blkio>` parameters, refer to the following command and replace values according to your environment:

```
# virsh blkio_tune virtual_machine [--weight number] [--device-weights string] [--config] [--live] [--current]
```

Parameters include:

- **weight**
  The I/O weight, within the range 100 to 1000.

- **device-weights**
  A single string listing one or more device/weight pairs, in the format of `/path/to/device,weight,/path/to/device,weight`. Each weight must be within the range 100-1000, or the value 0 to remove that device from per-device listings. Only the devices listed in the string are modified; any existing per-device weights for other devices remain unchanged.

- **config**
  Add the `--config` option for changes to take effect at next boot.

- **live**
  Add the `--live` option to apply the changes to the running virtual machine.

**Note**

The `--live` option requires the hypervisor to support this action. Not all hypervisors allow live changes of the maximum memory limit.

- **current**
  Add the `--current` option to apply the changes to the current virtual machine.
6.3. Caching

Caching options can be configured with virt-manager during guest installation, or on an existing guest virtual machine by editing the guest XML configuration.

Table 6.1. Caching options

<table>
<thead>
<tr>
<th>Caching Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache=none</td>
<td>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.</td>
</tr>
<tr>
<td>Cache=writethrough</td>
<td>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.</td>
</tr>
<tr>
<td>Cache=writeback</td>
<td>I/O from the guest is cached on the host.</td>
</tr>
</tbody>
</table>

In virt-manager, the caching mode can be specified under Virtual Disk. For information on using virt-manager to change the cache mode, see Section 2.4, “Virtual Disk Performance Options”.

To configure the cache mode in the guest XML, edit the cache setting inside the driver tag, specifying none, writeback, or writethrough. For example, to set the cache as writeback:

```xml
<disk type='file' device='disk'>
  <driver name='qemu' type='raw' cache='writeback'/>
</disk>
```

6.4. I/O Mode

I/O mode options can be configured with virt-manager during guest installation, or on an existing guest virtual machine by editing the guest XML configuration.

Table 6.2. I/O mode options

<table>
<thead>
<tr>
<th>Caching Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO=native</td>
<td>The default for Red Hat Enterprise Virtualization environments. This mode refers to kernel asynchronous I/O with direct I/O options.</td>
</tr>
<tr>
<td>IO=threads</td>
<td>The default are host user-mode based threads.</td>
</tr>
</tbody>
</table>
The default in Red Hat Enterprise Linux 7 is threads mode.

<table>
<thead>
<tr>
<th>Caching Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO=default</td>
<td>The default in Red Hat Enterprise Linux 7 is threads mode.</td>
</tr>
</tbody>
</table>

In **virt-manager**, the I/O mode can be specified under **Virtual Disk**. For information on using **virt-manager** to change the I/O mode, see Section 2.4, “Virtual Disk Performance Options”.

To configure the I/O mode in the guest XML, edit the `io` setting inside the `driver` tag, specifying `native`, `threads`, or `default`. For example, to set the I/O mode to `threads`:

```xml
<disk type='file' device='disk'>
  <driver name='qemu' type='raw' io='threads'/>
</disk>
```

### 6.5. Block I/O Tuning Techniques

This section describes more techniques for tuning block I/O performance in virtualized environments.

#### 6.5.1. Disk I/O Throttling

When several virtual machines are running simultaneously, they can interfere with system performance by using excessive disk I/O. Disk I/O throttling in KVM provides the ability to set a limit on disk I/O requests sent from virtual machines to the host machine. This can prevent a virtual machine from over-utilizing shared resources and impacting the performance of other virtual machines.

Disk I/O throttling can be useful in various situations, for example, where guest virtual machines belonging to different customers are running on the same host, or when quality of service guarantees are given for different guests. Disk I/O throttling can also be used to simulate slower disks.

I/O throttling can be applied independently to each block device attached to a guest and supports limits on throughput and I/O operations. Use the **virsh blkdeviotune** command to set I/O limits for a virtual machine. Refer to the following example:

```bash
# virsh blkdeviotune virtual_machine device --parameter limit
```

Optional parameters include:

- **total-bytes-sec**
  
  The total throughput limit in bytes per second.

- **read-bytes-sec**
  
  The read throughput limit in bytes per second.

- **write-bytes-sec**
  
  The write throughput limit in bytes per second.

- **total-iops-sec**
  
  The total I/O operations limit per second.

- **read-iops-sec**
  
  The read I/O operations limit per second.
write-iops-sec

The write I/O operations limit per second.

For example, to throttle disk0 on virtual_machine to 1000 I/O operations per second and 50 MB per second throughput, run this command:

```
# virsh blkdeviotune virtual_machine disk0 --total-iops-sec 1000 --total-bytes-sec 52428800
```

6.5.2. Multi-Queue virtio-scsi

Multi-queue virtio-scsi provides improved storage performance and scalability in the virtio-scsi driver. It enables each virtual CPU to have a separate queue and interrupt to use without affecting other vCPUs.

6.5.2.1. Configuring Multi-Queue virtio-scsi

Multi-queue virtio-scsi is disabled by default on Red Hat Enterprise Linux 7.

To enable multi-queue virtio-scsi support in the guest, add the following to the guest XML configuration, where N is the total number of vCPU queues:

```
<controller type='scsi' index='0' model='virtio-scsi'>
  <driver queues='N' />
</controller>
```
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 across NUMA nodes, which correspond to sockets or to a particular set of CPUs that have identical access latency to the local subset of system memory.

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:

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

7.3. Automatic NUMA Balancing

Automatic NUMA balancing improves the performance of applications running on NUMA hardware systems. It is enabled by default on Red Hat Enterprise Linux 7 systems.

An application will generally perform best when the threads of its processes are accessing memory on the same NUMA node as the threads are scheduled. Automatic NUMA balancing moves tasks (which can be threads or processes) closer to the memory they are accessing. It also moves application data to memory closer to the tasks that reference it. This is all done automatically by the kernel when automatic NUMA balancing is active.

Automatic NUMA balancing uses a number of algorithms and data structures, which are only active and allocated if automatic NUMA balancing is active on the system:

» Periodic NUMA unmapping of process memory
NUMA hinting fault

Migrate-on-Fault (MoF) - moves memory to where the program using it runs

task_numa_placement - moves running programs closer to their memory

### 7.3.1. Configuring Automatic NUMA Balancing

Automatic NUMA balancing is enabled by default in Red Hat Enterprise Linux 7, and will automatically activate when booted on hardware with NUMA properties.

Automatic NUMA balancing is enabled when both of the following conditions are met:

- `numactl --hardware` shows multiple nodes, and
- `cat /sys/kernel/debug/sched_features` shows NUMA in the flags.

Manual NUMA tuning of applications will override automatic NUMA balancing, disabling periodic unmapping of memory, NUMA faults, migration, and automatic NUMA placement of those applications.

In some cases, system-wide manual NUMA tuning is preferred.

To disable automatic NUMA balancing, use the following command:

```
# echo 0 > /proc/sys/kernel/numa_balancing
```

To enable automatic NUMA balancing, use the following command:

```
# echo 1 > /proc/sys/kernel/numa_balancing
```

### 7.4. libvirt NUMA Tuning

Generally, best performance on NUMA systems is achieved by limiting guest size to the amount of resources on a single NUMA node. Avoid unnecessarily splitting resources across NUMA nodes.

Use the `numastat` tool to view per-NUMA-node memory statistics for processes and the operating system.

In the following example, the `numastat` tool shows four virtual machines with inoptimal memory alignment across NUMA nodes:

```
# numastat -c qemu-kvm
```

<table>
<thead>
<tr>
<th>Per-node process</th>
<th>memory usage (in MBs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID</td>
<td>Node 0</td>
</tr>
<tr>
<td>------------------</td>
<td>--------</td>
</tr>
<tr>
<td>51722 (qemu-kvm)</td>
<td>68</td>
</tr>
<tr>
<td>51747 (qemu-kvm)</td>
<td>245</td>
</tr>
<tr>
<td>53736 (qemu-kvm)</td>
<td>62</td>
</tr>
<tr>
<td>53773 (qemu-kvm)</td>
<td>1393</td>
</tr>
</tbody>
</table>

```
Total | 1769 | 463 | 2024 | 7462 | 10037 | 2672 | 169 | 7837 | 32434 |
```

Run `numad` to align the guests' CPUs and memory resources automatically.

Then run `numastat -c qemu-kvm` again to view the results of running `numad`. The following output shows that resources have been aligned:
# numastat -c qemu-kvm

<table>
<thead>
<tr>
<th>PID</th>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
<th>Node 5</th>
<th>Node 6</th>
<th>Node 7</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>51747 (qemu-kvm)</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>8072</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>8080</td>
</tr>
<tr>
<td>53736 (qemu-kvm)</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>8113</td>
<td>0</td>
<td>8120</td>
<td></td>
</tr>
<tr>
<td>53773 (qemu-kvm)</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8110</td>
<td>8118</td>
<td></td>
</tr>
<tr>
<td>59065 (qemu-kvm)</td>
<td>0</td>
<td>0</td>
<td>8050</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8051</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>0</td>
<td>8072</td>
<td>0</td>
<td>8072</td>
<td>0</td>
<td>8114</td>
<td>8110</td>
<td>32368</td>
</tr>
</tbody>
</table>

**Note**

Running `numastat` with `-c` provides compact output; adding the `-m` option adds system-wide memory information on a per-node basis to the output. Refer to the `numastat` man page for more information.

### 7.4.1. NUMA vCPU Pinning

vCPU pinning provides similar advantages to task pinning on bare metal systems. Since vCPUs run as userspace tasks on the host operating system, pinning increases cache efficiency. One example of this is an environment where all vCPU threads are running on the same physical socket, therefore sharing a L3 cache domain.

Combining vCPU pinning with `numatune` can avoid NUMA misses. The performance impacts of NUMA misses are significant, generally starting at a 10% performance hit or higher. vCPU pinning and `numatune` should be configured together.

If the virtual machine is performing storage or network I/O tasks, it can be beneficial to pin all vCPUs and memory to the same physical socket that is physically connected to the I/O adapter.

**Note**

The `lstopo` tool can be used to visualize NUMA topology. It can also help verify that vCPUs are binding to cores on the same physical socket. Refer to the following Knowledgebase article for more information on `lstopo`: [https://access.redhat.com/site/solutions/62879](https://access.redhat.com/site/solutions/62879).

**Important**

Pinning causes increased complexity where there are many more vCPUs than physical cores.

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:

```xml
<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'/>
```
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>`:

```xml
<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>
```

### Important

`<vcpupin>`, `<numatune>`, and `<emulatorpin>` should be configured together to achieve optimal, deterministic performance. For more information on the `<numatune>` tag, see Section 7.4.2, “Domain Processes”. For more information on the `<emulatorpin>` tag, see Section 7.4.4, “Using emulatorpin”.

#### 7.4.2. 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:

```xml
<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:

```xml
<vcpu placement='auto'>8</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.

It is also possible to configure your system with numad to boot a selected number of vCPUs without pinning all vCPUs at startup.

For example, to enable only 8 vCPUs at boot on a system with 32 vCPUs, configure the XML similar to the following:

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

**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

### 7.4.3. Domain vCPU Threads

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

```
<cpuset>
  <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"/>
</cpuset>
```

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

### 7.4.4. Using emulatorpin

Another way of tuning the domain process pinning policy is to use the `<emulatorpin>` tag inside of `<cpuset>`.

The `<emulatorpin>` tag specifies which host physical CPUs the emulator (a subset of a domain, not including vCPUs) will be pinned to. The `<emulatorpin>` tag provides a method of setting a precise affinity to emulator thread processes. As a result, vhost threads run on the same subset of physical CPUs and memory, and therefore benefit from cache locality. For example:
<cputune>
  <emulatorpin cpuset="1-3"/>
</cputune>

**Note**

In Red Hat Enterprise Linux 7, automatic NUMA balancing is enabled by default. Automatic NUMA balancing reduces the need for manually tuning `<emulatorpin>`, since the vhost-net emulator thread follows the vCPU tasks more reliably. For more information about automatic NUMA balancing, see Section 7.3, “Automatic NUMA Balancing”.

### 7.4.5. 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 (`rhel7`) which has an ID of 1 to the physical CPU 2:

```
% virsh vcpupin rhel7 1 2
```

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

```
% virsh vcpupin rhel7
```

### 7.4.6. 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 rhel7 3-4
```

### 7.4.7. Tuning Domain Process Memory Policy with virsh

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

```
% virsh numatune rhel7 --nodeset 0-10
```

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

### 7.4.8. Guest NUMA Topology
Guest NUMA topology can be specified using the `<numa>` tag inside the `<cpu>` tag in the guest virtual machine's XML. Refer to the following example, and replace values accordingly:

```xml
<cpu>
  ...
  <numa>
    <cell cpus='0-3' memory='512000'/>
    <cell cpus='4-7' memory='512000'/>
  </numa>
  ...
</cpu>
```

Each `<cell>` element specifies a NUMA cell or a NUMA node. `cpus` specifies the CPU or range of CPUs that are part of the node, and `memory` specifies the node memory in kibibytes (blocks of 1024 bytes). Each cell or node is assigned a `cellid` or `nodeid` in increasing order starting from 0.

### 7.5. NUMA-Aware Kernel SamePage Merging (KSM)

Kernel SamePage Merging (KSM) allows virtual machines to share identical memory pages. KSM can detect that a system is using NUMA memory and control merging pages across different NUMA nodes.

Use the `sysfs /sys/kernel/mm/ksm/merge_across_nodes` parameter to control merging of pages across different NUMA nodes. By default, pages from all nodes can be merged together. When this parameter is set to zero, only pages from the same node are merged.

**Important**

When KSM merges across nodes on a NUMA host with multiple guest virtual machines, guests and CPUs from more distant nodes can suffer a significant increase of access latency to the merged KSM page.

To instruct the hypervisor to disable share pages for a guest, add the following to the guest's XML:

```xml
<memoryBacking>
  <nosharepages/>
</memoryBacking>
```
Chapter 8. Performance Monitoring Tools

8.1. Introduction

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

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:

```bash
# 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`:

   ```bash
   # 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:

   ```bash
   # 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`. 
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.

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 [.] __strlen_sse42
0.32% vi libc-2.12.so [.] __strlnc_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**
# 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

8.3. Virtual Performance Monitoring Unit (vPMU)

The virtual performance monitoring unit (vPMU) displays statistics which indicate how a guest virtual 
machine is functioning.

The virtual performance monitoring unit allows users to identify sources of possible performance problems 
in their guest virtual machines. The vPMU is based on Intel's PMU (Performance Monitoring Units) and can 
only be used on Intel machines.

This feature is only supported with guest virtual machines running Red Hat Enterprise Linux 6 or Red Hat 
Enterprise Linux 7 and is disabled by default.

To verify if the vPMU is supported on your system, check for the arch_perfmon flag on the host CPU by 
running:

```
# cat /proc/cpuinfo|grep arch_perfmon
```

To enable the vPMU, specify the cpu mode in the guest XML as host-passthrough:

```
# virsh dumpxml guest_name |grep "cpu mode"
<cpu mode='host-passthrough'>
```

After the vPMU is enabled, display a virtual machine's performance statistics by running the perf 
command from the guest virtual machine.
## Revision History

<table>
<thead>
<tr>
<th>Revision 0.1-50.405</th>
<th>Thu Jul 7 2014</th>
<th>Rüdiger Landmann</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add html-single and epub formats</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-50</th>
<th>Tues June 3 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version for 7.0 GA release</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-49</th>
<th>Tues June 3 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated screenshots in Virtual Machine Manager chapter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Updated support limit URLs in first chapter for <a href="https://example.com">BZ#1064612</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Added note about the nosharepages parameter to memory chapter for QE feedback in <a href="https://example.com">BZ#1064616</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-48</th>
<th>Fri May 23 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changed terminology and guest names in NUMA chapter for QE feedback in <a href="https://example.com">BZ#1064618</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edited blkiotune command, added the &quot;live&quot; and &quot;current&quot; options in Block I/O chapter for QE feedback in <a href="https://example.com">BZ#1064617</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edited tuned-adm list command output and added reference to tuned.conf man page in tuned chapter for QE feedback in <a href="https://example.com">BZ#1096793</a> and <a href="https://example.com">BZ#1096795</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-47</th>
<th>Tues May 20 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated support limit URLs in Introduction chapter.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-46</th>
<th>Mon May 19 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Removed documentation suite list from introduction chapter and preface material.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-45</th>
<th>Fri May 9 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rebuild for style changes.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-43</th>
<th>Wed April 30 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added description of emulatorpin; expanded note about configuring vcpupin, numatune, and emulatorpin together to NUMA chapter for <a href="https://example.com">BZ#1031911</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-42</th>
<th>Thurs April 17 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added numastat examples to NUMA chapter for <a href="https://example.com">BZ#1011786</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-41</th>
<th>Fri April 11 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edited overcommitting explanation, added numastat command to libvirt NUMA tuning section, and corrected wording on KSM for SME feedback in <a href="https://example.com">BZ#1011786</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-40</th>
<th>Wed April 9 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build for beta refresh.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-38</th>
<th>Fri April 4 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edits to memory chapter for <a href="https://example.com">BZ#1057397</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-37</th>
<th>Thurs April 3 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edits to memory chapter for <a href="https://example.com">BZ#1057397</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-36</th>
<th>Wed April 2 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added perf command to vPMU section in <a href="https://example.com">BZ#799750</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revision 0.1-35</td>
<td>Tues April 1 2014</td>
<td>Dayle Parker</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Edited IO mode options in Block I/O chapter; added cross-reference and new screenshot in virt-manager chapter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edited zero copy transmit description for SME feedback in <a href="#">BZ#794627</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edited vPMU commands for QE feedback in <a href="#">BZ#1059016</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-33</th>
<th>Thurs March 27 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edited author group.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-organized subsections within chapters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Added reference to Red Hat Enterprise Linux Performance Tuning Guide in introduction.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edited virt-manager chapter introduction.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edits to NUMA section for <a href="#">BZ#1031911</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-32</th>
<th>Wed March 26 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edited disk I/O throttling section for <a href="#">BZ#799753</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Added to blkiotune section for <a href="#">BZ#1031912</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edits throughout guide for SME feedback in <a href="#">BZ#1011786</a> and <a href="#">BZ#1031911</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-31</th>
<th>Mon March 24 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edited block I/O chapter for <a href="#">BZ#799753</a> and <a href="#">BZ#1031912</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edits throughout guide for SME feedback in <a href="#">BZ#1011786</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-30</th>
<th>Wed March 19 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated virtualization docs suite list to include new Red Hat Enterprise Virtualization documentation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edited tuned chapter based on SME feedback for <a href="#">BZ#952966</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edited vPMU section for QE feedback in <a href="#">BZ#1059016</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-29</th>
<th>Tues Mar 11 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Included virsh memtune command in memory chapter for <a href="#">BZ#1056915</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-28</th>
<th>Fri Mar 7 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edited transparent huge pages section and NUMA chapter on SME feedback.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-27</th>
<th>Thurs Mar 6 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edited guest NUMA topology section for <a href="#">BZ#1056917</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-26</th>
<th>Tues Mar 4 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edited memory chapter for <a href="#">BZ#1056915</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-25</th>
<th>Mon Mar 3 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added text about cgroups.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Added guest NUMA topology section for <a href="#">BZ#1056917</a>.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 0.1-24</th>
<th>Tues Feb 25 2014</th>
<th>Dayle Parker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edited multi-queue virtio-scsi description in introduction for <a href="#">BZ#1034020</a>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moved tuning-related content from Virtualization Deployment and Administration Guide to this guide.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Added Virtual Performance Monitoring Unit section.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Revision 0.1-22 | Fri Feb 14 2014 | Dayle Parker |
Minor edits for NUMA aware KSM section [BZ#949244](https://bugzilla.redhat.com/949244).
Edited terminology in NUMA introduction following SME review.

**Revision 0.1-21**  
**Thurs Feb 13 2014**  
Dayle Parker

Minor edits for [BZ#1056909](https://bugzilla.redhat.com/1056909).  
Minor edits for [BZ#1032535](https://bugzilla.redhat.com/1032535).

**Revision 0.1-19**  
**Fri Feb 7 2014**  
Dayle Parker

Created section for zero copy transmit and edited brief description for [BZ#794627](https://bugzilla.redhat.com/794627).  
Edited NUMA chapter for [BZ#1032535](https://bugzilla.redhat.com/1032535).

**Revision 0.1-17**  
**Wed Feb 5 2014**  
Dayle Parker

Edited multi-queue virtio-net section for [BZ#794608](https://bugzilla.redhat.com/794608), and [BZ#1056909](https://bugzilla.redhat.com/1056909).  
Edited NUMA chapter for [BZ#1032535](https://bugzilla.redhat.com/1032535) and [BZ#1056917](https://bugzilla.redhat.com/1056917).

**Revision 0.1-16**  
**Fri Jan 24 2014**  
Dayle Parker

Applied SME feedback to descriptions of Para-virtualized Time vsyscall Optimization ([BZ#1034029](https://bugzilla.redhat.com/1034029)), multi-queue virtio-net ([BZ#794608](https://bugzilla.redhat.com/794608)), and APIC virtualization ([BZ#799749](https://bugzilla.redhat.com/799749)).  
Corrected commands in Hugepages and tuned sections.

**Revision 0.1-13**  
**Tues Dec 10 2013**  
Dayle Parker

Edited automatic NUMA balancing section based on SME feedback in [BZ#1032535](https://bugzilla.redhat.com/1032535).

**Revision 0.1-10**  
**Fri Dec 6 2013**  
Dayle Parker

Version for beta release.

**Revision 0.1-9**  
**Thurs Dec 5 2013**  
Dayle Parker

Edited list of Red Hat Enterprise Linux 7 performance features.  
Edited automatic NUMA balancing section based on SME feedback in [BZ#1032535](https://bugzilla.redhat.com/1032535).

**Revision 0.1-8**  
**Wed Nov 27 2013**  
Dayle Parker

Edited huge pages section.  
Added disk I/O throttling to Performance Features list for [BZ#799753](https://bugzilla.redhat.com/799753).  
Added note to About This Guide about chapter structure.

**Revision 0.1-7**  
**Mon Nov 25 2013**  
Dayle Parker

Added multi-queue NIC support description for [BZ#794608](https://bugzilla.redhat.com/794608).  
Expanded list of Red Hat Enterprise Linux 7 performance features.

**Revision 0.1-6**  
**Wed Nov 20 2013**  
Dayle Parker

Added section for performance improvements in Red Hat Enterprise Linux 7.  
Added sections for automatic NUMA balancing and NUMA-aware KSM for [BZ#1032535](https://bugzilla.redhat.com/1032535) and [BZ#949244](https://bugzilla.redhat.com/949244).  
Added zero copy transmit description for [BZ#794627](https://bugzilla.redhat.com/794627).

**Revision 0.1-5**  
**Mon Nov 4 2013**  
Dayle Parker

Added performance profile information to tuned section for [BZ#952966](https://bugzilla.redhat.com/952966).

**Revision 0.1-4**  
**Thurs Oct 31 2013**  
Dayle Parker

Updated virtualization documentation suite list in Chapter 1.

**Revision 0.1-3**  
**Wed Oct 30 2013**  
Dayle Parker
Included updates and wording improvements throughout from Red Hat Enterprise Linux 6 version of guide. Created tuned chapter from existing content. Adjusted image scaling parameters throughout guide.

**Revision 0.1-2**  **Wed Jan 23 2013**  **Scott Radvan**
Send to internal preview site. Initial build for version 7.

**Revision 0.1-1**  **Thu Jan 17 2013**  **Scott Radvan**
Branched from the Red Hat Enterprise Linux 6 version of the document.