Red Hat Enterprise Linux 8

Monitoring and managing system status and performance

Optimizing system throughput, latency, and power consumption

Last Updated: 2019-09-12

Red Hat Enterprise Linux 8 Monitoring and managing system status and performance

Optimizing system throughput, latency, and power consumption

Legal Notice

Abstract

This documentation collection provides instructions on how to monitor and optimize the throughput, latency, and power consumption of Red Hat Enterprise Linux 8 in different scenarios.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages, make sure you are viewing the documentation in the Multi-page HTML format. Highlight the part of text that you want to comment on. Then, click the **Add Feedback** pop-up that appears below the highlighted text, and follow the displayed instructions.

- For submitting more complex feedback, create a Bugzilla ticket:

  1. Go to the Bugzilla website.

  2. As the Component, use **Documentation**.

  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.

  4. Click **Submit Bug**.

CHAPTER 1. GETTING STARTED WITH TUNED

As a system administrator, you can use the **Tuned** application to optimize the performance profile of your system for a variety of use cases.

1.1. THE PURPOSE OF TUNED

Tuned is a service that monitors your system and optimizes the performance under certain workloads. The core of **Tuned** are *profiles*, which tune your system for different use cases.

Tuned is distributed with a number of predefined profiles for use cases such as:

- High throughput
- Low latency
- Saving power

It is possible to modify the rules defined for each profile and customize how to tune a particular device. When you switch to another profile or deactivate **Tuned**, all changes made to the system settings by the previous profile revert back to their original state.

You can also configure **Tuned** to react to changes in device usage and adjusts settings to improve performance of active devices and reduce power consumption of inactive devices.

1.2. TUNED PROFILES

A detailed analysis of a system can be very time-consuming. **Tuned** provides a number of predefined profiles for typical use cases. You can also create, modify, and delete profiles.

The profiles provided with **Tuned** are divided into the following categories:

- Power-saving profiles
- Performance-boosting profiles

The performance-boosting profiles include profiles that focus on the following aspects:

- Low latency for storage and network
- High throughput for storage and network
- Virtual machine performance
- Virtualization host performance

The default profile

During the installation, the best profile for your system is selected automatically. Currently, the default profile is selected according to the following customizable rules:

Environment	Default profile	Goal
Compute nodes	**throughput-performance**	The best throughput performance

Environment	Default profile	Goal
Virtual machines	**virtual-guest**	The best performance. If you are not interested in the best performance, you can change it to the **balanced** or **powersave** profile.
Other cases	**balanced**	Balanced performance and power consumption

Merged profiles

As an experimental feature, it is possible to select more profiles at once. **Tuned** will try to merge them during the load.

If there are conflicts, the settings from the last specified profile takes precedence.

Example 1.1. Low power consumption in a virtual guest

The following example optimizes the system to run in a virtual machine for the best performance and concurrently tunes it for low power consumption, while the low power consumption is the priority:

```
# tuned-adm profile virtual-guest powersave
```

WARNING

Merging is done automatically without checking whether the resulting combination of parameters makes sense. Consequently, the feature might tune some parameters the opposite way, which might be counterproductive: for example, setting the disk for high throughput by using the **throughput-performance** profile and concurrently setting the disk spindown to the low value by the **spindown-disk** profile.

The location of profiles

Tuned stores profiles in the following directories:

/usr/lib/tuned/

Distribution-specific profiles are stored in the directory. Each profile has its own directory. The profile consists of the main configuration file called **tuned.conf**, and optionally other files, for example helper scripts.

/etc/tuned/

If you need to customize a profile, copy the profile directory into the directory, which is used for custom profiles. If there are two profiles of the same name, the custom profile located in **/etc/tuned/** is used.

The syntax of profile configuration

The **tuned.conf** file can contain one **[main]** section and other sections for configuring plug-in instances. However, all sections are optional.

Lines starting with the hash sign (**#**) are comments.

Additional resources

- The **tuned.conf(5)** man page.

1.3. TUNED PROFILES DISTRIBUTED WITH RHEL

The following is a list of profiles that are installed with **Tuned** on Red Hat Enterprise Linux.

NOTE

There might be more product-specific or third-party **Tuned** profiles available. Such profiles are usually provided by separate RPM packages.

balanced

The default power-saving profile. It is intended to be a compromise between performance and power consumption. It uses auto-scaling and auto-tuning whenever possible. The only drawback is the increased latency. In the current **Tuned** release, it enables the CPU, disk, audio, and video plugins, and activates the **conservative** CPU governor. The **radeon_powersave** option uses the **dpm-balanced** value if it is supported, otherwise it is set to **auto**.

powersave

A profile for maximum power saving performance. It can throttle the performance in order to minimize the actual power consumption. In the current **Tuned** release it enables USB autosuspend, WiFi power saving, and Aggressive Link Power Management (ALPM) power savings for SATA host adapters. It also schedules multi-core power savings for systems with a low wakeup rate and activates the **ondemand** governor. It enables AC97 audio power saving or, depending on your system, HDA-Intel power savings with a 10 seconds timeout. If your system contains a supported Radeon graphics card with enabled KMS, the profile configures it to automatic power saving. On ASUS Eee PCs, a dynamic Super Hybrid Engine is enabled.

NOTE

In certain cases, the **balanced** profile is more efficient compared to the **powersave** profile.

Consider there is a defined amount of work that needs to be done, for example a video file that needs to be transcoded. Your machine might consume less energy if the transcoding is done on the full power, because the task is finished quickly, the machine starts to idle, and it can automatically step-down to very efficient power save modes. On the other hand, if you transcode the file with a throttled machine, the machine consumes less power during the transcoding, but the process takes longer and the overall consumed energy can be higher.

That is why the **balanced** profile can be generally a better option.

throughput-performance

A server profile optimized for high throughput. It disables power savings mechanisms and enables **sysctl** settings that improve the throughput performance of the disk and network IO. CPU governor is set to **performance**.

latency-performance

A server profile optimized for low latency. It disables power savings mechanisms and enables **sysctl** settings that improve latency. CPU governor is set to **performance** and the CPU is locked to the low C states (by PM QoS).

network-latency

A profile for low latency network tuning. It is based on the **latency-performance** profile. It additionally disables transparent huge pages and NUMA balancing, and tunes several other network-related **sysctl** parameters.

network-throughput

A profile for throughput network tuning. It is based on the **throughput-performance** profile. It additionally increases kernel network buffers.

virtual-guest

A profile designed for virtual guests based on the **throughput-performance** profile that, among other tasks, decreases virtual memory swappiness and increases disk readahead values. It does not disable disk barriers.

virtual-host

A profile designed for virtual hosts based on the **throughput-performance** profile that, among other tasks, decreases virtual memory swappiness, increases disk readahead values, and enables a more aggressive value of dirty pages writeback.

oracle

A profile optimized for Oracle databases loads based on **throughput-performance** profile. It additionally disables transparent huge pages and modifies other performance-related kernel parameters. This profile is provided by the **tuned-profiles-oracle** package.

desktop

A profile optimized for desktops, based on the **balanced** profile. It additionally enables scheduler autogroups for better response of interactive applications.

Real-time profiles

Real-time profiles are intended for systems running the real-time kernel. Without a special kernel build, they do not configure the system to be real-time. On RHEL, the profiles are available from additional repositories.

The following real-time profiles are available:

realtime

Use on bare-metal real-time systems.
Provided by the **tuned-profiles-realtime** package, which is available from the RT or NFV repositories.

realtime-virtual-host

Use in a virtualization host configured for real-time.
Provided by the **tuned-profiles-nfv-host** package, which is available from the NFV repository.

realtime-virtual-guest

Use in a virtualization guest configured for real-time.
Provided by the **tuned-profiles-nfv-guest** package, which is available from the NFV repository.

1.4. STATIC AND DYNAMIC TUNING IN TUNED

This section explains the difference between the two categories of system tuning that **Tuned** applies: *static* and *dynamic*.

Static tuning

Mainly consists of the application of predefined **sysctl** and **sysfs** settings and one-shot activation of several configuration tools such as **ethtool**.

Dynamic tuning

Watches how various system components are used throughout the uptime of your system. **Tuned** adjusts system settings dynamically based on that monitoring information.
For example, the hard drive is used heavily during startup and login, but is barely used later when the user might mainly work with applications such as web browsers or email clients. Similarly, the CPU and network devices are used differently at different times. **Tuned** monitors the activity of these components and reacts to the changes in their use.

By default, dynamic tuning is disabled. To enable it, edit the **/etc/tuned/tuned-main.conf** file and change the **dynamic_tuning** option to **1**. **Tuned** then periodically analyzes system statistics and uses them to update your system tuning settings. To configure the time interval in seconds between these updates, use the **update_interval** option.

Currently implemented dynamic tuning algorithms try to balance the performance and powersave, and are therefore disabled in the performance profiles. Dynamic tuning for individual plug-ins can be enabled or disabled in the **Tuned** profiles.

Example 1.2. Static and dynamic tuning on a workstation

On a typical office workstation, the Ethernet network interface is inactive most of the time. Only a few emails go in and out or some web pages might be loaded.

For those kinds of loads, the network interface does not have to run at full speed all the time, as it does by default. **Tuned** has a monitoring and tuning plug-in for network devices that can detect this low activity and then automatically lower the speed of that interface, typically resulting in a lower power usage.

If the activity on the interface increases for a longer period of time, for example because a DVD image is being downloaded or an email with a large attachment is opened, **Tuned** detects this and sets the interface speed to maximum to offer the best performance while the activity level is high.

This principle is used for other plug-ins for CPU and disks as well.

1.5. TUNED NO-DAEMON MODE

You can run **Tuned** in **no-daemon** mode, which does not require any resident memory. In this mode, **Tuned** applies the settings and exits.

By default, **no-daemon** mode is disabled because a lot of **Tuned** functionality is missing in this mode, including:

- D-Bus support
- Hot-plug support

- Rollback support for settings

To enable **no-daemon** mode, include the following line in the **/etc/tuned/tuned-main.conf** file:

```
daemon = 0
```

1.6. INSTALLING AND ENABLING TUNED

This procedure installs and enables the **Tuned** application, installs **Tuned** profiles, and presets a default **Tuned** profile for your system.

Procedure

1. Install the **tuned** package:

   ```
   # yum install tuned
   ```

2. Enable and start the **tuned** service:

   ```
   # systemctl enable --now tuned
   ```

3. Optionally, install **Tuned** profiles for real-time systems:

   ```
   # yum install tuned-profiles-realtime tuned-profiles-nfv
   ```

4. Verify that a **Tuned** profile is active and applied:

   ```
   $ tuned-adm active

   Current active profile: balanced
   ```

   ```
   $ tuned-adm verify

   Verfication succeeded, current system settings match the preset profile.
   See tuned log file ('/var/log/tuned/tuned.log') for details.
   ```

1.7. LISTING AVAILABLE TUNED PROFILES

This procedure lists all **Tuned** profiles that are currently available on your system.

Procedure

- To list all available **Tuned** profiles on your system, use:

  ```
  $ tuned-adm list

  Available profiles:
  - balanced              - General non-specialized tuned profile
  - desktop               - Optimize for the desktop use-case
  - latency-performance   - Optimize for deterministic performance at the cost of increased
  power consumption
  - network-latency       - Optimize for deterministic performance at the cost of increased power
  ```

```
consumption, focused on low latency network performance
- network-throughput     - Optimize for streaming network throughput, generally only
necessary on older CPUs or 40G+ networks
- powersave              - Optimize for low power consumption
- throughput-performance - Broadly applicable tuning that provides excellent performance
across a variety of common server workloads
- virtual-guest          - Optimize for running inside a virtual guest
- virtual-host           - Optimize for running KVM guests
Current active profile: balanced
```

- To display only the currently active profile, use:

```
$ tuned-adm active

Current active profile: balanced
```

Additional resources

- The **tuned-adm(8)** man page.

1.8. SETTING A TUNED PROFILE

This procedure activates a selected **Tuned** profile on your system.

Prerequisites

- The **tuned** service is running. See Section 1.6, "Installing and enabling Tuned" for details.

Procedure

1. Optionally, you can let **Tuned** recommend the most suitable profile for your system:

```
# tuned-adm recommend

balanced
```

2. Activate a profile:

```
# tuned-adm profile selected-profile
```

Alternatively, you can activate a combination of multiple profiles:

```
# tuned-adm profile profile1 profile2
```

Example 1.3. A virtual machine optimized for low power consumption

The following example optimizes the system to run in a virtual machine with the best performance and concurrently tunes it for low power consumption, while the low power consumption is the priority:

```
# tuned-adm profile virtual-guest powersave
```

3. Verify that the **Tuned** profile is active and applied:

```
$ tuned-adm active

Current active profile: selected-profile
```

```
$ tuned-adm verify

Verfication succeeded, current system settings match the preset profile.
See tuned log file ('/var/log/tuned/tuned.log') for details.
```

Additional resources

- The **tuned-adm(8)** man page

1.9. DISABLING TUNED

This procedure disables **Tuned** and resets all affected system settings to their original state before **Tuned** modified them.

Procedure

- To disable all tunings temporarily:

  ```
  # tuned-adm off
  ```

  The tunings are applied again after the **tuned** service restarts.

- Alternatively, to stop and disable the **tuned** service permanently:

  ```
  # systemctl disable --now tuned
  ```

Additional resources

- The **tuned-adm(8)** man page.

1.10. RELATED INFORMATION

- The **tuned(8)** man page
- The **tuned-adm(8)** man page
- The **Tuned** project website: https://tuned-project.org/

CHAPTER 2. CUSTOMIZING TUNED PROFILES

You can create or modify **Tuned** profiles to optimize system performance for your intended use case.

2.1. PREREQUISITES

- Install and enable **Tuned** as described in Section 1.6, "Installing and enabling Tuned" .

2.2. TUNED PROFILES

A detailed analysis of a system can be very time-consuming. **Tuned** provides a number of predefined profiles for typical use cases. You can also create, modify, and delete profiles.

The profiles provided with **Tuned** are divided into the following categories:

- Power-saving profiles
- Performance-boosting profiles

The performance-boosting profiles include profiles that focus on the following aspects:

- Low latency for storage and network
- High throughput for storage and network
- Virtual machine performance
- Virtualization host performance

The default profile

During the installation, the best profile for your system is selected automatically. Currently, the default profile is selected according to the following customizable rules:

Environment	Default profile	Goal
Compute nodes	**throughput-performance**	The best throughput performance
Virtual machines	**virtual-guest**	The best performance. If you are not interested in the best performance, you can change it to the **balanced** or **powersave** profile.
Other cases	**balanced**	Balanced performance and power consumption

Merged profiles

As an experimental feature, it is possible to select more profiles at once. **Tuned** will try to merge them during the load.

If there are conflicts, the settings from the last specified profile takes precedence.

Example 2.1. Low power consumption in a virtual guest

The following example optimizes the system to run in a virtual machine for the best performance and concurrently tunes it for low power consumption, while the low power consumption is the priority:

```
# tuned-adm profile virtual-guest powersave
```

WARNING

Merging is done automatically without checking whether the resulting combination of parameters makes sense. Consequently, the feature might tune some parameters the opposite way, which might be counterproductive: for example, setting the disk for high throughput by using the **throughput-performance** profile and concurrently setting the disk spindown to the low value by the **spindown-disk** profile.

The location of profiles

Tuned stores profiles in the following directories:

/usr/lib/tuned/

Distribution-specific profiles are stored in the directory. Each profile has its own directory. The profile consists of the main configuration file called **tuned.conf**, and optionally other files, for example helper scripts.

/etc/tuned/

If you need to customize a profile, copy the profile directory into the directory, which is used for custom profiles. If there are two profiles of the same name, the custom profile located in **/etc/tuned/** is used.

The syntax of profile configuration

The **tuned.conf** file can contain one **[main]** section and other sections for configuring plug-in instances. However, all sections are optional.

Lines starting with the hash sign (**#**) are comments.

Additional resources

- The **tuned.conf(5)** man page.

2.3. INHERITANCE BETWEEN TUNED PROFILES

Tuned profiles can be based on other profiles and modify only certain aspects of their parent profile.

The **[main]** section of **Tuned** profiles recognizes the **include** option:

```
[main]
include=parent
```

All settings from the *parent* profile are loaded in this *child* profile. In the following sections, the *child* profile can override certain settings inherited from the *parent* profile or add new settings not present in the *parent* profile.

You can create your own *child* profile in the **/etc/tuned/** directory based on a pre-installed profile in **/usr/lib/tuned/** with only some parameters adjusted.

If the *parent* profile is updated, such as after a **Tuned** upgrade, the changes are reflected in the *child* profile.

Example 2.2. A power-saving profile based on balanced

The following is an example of a custom profile that extends the **balanced** profile and sets Aggressive Link Power Management (ALPM) for all devices to the maximum powersaving.

```
[main]
include=balanced

[scsi_host]
alpm=min_power
```

Additional resources

- The **tuned.conf(5)** man page

2.4. STATIC AND DYNAMIC TUNING IN TUNED

This section explains the difference between the two categories of system tuning that **Tuned** applies: *static* and *dynamic*.

Static tuning

Mainly consists of the application of predefined **sysctl** and **sysfs** settings and one-shot activation of several configuration tools such as **ethtool**.

Dynamic tuning

Watches how various system components are used throughout the uptime of your system. **Tuned** adjusts system settings dynamically based on that monitoring information.
For example, the hard drive is used heavily during startup and login, but is barely used later when the user might mainly work with applications such as web browsers or email clients. Similarly, the CPU and network devices are used differently at different times. **Tuned** monitors the activity of these components and reacts to the changes in their use.

By default, dynamic tuning is disabled. To enable it, edit the **/etc/tuned/tuned-main.conf** file and change the **dynamic_tuning** option to **1**. **Tuned** then periodically analyzes system statistics and uses them to update your system tuning settings. To configure the time interval in seconds between these updates, use the **update_interval** option.

Currently implemented dynamic tuning algorithms try to balance the performance and powersave, and are therefore disabled in the performance profiles. Dynamic tuning for individual plug-ins can be enabled or disabled in the **Tuned** profiles.

Example 2.3. Static and dynamic tuning on a workstation

> On a typical office workstation, the Ethernet network interface is inactive most of the time. Only a few emails go in and out or some web pages might be loaded.
>
> For those kinds of loads, the network interface does not have to run at full speed all the time, as it does by default. **Tuned** has a monitoring and tuning plug-in for network devices that can detect this low activity and then automatically lower the speed of that interface, typically resulting in a lower power usage.
>
> If the activity on the interface increases for a longer period of time, for example because a DVD image is being downloaded or an email with a large attachment is opened, **Tuned** detects this and sets the interface speed to maximum to offer the best performance while the activity level is high.
>
> This principle is used for other plug-ins for CPU and disks as well.

2.5. TUNED PLUG-INS

Plug-ins are modules in **Tuned** profiles that **Tuned** uses to monitor or optimize different devices on the system.

Tuned uses two types of plug-ins:

- monitoring plug-ins
- tuning plug-ins

Monitoring plug-ins

Monitoring plug-ins are used to get information from a running system. The output of the monitoring plug-ins can be used by tuning plug-ins for dynamic tuning.

Monitoring plug-ins are automatically instantiated whenever their metrics are needed by any of the enabled tuning plug-ins. If two tuning plug-ins require the same data, only one instance of the monitoring plug-in is created and the data is shared.

Tuning plug-ins

Each tuning plug-in tunes an individual subsystem and takes several parameters that are populated from the tuned profiles. Each subsystem can have multiple devices, such as multiple CPUs or network cards, that are handled by individual instances of the tuning plug-ins. Specific settings for individual devices are also supported.

Syntax for plug-ins in Tuned profiles

Sections describing plug-in instances are formatted in the following way:

```
[NAME]
type=TYPE
devices=DEVICES
```

NAME

: is the name of the plug-in instance as it is used in the logs. It can be an arbitrary string.

TYPE

: is the type of the tuning plug-in.

DEVICES

: is the list of devices that this plug-in instance handles.

The **devices** line can contain a list, a wildcard (*****), and negation (**!**). If there is no **devices** line, all devices present or later attached on the system of the *TYPE* are handled by the plug-in instance. This is same as using the **devices=*** option.

Example 2.4. Matching block devices with a plug-in

The following example matches all block devices starting with **sd**, such as **sda** or **sdb**, and does not disable barriers on them:

```
[data_disk]
type=disk
devices=sd*
disable_barriers=false
```

The following example matches all block devices except **sda1** and **sda2**:

```
[data_disk]
type=disk
devices=!sda1, !sda2
disable_barriers=false
```

If no instance of a plug-in is specified, the plug-in is not enabled.

If the plug-in supports more options, they can be also specified in the plug-in section. If the option is not specified and it was not previously specified in the included plug-in, the default value is used.

Short plug-in syntax

If you do not need custom names for the plug-in instance and there is only one definition of the instance in your configuration file, **Tuned** supports the following short syntax:

```
[TYPE]
devices=DEVICES
```

In this case, it is possible to omit the **type** line. The instance is then referred to with a name, same as the type. The previous example could be then rewritten into:

Example 2.5. Matching block devices using the short syntax

```
[disk]
devices=sdb*
disable_barriers=false
```

Conflicting plug-in definitions in a profile

If the same section is specified more than once using the **include** option, the settings are merged. If they cannot be merged due to a conflict, the last conflicting definition overrides the previous settings. If you do not know what was previously defined, you can use the **replace** Boolean option and set it to **true**. This causes all the previous definitions with the same name to be overwritten and the merge does not happen.

You can also disable the plug-in by specifying the **enabled=false** option. This has the same effect as if the instance was never defined. Disabling the plug-in is useful if you are redefining the previous definition from the **include** option and do not want the plug-in to be active in your custom profile.

Functionality not implemented in any plug-in

Tuned includes the ability to run any shell command as part of enabling or disabling a tuning profile. This enables you to extend **Tuned** profiles with functionality that has not been integrated into Tuned yet.

You can specify arbitrary shell commands using the **script** plug-in.

Additional resources

- The **tuned.conf(5)** man page

2.6. AVAILABLE TUNED PLUG-INS

This section lists all monitoring and tuning plug-ins currently available in **Tuned**.

Monitoring plug-ins

Currently, the following monitoring plug-ins are implemented:

disk

Gets disk load (number of IO operations) per device and measurement interval.

net

Gets network load (number of transferred packets) per network card and measurement interval.

load

Gets CPU load per CPU and measurement interval.

Tuning plug-ins

Currently, the following tuning plug-ins are implemented. Only some of these plug-ins implement dynamic tuning. Options supported by plug-ins are also listed:

cpu

Sets the CPU governor to the value specified by the **governor** option and dynamically changes the Power Management Quality of Service (PM QoS) CPU Direct Memory Access (DMA) latency according to the CPU load.
If the CPU load is lower than the value specified by the **load_threshold** option, the latency is set to the value specified by the **latency_high** option, otherwise it is set to the value specified by **latency_low**.

You can also force the latency to a specific value and prevent it from dynamically changing further. To do so, set the **force_latency** option to the required latency value.

eeepc_she

Dynamically sets the front-side bus (FSB) speed according to the CPU load.
This feature can be found on some netbooks and is also known as the ASUS Super Hybrid Engine (SHE).

If the CPU load is lower or equal to the value specified by the **load_threshold_powersave** option, the plug-in sets the FSB speed to the value specified by the **she_powersave** option. If the CPU load is higher or equal to the value specified by the **load_threshold_normal** option, it sets the FSB speed to the value specified by the **she_normal** option.

Static tuning is not supported and the plug-in is transparently disabled if **Tuned** does not detect the hardware support for this feature.

net

Configures the Wake-on-LAN functionality to the values specified by the **wake_on_lan** option. It uses the same syntax as the **ethtool** utility. It also dynamically changes the interface speed according to the interface utilization.

sysctl

Sets various **sysctl** settings specified by the plug-in options.
The syntax is ***name*=*value***, where *name* is the same as the name provided by the **sysctl** utility.

Use the **sysctl** plug-in if you need to change system settings that are not covered by other plug-ins available in **Tuned**. If the settings are covered by some specific plug-ins, prefer these plug-ins.

usb

Sets autosuspend timeout of USB devices to the value specified by the **autosuspend** parameter.
The value **0** means that autosuspend is disabled.

vm

Enables or disables transparent huge pages depending on the Boolean value of the **transparent_hugepages** option.

audio

Sets the autosuspend timeout for audio codecs to the value specified by the **timeout** option.
Currently, the **snd_hda_intel** and **snd_ac97_codec** codecs are supported. The value **0** means that the autosuspend is disabled. You can also enforce the controller reset by setting the Boolean option **reset_controller** to **true**.

disk

Sets the disk elevator to the value specified by the **elevator** option.
It also sets:

- APM to the value specified by the **apm** option
- Scheduler quantum to the value specified by the **scheduler_quantum** option
- Disk spindown timeout to the value specified by the **spindown** option
- Disk readahead to the value specified by the **readahead** parameter
- The current disk readahead to a value multiplied by the constant specified by the **readahead_multiply** option

In addition, this plug-in dynamically changes the advanced power management and spindown timeout setting for the drive according to the current drive utilization. The dynamic tuning can be controlled by the Boolean option **dynamic** and is enabled by default.

scsi_host

Tunes options for SCSI hosts.
It sets Aggressive Link Power Management (ALPM) to the value specified by the **alpm** option.

mounts

Enables or disables barriers for mounts according to the Boolean value of the **disable_barriers** option.

script

Executes an external script or binary when the profile is loaded or unloaded. You can choose an arbitrary executable.

IMPORTANT

The **script** plug-in is provided mainly for compatibility with earlier releases. Prefer other **Tuned** plug-ins if they cover the required functionality.

Tuned calls the executable with one of the following arguments:

- **start** when loading the profile
- **stop** when unloading the profile

You need to correctly implement the **stop** action in your executable and revert all settings that you changed during the **start** action. Otherwise, the roll-back step after changing your **Tuned** profile will not work.

Bash scripts can import the **/usr/lib/tuned/functions** Bash library and use the functions defined there. Use these functions only for functionality that is not natively provided by **Tuned**. If a function name starts with an underscore, such as **_wifi_set_power_level**, consider the function private and do not use it in your scripts, because it might change in the future.

Specify the path to the executable using the **script** parameter in the plug-in configuration.

Example 2.6. Running a Bash script from a profile

To run a Bash script named **script.sh** that is located in the profile directory, use:

```
[script]
script=${i:PROFILE_DIR}/script.sh
```

sysfs

Sets various **sysfs** settings specified by the plug-in options.
The syntax is ***name=value***, where *name* is the **sysfs** path to use.

Use this plugin in case you need to change some settings that are not covered by other plug-ins. Prefer specific plug-ins if they cover the required settings.

video

Sets various powersave levels on video cards. Currently, only the Radeon cards are supported.
The powersave level can be specified by using the **radeon_powersave** option. Supported values are:

- **default**
- **auto**
- **low**

- **mid**
- **high**
- **dynpm**
- **dpm-battery**
- **dpm-balanced**
- **dpm-perfomance**

For details, see www.x.org. Note that this plug-in is experimental and the option might change in future releases.

bootloader

Adds options to the kernel command line. This plug-in supports only the GRUB 2 boot loader. Customized non-standard location of the GRUB 2 configuration file can be specified by the **grub2_cfg_file** option.

The kernel options are added to the current GRUB configuration and its templates. The system needs to be rebooted for the kernel options to take effect.

Switching to another profile or manually stopping the **tuned** service removes the additional options. If you shut down or reboot the system, the kernel options persist in the **grub.cfg** file.

The kernel options can be specified by the following syntax:

```
cmdline=arg1 arg2 ... argN
```

Example 2.7. Modifying the kernel command line

For example, to add the **quiet** kernel option to a **Tuned** profile, include the following lines in the **tuned.conf** file:

```
[bootloader]
cmdline=quiet
```

The following is an example of a custom profile that adds the **isolcpus=2** option to the kernel command line:

```
[bootloader]
cmdline=isolcpus=2
```

2.7. VARIABLES AND BUILT-IN FUNCTIONS IN TUNED PROFILES

Variables and built-in functions expand at run time when a **Tuned** profile is activated.

Using **Tuned** variables reduces the amount of necessary typing in **Tuned** profiles. You can also:

- Use various built-in functions together with **Tuned** variables

- Create custom functions in Python and add them to **Tuned** in the form of plug-ins

Variables

There are no predefined variables in **Tuned** profiles. You can define your own variables by creating the **[variables]** section in a profile and using the following syntax:

```
[variables]

variable_name=value
```

To expand the value of a variable in a profile, use the following syntax:

```
${variable_name}
```

Example 2.8. Isolating CPU cores using variables

In the following example, the **${isolated_cores}** variable expands to **1,2**; hence the kernel boots with the **isolcpus=1,2** option:

```
[variables]
isolated_cores=1,2

[bootloader]
cmdline=isolcpus=${isolated_cores}
```

The variables can be specified in a separate file. For example, you can add the following lines to **tuned.conf**:

```
[variables]
include=/etc/tuned/my-variables.conf

[bootloader]
cmdline=isolcpus=${isolated_cores}
```

If you add the **isolated_cores=1,2** option to the **/etc/tuned/my-variables.conf** file, the kernel boots with the **isolcpus=1,2** option.

Functions

To call a function, use the following syntax:

```
${f:function_name:argument_1:argument_2}
```

To expand the directory path where the profile and the **tuned.conf** file are located, use the **PROFILE_DIR** function, which requires special syntax:

```
${i:PROFILE_DIR}
```

Example 2.9. Isolating CPU cores using variables and built-in functions

In the following example, the **${non_isolated_cores}** variable expands to **0,3-5**, and the **cpulist_invert** built-in function is called with the **0,3-5** argument:

```
[variables]
non_isolated_cores=0,3-5

[bootloader]
cmdline=isolcpus=${f:cpulist_invert:${non_isolated_cores}}
```

The **cpulist_invert** function inverts the list of CPUs. For a 6-CPU machine, the inversion is **1,2**, and the kernel boots with the **isolcpus=1,2** command-line option.

Additional resources

- The **tuned.conf(5)** man page

2.8. BUILT-IN FUNCTIONS AVAILABLE IN TUNED PROFILES

The following built-in functions are available in all **Tuned** profiles:

PROFILE_DIR
: Returns the directory path where the profile and the **tuned.conf** file are located.

exec
: Executes a process and returns its output.

assertion
: Compares two arguments. If they *do not match*, the function logs text from the first argument and aborts profile loading.

assertion_non_equal
: Compares two arguments. If they *match*, the function logs text from the first argument and aborts profile loading.

kb2s
: Converts kilobytes to disk sectors.

s2kb
: Converts disk sectors to kilobytes.

strip
: Creates a string from all passed arguments and deletes both leading and trailing white space.

virt_check
: Checks whether **Tuned** is running inside a virtual machine (VM) or on bare metal:

  - Inside a VM, the function returns the first argument.
  - On bare metal, the function returns the second argument, even in case of an error.

cpulist_invert
: Inverts a list of CPUs to make its complement. For example, on a system with 4 CPUs, numbered from 0 to 3, the inversion of the list **0,2,3** is **1**.

cpulist2hex
: Converts a CPU list to a hexadecimal CPU mask.

cpulist2hex_invert
: Converts a CPU list to a hexadecimal CPU mask and inverts it.

hex2cpulist

Converts a hexadecimal CPU mask to a CPU list.

cpulist_online

Checks whether the CPUs from the list are online. Returns the list containing only online CPUs.

cpulist_present

Checks whether the CPUs from the list are present. Returns the list containing only present CPUs.

cpulist_unpack

Unpacks a CPU list in the form of **1-3,4** to **1,2,3,4**.

cpulist_pack

Packs a CPU list in the form of **1,2,3,5** to **1-3,5**.

2.9. CREATING NEW TUNED PROFILES

This procedure creates a new **Tuned** profile with custom performance rules.

Prerequisites

- The **tuned** service is installed and running. See Section 1.6, “Installing and enabling Tuned” for details.

Procedure

1. In the **/etc/tuned/** directory, create a new directory named the same as the profile that you want to create:

```
# mkdir /etc/tuned/my-profile
```

2. In the new directory, create a file named **tuned.conf**. Add a **[main]** section and plug-in definitions in it, according to your requirements.
For example, see the configuration of the **balanced** profile:

```
[main]
summary=General non-specialized tuned profile

[cpu]
governor=conservative
energy_perf_bias=normal

[audio]
timeout=10

[video]
radeon_powersave=dpm-balanced, auto

[scsi_host]
alpm=medium_power
```

3. To activate the profile, use:

```
# tuned-adm profile my-profile
```

4. Verify that the **Tuned** profile is active and the system settings are applied:

```
$ tuned-adm active

Current active profile: my-profile
```

```
$ tuned-adm verify

Verfication succeeded, current system settings match the preset profile.
See tuned log file ('/var/log/tuned/tuned.log') for details.
```

Additional resources

- The **tuned.conf(5)** man page

2.10. MODIFYING EXISTING TUNED PROFILES

This procedure creates a modified child profile based on an existing **Tuned** profile.

Prerequisites

- The **tuned** service is installed and running. See Section 1.6, "Installing and enabling Tuned" for details.

Procedure

1. In the **/etc/tuned/** directory, create a new directory named the same as the profile that you want to create:

```
# mkdir /etc/tuned/modified-profile
```

2. In the new directory, create a file named **tuned.conf**, and set the **[main]** section as follows:

```
[main]
include=parent-profile
```

Replace *parent-profile* with the name of the profile you are modifying.

3. Include your profile modifications.

Example 2.10. Lowering swappiness in the throughput-performance profile

To use the settings from the **throughput-performance** profile and change the value of **vm.swappiness** to 5, instead of the default 10, use:

```
[main]
include=throughput-performance

[sysctl]
vm.swappiness=5
```

4. To activate the profile, use:

```
# tuned-adm profile modified-profile
```

5. Verify that the **Tuned** profile is active and the system settings are applied:

```
$ tuned-adm active

Current active profile: my-profile
```

```
$ tuned-adm verify

Verfication succeeded, current system settings match the preset profile.
See tuned log file ('/var/log/tuned/tuned.log') for details.
```

Additional resources

- The **tuned.conf(5)** man page

2.11. RELATED INFORMATION

- The **tuned.conf(5)** man page
- The **Tuned** project website: https://tuned-project.org/

CHAPTER 3. USING THE WEB CONSOLE FOR SELECTING PERFORMANCE PROFILES

Red Hat Enterprise Linux 8 includes performance profiles optimizing:

- Systems using Desktop
- Latency performance
- Network performance
- Low power consumption
- Virtual machines

The following procedure describes setting up performance profiles in the web console.

The RHEL 8 web console configures the **tuned** service.

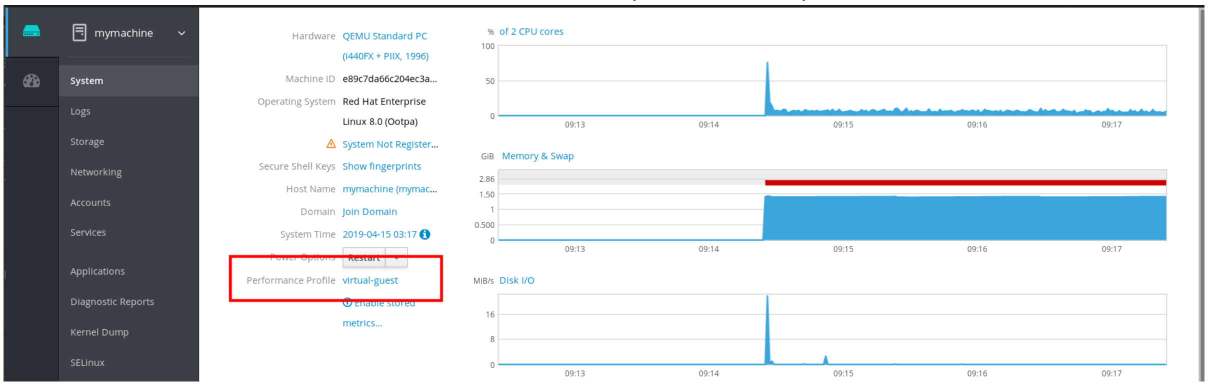
For details about the **tuned** service, see Monitoring and managing system status and performance .

Prerequisites

- The web console must be installed and accessible.
  For details, see Installing the web console.

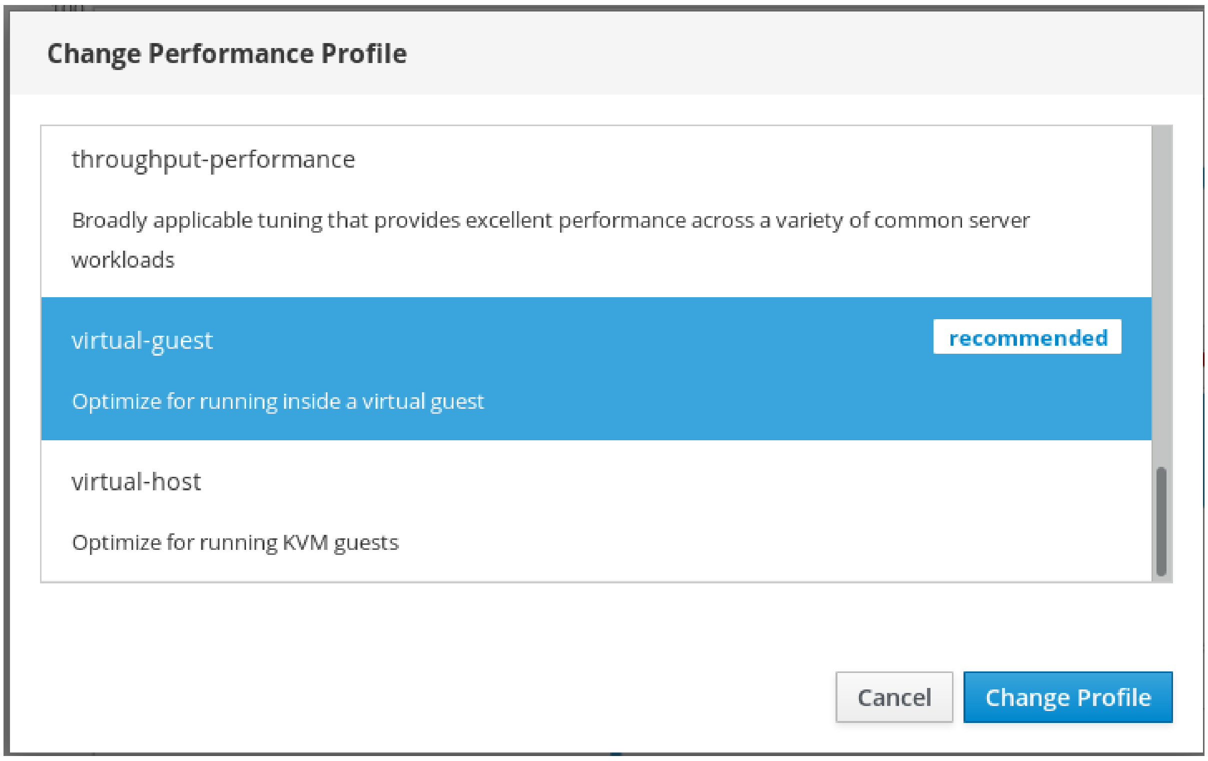
Procedure

1. Log in to the RHEL 8 web console.
   For details, see Logging in to the web console .
2. Click **System**.
3. In the **Performance Profile** field, click the current performance profile.



Hardware QEMU Standard PC (i440FX + PIIX, 1996)
Machine ID e89c7da66c204ec3a...
Operating System Red Hat Enterprise Linux 8.0 (Ootpa)
System Not Register...
Secure Shell Keys Show fingerprints
Host Name mymachine (mymac...
Domain Join Domain
System Time 2019-04-15 03:17
Power Options Restart
Performance Profile virtual-guest
Enable stored metrics...

4. In the **Change Performance Profile** dialog box, change the profile if necessary.
5. Click **Change**.


Change Performance Profile

throughput-performance

Broadly applicable tuning that provides excellent performance across a variety of common server workloads

virtual-guest — recommended

Optimize for running inside a virtual guest

virtual-host

Optimize for running KVM guests

The change is now available in the **System** tab.

CHAPTER 4. TUNING THE PERFORMANCE OF A SAMBA SERVER

This chapter describes what settings can improve the performance of Samba in certain situations, and which settings can have a negative performance impact.

Parts of this section were adopted from the Performance Tuning documentation published in the Samba Wiki. License: CC BY 4.0. Authors and contributors: See the history tab on the Wiki page.

Prerequisites

- Samba is set up as a file or print server
  See Using Samba as a server .

4.1. SETTING THE SMB PROTOCOL VERSION

Each new SMB version adds features and improves the performance of the protocol. The recent Windows and Windows Server operating systems always supports the latest protocol version. If Samba also uses the latest protocol version, Windows clients connecting to Samba benefit from the performance improvements. In Samba, the default value of the server max protocol is set to the latest supported stable SMB protocol version.

NOTE

To always have the latest stable SMB protocol version enabled, do not set the **server max protocol** parameter. If you set the parameter manually, you will need to modify the setting with each new version of the SMB protocol, to have the latest protocol version enabled.

The following procedure explains how to use the default value in the **server max protocol** parameter.

Procedure

1. Remove the **server max protocol** parameter from the **[global]** section in the **/etc/samba/smb.conf** file.

2. Reload the Samba configuration

```
# smbcontrol all reload-config
```

4.2. TUNING SHARES WITH DIRECTORIES THAT CONTAIN A LARGE NUMBER OF FILES

Linux supports case-sensitive file names. For this reason, Samba needs to scan directories for uppercase and lowercase file names when searching or accessing a file. You can configure a share to create new files only in lowercase or uppercase, which improves the performance.

Prerequisites

- Samba is configured as a file server

Procedure

1. Rename all files on the share to lowercase.

   NOTE

   Using the settings in this procedure, files with names other than in lowercase will no longer be displayed.

2. Set the following parameters in the share's section:

   ```
   case sensitive = true
   default case = lower
   preserve case = no
   short preserve case = no
   ```

   For details about the parameters, see their descriptions in the **smb.conf(5)** man page.

3. Verify the **/etc/samba/smb.conf** file:

   ```
   # testparm
   ```

4. Reload the Samba configuration:

   ```
   # smbcontrol all reload-config
   ```

After you applied these settings, the names of all newly created files on this share use lowercase. Because of these settings, Samba no longer needs to scan the directory for uppercase and lowercase, which improves the performance.

Additional resources

- Verifying the smb.conf file by using the testparm utility

4.3. SETTINGS THAT CAN HAVE A NEGATIVE PERFORMANCE IMPACT

By default, the kernel in Red Hat Enterprise Linux is tuned for high network performance. For example, the kernel uses an auto-tuning mechanism for buffer sizes. Setting the **socket options** parameter in the **/etc/samba/smb.conf** file overrides these kernel settings. As a result, setting this parameter decreases the Samba network performance in most cases.

To use the optimized settings from the Kernel, remove the **socket options** parameter from the **[global]** section in the **/etc/samba/smb.conf**.

CHAPTER 5. OPTIMIZING VIRTUAL MACHINE PERFORMANCE IN RHEL 8

Virtual machines (VMs) always experience some degree of performance deterioration in comparison to the host. The following sections describe the reasons why that is and provide instructions how to minimize the performance impact of virtualization, so that your hardware infrastructure resources can be used as efficiently as possible.

5.1. WHAT INFLUENCES VIRTUAL MACHINE PERFORMANCE

VMs are run as user-space processes on the host. The hypervisor therefore needs to convert the host's system resources so that the VMs can use them. As a consequence, a portion of the resources is consumed by the conversion, and the VM therefore cannot achieve the same performance as the host.

More specific **reasons for VM performance loss**include:

- Virtual CPUs (vCPUs) are implemented as threads on the host, handled by the Linux scheduler.
- VMs do not automatically inherit optimization features such as NUMA and huge pages from the host kernel.
- Disk and network I/O settings of the host might have a significant performance impact on the VM.
- Network traffic typically travels to a VM through a software-based bridge.
- Depending on the host devices and their models, there might be significant overhead due to emulation of particular hardware.

Nevertheless, a variety of features are available that you can use to **reduce the negative performance effects** of virtualization. Notably:

- The **tuned** service can automatically optimize the resource distribution and performance of your VMs.
- Block I/O tuning can improve the performances of the VM's block devices, such as disks.
- NUMA tuning can increase vCPU performance.
- Virtual networking can be optimized in various ways.

NOTE

Tuning VM performance can have adverse effects on other virtualization functions. For example, it can make migrating the modified VM more difficult.

5.2. OPTIMIZING VIRTUAL MACHINE PERFORMANCE USING TUNED

The **tuned** utility is a tuning profile delivery mechanism that adapts RHEL for certain workload characteristics, such as requirements for CPU-intensive tasks or storage-network throughput responsiveness. It provides a number of tuning profiles that are pre-configured to enhance performance and reduce power consumption in a number of specific use cases. You can edit these profiles or create new profiles to create performance solutions tailored to your environment, including virtualized environments.

Red Hat recommends using the following profiles when using virtualization in RHEL 8:

- For RHEL 8 virtual machines, use the **virtual-guest** profile. It is based on the generally applicable ***throughput-performance*** profile, but also decreases the swappiness of virtual memory.
- For RHEL 8 virtualization hosts, use the **virtual-host** profile. This enables more aggressive writeback of dirty memory pages, which benefits the host performance.

Prerequisites

- The **tuned** service must be installed and enabled.

Procedure

To enable a specific **tuned** profile:

1. List the available **tuned** profiles.

```
# tuned-adm list

Available profiles:
- balanced             - General non-specialized tuned profile
- desktop              - Optimize for the desktop use-case
[...]
- virtual-guest        - Optimize for running inside a virtual guest
- virtual-host         - Optimize for running KVM guests
Current active profile: balanced
```

2. **[Optional]** Create a new **tuned** profile or edit an existing **tuned** profile.
   For more information, see Customizing tuned profiles.

3. Activate a **tuned** profile.

```
# tuned-adm profile selected-profile
```

   - To optimize a virtualization host, use the *virtual-host* profile.

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

   - On a RHEL guest operating system, use the *virtual-guest* profile.

   ```
   # tuned-adm profile virtual-guest
   ```

Additional resources

- For more information about **tuned** and **tuned** profiles, see Monitoring and managing system status and performance.

5.3. OPTIMIZING VIRTUAL MACHINE I/O PERFORMANCE

The input and output (I/O) capabilities of a virtual machine (VM) can create a significant limitation to the VM's overall efficiency. To address this, you can optimize a VM's I/O by configuring block I/O parameters.

5.3.1. Tuning block I/O in virtual machines

When multiple block devices are being used by one or more VMs, it might be important to adjust the I/O priority of specific virtual devices by modifying their *I/O weights*.

Increasing the I/O weight of a device increases its priority for I/O bandwidth, and therefore provides it with more host resources. Similarly, reducing a device's weight makes it consume less host resources.

NOTE

Each device's **weight** value must be within the **100** to **1000 range. Alternatively, the value can be `0**, which removes that device from per-device listings.

Procedure
To display and set a VM's block I/O parameters:

1. Display the current **<blkio>** parameters for a VM:
   # virsh blkiotune *virtual_machine*

   ```
   <domain>
    ...
    <blkiotune>
      <weight>800</weight>
      <device>
        <path>/dev/sda</path>
        <weight>1000</weight>
      </device>
      <device>
        <path>/dev/sdb</path>
        <weight>500</weight>
      </device>
    </blkiotune>
    ...
   </domain>
   ```

2. Edit the I/O weight of a specified device:

   ```
   # virsh blkiotune VM-name --device-weights device, I/O-weight
   ```

   For example, the following changes the weight of the */dev/sda* device in the *liftrul* VM to 500.

   ```
   # *virsh blkiotune liftbrul --device-weights /dev/sda, 500
   ```

5.3.2. Disk I/O throttling in virtual machines

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

To enable disk I/O throttling, set a limit on disk I/O requests sent from each block device attached to VMs to the host machine.

Procedure

1. Use the **virsh domblklist** command for a list of disk device names on a specified VM.

   ```
   # virsh domblklist rollin-coal
   Target     Source
   ------------------------------------------------
   vda        /var/lib/libvirt/images/rollin-coal.qcow2
   sda        -
   sdb        /home/horridly-demanding-processes.iso
   ```

2. Set I/O limits for a block device attached to a VM using the **virsh blkdeviotune** command:

   ```
   # virsh blkdeviotune VM-name device --parameter limit
   ```

   For example, to throttle the **sdb** device on the **rollin-coal** VM to 1000 I/O operations per second and 50 MB per second throughput:

   ```
   # virsh blkdeviotune rollin-coal sdb --total-iops-sec 1000 --total-bytes-sec 52428800
   ```

Additional information

- Disk I/O throttling can be useful in various situations, for example when VMs belonging to different customers are running on the same host, or when quality of service guarantees are given for different VMs. 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 VM and supports limits on throughput and I/O operations.

5.3.3. Enabling multi-queue virtio-scsi

When using **virtio-scsi** storage devices in your virtual machines (VMs), the *multi-queue virtio-scsi* feature provides improved storage performance and scalability. It enables each virtual CPU to have a separate queue and interrupt to use without affecting other vCPUs.

Procedure

- To enable multi-queue virtio-scsi support for a specific VM, add the following to the VM's XML configuration, where *N* is the total number of vCPU queues:

  ```
  <controller type='scsi' index='0' model='virtio-scsi'>
     <driver queues='N' />
  </controller>
  ```

5.4. OPTIMIZING VIRTUAL MACHINE CPU PERFORMANCE

Much like physical CPUs in host machines, vCPUs are critical to virtual machine (VM) performance. As a result, optimizing vCPUs can have a significant impact on the resource efficiency of your VMs. To optimize your vCPU:

1. Ensure that the vCPU model is aligned with the CPU model of the host. For example, to set the *testguest1* VM to use the CPU model of the host:

   ```
   # virt-xml testguest1 --edit --cpu host-model
   ```

2. If your host machine uses Non-Uniform Memory Access (NUMA), you can also **configure NUMA**

for its VMs. This maps the host's CPU and memory processes to CPU and memory processes on the VM as closely as possible. In effect, NUMA tuning provides the vCPU with a more streamlined access to the system memory allocated to the VM, which can improve the vCPU processing effectiveness.
For details, see Section 5.4.1, "Configuring NUMA in a virtual machine" and Section 5.4.2, "Sample vCPU performance tuning scenario".

5.4.1. Configuring NUMA in a virtual machine

The following methods can be used to configure Non-Uniform Memory Access (NUMA) settings of a virtual machine (VM) on a RHEL 8 host.

Prerequisites

- The host must be a NUMA-compatible machine. To detect whether this is the case, use the **virsh nodeinfo** command and see the **NUMA cell(s)** line:

```
# virsh nodeinfo
CPU model:           x86_64
CPU(s):              48
CPU frequency:       1200 MHz
CPU socket(s):       1
Core(s) per socket:  12
Thread(s) per core:  2
NUMA cell(s):        2
Memory size:         67012964 KiB
```

  If the value of the line is 2 or greater, the host is NUMA-compatible.

Procedure

For ease of use, you can set up a VM's NUMA configuration using automated utilities and services. However, manual NUMA setup is more likely to yield a significant performance improvement.

Automatic methods

- Set the VM's NUMA policy to **Preferred**. For example, to do so for the *testguest5* VM:

```
# virt-xml testguest5 --edit --vcpus placement=auto
# virt-xml testguest5 --edit --numatune mode=preferred
```

- Enable automatic NUMA balancing on the host:

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

- Use the **numad** command to automatically align the VM CPU with memory resources.

```
# numad
```

Manual methods

1. Pin specific vCPU threads to a specific host CPU or range of CPUs. This is possible also on non-NUMA hosts and VMs, and is recommended as a safe method of vCPU performance improvement.
   For example, the following commands pin vCPU threads 0 to 5 of the *testguest6* VM to host CPUs 1, 3, 5, 7, 9, and 11, respectively:

```
# virsh vcpupin testguest6 0 1
# virsh vcpupin testguest6 1 3
# virsh vcpupin testguest6 2 5
# virsh vcpupin testguest6 3 7
# virsh vcpupin testguest6 4 9
# virsh vcpupin testguest6 5 11
```

Afterwards, you can verify whether this was successful:

```
# virsh vcpupin testguest6
VCPU   CPU Affinity
----------------------
0      1
1      3
2      5
3      7
4      9
5      11
```

2. After pinning vCPU threads, you can also pin QEMU process threads associated with a specified VM to a specific host CPU or range of CPUs. For example, the following commands pin the QEMU process thread of *testguest6* to CPUs 13 and 15, and verify this was successful:

```
# virsh emulatorpin testguest6 13,15
# virsh emulatorpin testguest6
emulator: CPU Affinity
----------------------------------
       *: 13,15
```

3. Finally, you can also specify which host NUMA nodes will be assigned specifically to a certain VM. This can improve the host memory usage by the VM's vCPU. For example, the following commands set *testguest6* to use host NUMA nodes 3 to 5, and verify this was successful:

```
# virsh numatune testguest6 --nodeset 3-5
# virsh numatune testguest6
```

Additional resources

- Note that for best performance results, it is recommended to use all of the manual tuning methods listed above. For an example of such a configuration, see Section 5.4.2, "Sample vCPU performance tuning scenario".
- To see the current NUMA configuration of your system, you can use the **numastat** utility. For details on using **numastat**, see Section 5.6, "Virtual machine performance monitoring tools" .

5.4.2. Sample vCPU performance tuning scenario

To obtain the best vCPU performance possible, Red Hat recommends using manual **vcpupin**, **emulatorpin**, and **numatune** settings together, for example like in the following scenario.

Starting scenario

- Your host has the following hardware specifics:
  - 2 NUMA nodes

- 3 CPU cores on each node
- 2 threads on each core

The output of **virsh nodeinfo** of such a machine would look similar to:

```
# virsh nodeinfo
CPU model:           x86_64
CPU(s):              12
CPU frequency:       3661 MHz
CPU socket(s):       2
Core(s) per socket:  3
Thread(s) per core:  2
NUMA cell(s):        2
Memory size:         31248692 KiB
```

- You intend to modify an existing VM to have 8 vCPUs, which means that it will not fit in a single NUMA node.
  Therefore, you should distribute 4 vCPUs on each NUMA node and make the vCPU topology resemble the host topology as closely as possible. This means that vCPUs that run as sibling threads of a given physical CPU should be pinned to host threads on the same core. For details, see the *Solution* below:

Solution

1. Obtain the information on the host topology:

```
# virsh capabilities
```

The output should include a section that looks similar to the following:

```
<topology>
  <cells num="2">
    <cell id="0">
      <memory unit="KiB">15624346</memory>
      <pages unit="KiB" size="4">3906086</pages>
      <pages unit="KiB" size="2048">0</pages>
      <pages unit="KiB" size="1048576">0</pages>
      <distances>
        <sibling id="0" value="10" />
        <sibling id="1" value="21" />
      </distances>
      <cpus num="6">
        <cpu id="0" socket_id="0" core_id="0" siblings="0,3" />
        <cpu id="1" socket_id="0" core_id="1" siblings="1,4" />
        <cpu id="2" socket_id="0" core_id="2" siblings="2,5" />
        <cpu id="3" socket_id="0" core_id="0" siblings="0,3" />
        <cpu id="4" socket_id="0" core_id="1" siblings="1,4" />
        <cpu id="5" socket_id="0" core_id="2" siblings="2,5" />
      </cpus>
    </cell>
    <cell id="1">
      <memory unit="KiB">15624346</memory>
      <pages unit="KiB" size="4">3906086</pages>
      <pages unit="KiB" size="2048">0</pages>
      <pages unit="KiB" size="1048576">0</pages>
      <distances>
        <sibling id="0" value="21" />
        <sibling id="1" value="10" />
      </distances>
      <cpus num="6">
        <cpu id="6" socket_id="1" core_id="3" siblings="6,9" />
        <cpu id="7" socket_id="1" core_id="4" siblings="7,10" />
        <cpu id="8" socket_id="1" core_id="5" siblings="8,11" />
        <cpu id="9" socket_id="1" core_id="3" siblings="6,9" />
        <cpu id="10" socket_id="1" core_id="4" siblings="7,10" />
        <cpu id="11" socket_id="1" core_id="5" siblings="8,11" />
      </cpus>
    </cell>
  </cells>
</topology>
```

2. **[Optional]** Test the performance of the VM using the applicable tools and utilities.

3. Set up and mount 1 GiB huge pages on the host:

   a. Add the following line to the host's kernel command line:

   ```
   default_hugepagesz=1G hugepagesz=1G
   ```

   b. Create the **/etc/systemd/system/hugetlb-gigantic-pages.service** file with the following content:

   ```
   [Unit]
   Description=HugeTLB Gigantic Pages Reservation
   DefaultDependencies=no
   Before=dev-hugepages.mount
   ConditionPathExists=/sys/devices/system/node
   ConditionKernelCommandLine=hugepagesz=1G

   [Service]
   Type=oneshot
   RemainAfterExit=yes
   ExecStart=/etc/systemd/hugetlb-reserve-pages.sh

   [Install]
   WantedBy=sysinit.target
   ```

   c. Create the **/etc/systemd/hugetlb-reserve-pages.sh** file with the following content:

   ```
   #!/bin/sh

   nodes_path=/sys/devices/system/node/
   if [ ! -d $nodes_path ]; then
    echo "ERROR: $nodes_path does not exist"
    exit 1
   fi

   reserve_pages()
   {
   ```

```
 echo $1 > $nodes_path/$2/hugepages/hugepages-1048576kB/nr_hugepages
}

reserve_pages 4 node1
reserve_pages 4 node2
```

This reserves four 1GiB huge pages from *node1* and four 1GiB huge pages from *node2*.

d. Make the script created in the previous step executable:

```
# chmod +x /etc/systemd/hugetlb-reserve-pages.sh
```

e. Enable huge page reservation on boot:

```
# systemctl enable hugetlb-gigantic-pages
```

4. Use the **virsh edit** command to edit the XML configuration of the VM you wish to optimize, in this example *super-VM*:

```
# virsh edit super-vm
```

5. Adjust the XML configuration of the VM in the following way:

- Set the VM to use 8 static vCPUs. Use the **<vcpu/>** element to do this.
- Pin each of the vCPU threads to the corresponding host CPU threads that it mirrors in topology. To do so, use the **<vcpupin/>** elements in the **<cputune>** section.
  Note that, as shown by **virsh capabilities** above, host CPU threads are not ordered sequentially in their respective cores. In addition, the vCPU threads should be pinned to the highest available set of host cores on the same NUMA node. For a table illustration, see the **Additional Resources** section below.
- Configure the VM's NUMA nodes to use memory from the corresponding NUMA nodes on the host. To do so, use the **<memnode/>** elements in the **<numatune/>** section.
- Ensure the CPU mode is set to **host-passthrough**, and that the CPU uses cache in **passthrough** mode.

The resulting XML configuration should include a section similar to the following:

```
[...]
  <memoryBacking>
    <hugepages>
      <page size='1' unit='GiB'/>
    </hugepages>
  </memoryBacking>
  <vcpu placement='static'>8</vcpu>
  <cputune>
    <vcpupin vcpu='0' cpuset='1'/>
    <vcpupin vcpu='1' cpuset='4'/>
    <vcpupin vcpu='2' cpuset='2'/>
    <vcpupin vcpu='3' cpuset='5'/>
    <vcpupin vcpu='4' cpuset='7'/>
    <vcpupin vcpu='5' cpuset='10'/>
    <vcpupin vcpu='6' cpuset='8'/>
    <vcpupin vcpu='7' cpuset='11'/>
    <emulatorpin cpuset='6,9'/>
  </cputune>
  <numatune>
    <memory mode="preferred" nodeset="1"/>
    <memnode cellid="0" mode="strict" nodeset="0"/>
    <memnode cellid="1" mode="strict" nodeset="1"/>
  </numatune>
  <cpu mode="host-passthrough">
    <topology sockets="2" cores="2" threads="2"/>
    <cache mode="passthrough"/>
    <numa>
      <cell id="0" cpus="0-3" memory="2" unit="GiB">
        <distances>
          <sibling id="0" value="10"/>
          <sibling id="1" value="21"/>
        </distances>
      </cell>
      <cell id="1" cpus="4-7" memory="2" unit="GiB">
        <distances>
          <sibling id="0" value="21"/>
          <sibling id="1" value="10"/>
        </distances>
      </cell>
    </numa>
  </cpu>
</domain>
```

6. **[Optional]** Test the performance of the VM using the applicable tools and utilities to evaluate the impact of the VM's optimization.

Additional resources

- The following tables illustrate the connections between the vCPUs and the host CPUs they should be pinned to:

Table 5.1. Host topology

CPU threads	0	3	1	4	2	5	6	9	7	10	8	11
Cores	0		1		2		3		4		5	
Sockets	0						1					
NUMA nodes	0						1					

Table 5.2. VM topology

vCPU threads	0	1	2	3	4	5	6	7
Cores	0		1		2		3	
Sockets	0				1			

NUMA nodes	0	1

Table 5.3. Combined host and VM topology

vCPU threads			0	1	2	3			4	5	6	7
Host CPU threads	0	3	1	4	2	5	6	9	7	10	8	11
Cores	0		1		2		3		4		5	
Sockets	0						1					
NUMA nodes	0						1					

In this scenario, there are 2 NUMA nodes and 8 vCPUs. Therefore, 4 vCPU threads should be pinned to each node.

In addition, Red Hat recommends leaving at least a single CPU thread available on each node for host system operations.

Because in this example, each NUMA node houses 3 cores, each with 2 host CPU threads, the set for node 0 translates as follows:

```
<vcpupin vcpu='0' cpuset='1'/>
<vcpupin vcpu='1' cpuset='4'/>
<vcpupin vcpu='2' cpuset='2'/>
<vcpupin vcpu='3' cpuset='5'/>
```

5.5. OPTIMIZING VIRTUAL MACHINE NETWORK PERFORMANCE

Due to the virtual nature of a VM's network interface card (NIC), the VM loses a portion of the host network bandwidth it is allocated, which can reduce the overall workload efficiency of the VM. The following tips can minimize the negative impact of virtualization on the virtual NIC (vNIC) throughput.

Procedure

Use any of the following methods and observe if it has a beneficial effect on your VM network performance:

Enable the vhost_net module

On the host, ensure the **vhost_net** kernel feature is enabled:

```
# lsmod | grep vhost
vhost_net              32768  1
vhost                  53248  1 vhost_net
tap                    24576  1 vhost_net
tun                    57344  6 vhost_net
```

If the output of this command is blank, enable the **vhost_net** kernel module:

```
# modprobe vhost_net
```

Set up multi-queue virtio-net

To set up the *multi-queue virtio-net* feature for a VM, use the **virsh edit** command to edit to the XML configuration of the VM. In the XML, add the following to the **<devices>** section, and replace **N** with the number of vCPUs in the VM, up to 16:

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

If the VM is running, restart it for the changes to take effect.

Set up vhost zero-copy transmit

If using a network with large packet size, enable the *vhost zero-copy transmit* feature.
Note that this feature only improves the performance when transmitting large packets between a guest network and an external network. It does not affect performance for guest-to-guest and guest-to-host workloads. In addition, it is likely to have a negative impact on the performance of small packet workloads.

Also, enabling zero-copy transmit can cause head-of-line blocking of packets, which may create a potential security risk.

To enable vhost zero-copy transmit:

1. On the host, disable the vhost-net kernel module:

   ```
   # modprobe -r vhost_net
   ```

2. Re-enable the vhost-net module with the zero-copy parameter turned on:

   ```
   # modprobe vhost-net experimental_zcopytx=1
   ```

3. Check whether zero-copy transmit was enabled successfully:

   ```
   # cat /sys/module/vhost_net/parameters/experimental_zcopytx
   1
   ```

Batching network packets

In Linux VM configurations with a long transmission path, batching packets before submitting them to the kernel may improve cache utilization. To set up packet batching, use the following command on the host, and replace *tap0* with the name of the network interface that the VMs use:

```
# ethtool -C tap0 rx-frames 128
```

Additional resources

- For additional information on virtual network connection types and tips for usage, see Understanding virtual networking.

5.6. VIRTUAL MACHINE PERFORMANCE MONITORING TOOLS

To recognize what consumes the most VM resources and which aspect of VM performance needs optimization, both general and VM-specific performance diagnostic tools can be used.

Default OS performance monitoring tools

For standard performance evaluation, you can use the utilities provided by default by your host and guest operating systems:

- On your RHEL 8 host, as root, use the **top** utility or the **system monitor** application, and look for **qemu** and **virt** in the output. This shows how much host system resources your VMs are consuming.
  - If the monitoring tool displays that any of the **qemu** or **virt** processes consume a large portion of the host CPU or memory capacity, use the **perf** utility to investigate. For details, see below.
  - In addition, if a **vhost_net** thread process, named for example *vhost_net-1234*, is displayed as consuming an excessive amount of host CPU capacity, consider using virtual network optimization features, such as **multi-queue virtio-net**.
- On the guest operating system, use performance utilities and applications available on the system to evaluate which processes consume the most system resources.
  - On Linux systems, you can use the **top** utility.
  - On Windows systems, you can use the **Task Manager** application.

perf kvm

You can use the **perf** utility to collect and analyze virtualization-specific statistics about the performance of your RHEL 8 host. To do so:

1. On the host, install the *perf* package:

```
# yum install perf
```

2. Use the **perf kvm stat** command to display perf statistics for your virtualization host:
   - For real-time monitoring of your hypervisor, use the **perf kvm stat live** command.
   - To log the perf data of your hypervisor over a period of time, activate the logging using the **perf kvm stat record** command. After the command is canceled or interrupted, the data is saved in the **perf.data.guest** file, which can be analyzed using the **perf kvm stat report** command.

3. Analyze the **perf** output for types of **VM-EXIT** events and their distribution. For example, the **PAUSE_INSTRUCTION** events should be infrequent, but in the following output, the high occurrence of this event suggests that the host CPUs are not handling the running vCPUs well. In such a scenario, consider powering down some of your active VMs, removing vCPUs from these VMs, or tuning the performance of the vCPUs.

```
# perf kvm stat report

Analyze events for all VMs, all VCPUs:


         VM-EXIT   Samples  Samples%     Time%    Min Time    Max Time        Avg time
```

```
  EXTERNAL_INTERRUPT    365634    31.59%    18.04%      0.42us  58780.59us
204.08us ( +-   0.99% )
          MSR_WRITE     293428    25.35%     0.13%      0.59us  17873.02us      1.80us ( +-
4.63% )
    PREEMPTION_TIMER    276162    23.86%     0.23%      0.51us  21396.03us      3.38us (
+-   5.19% )
   PAUSE_INSTRUCTION    189375    16.36%    11.75%      0.72us  29655.25us    256.77us
( +-   0.70% )
                HLT      20440     1.77%    69.83%      0.62us  79319.41us  14134.56us ( +-   0.79%
)
             VMCALL      12426     1.07%     0.03%      1.02us   5416.25us      8.77us ( +-   7.36%
)
      EXCEPTION_NMI         27     0.00%     0.00%      0.69us      1.34us      0.98us ( +-
3.50% )
      EPT_MISCONFIG          5     0.00%     0.00%      5.15us     10.85us      7.88us ( +-
11.67% )

Total Samples:1157497, Total events handled time:413728274.66us.
```

Other event types that can signal problems in the output of **perf kvm stat** include:

- **INSN_EMULATION** - suggests suboptimal VM I/O configuration.

For more information on using **perf** to monitor virtualization performance, see the **perf-kvm** man page.

numastat

To see the current NUMA configuration of your system, you can use the **numastat** utility, which is provided by installing the **numactl** package.

The following shows a host with 4 running VMs, each obtaining memory from multiple NUMA nodes. This is not optimal for vCPU performance, and warrants adjusting:

```
# numastat -c qemu-kvm

Per-node process memory usage (in MBs)
PID              Node 0 Node 1 Node 2 Node 3 Node 4 Node 5 Node 6 Node 7 Total
---------------  ------ ------ ------ ------ ------ ------ ------ ------ -----
51722 (qemu-kvm)     68     16    357   6936      2      3    147    598  8128
51747 (qemu-kvm)    245     11      5     18   5172   2532      1     92  8076
53736 (qemu-kvm)     62    432   1661    506   4851    136     22    445  8116
53773 (qemu-kvm)   1393      3      1      2     12      0      0   6702  8114
---------------  ------ ------ ------ ------ ------ ------ ------ ------ -----
Total              1769    463   2024   7462  10037   2672    169   7837 32434
```

In contrast, the following shows memory being provided to each VM by a single node, which is significantly more efficient.

```
# numastat -c qemu-kvm

Per-node process memory usage (in MBs)
PID              Node 0 Node 1 Node 2 Node 3 Node 4 Node 5 Node 6 Node 7 Total
---------------  ------ ------ ------ ------ ------ ------ ------ ------ -----
51747 (qemu-kvm)      0      0      7      0   8072      0      1      0  8080
53736 (qemu-kvm)      0      0      7      0      0      0   8113      0  8120
53773 (qemu-kvm)      0      0      7      0      0      0      1   8110  8118
```

```
59065 (qemu-kvm)      0      0   8050      0      0      0      0    0  8051
---------------  ------ ------ ------ ------ ------ ------ ------ ------ -----
Total                 0      0   8072      0   8072      0   8114   8110 32368
```

CHAPTER 6. MANAGING POWER CONSUMPTION WITH POWERTOP

As a system administrator, you can use the **PowerTOP** tool to analyze and manage power consumption.

6.1. THE PURPOSE OF POWERTOP

PowerTOP is a program that diagnoses issues related to power consumption and provides suggestions on how to extend battery lifetime.

The **PowerTOP** tool can provide an estimate of the total power usage of the system and also individual power usage for each process, device, kernel worker, timer, and interrupt handler. The tool can also identify specific components of kernel and user-space applications that frequently wake up the CPU.

Red Hat Enterprise Linux 8 uses version 2.x of **PowerTOP**.

6.2. USING POWERTOP

6.2.1. Prerequisites

- To be able to use **PowerTOP**, make sure that the **powertop** package has been installed on your system:

  ```
  # yum install powertop
  ```

6.2.2. Starting PowerTOP

Procedure

- To run **PowerTOP**, use the following command:

  ```
  # powertop
  ```

IMPORTANT

Laptops should run on battery power when running the **powertop** command.

6.2.3. Calibrating PowerTOP

Procedure

1. On a laptop, you can calibrate the power estimation engine by running the following command:

   ```
   # powertop --calibrate
   ```

2. Let the calibration finish without interacting with the machine during the process.
   Calibration takes time because the process performs various tests, cycles through brightness levels and switches devices on and off.

3. When the calibration process is completed, **PowerTOP** starts as normal. Let it run for approximately an hour to collect data.

When enough data is collected, power estimation figures will be displayed in the first column of the output table.

NOTE

Note that **powertop --calibrate** can only be used on laptops.

6.2.4. Setting the measuring interval

By default, **PowerTOP** takes measurements in 20 seconds intervals.

If you want to change this measuring frequency, use the following procedure:

Procedure

- Run the **powertop** command with the **--time** option:

```
# powertop --time=time in seconds
```

6.2.5. Related information

For more details on how to use **PowerTOP**, see the **powertop** man page.

6.3. POWERTOP STATISTICS

While it runs, **PowerTOP** gathers statistics from the system.

PowerTOP's output provides multiple tabs:

- **Overview**
- **Idle stats**
- **Frequency stats**
- **Device stats**
- **Tunables**

You can use the **Tab** and **Shift+Tab** keys to cycle through these tabs.

6.3.1. The Overview tab

In the **Overview** tab, you can view a list of the components that either send wakeups to the CPU most frequently or consume the most power. The items within the **Overview** tab, including processes, interrupts, devices, and other resources, are sorted according to their utilization.

The adjacent columns within the **Overview** tab provide the following pieces of information:

Usage

Power estimation of how the resource is being used.

Events/s

Wakeups per second. The number of wakeups per second indicates how efficiently the services or the devices and drivers of the kernel are performing. Less wakeups means that less power is consumed. Components are ordered by how much further their power usage can be optimized.

Category

Classification of the component; such as process, device, or timer.

Description

Description of the component.

If properly calibrated, a power consumption estimation for every listed item in the first column is shown as well.

Apart from this, the **Overview** tab includes the line with summary statistics such as:

- Total power consumption
- Remaining battery life (only if applicable)
- Summary of total wakeups per second, GPU operations per second, and virtual file system operations per second

6.3.2. The Idle stats tab

The **Idle stats** tab shows usage of C-states for all processors and cores, while the **Frequency stats** tab shows usage of P-states including the Turbo mode, if applicable, for all processors and cores. The duration of C- or P-states is an indication of how well the CPU usage has been optimized. The longer the CPU stays in the higher C- or P-states (for example C4 is higher than C3), the better the CPU usage optimization is. Ideally, residency is 90% or more in the highest C- or P-state when the system is idle.

6.3.3. The Device stats tab

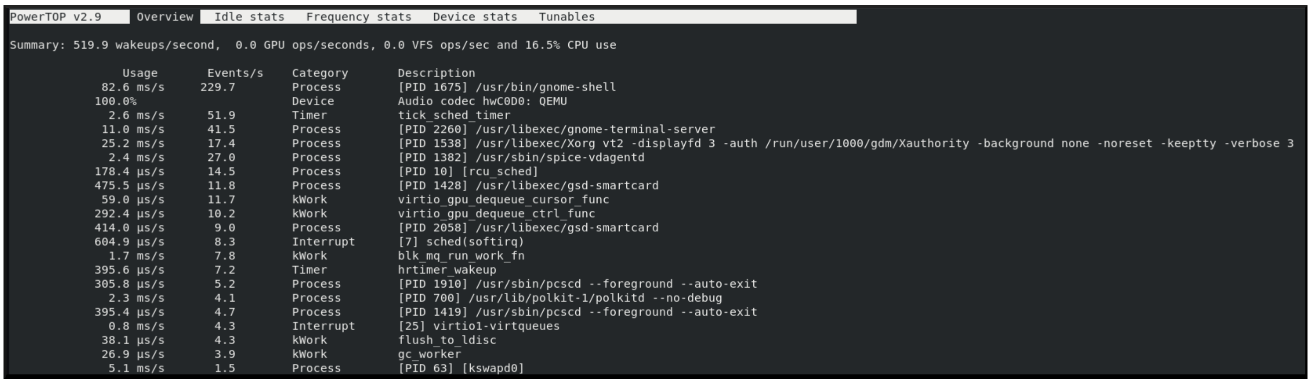
The **Device stats** tab provides similar information to the **Overview** tab but only for devices.

6.3.4. The Tunables tab

The **Tunables** tab contains **PowerTOP**'s suggestions for optimizing the system for lower power consumption.

Use the **up** and **down** keys to move through suggestions, and the **enter** key to toggle the suggestion on or off.

Figure 6.1. PowerTOP output



```
PowerTOP v2.9     Overview   Idle stats   Frequency stats   Device stats   Tunables

Summary: 519.9 wakeups/second,  0.0 GPU ops/seconds, 0.0 VFS ops/sec and 16.5% CPU use

                Usage       Events/s    Category       Description
             82.6 ms/s     229.7        Process        [PID 1675] /usr/bin/gnome-shell
            100.0%                      Device         Audio codec hwC0D0: QEMU
              2.6 ms/s      51.9        Timer          tick_sched_timer
             11.0 ms/s      41.5        Process        [PID 2260] /usr/libexec/gnome-terminal-server
             25.2 ms/s      17.4        Process        [PID 1538] /usr/libexec/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthority -background none -noreset -keeptty -verbose 3
              2.4 ms/s      27.0        Process        [PID 1382] /usr/sbin/spice-vdagentd
            178.4 µs/s      14.5        Process        [PID 10] [rcu_sched]
            475.5 µs/s      11.8        Process        [PID 1428] /usr/libexec/gsd-smartcard
             59.0 µs/s      11.7        kWork          virtio_gpu_dequeue_cursor_func
            292.4 µs/s      10.2        kWork          virtio_gpu_dequeue_ctrl_func
            414.0 µs/s       9.0        Process        [PID 2058] /usr/libexec/gsd-smartcard
            604.9 µs/s       8.3        Interrupt      [7] sched(softirq)
              1.7 ms/s       7.8        kWork          blk_mq_run_work_fn
            395.6 µs/s       7.2        Timer          hrtimer_wakeup
            305.8 µs/s       5.2        Process        [PID 1910] /usr/sbin/pcscd --foreground --auto-exit
              2.3 ms/s       4.1        Process        [PID 700] /usr/lib/polkit-1/polkitd --no-debug
            395.4 µs/s       4.7        Process        [PID 1419] /usr/sbin/pcscd --foreground --auto-exit
              0.8 ms/s       4.3        Interrupt      [25] virtio1-virtqueues
             38.1 µs/s       4.3        kWork          flush_to_ldisc
             26.9 µs/s       3.9        kWork          gc_worker
              5.1 ms/s       1.5        Process        [PID 63] [kswapd0]
```

Additional resources

For more details on **PowerTOP**, see PowerTOP's home page.

6.4. GENERATING AN HTML OUTPUT

Apart from the **powertop's** output in terminal, you can also generate an HTML report.

Procedure

- Run the **powertop** command with the **--html** option:

  ```
  # powertop --html=htmlfile.html
  ```

  Replace the **htmlfile.html** parameter with the required name for the output file.

6.5. OPTIMIZING POWER CONSUMPTION

To optimize power consumption, you can use either the **powertop** service or the **powertop2tuned** utility.

6.5.1. Optimizing power consumption using the powertop service

You can use the **powertop** service to automatically enable all **PowerTOP**'s suggestions from the **Tunables** tab on the boot:

Procedure

- Enable the **powertop** service:

  ```
  # systemctl enable powertop
  ```

6.5.2. The powertop2tuned utility

The **powertop2tuned** utility allows you to create custom **Tuned** profiles from **PowerTOP** suggestions.

By default, **powertop2tuned** creates profiles in the **/etc/tuned/** directory, and bases the custom profile on the currently selected **Tuned** profile. For safety reasons, all **PowerTOP** tunings are initially disabled in the new profile.

To enable the tunings, you can:

- Uncomment them in the **/etc/tuned/profile_name/tuned.conf file**.
- Use the **--enable** or **-e** option to generate a new profile that enables most of the tunings suggested by **PowerTOP**.
  Certain potentially problematic tunings, such as the USB autosuspend, are disabled by default and need to be uncommented manually.

6.5.3. Optimizing power consumption using the powertop2tuned utility

Prerequisites

- The **powertop2tuned** utility is installed on the system:

```
# yum install tuned-utils
```

Procedure

1. Create a custom profile:

```
# powertop2tuned new_profile_name
```

2. Activate the new profile:

```
# tuned-adm profile new_profile_name
```

Additional information

- For a complete list of options that **powertop2tuned** supports, use:

```
$ powertop2tuned --help
```

6.5.4. Comparison of powertop.service and powertop2tuned

Optimizing power consumption with **powertop2tuned** is preferred over **powertop.service** for the following reasons:

- The **powertop2tuned** utility represents integration of **PowerTOP** into **Tuned**, which enables to benefit of advantages of both tools.
- The **powertop2tuned** utility allows for fine-grained control of enabled tuning.
- With **powertop2tuned**, potentially dangerous tuning are not automatically enabled.
- With **powertop2tuned**, rollback is possible without reboot.