Red Hat Enterprise Linux 8

Monitoring and managing system status and performance

Optimizing system throughput, latency, and power consumption
Red Hat Enterprise Linux 8 Monitoring and managing system status and performance

Optimizing system throughput, latency, and power consumption
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 .............................................. 5

CHAPTER 1. GETTING STARTED WITH TUNED ......................................................... 6
1.1. THE PURPOSE OF TUNED 6
1.2. TUNED PROFILES 6
   The default profile 6
   Merged profiles 7
   The location of profiles 7
   The syntax of profile configuration 8
   Additional resources 8
1.3. TUNED PROFILES DISTRIBUTED WITH RHEL 8
   Real-time profiles 9
1.4. STATIC AND DYNAMIC TUNING IN TUNED 10
1.5. TUNED NO-DAEMON MODE 10
1.6. INSTALLING AND ENABLING TUNED 11
   Procedure 11
1.7. LISTING AVAILABLE TUNED PROFILES 11
   Procedure 11
   Additional resources 12
1.8. SETTING A TUNED PROFILE 12
   Prerequisites 12
   Procedure 12
   Additional resources 13
1.9. DISABLING TUNED 13
   Procedure 13
   Additional resources 13
1.10. RELATED INFORMATION 13

CHAPTER 2. CUSTOMIZING TUNED PROFILES ....................................................... 14
2.1. PREREQUISITES 14
2.2. TUNED PROFILES 14
   The default profile 14
   Merged profiles 14
   The location of profiles 15
   The syntax of profile configuration 15
   Additional resources 15
2.3. INHERITANCE BETWEEN TUNED PROFILES 15
   Additional resources 16
2.4. STATIC AND DYNAMIC TUNING IN TUNED 16
2.5. TUNED PLUG-INS 17
   Monitoring plug-ins 17
   Tuning plug-ins 17
   Syntax for plug-ins in Tuned profiles 17
   Short plug-in syntax 18
   Conflicting plug-in definitions in a profile 18
   Functionality not implemented in any plug-in 19
   Additional resources 19
2.6. AVAILABLE TUNED PLUG-INS 19
   Monitoring plug-ins 19
   Tuning plug-ins 19
2.7. VARIABLES AND BUILT-IN FUNCTIONS IN TUNED PROFILES 22
Variables 23
Functions 23
Additional resources 24
2.8. BUILT-IN FUNCTIONS AVAILABLE IN TUNED PROFILES 24
2.9. CREATING NEW TUNED PROFILES 25
Prerequisites 25
Procedure 25
Additional resources 26
2.10. MODIFYING EXISTING TUNED PROFILES 26
Prerequisites 26
Procedure 26
Additional resources 27
2.11. RELATED INFORMATION 27

CHAPTER 3. USING THE WEB CONSOLE FOR SELECTING PERFORMANCE PROFILES 28
Prerequisites 28
Procedure 28

CHAPTER 4. TUNING THE PERFORMANCE OF A SAMBA SERVER 30
Prerequisites 30
4.1. SETTING THE SMB PROTOCOL VERSION 30
Procedure 30
4.2. TUNING SHARES WITH DIRECTORIES THAT CONTAIN A LARGE NUMBER OF FILES 30
Prerequisites 30
Procedure 30
Additional resources 31
4.3. SETTINGS THAT CAN HAVE A NEGATIVE PERFORMANCE IMPACT 31

CHAPTER 5. MANAGING POWER CONSUMPTION WITH POWERTOP 32
5.1. THE PURPOSE OF POWERTOP 32
5.2. USING POWERTOP 32
5.2.1. Prerequisites 32
5.2.2. Starting PowerTOP 32
Procedure 32
5.2.3. Calibrating PowerTOP 32
Procedure 32
5.2.4. Setting the measuring interval 33
Procedure 33
5.2.5. Related information 33
5.3. POWERTOP STATISTICS 33
5.3.1. The Overview tab 33
5.3.2. The Idle stats tab 34
5.3.3. The Device stats tab 34
5.3.4. The Tunables tab 34
Additional resources 35
5.4. GENERATING AN HTML OUTPUT 35
Procedure 35
5.5. OPTIMIZING POWER CONSUMPTION 35
5.5.1. Optimizing power consumption using the powertop service 35
Procedure 35
5.5.2. The powertop2tuned utility 35
5.5.3. Optimizing power consumption using the powertop2tuned utility 35
Prerequisites 35
Procedure 36
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional information</td>
<td>36</td>
</tr>
<tr>
<td>5.5.4. Comparison of powertop.service and powertop2tuned</td>
<td>36</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Environment</th>
<th>Default profile</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute nodes</td>
<td>throughput-performance</td>
<td>The best throughput performance</td>
</tr>
</tbody>
</table>
Virtual machines

<table>
<thead>
<tr>
<th>Environment</th>
<th>Default profile</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual machines</td>
<td><strong>virtual-guest</strong></td>
<td>The best performance. If you are not interested in the best performance, you can change it to the <strong>balanced</strong> or <strong>powersave</strong> profile.</td>
</tr>
<tr>
<td>Other cases</td>
<td><strong>balanced</strong></td>
<td>Balanced performance and power consumption</td>
</tr>
</tbody>
</table>

**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:

```bash
# 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 \texttt{sysctl} settings that improve the throughput performance of the disk and network IO. CPU governor is set to \texttt{performance}.

\textbf{latency-performance}

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

\textbf{network-latency}

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

\textbf{network-throughput}

A profile for throughput network tuning. It is based on the \texttt{throughput-performance} profile. It additionally increases kernel network buffers.

\textbf{virtual-guest}

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

\textbf{virtual-host}

A profile designed for virtual hosts based on the \texttt{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.

\textbf{oracle}

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

\textbf{desktop}

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

\textbf{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:

\textbf{realtime}

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

\textbf{realtime-virtual-host}

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

\textbf{realtime-virtual-guest}

Use in a virtualization guest configured for real-time. Provided by the \texttt{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:

```bash
$ 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:

```bash
# tuned-adm recommend
balanced
```

2. Activate a profile:

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

Alternatively, you can activate a combination of multiple profiles:

```bash
# 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:

```bash
# 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
Verification 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:

<table>
<thead>
<tr>
<th>Environment</th>
<th>Default profile</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute nodes</td>
<td>throughput-performance</td>
<td>The best throughput performance</td>
</tr>
<tr>
<td>Virtual machines</td>
<td>virtual-guest</td>
<td>The best performance. If you are not interested in the best performance, you can change it to the balanced or powersave profile.</td>
</tr>
<tr>
<td>Other cases</td>
<td>balanced</td>
<td>Balanced performance and power consumption</td>
</tr>
</tbody>
</table>

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.

```plaintext
[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:

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

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

```yaml
[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:

```yaml
[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**

```yaml
[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**
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:
The \texttt{cpulist\_invert} function inverts the list of CPUs. For a 6-CPU machine, the inversion is \texttt{1,2}, and the kernel boots with the \texttt{isolcpus=1,2} command-line option.

Additional resources

- The \texttt{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:

\textbf{PROFILE\_DIR}

Returns the directory path where the profile and the \texttt{tuned.conf} file are located.

\textbf{exec}

Executes a process and returns its output.

\textbf{assertion}

Compares two arguments. If they \textit{do not match}, the function logs text from the first argument and aborts profile loading.

\textbf{assertion\_non\_equal}

Compares two arguments. If they \textit{match}, the function logs text from the first argument and aborts profile loading.

\textbf{kb2s}

Converts kilobytes to disk sectors.

\textbf{s2kb}

Converts disk sectors to kilobytes.

\textbf{strip}

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

\textbf{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.

\textbf{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 \texttt{0,2,3} is \texttt{1}.

\textbf{cpulist2hex}

Converts a CPU list to a hexadecimal CPU mask.

\textbf{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 \texttt{1-3,4} to \texttt{1,2,3,4}.

cpulist_pack

Packs a CPU list in the form of \texttt{1,2,3,5} to \texttt{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 \texttt{/etc/tuned/} directory, create a new directory named the same as the profile that you want to create:

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

2. In the new directory, create a file named \texttt{tuned.conf}. Add a \texttt{[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:

   ```bash
   # 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
Verification 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:
5. Verify that the Tuned profile is active and the system settings are applied:

```bash
$ tuned-adm active
Current active profile: my-profile
```

```bash
$ tuned-adm verify
Verification 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](https://man7.org/linux/man-pages/man5/tuned.conf.5.html)

### 2.11. RELATED INFORMATION

- The [tuned.conf(5) man page](https://man7.org/linux/man-pages/man5/tuned.conf.5.html)
- The Tuned project website: [https://tuned-project.org/](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.

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

5. Click Change.
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. MANAGING POWER CONSUMPTION WITH POWERTOP

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

5.1. 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.

5.2. USING POWERTOP

5.2.1. Prerequisites

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

```
# yum install powertop
```

5.2.2. Starting PowerTOP

Procedure

- To run PowerTOP, use the following command:

```
# powertop
```

**IMPORTANT**

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

5.2.3. 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.

### 5.2.4. 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:

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

### 5.2.5. Related information

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

### 5.3. 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.

#### 5.3.1. 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

### 5.3.2. 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.

### 5.3.3. The Device stats tab

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

### 5.3.4. 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 5.1. PowerTOP output
Additional resources
For more details on PowerTOP, see PowerTOP’s home page.

5.4. 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.

5.5. OPTIMIZING POWER CONSUMPTION

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

5.5.1. 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

5.5.2. 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.

5.5.3. 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
  ```

5.5.4. 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.