Red Hat Enterprise Linux 9.0

Monitoring and managing system status and performance

Optimizing system throughput, latency, and power consumption
Red Hat Enterprise Linux 9.0 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 9 in different scenarios.
Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE .......................................................... 6

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION ............................................. 7

CHAPTER 1. GETTING STARTED WITH TUNED ............................................................ 8
  1.1. THE PURPOSE OF TUNED .............................................................................. 8
  1.2. TUNED PROFILES ..................................................................................... 8
    Syntax of profile configuration ........................................................................ 8
  1.3. THE DEFAULT TUNED PROFILE ................................................................ 9
  1.4. MERGED TUNED PROFILES ..................................................................... 9
  1.5. THE LOCATION OF TUNED PROFILES ...................................................... 10
  1.6. TUNED PROFILES DISTRIBUTED WITH RHEL .............................................. 10
  1.7. TUNED CPU-PARTITIONING PROFILE ...................................................... 12
  1.8. USING THE TUNED CPU-PARTITIONING PROFILE FOR LOW-LATENCY TUNING 13
  1.9. CUSTOMIZING THE CPU-PARTITIONING TUNED PROFILE ......................... 14
  1.10. REAL-TIME TUNED PROFILES DISTRIBUTED WITH RHEL ...................... 15
  1.11. STATIC AND DYNAMIC TUNING IN TUNED .............................................. 15
  1.12. TUNED NO-DAEMON MODE .................................................................... 16
  1.13. INSTALLING AND ENABLING TUNED ...................................................... 16
  1.14. LISTING AVAILABLE TUNED PROFILES ................................................... 17
  1.15. SETTING A TUNED PROFILE ................................................................... 18
  1.16. DISABLING TUNED .................................................................................. 19

CHAPTER 2. CUSTOMIZING TUNED PROFILES ............................................................ 20
  2.1. TUNED PROFILES ..................................................................................... 20
    Syntax of profile configuration ........................................................................ 20
  2.2. THE DEFAULT TUNED PROFILE ................................................................ 20
  2.3. MERGED TUNED PROFILES ..................................................................... 21
  2.4. THE LOCATION OF TUNED PROFILES ...................................................... 21
  2.5. INHERITANCE BETWEEN TUNED PROFILES .............................................. 22
  2.6. STATIC AND DYNAMIC TUNING IN TUNED .............................................. 22
  2.7. TUNED PLUG-INS ..................................................................................... 23
    Syntax for plug-ins in TuneD profiles ................................................................. 24
    Short plug-in syntax .......................................................................................... 24
    Conflicting plug-in definitions in a profile ......................................................... 25
  2.8. AVAILABLE TUNED PLUG-INS .................................................................. 25
    Monitoring plug-ins ......................................................................................... 25
    Tuning plug-ins ................................................................................................. 25
  2.9. VARIABLES IN TUNED PROFILES ............................................................. 29
  2.10. BUILT-IN FUNCTIONS IN TUNED PROFILES .......................................... 30
  2.11. BUILT-IN FUNCTIONS AVAILABLE IN TUNED PROFILES .................... 30
  2.12. CREATING NEW TUNED PROFILES ....................................................... 31
  2.13. MODIFYING EXISTING TUNED PROFILES ............................................. 32
  2.14. SETTING THE DISK SCHEDULER USING TUNED ..................................... 33

CHAPTER 3. MONITORING PERFORMANCE USING RHEL SYSTEM ROLES ..................... 36
  3.1. INTRODUCTION TO RHEL SYSTEM ROLES ............................................. 36
  3.2. RHEL SYSTEM ROLES TERMINOLOGY .................................................... 36
  3.3. INSTALLING RHEL SYSTEM ROLES IN YOUR SYSTEM ......................... 37
  3.4. APPLYING A ROLE .................................................................................... 37
  3.5. INTRODUCTION TO THE METRICS SYSTEM ROLE .................................... 40
  3.6. USING THE METRICS SYSTEM ROLE TO MONITOR YOUR LOCAL SYSTEM WITH VISUALIZATION 41
### Chapter 4. Setting Up PCP

1. Overview of PCP
2. Installing and Enabling PCP
3. Deploying a Minimal PCP Setup
4. System Services Distributed with PCP
5. Tools Distributed with PCP
6. PCP Deployment Architectures
7. Recommended Deployment Architecture
8. Sizing Factors
9. Configuration Options for PCP Scaling
10. Example: Analyzing the Centralized Logging Deployment
11. Example: Analyzing the Federated Setup Deployment
12. Troubleshooting High Memory Usage

### Chapter 5. Logging Performance Data with PMLogger

1. Modifying the PMLogger Configuration File with PMLogConf
2. Editing the PMLogger Configuration File Manually
3. Enabling the PMLogger Service
4. Setting Up a Client System for Metrics Collection
5. Setting Up a Central Server to Collect Data
6. Replaying the PCP Log Archives with PMRep

### Chapter 6. Monitoring Performance with Performance Co-Pilot

1. Monitoring Postfix with PMDA-Postfix
2. Visually Tracing PCP Log Archives with the PCP Charts Application
3. Collecting Data from SQL Server Using PCP
4. Generating PCP Archives from SADC Archives

### Chapter 7. Performance Analysis of XFS with PCP

1. Installing XFS PMDA Manually
2. Examining XFS Performance Metrics with PMINFO
3. Resetting XFS Performance Metrics with PMSTORE
4. PCP Metric Groups for XFS
5. Per-Device PCP Metric Groups for XFS

### Chapter 8. Setting Up Graphical Representation of PCP Metrics

1. Setting Up PCP with PCP-Zerconf
2. Setting Up a Grafana-Server
3. Accessing the Grafana Web UI
4. Configuring PCP Redis
5. Creating Panels and Alert in PCP Redis Data Source
6. Adding Notification Channels for Alerts
7. Setting Up Authentication Between PCP Components
8. Installing PCP BPFtrace
9. Viewing the PCP BPFtrace System Analysis Dashboard
10. Installing PCP Vector
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.3. ROUND ROBIN PRIORITY SCHEDULING WITH SCHED_RR</td>
<td>119</td>
</tr>
<tr>
<td>14.4. NORMAL SCHEDULING WITH SCHED_OTHER</td>
<td>119</td>
</tr>
<tr>
<td>14.5. SETTING SCHEDULER POLICIES</td>
<td>119</td>
</tr>
<tr>
<td>14.6. POLICY OPTIONS FOR THE CHRT COMMAND</td>
<td>120</td>
</tr>
<tr>
<td>14.7. CHANGING THE PRIORITY OF SERVICES DURING THE BOOT PROCESS</td>
<td>121</td>
</tr>
<tr>
<td>14.8. PRIORITY MAP</td>
<td>122</td>
</tr>
<tr>
<td>14.9. TUNED CPU-PARTITIONING PROFILE</td>
<td>123</td>
</tr>
<tr>
<td>14.10. USING THE TUNED CPU-PARTITIONING PROFILE FOR LOW-LATENCY TUNING</td>
<td>124</td>
</tr>
<tr>
<td>14.11. CUSTOMIZING THE CPU-PARTITIONING TUNED PROFILE</td>
<td>125</td>
</tr>
</tbody>
</table>

CHAPTER 15. CONFIGURING RESOURCE MANAGEMENT USING CGROUPS VERSION 2 WITH SYSTEMD
15.1. PREREQUISITES                                                   127
15.2. INTRODUCTION TO RESOURCE DISTRIBUTION MODELS                    127
15.3. ALLOCATING CPU RESOURCES USING SYSTEMD                           128
15.4. CPU TIME ALLOCATION POLICY OPTIONS FOR SYSTEMD                   128
15.5. ALLOCATING MEMORY RESOURCES USING SYSTEMD                        129
15.6. MEMORY ALLOCATION CONFIGURATION OPTIONS FOR SYSTEMD              130
15.7. CONFIGURING I/O BANDWIDTH USING SYSTEMD                          130
15.8. I/O BANDWIDTH CONFIGURATION OPTIONS FOR SYSTEMD                  131

CHAPTER 16. CONFIGURING HUGE PAGES
16.1. AVAILABLE HUGE PAGE FEATURES                                     133
16.2. PARAMETERS FOR RESERVING HUGETLB PAGES AT BOOT TIME               134
16.3. CONFIGURING HUGETLB AT BOOT TIME                                 134
16.4. PARAMETERS FOR RESERVING HUGETLB PAGES AT RUN TIME                136
16.5. CONFIGURING HUGETLB AT RUN TIME                                  137
16.6. ENABLING TRANSPARENT HUGEPAGES                                   138
16.7. DISABLING TRANSPARENT HUGEPAGES                                  138
16.8. IMPACT OF PAGE SIZE ON TRANSLATION LOOKASIDE BUFFER SIZE         139
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the Multi-page HTML format. In addition, ensure you see the Feedback button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the Add Feedback pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.

- For submitting feedback via Bugzilla, create a new 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

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

- tuned.conf(5) man page.
1.3. THE DEFAULT TUNED 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><strong>throughput-performance</strong></td>
<td>The best throughput performance</td>
</tr>
<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>

Additional resources

- `tuned.conf(5)` man page.

1.4. MERGED TUNED 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.

Additional resources
1.5. THE LOCATION OF TUNED 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.

Additional resources
- `tuned.conf(5)` man page.

1.6. 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`.

It changes the `energy_performance_preference` attribute to the `normal` energy setting. It also changes the `scaling_governor` policy attribute to either the `conservative` or `powersave` CPU governor.

**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.

It changes the `energy_performance_preference` attribute to the `powersave` or `power` energy setting. It also changes the `scaling_governor` policy attribute to either the `ondemand` or `powersave` CPU governor.
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.
It changes the energy_performance_preference and scaling_governor attribute to the performance profile.

accelerator-performance

The accelerator-performance profile contains the same tuning as the throughput-performance profile. Additionally, it locks the CPU to low C states so that the latency is less than 100us. This improves the performance of certain accelerators, such as GPUs.

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).
It changes the energy_performance_preference and scaling_governor attribute to the performance profile.

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.
It inherits the latency-performance profile which changes the energy_performance_preference and scaling_governor attribute to the performance profile.

hpc-compute

A profile optimized for high-performance computing. It is based on the latency-performance profile.

network-throughput

A profile for throughput network tuning. It is based on the throughput-performance profile. It additionally increases kernel network buffers.
It inherits either the latency-performance or throughput-performance profile, and changes the energy_performance_preference and scaling_governor attribute to the performance profile.

virtual-guest
A profile designed for Red Hat Enterprise Linux 9 virtual machines and VMWare 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. It inherits the `throughput-performance` profile and changes the `energy_performance_preference` and `scaling_governor` attribute to the `performance` profile.

**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. It inherits the `throughput-performance` profile and changes the `energy_performance_preference` and `scaling_governor` attribute to the `performance` profile.

**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.

**optimize-serial-console**

A profile that tunes down I/O activity to the serial console by reducing the printk value. This should make the serial console more responsive. This profile is intended to be used as an overlay on other profiles. For example:

```
# tuned-adm profile throughput-performance optimize-serial-console
```

**mssql**

A profile provided for Microsoft SQL Server. It is based on the `throughput-performance` profile.

**intel-sst**

A profile optimized for systems with user-defined Intel Speed Select Technology configurations. This profile is intended to be used as an overlay on other profiles. For example:

```
# tuned-adm profile cpu-partitioning intel-sst
```

### 1.7. TUNED CPU-PARTITIONING PROFILE

For tuning Red Hat Enterprise Linux 9 for latency-sensitive workloads, Red Hat recommends to use the `cpu-partitioning` TuneD profile.

Prior to Red Hat Enterprise Linux 9, the low-latency Red Hat documentation described the numerous low-level steps needed to achieve low-latency tuning. In Red Hat Enterprise Linux 9, you can perform low-latency tuning more efficiently by using the `cpu-partitioning` TuneD profile. This profile is easily customizable according to the requirements for individual low-latency applications.

The following figure is an example to demonstrate how to use the `cpu-partitioning` profile. This example uses the CPU and node layout.
You can configure the cpu-partitioning profile in the `/etc/tuned/cpu-partitioning-variables.conf` file using the following configuration options:

**Isolated CPUs with load balancing**

In the cpu-partitioning figure, the blocks numbered from 4 to 23, are the default isolated CPUs. The kernel scheduler’s process load balancing is enabled on these CPUs. It is designed for low-latency processes with multiple threads that need the kernel scheduler load balancing.

You can configure the cpu-partitioning profile in the `/etc/tuned/cpu-partitioning-variables.conf` file using the `isolated_cores=cpu-list` option, which lists CPUs to isolate that will use the kernel scheduler load balancing.

The list of isolated CPUs is comma-separated or you can specify a range using a dash, such as `3-5`. This option is mandatory. Any CPU missing from this list is automatically considered a housekeeping CPU.

**Isolated CPUs without load balancing**

In the cpu-partitioning figure, the blocks numbered 2 and 3, are the isolated CPUs that do not provide any additional kernel scheduler process load balancing.

You can configure the cpu-partitioning profile in the `/etc/tuned/cpu-partitioning-variables.conf` file using the `no_balance_cores=cpu-list` option, which lists CPUs to isolate that will not use the kernel scheduler load balancing.

Specifying the `no_balance_cores` option is optional, however any CPUs in this list must be a subset of the CPUs listed in the `isolated_cores` list.

Application threads using these CPUs need to be pinned individually to each CPU.

**Housekeeping CPUs**

Any CPU not isolated in the `cpu-partitioning-variables.conf` file is automatically considered a housekeeping CPU. On the housekeeping CPUs, all services, daemons, user processes, movable kernel threads, interrupt handlers, and kernel timers are permitted to execute.

**Additional resources**

- `tuned-profiles-cpu-partitioning(7)` man page

**1.8. USING THE TUNED CPU-PARTITIONING PROFILE FOR LOW-LATENCY TUNING**
This procedure describes how to tune a system for low-latency using the TuneD’s `cpu-partitioning` profile. It uses the example of a low-latency application that can use `cpu-partitioning` and the CPU layout as mentioned in the `cpu-partitioning` figure.

The application in this case uses:

- One dedicated reader thread that reads data from the network will be pinned to CPU 2.
- A large number of threads that process this network data will be pinned to CPUs 4-23.
- A dedicated writer thread that writes the processed data to the network will be pinned to CPU 3.

**Prerequisites**

- You have installed the `cpu-partitioning` TuneD profile by using the `dnf install tuned-profiles-cpu-partitioning` command as root.

**Procedure**

1. Edit `/etc/tuned/cpu-partitioning-variables.conf` file and add the following information:

```plaintext
# Isolated CPUs with the kernel’s scheduler load balancing:
isolated_cores=2-23
# Isolated CPUs without the kernel’s scheduler load balancing:
no_balance_cores=2,3
```

2. Set the `cpu-partitioning` TuneD profile:

```plaintext
# tuned-adm profile cpu-partitioning
```

3. Reboot

After rebooting, the system is tuned for low-latency, according to the isolation in the `cpu-partitioning` figure. The application can use `taskset` to pin the reader and writer threads to CPUs 2 and 3, and the remaining application threads on CPUs 4-23.

**Additional resources**

- `tuned-profiles-cpu-partitioning(7)` man page

### 1.9. CUSTOMIZING THE CPU-PARTITIONING TUNED PROFILE

You can extend the TuneD profile to make additional tuning changes.

For example, the `cpu-partitioning` profile sets the CPUs to use `cstate=1`. In order to use the `cpu-partitioning` profile but to additionally change the CPU cstate from `cstate1` to `cstate0`, the following procedure describes a new TuneD profile named `my_profile`, which inherits the `cpu-partitioning` profile and then sets C state-0.

**Procedure**

1. Create the `/etc/tuned/my_profile` directory:

```plaintext
# mkdir /etc/tuned/my_profile
```

Red Hat Enterprise Linux 9.0 Monitoring and managing system status and performance
2. Create a tuned.conf file in this directory, and add the following content:

```
# vi /etc/tuned/my_profile/tuned.conf
[main]
summary=Customized tuning on top of cpu-partitioning
include=cpu-partitioning
[cpu]
force_latency=cstate.id:0|1
```

3. Use the new profile:

```
# tuned-adm profile my_profile
```

**NOTE**

In the shared example, a reboot is not required. However, if the changes in the `my_profile` profile require a reboot to take effect, then reboot your machine.

Additional resources

- tuned-profiles-cpu-partitioning(7) man page

1.10. REAL-TIME TUNED PROFILES DISTRIBUTED WITH RHEL

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.

1.11. 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.

---

### 1.12. 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
```

### 1.13. 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:

   # dnf install tuned

2. Enable and start the tuned service:

   # systemctl enable --now tuned

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

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

1.14. 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:
   - accelerator-performance - Throughput performance based tuning with disabled higher latency STOP states
   - 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

• tuned-adm(8) man page.

1.15. SETTING A TUNED PROFILE

This procedure activates a selected TuneD profile on your system.

Prerequisites

• The tuned service is running. See 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. View the current active TuneD profile on your system:

   # tuned-adm active

   Current active profile: selected-profile
4. Reboot the system:

```bash
# reboot
```

**Verification steps**

- Verify that the TuneD profile is active and applied:

```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**

- [tuned-adm(8) man page](#)

### 1.16. 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:

```bash
# tuned-adm off
```

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

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

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

**Additional resources**

- [tuned-adm(8) man page](#)
CHAPTER 2. CUSTOMIZING TUNED PROFILES

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

Prerequisites

- Install and enable TuneD as described in Installing and Enabling TuneD for details.

2.1. 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

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

- tuned.conf(5) man page.

2.2. THE DEFAULT TUNED 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>
</tbody>
</table>
Other cases  balanced  Balanced performance and power consumption

Additional resources

- tuned.conf(5) man page.

### 2.3. MERGED TUNED 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:

```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.

Additional resources

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

### 2.4. THE LOCATION OF TUNED 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.
2.5. 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
```

2.6. 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.

2.7. 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

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**
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.

NOTE
TunedD includes the ability to run any shell command as part of enabling or disabling a tuning profile. This enables you to extend TunedD profiles with functionality that has not been integrated into TunedD yet.
You can specify arbitrary shell commands using the `script` plug-in.

Additional resources
- `tuned.conf(5)` man page

2.8. AVAILABLE TUNED PLUG-INS
This section lists all monitoring and tuning plug-ins currently available in TunedD.

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 value of the `transparent_hugepages` option. Valid values of the `transparent_hugepages` option are:

- "always"
- "never"
- "madvise"

**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-performance

For details, see [www.x.org](http://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

### 2.9. VARIABLES IN TUNED PROFILES

Variables expand at run time when a **TuneD** profile is activated.

Using **TuneD** variables reduces the amount of necessary typing in **TuneD** profiles.

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.
2.10. BUILT-IN FUNCTIONS IN TUNED PROFILES

Built-in functions expand at run time when a TuneD profile is activated.

You can:

- 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

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

- tuned.conf(5) man page

2.11. 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.

### 2.12. CREATING NEW TUNED PROFILES

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

**Prerequisites**
- The **tuned** service is running. See **Installing and Enabling TuneD** for details.
1. In the `/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 `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:

```conf
[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:

```bash
$ 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

- `tuned.conf(5)` man page

2.13. MODIFYING EXISTING TUNED PROFILES

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

Prerequisites

- The `tuned` service is running. See 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:

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

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

   ```conf
   [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:

   ```conf
   [main]
   include=throughput-performance
   [sysctl]
   vm.swappiness=5
   ```

4. To activate the profile, use:

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

5. Verify that the `TuneD` profile is active and the system settings are applied:

   ```bash
   $ 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

- `tuned.conf(5)` man page

**2.14. SETTING THE DISK SCHEDULER USING TUNED**

This procedure creates and enables a `TuneD` profile that sets a given disk scheduler for selected block devices. The setting persists across system reboots.

In the following commands and configuration, replace:
• device with the name of the block device, for example sdf

• selected-scheduler with the disk scheduler that you want to set for the device, for example bfq

Prerequisites

• The tuned service is installed and enabled. For details, see Installing and enabling TuneD.

Procedure

1. Optional: Select an existing TuneD profile on which your profile will be based. For a list of available profiles, see TuneD profiles distributed with RHEL. To see which profile is currently active, use:

   $ tuned-adm active

2. Create a new directory to hold your TuneD profile:

   # mkdir /etc/tuned/my-profile

3. Find the system unique identifier of the selected block device:

   $ udevadm info --query=property --name=/dev/device | grep -E '(WWN|SERIAL)'

   ID_WWN=0x5002538d00000000_
   ID_SERIAL=Generic-_SD_MMC_20120501030900000-0:0
   ID_SERIAL_SHORT=20120501030900000

   NOTE

   The command in the this example will return all values identified as a World Wide Name (WWN) or serial number associated with the specified block device. Although it is preferred to use a WWN, the WWN is not always available for a given device and any values returned by the example command are acceptable to use as the device system unique ID.

4. Create the /etc/tuned/my-profile/tuned.conf configuration file. In the file, set the following options:

   a. Optional: Include an existing profile:

      [main]
      include=existing-profile

   b. Set the selected disk scheduler for the device that matches the WWN identifier:

      [disk]
      devices_udev_regex=IDNAME=device system unique id
      elevator=selected-scheduler

      Here:

      • Replace IDNAME with the name of the identifier being used (for example, ID_WWN).
• Replace device system unique id with the value of the chosen identifier (for example, 0x5002538d00000000).
To match multiple devices in the devices_udev_regex option, enclose the identifiers in parentheses and separate them with vertical bars:

```
devices_udev_regex=(ID_WWN=0x5002538d00000000)
(ID_WWN=0x1234567800000000)
```

5. Enable your profile:

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

Verification steps

1. Verify that the TuneD profile is active and 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.
```

2. Read the contents of the /sys/block/device/queue/scheduler file:

```
# cat /sys/block/device/queue/scheduler
[mq-deadline] kyber bfq none
```

In the file name, replace device with the block device name, for example sdc.

The active scheduler is listed in square brackets ([]).

Additional resources

• Customizing TuneD profiles.
CHAPTER 3. MONITORING PERFORMANCE USING RHEL SYSTEM ROLES

As a system administrator, you can use the Metrics RHEL System Role to monitor the performance of a system.

3.1. INTRODUCTION TO RHEL SYSTEM ROLES

RHEL System Roles is a collection of Ansible roles and modules. RHEL System Roles provide a configuration interface to remotely manage multiple RHEL systems. The interface enables managing system configurations across multiple versions of RHEL, as well as adopting new major releases.

On Red Hat Enterprise Linux 9, the interface currently consists of the following roles:

- Certificate Issuance and Renewal
- Kernel Settings
- Metrics
- Network Bound Disk Encryption client and Network Bound Disk Encryption server
- Networking
- Postfix
- SSH client
- SSH server
- System-wide Cryptographic Policies
- Terminal Session Recording

All these roles are provided by the `rhel-system-roles` package available in the `AppStream` repository.

Additional resources

- Red Hat Enterprise Linux (RHEL) System Roles
- Documentation in the `/usr/share/doc/rhel-system-roles/` directory

3.2. RHEL SYSTEM ROLES TERMINOLOGY

You can find the following terms across this documentation:

Ansible playbook

Playbooks are Ansible’s configuration, deployment, and orchestration language. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process.

Control node

Any machine with Ansible installed. You can run commands and playbooks, invoking `/usr/bin/ansible` or `/usr/bin/ansible-playbook`, from any control node. You can use any computer that has Python installed on it as a control node – laptops, shared desktops, and servers can all run Ansible. However, you cannot use a Windows machine as a control node. You can have multiple control nodes.
Inventory
A list of managed nodes. An inventory file is also sometimes called a “hostfile”. Your inventory can specify information like IP address for each managed node. An inventory can also organize managed nodes, creating and nesting groups for easier scaling. To learn more about inventory, see the Working with Inventory section.

Managed nodes
The network devices, servers, or both that you manage with Ansible. Managed nodes are also sometimes called “hosts”. Ansible is not installed on managed nodes.

3.3. INSTALLING RHEL SYSTEM ROLES IN YOUR SYSTEM

To use the RHEL System Roles, install the required packages in your system.

Prerequisites
- The Ansible Core package is installed on the control machine.
- You have Ansible packages installed in the system you want to use as a control node.

Procedure
1. Install the `rhel-system-roles` package on the system that you want to use as a control node:

```
# dnf install rhel-system-roles
```

2. Install the Ansible Core package:

```
# dnf install ansible-core
```

The Ansible Core package provides the `ansible-playbook` CLI, the Ansible Vault functionality, and the basic modules and filters required by RHEL Ansible content.

As a result, you are able to create an Ansible playbook.

Additional resources
- The Red Hat Enterprise Linux (RHEL) System Roles
- The `ansible-playbook` man page.

3.4. APPLYING A ROLE

The following procedure describes how to apply a particular role.

Prerequisites
- Ensure that the `rhel-system-roles` package is installed on the system that you want to use as a control node:

```
# dnf install rhel-system-roles
```

1. Install the Ansible Core package:
# dnf install ansible-core

The Ansible Core package provides the `ansible-playbook` CLI, the Ansible Vault functionality, and the basic modules and filters required by RHEL Ansible content.

- Ensure that you are able to create an Ansible inventory. Inventories represent the hosts, host groups, and some of the configuration parameters used by the Ansible playbooks.

Playbooks are typically human-readable, and are defined in `ini`, `yaml`, `json`, and other file formats.

- Ensure that you are able to create an Ansible playbook. Playbooks represent Ansible’s configuration, deployment, and orchestration language. By using playbooks, you can declare and manage configurations of remote machines, deploy multiple remote machines or orchestrate steps of any manual ordered process.

A playbook is a list of one or more **plays**. Every **play** can include Ansible variables, tasks, or roles.

Playbooks are human-readable, and are defined in the `yaml` format.

**Procedure**

1. Create the required Ansible inventory containing the hosts and groups that you want to manage. Here is an example using a file called `inventory.ini` of a group of hosts called `webservers`:

   ```ini
   [webservers]
   host1
   host2
   host3
   ```

2. Create an Ansible playbook including the required role. The following example shows how to use roles through the `roles:` option for a playbook:

   The following example shows how to use roles through the `roles:` option for a given **play**:

   ```yaml
   ---
   - hosts: webservers
     roles:
       - rhel-system-roles.network
       - rhel-system-roles.postfix
   ```
NOTE

Every role includes a README file, which documents how to use the role and supported parameter values. You can also find an example playbook for a particular role under the documentation directory of the role. Such documentation directory is provided by default with the `rhel-system-roles` package, and can be found in the following location:

```
/usr/share/doc/rhel-system-roles/SUBSYSTEM/
```

Replace `SUBSYSTEM` with the name of the required role, such as `postfix`, `metrics`, `network`, `tlog`, or `ssh`.

3. To execute the playbook on specific hosts, you must perform one of the following:

- Edit the playbook to use `hosts: host1[,host2,...]`, or `hosts: all`, and execute the command:

  ```
  # ansible-playbook name.of.the.playbook
  ```

- Edit the inventory to ensure that the hosts you want to use are defined in a group, and execute the command:

  ```
  # ansible-playbook -i name.of.the.inventory name.of.the.playbook
  ```

- Specify all hosts when executing the `ansible-playbook` command:

  ```
  # ansible-playbook -i host1,host2,... name.of.the.playbook
  ```

IMPORTANT

Be aware that the `-i` flag specifies the inventory of all hosts that are available. If you have multiple targeted hosts, but want to select a host against which you want to run the playbook, you can add a variable in the playbook to be able to select a host. For example:

Ansible Playbook | example-playbook.yml:

```yaml
- hosts: "{{ target_host }}"
  roles:
    - rhel-system-roles.network
    - rhel-system-roles.postfix
```

Playbook execution command:

```
# ansible-playbook -i host1,..hostn -e target_host=host5 example-playbook.yml
```

Additional resources

- Ansible playbooks
- Using roles in Ansible playbook
3.5. INTRODUCTION TO THE METRICS SYSTEM ROLE

RHEL System Roles is a collection of Ansible roles and modules that provide a consistent configuration interface to remotely manage multiple RHEL systems. The Metrics System Role configures performance analysis services for the local system and, optionally, includes a list of remote systems to be monitored by the local system. The Metrics System Role enables you to use `pcp` to monitor your systems performance without having to configure `pcp` separately, as the set-up and deployment of `pcp` is handled by the playbook.

<table>
<thead>
<tr>
<th>Role variable</th>
<th>Description</th>
<th>Example usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>metrics_monitored_hosts</td>
<td>List of remote hosts to be analyzed by the target host. These hosts will have metrics recorded on the target host, so ensure enough disk space exists below <code>/var/log</code> for each host.</td>
<td><code>metrics_monitored_hosts: [&quot;webserver.example.com&quot;, &quot;database.example.com&quot;]</code></td>
</tr>
<tr>
<td>metrics_retention_days</td>
<td>Configures the number of days for performance data retention before deletion.</td>
<td><code>metrics_retention_days: 14</code></td>
</tr>
<tr>
<td>metrics_graph_service</td>
<td>A boolean flag that enables the host to be set up with services for performance data visualization via <code>pcp</code> and <code>grafana</code>. Set to false by default.</td>
<td><code>metrics_graph_service: no</code></td>
</tr>
<tr>
<td>metrics_query_service</td>
<td>A boolean flag that enables the host to be set up with time series query services for querying recorded <code>pcp</code> metrics via <code>redis</code>. Set to false by default.</td>
<td><code>metrics_query_service: no</code></td>
</tr>
<tr>
<td>metrics_provider</td>
<td>Specifies which metrics collector to use to provide metrics. Currently, <code>pcp</code> is the only supported metrics provider.</td>
<td><code>metrics_provider: &quot;pcp&quot;</code></td>
</tr>
</tbody>
</table>

**NOTE**

For details about the parameters used in `metrics_connections` and additional information about the Metrics System Role, see the `/usr/share/ansible/roles/rhel-system-roles.metrics/README.md` file.
3.6. USING THE METRICS SYSTEM ROLE TO MONITOR YOUR LOCAL SYSTEM WITH VISUALIZATION

This procedure describes how to use the Metrics RHEL System Role to monitor your local system while simultaneously provisioning data visualization via Grafana.

Prerequisites

- The Ansible Core package is installed on the control machine.
- You have the rhel-system-roles package installed on the machine you want to monitor.

Procedure

1. Configure localhost in the the /etc/ansible/hosts Ansible inventory by adding the following content to the inventory:

```yaml
localhost ansible_connection=local
```

2. Create an Ansible playbook with the following content:

```yaml
---
- hosts: localhost
  vars:
    metrics_graph_service: yes
  roles:
    - rhel-system-roles.metrics
```

3. Run the Ansible playbook:

```
# ansible-playbook name_of_your_playbook.yml
```

**NOTE**

Since the metrics_graph_service boolean is set to value="yes", Grafana is automatically installed and provisioned with pcp added as a data source.

4. To view visualization of the metrics being collected on your machine, access the grafana web interface as described in Accessing the Grafana web UI.

3.7. USING THE METRICS SYSTEM ROLE TO SETUP A FLEET OF INDIVIDUAL SYSTEMS TO MONITOR THEMSELVES

This procedure describes how to use the Metrics System Role to set up a fleet of machines to monitor themselves.

Prerequisites

- The Ansible Core package is installed on the control machine.
- You have the rhel-system-roles package installed on the machine you want to use to run the playbook.
• You have the SSH connection established.

Procedure

1. Add the name or IP of the machines you wish to monitor via the playbook to the `/etc/ansible/hosts` Ansible inventory file under an identifying group name enclosed in brackets:

   [remotes]
   webserver.example.com
database.example.com

2. Create an Ansible playbook with the following content:

   ```yml
   ---
   - hosts: remotes
     vars:
       metrics_retention_days: 0
     roles:
       - rhel-system-roles.metrics
   ```

3. Run the Ansible playbook:

   ```bash
   # ansible-playbook name_of_your_playbook.yml -k
   ```

Where the `-k` prompt for password to connect to remote system.

### 3.8. USING THE METRICS SYSTEM ROLE TO MONITOR A FLEET OF MACHINES CENTRALLY VIA YOUR LOCAL MACHINE

This procedure describes how to use the Metrics System Role to set up your local machine to centrally monitor a fleet of machines while also provisioning visualization of the data via grafana and querying of the data via redis.

**Prerequisites**

- The Ansible Core package is installed on the control machine.
- You have the `rhel-system-roles` package installed on the machine you want to use to run the playbook.

**Procedure**

1. Create an Ansible playbook with the following content:

   ```yml
   ---
   - hosts: localhost
     vars:
       metrics_graph_service: yes
       metrics_query_service: yes
       metrics_retention_days: 10
       metrics_monitored_hosts: ["database.example.com", "webserver.example.com"]
     roles:
       - rhel-system-roles.metrics
   ```
2. Run the Ansible playbook:

```
# ansible-playbook name_of_your_playbook.yml
```

**NOTE**

Since the `metrics_graph_service` and `metrics_query_service` booleans are set to value="yes", Grafana is automatically installed and provisioned with PCP added as a data source with the PCP data recording indexed into Redis, allowing the PCP querying language to be used for complex querying of the data.

3. To view graphical representation of the metrics being collected centrally by your machine and to query the data, access the Grafana web interface as described in Accessing the Grafana web UI.

### 3.9. SETTING UP AUTHENTICATION WHILE MONITORING A SYSTEM USING THE METRICS SYSTEM ROLE

PCP supports the **scram-sha-256** authentication mechanism through the Simple Authentication Security Layer (SASL) framework. The Metrics RHEL System Role automates the steps to setup authentication using the **scram-sha-256** authentication mechanism. This procedure describes how to setup authentication using the Metrics RHEL System Role.

**Prerequisites**

- The Ansible Core package is installed on the control machine.
- You have the `rhel-system-roles` package installed on the machine you want to use to run the playbook.

**Procedure**

1. Include the following variables in the Ansible playbook you want to setup authentication for:

   ```
   vars:
   metrics_username: your_username
   metrics_password: your_password
   ```

2. Run the Ansible playbook:

   ```
   # ansible-playbook name_of_your_playbook.yml
   ```

**Verification steps**

- Verify the **sasl** configuration:

  ```
  # pminfo -f -h "pcp://ip_adress?username=your_username" disk.dev.read
  Password:
  disk.dev.read
  inst [0 or "sda"] value 19540
  ```

  `ip_adress` should be replaced by the IP address of the host.
3.10. USING THE METRICS SYSTEM ROLE TO CONFIGURE AND ENABLE METRICS COLLECTION FOR SQL SERVER

This procedure describes how to use the Metrics RHEL System Role to automate the configuration and enabling of metrics collection for Microsoft SQL Server via `pcp` on your local system.

Prerequisites

- The Ansible Core package is installed on the control machine.
- You have the `rhel-system-roles` package installed on the machine you want to monitor.
- You have installed Microsoft SQL Server for Red Hat Enterprise Linux and established a ‘trusted’ connection to an SQL server.
- You have installed the Microsoft ODBC driver for SQL Server for Red Hat Enterprise Linux.

Procedure

1. Configure `localhost` in the the `/etc/ansible/hosts` Ansible inventory by adding the following content to the inventory:

   ```
   localhost ansible_connection=local
   ```

2. Create an Ansible playbook that contains the following content:

   ```
   ---
   - hosts: localhost
     roles:
       - role: rhel-system-roles.metrics
     vars:
       metrics_from_mssql: yes
   ```

3. Run the Ansible playbook:

   ```
   # ansible-playbook name_of_your_playbook.yml
   ```

Verification steps

- Use the `pcp` command to verify that SQL Server PMDA agent (mssql) is loaded and running:

  ```
  # pcp
  platform: Linux rhel82-2.local 4.18.0-167.el8.x86_64 #1 SMP Sun Dec 15 01:24:23 UTC 2019 x86_64
  hardware: 2 cpus, 1 disk, 1 node, 2770MB RAM
  timezone: PDT+7
  services: pmcd pmproxy
  pmcd: Version 5.0.2-1, 12 agents, 4 clients
  pmda: root pmcd proc pmproxy xfs linux nfsclient mmv kvm mssql jbd2 dm
  pmlogger: primary logger: /var/log/pcp/pmlogger/rhel82-2.local/20200326.16.31
  pmie: primary engine: /var/log/pcp/pmie/rhel82-2.local/pmie.log
  ```
Additional resources

- For more information about using Performance Co-Pilot for Microsoft SQL Server, see this Red Hat Developers Blog post.

[[1]] This documentation is installed automatically with the `rhel-system-roles` package.
CHAPTER 4. SETTING UP PCP

Performance Co-Pilot (PCP) is a suite of tools, services, and libraries for monitoring, visualizing, storing, and analyzing system-level performance measurements.

This section describes how to install and enable PCP on your system.

4.1. OVERVIEW OF PCP

You can add performance metrics using Python, Perl, C++, and C interfaces. Analysis tools can use the Python, C++, C client APIs directly, and rich web applications can explore all available performance data using a JSON interface.

You can analyze data patterns by comparing live results with archived data.

Features of PCP:

- Light-weight distributed architecture, which is useful during the centralized analysis of complex systems.
- It allows the monitoring and management of real-time data.
- It allows logging and retrieval of historical data.

PCP has the following components:

- The Performance Metric Collector Daemon (pmcd) collects performance data from the installed Performance Metric Domain Agents (pmda). PMDAs can be individually loaded or unloaded on the system and are controlled by the PMCD on the same host.
- Various client tools, such as pminfo or pmstat, can retrieve, display, archive, and process this data on the same host or over the network.
- The pcp package provides the command-line tools and underlying functionality.
- The pcp-gui package provides the graphical application. Install the pcp-gui package by executing the dnf install pcp-gui command. For more information, see Visually tracing PCP log archives with the PCP Charts application.

Additional resources

- pcp(1) man page
- /usr/share/doc/pcp-doc/ directory
- Tools distributed with PCP
- Index of Performance Co-Pilot (PCP) articles, solutions, tutorials, and white papers from on Red Hat Customer Portal
- Side-by-side comparison of PCP tools with legacy tools Red Hat Knowledgebase article
- PCP upstream documentation

4.2. INSTALLING AND ENABLING PCP
To begin using PCP, install all the required packages and enable the PCP monitoring services.

This procedure describes how to install PCP using the `pcp` package. If you want to automate the PCP installation, install it using the `pcp-zeroconf` package. For more information on installing PCP by using `pcp-zeroconf`, see Setting up PCP with pcp-zeroconf.

**Procedure**

1. Install the `pcp` package:
   ```
   # dnf install pcp
   ```

2. Enable and start the `pmcd` service on the host machine:
   ```
   # systemctl enable pmcd
   # systemctl start pmcd
   ```

**Verification steps**

- Verify if the `pmcd` process is running on the host:
  ```
  # pcp
  ```

**Performance Co-Pilot configuration on workstation:**

- **platform:** Linux workstation 4.18.0-80.el8.x86_64 #1 SMP Wed Mar 13 12:02:46 UTC 2019 x86_64
- **hardware:** 12 cpus, 2 disks, 1 node, 36023MB RAM
- **timezone:** CEST-2
- **services:** pmcd
- **pmcd:** Version 4.3.0-1, 8 agents
- **pmda:** root pmcd proc xfs linux mmv kvm jbd2

**Additional resources**

- `pmcd(1)` man page
- Tools distributed with PCP

### 4.3. DEPLOYING A MINIMAL PCP SETUP

The minimal PCP setup collects performance statistics on Red Hat Enterprise Linux. The setup involves adding the minimum number of packages on a production system needed to gather data for further analysis.

You can analyze the resulting `tar.gz` file and the archive of the `pmlogger` output using various PCP tools and compare them with other sources of performance information.

**Prerequisites**

- PCP is installed. For more information, see Installing and enabling PCP.
Procedure

1. Update the `pmlogger` configuration:
   
   ```
   # pmlogconf -r /var/lib/pcp/config/pmlogger/config.default
   ```

2. Start the `pmcd` and `pmlogger` services:
   
   ```
   # systemctl start pmcd.service
   # systemctl start pmlogger.service
   ```

3. Execute the required operations to record the performance data.

4. Stop the `pmcd` and `pmlogger` services:
   
   ```
   # systemctl stop pmcd.service
   # systemctl stop pmlogger.service
   ```

5. Save the output and save it to a `tar.gz` file named based on the host name and the current date and time:
   
   ```
   # cd /var/log/pcp/pmlogger/
   # tar -czf $(hostname).$(date +%F-%Hh%M).pcp.tar.gz $(hostname)
   ```

   Extract this file and analyze the data using PCP tools.

Additional resources

- `pmlogconf(1)`, `pmlogger(1)`, and `pmcd(1)` man pages
- Tools distributed with PCP
- System services distributed with PCP

4.4. SYSTEM SERVICES DISTRIBUTED WITH PCP

The following table describes roles of various system services, which are distributed with PCP.

Table 4.1. Roles of system services distributed with PCP

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmcd</td>
<td>The Performance Metric Collector Daemon (PMCD).</td>
</tr>
<tr>
<td>pmie</td>
<td>The Performance Metrics Inference Engine.</td>
</tr>
<tr>
<td>pmlogger</td>
<td>The performance metrics logger.</td>
</tr>
</tbody>
</table>
pmproxy | The realtime and historical performance metrics proxy, time series query and REST API service.

### 4.5. TOOLS DISTRIBUTED WITH PCP

The following table describes usage of various tools, which are distributed with PCP.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pcp</td>
<td>Displays the current status of a Performance Co-Pilot installation.</td>
</tr>
<tr>
<td>pcp-atop</td>
<td>Shows the system-level occupation of the most critical hardware resources from the performance point of view: CPU, memory, disk, and network.</td>
</tr>
<tr>
<td>pcp-atopsar</td>
<td>Generates a system-level activity report over a variety of system resource utilization. The report is generated from a raw logfile previously recorded using pmlogger or the -w option of pcp-atop.</td>
</tr>
<tr>
<td>pcp-dmcache</td>
<td>Displays information about configured Device Mapper Cache targets, such as: device IOPs, cache and metadata device utilization, as well as hit and miss rates and ratios for both reads and writes for each cache device.</td>
</tr>
<tr>
<td>pcp-dstat</td>
<td>Displays metrics of one system at a time. To display metrics of multiple systems, use --host option.</td>
</tr>
<tr>
<td>pcp-free</td>
<td>Reports on free and used memory in a system.</td>
</tr>
<tr>
<td>pcp-htop</td>
<td>Displays all processes running on a system along with their command line arguments in a manner similar to the top command, but allows you to scroll vertically and horizontally as well as interact using a mouse. You can also view processes in a tree format and select and act on multiple processes at once.</td>
</tr>
<tr>
<td>pcp-ipcs</td>
<td>Displays information on the inter-process communication (IPC) facilities that the calling process has read access for.</td>
</tr>
<tr>
<td>pcp-numastat</td>
<td>Displays NUMA allocation statistics from the kernel memory allocator.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>pcp-pidstat</code></td>
<td>Displays information about individual tasks or processes running on the system such as: CPU percentage, memory and stack usage, scheduling, and priority. Reports live data for the local host by default.</td>
</tr>
<tr>
<td><code>pcp-ss</code></td>
<td>Displays socket statistics collected by the pmdasockets Performance Metrics Domain Agent (PMDA).</td>
</tr>
<tr>
<td><code>pcp-uptime</code></td>
<td>Displays how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.</td>
</tr>
<tr>
<td><code>pcp-vmstat</code></td>
<td>Provides a high-level system performance overview every 5 seconds. Displays information about processes, memory, paging, block IO, traps, and CPU activity.</td>
</tr>
<tr>
<td><code>pmchart</code></td>
<td>Plots performance metrics values available through the facilities of the Performance Co-Pilot.</td>
</tr>
<tr>
<td><code>pmclient</code></td>
<td>Displays high-level system performance metrics by using the Performance Metrics Application Programming Interface (PMAPI).</td>
</tr>
<tr>
<td><code>pmconfig</code></td>
<td>Displays the values of configuration parameters.</td>
</tr>
<tr>
<td><code>pmdbg</code></td>
<td>Displays available Performance Co-Pilot debug control flags and their values.</td>
</tr>
<tr>
<td><code>pmdiff</code></td>
<td>Compares the average values for every metric in either one or two archives, in a given time window, for changes that are likely to be of interest when searching for performance regressions.</td>
</tr>
<tr>
<td><code>pmdumplog</code></td>
<td>Displays control, metadata, index, and state information from a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td><code>pmdumptext</code></td>
<td>Outputs the values of performance metrics collected live or from a Performance Co-Pilot archive.</td>
</tr>
<tr>
<td><code>pmerr</code></td>
<td>Displays available Performance Co-Pilot error codes and their corresponding error messages.</td>
</tr>
<tr>
<td><code>pmfind</code></td>
<td>Finds PCP services on the network.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pmie</td>
<td>An inference engine that periodically evaluates a set of arithmetic, logical, and rule expressions. The metrics are collected either from a live system, or from a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmieconf</td>
<td>Displays or sets configurable pmie variables.</td>
</tr>
<tr>
<td>pmiectl</td>
<td>Manages non-primary instances of pmie.</td>
</tr>
<tr>
<td>pminfo</td>
<td>Displays information about performance metrics. The metrics are collected either from a live system, or from a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmiostat</td>
<td>Reports I/O statistics for SCSI devices (by default) or device-mapper devices (with the -x dm option).</td>
</tr>
<tr>
<td>pmlc</td>
<td>Interactively configures active pmlogger instances.</td>
</tr>
<tr>
<td>pmlogcheck</td>
<td>Identifies invalid data in a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmlogconf</td>
<td>Creates and modifies a pmlogger configuration file.</td>
</tr>
<tr>
<td>pmlogctl</td>
<td>Manages non-primary instances of pmlogger.</td>
</tr>
<tr>
<td>pmloglabel</td>
<td>Verifies, modifies, or repairs the label of a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmlogsummary</td>
<td>Calculates statistical information about performance metrics stored in a Performance Co-Pilot archive file.</td>
</tr>
<tr>
<td>pmprobe</td>
<td>Determines the availability of performance metrics.</td>
</tr>
<tr>
<td>pmrep</td>
<td>Reports on selected, easily customizable, performance metrics values.</td>
</tr>
<tr>
<td>pmsocks</td>
<td>Allows access to a Performance Co-Pilot hosts through a firewall.</td>
</tr>
<tr>
<td>pmstat</td>
<td>Periodically displays a brief summary of system performance.</td>
</tr>
<tr>
<td>pmstore</td>
<td>Modifies the values of performance metrics.</td>
</tr>
<tr>
<td>pmtrace</td>
<td>Provides a command line interface to the trace PMDA.</td>
</tr>
<tr>
<td>pmval</td>
<td>Displays the current value of a performance metric.</td>
</tr>
</tbody>
</table>
4.6. PCP DEPLOYMENT ARCHITECTURES

Performance Co-Pilot (PCP) offers many options to accomplish advanced setups. From the huge variety of possible architectures, this section describes how to scale your PCP deployment based on the recommended deployment setup by Red Hat, sizing factors, and configuration options.

PCP supports multiple deployment architectures, based on the scale of the PCP deployment.

Available scaling deployment setup variants:

**Localhost**
Each service runs locally on the monitored machine. When you start a service without any configuration changes, this is the default deployment. Scaling beyond the individual node is not possible in this case.
By default, the deployment setup for Redis is standalone, localhost. However, Redis can optionally perform in a highly-available and highly scalable clustered fashion, where data is shared across multiple hosts. Another viable option is to deploy a Redis cluster in the cloud, or to utilize a managed Redis cluster from a cloud vendor.

**Decentralized**
The only difference between localhost and decentralized setup is the centralized Redis service. In this model, the host executes `pmlogger` service on each monitored host and retrieves metrics from a local `pmcd` instance. A local `pmproxy` service then exports the performance metrics to a central Redis instance.

**Figure 4.1. Decentralized logging**

![Decentralized logging diagram]

**Centralized logging - pmlogger farm**
When the resource usage on the monitored hosts is constrained, another deployment option is a `pmlogger` farm, which is also known as centralized logging. In this setup, a single logger host executes multiple `pmlogger` processes, and each is configured to retrieve performance metrics from a different remote `pmcd` host. The centralized logger host is also configured to execute the `pmproxy` service, which discovers the resulting PCP archives logs and loads the metric data into a Redis instance.
Federated - multiple pmlogger farms

For large scale deployments, Red Hat recommends to deploy multiple pmlogger farms in a federated fashion. For example, one pmlogger farm per rack or data center. Each pmlogger farm loads the metrics into a central Redis instance.
NOTE

By default, the deployment setup for Redis is standalone, localhost. However, Redis can optionally perform in a highly-available and highly scalable clustered fashion, where data is shared across multiple hosts. Another viable option is to deploy a Redis cluster in the cloud, or to utilize a managed Redis cluster from a cloud vendor.
Additional resources

- `pcp(1)`, `pmlogger(1)`, `pmproxy(1)`, and `pmcd(1)` man pages
- Recommended deployment architecture

### 4.7. RECOMMENDED DEPLOYMENT ARCHITECTURE

The following table describes the recommended deployment architectures based on the number of monitored hosts.

Table 4.3. Recommended deployment architecture

<table>
<thead>
<tr>
<th>Number of hosts (N)</th>
<th>1-10</th>
<th>10-100</th>
<th>100-1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>pmcd servers</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>pmlogger servers</td>
<td>1 to N</td>
<td>N/10 to N</td>
<td>N/100 to N</td>
</tr>
<tr>
<td>pmproxy servers</td>
<td>1 to N</td>
<td>1 to N/10</td>
<td>N/100 to N</td>
</tr>
<tr>
<td>Redis servers</td>
<td>1 to N</td>
<td>1 to N/10</td>
<td>N/100 to N</td>
</tr>
<tr>
<td>Redis cluster</td>
<td>No</td>
<td>Maybe</td>
<td>Yes</td>
</tr>
<tr>
<td>Recommended deployment setup</td>
<td>Localhost, Decentralized, or Centralized logging</td>
<td>Decentralized, Centralized logging, or Federated</td>
<td>Decentralized or Federated</td>
</tr>
</tbody>
</table>

### 4.8. SIZING FACTORS

The following are the sizing factors required for scaling:

**Remote system size**

The number of CPUs, disks, network interfaces, and other hardware resources affects the amount of data collected by each `pmlogger` on the centralized logging host.

**Logged Metrics**

The number and types of logged metrics play an important role. In particular, the per-process proc.* metrics require a large amount of disk space, for example, with the standard `pcp-zeroconf` setup, 10s logging interval, 11 MB without proc metrics versus 155 MB with proc metrics - a factor of 10 times more. Additionally, the number of instances for each metric, for example the number of CPUs, block devices, and network interfaces also impacts the required storage capacity.

**Logging Interval**

The interval how often metrics are logged, affects the storage requirements. The expected daily PCP archive file sizes are written to the `pmlogger.log` file for each `pmlogger` instance. These values are uncompressed estimates. Since PCP archives compress very well, approximately 10:1, the actual long term disk space requirements can be determined for a particular site.

**pmlogrewrite**
After every PCP upgrade, the `pmlogrewrite` tool is executed and rewrites old archives if there were changes in the metric metadata from the previous version and the new version of PCP. This process duration scales linear with the number of archives stored.

**Additional resources**

- `pmlogrewrite(1)` and `pmlogger(1)` man pages

### 4.9. CONFIGURATION OPTIONS FOR PCP SCALING

The following are the configuration options, which are required for scaling:

**sysctl and rlimit settings**

When archive discovery is enabled, `pmproxy` requires four descriptors for every `pmlogger` that it is monitoring or log-tailing, along with the additional file descriptors for the service logs and `pmproxy` client sockets, if any. Each `pmlogger` process uses about 20 file descriptors for the remote `pmcd` socket, archive files, service logs, and others. In total, this can exceed the default 1024 soft limit on a system running around 200 `pmlogger` processes. The `pmproxy` service in `pcp-5.3.0` and later automatically increases the soft limit to the hard limit. On earlier versions of PCP, tuning is required if a high number of `pmlogger` processes are to be deployed, and this can be accomplished by increasing the soft or hard limits for `pmlogger`. For more information, see [How to set limits (ulimit)](https://www.example.com) for services run by systemd.

**Local Archives**

The `pmlogger` service stores metrics of local and remote `pmcds` in the `/var/log/pcp/pmlogger/` directory. To control the logging interval of the local system, update the `/etc/pcp/pmlogger/control.d/configfile` file and add `-t X` in the arguments, where `X` is the logging interval in seconds. To configure which metrics should be logged, execute `pmlogconf /var/lib/pcp/config/pmlogger/config.clienthostname`. This command deploys a configuration file with a default set of metrics, which can optionally be further customized. To specify retention settings, that is when to purge old PCP archives, update the `/etc/sysconfig/pmlogger_timers` file and specify `PMLOGGER_DAILY_PARAMS="-E -k X"`, where `X` is the amount of days to keep PCP archives.

**Redis**

The `pmproxy` service sends logged metrics from `pmlogger` to a Redis instance. The following are the available two options to specify the retention settings in the `/etc/pcp/pmproxy/pmproxy.conf` configuration file:

- **stream.expire** specifies the duration when stale metrics should be removed, that is metrics which were not updated in a specified amount of time in seconds.

- **stream.maxlen** specifies the maximum number of metric values for one metric per host. This setting should be the retention time divided by the logging interval, for example 20160 for 14 days of retention and 60s logging interval (60*60*24*14/60)

**Additional resources**

- `pmproxy(1)`, `pmlogger(1)`, and `sysctl(8)` man pages

### 4.10. EXAMPLE: ANALYZING THE CENTRALIZED LOGGING DEPLOYMENT

The following results were gathered on a centralized logging setup, also known as pmlogger farm Red Hat Enterprise Linux 9.0 Monitoring and managing system status and performance
The following results were gathered on a centralized logging setup, also known as pmlogger farm deployment, with a default `pcp-zeroconf 5.3.0` installation, where each remote host is an identical container instance running `pmcd` on a server with 64 CPU cores, 376 GB RAM, and one disk attached.

The logging interval is 10s, proc metrics of remote nodes are not included, and the memory values refer to the Resident Set Size (RSS) value.

### Table 4.4. Detailed utilization statistics for 10s logging interval

<table>
<thead>
<tr>
<th>Number of Hosts</th>
<th>10</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCP Archives Storage per Day</td>
<td>91 MB</td>
<td>522 MB</td>
</tr>
<tr>
<td><strong>pmlogger</strong> Memory</td>
<td>160 MB</td>
<td>580 MB</td>
</tr>
<tr>
<td><strong>pmlogger</strong> Network per Day (In)</td>
<td>2 MB</td>
<td>9 MB</td>
</tr>
<tr>
<td><strong>pmproxy</strong> Memory</td>
<td>1.4 GB</td>
<td>6.3 GB</td>
</tr>
<tr>
<td>Redis Memory per Day</td>
<td>2.6 GB</td>
<td>12 GB</td>
</tr>
</tbody>
</table>

### Table 4.5. Used resources depending on monitored hosts for 60s logging interval

<table>
<thead>
<tr>
<th>Number of Hosts</th>
<th>10</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCP Archives Storage per Day</td>
<td>20 MB</td>
<td>120 MB</td>
<td>271 MB</td>
</tr>
<tr>
<td><strong>pmlogger</strong> Memory</td>
<td>104 MB</td>
<td>524 MB</td>
<td>1049 MB</td>
</tr>
<tr>
<td><strong>pmlogger</strong> Network per Day (In)</td>
<td>0.38 MB</td>
<td>1.75 MB</td>
<td>3.48 MB</td>
</tr>
<tr>
<td><strong>pmproxy</strong> Memory</td>
<td>2.67 GB</td>
<td>5.5GB</td>
<td>9 GB</td>
</tr>
<tr>
<td>Redis Memory per Day</td>
<td>0.54 GB</td>
<td>2.65 GB</td>
<td>5.3 GB</td>
</tr>
</tbody>
</table>

**NOTE**

The **pmproxy** queues Redis requests and employs Redis pipelining to speed up Redis queries. This can result in high memory usage. For troubleshooting this issue, see **Troubleshooting high memory usage**.

### 4.11. EXAMPLE: ANALYZING THE FEDERATED SETUP DEPLOYMENT

The following results were observed on a federated setup, also known as multiple **pmlogger** farms, consisting of three centralized logging (**pmlogger** farm) setups, where each **pmlogger** farm was monitoring 100 remote hosts, that is 300 hosts in total.
This setup of the `pmlogger` farms is identical to the configuration mentioned in the Example: Analyzing the centralized logging deployment for 60s logging interval, except that the Redis servers were operating in cluster mode.

Table 4.6. Used resources depending on federated hosts for 60s logging interval

<table>
<thead>
<tr>
<th>PCP Archives Storage per Day</th>
<th>pmlogger Memory</th>
<th>Network per Day (In/Out)</th>
<th>pmproxy Memory</th>
<th>Redis Memory per Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>277 MB</td>
<td>1058 MB</td>
<td>15.6 MB / 12.3 MB</td>
<td>6-8 GB</td>
<td>5.5 GB</td>
</tr>
</tbody>
</table>

Here, all values are per host. The network bandwidth is higher due to the inter-node communication of the Redis cluster.

### 4.12. TROUBLESHOOTING HIGH MEMORY USAGE

The following scenarios can result in high memory usage:

- The `pmproxy` process is busy processing new PCP archives and does not have spare CPU cycles to process Redis requests and responses.
- The Redis node or cluster is overloaded and cannot process incoming requests on time.

The `pmproxy` service daemon uses Redis streams and supports the configuration parameters, which are PCP tuning parameters and affects Redis memory usage and key retention. The `/etc/pcp/pmproxy/pmproxy.conf` file lists the available configuration options for `pmproxy` and the associated APIs.

This section describes how to troubleshoot high memory usage issue.

**Prerequisites**

1. Install the `pcp-pmda-redis` package:

   ```
   # dnf install pcp-pmda-redis
   ```

2. Install the redis PMDA:

   ```
   # cd /var/lib/pcp/pmdas/redis && ./Install
   ```

**Procedure**

- To troubleshoot high memory usage, execute the following command and observe the `inflight` column:

  ```
  $ pmrep :pmproxy
  backlog inflight reqs/s resp/s wait req err resp err changed throttled
  byte count count/s count/s s/s count/s count/s count/s count/s
  14:59:08 0 0 N/A N/A N/A N/A N/A N/A N/A
  14:59:09 0 0 2268.9 2268.9 28 0 0 2.0 4.0
  14:59:10 0 0 0.0 0.0 0 0 0 0.0 0.0
  14:59:11 0 0 0.0 0.0 0 0 0 0.0 0.0
  ```
This column shows how many Redis requests are in-flight, which means they are queued or sent, and no reply was received so far.

A high number indicates one of the following conditions:

- The **pmproxy** process is busy processing new PCP archives and does not have spare CPU cycles to process Redis requests and responses.

- The Redis node or cluster is overloaded and cannot process incoming requests on time.

- To troubleshoot the high memory usage issue, reduce the number of **pmlogger** processes for this farm, and add another pmlogger farm. Use the federated - multiple pmlogger farms setup. If the Redis node is using 100% CPU for an extended amount of time, move it to a host with better performance or use a clustered Redis setup instead.

- To view the **pmproxy.redis.*** metrics, use the following command:

```bash
$ pminfo -ftd pmproxy.redis
pmproxy.redis.responses.wait [wait time for responses]
  Data Type: 64-bit unsigned int InDom: PM_INDOM_NULL 0xffffffff
  Semantics: counter Units: microsec
  value 546028367374
pmproxy.redis.responses.error [number of error responses]
  Data Type: 64-bit unsigned int InDom: PM_INDOM_NULL 0xffffffff
  Semantics: counter Units: count
  value 1164

pmproxy.redis.requests.inflight.bytes [bytes allocated for inflight requests]
  Data Type: 64-bit int InDom: PM_INDOM_NULL 0xffffffff
  Semantics: discrete Units: byte
  value 0

pmproxy.redis.requests.inflight.total [inflight requests]
  Data Type: 64-bit unsigned int InDom: PM_INDOM_NULL 0xffffffff
  Semantics: discrete Units: count
  value 0

[...]
```

To view how many Redis requests are inflight, see the `pmproxy.redis.requests.inflight.total` metric and `pmproxy.redis.requests.inflight.bytes` metric to view how many bytes are occupied by all current inflight Redis requests.

In general, the redis request queue would be zero but can build up based on the usage of large pmlogger farms, which limits scalability and can cause high latency for **pmproxy** clients.

- Use the **pminfo** command to view information about performance metrics. For example, to view the **redis.*** metrics, use the following command:

```bash
$ pminfo -ftd redis
redis.redis_build_id [Build ID]
  Data Type: string InDom: 24.0 0x6000000
  Semantics: discrete Units: count
  inst [0 or "localhost:6379"] value "87e335e57c7af755"
redis.total_commands_processed [Total number of commands processed by the server]
  Data Type: 64-bit unsigned int InDom: 24.0 0x6000000
  Semantics: counter Units: count
```
inst [0 or "localhost:6379"] value 595627069

redis.used_memory_peak [Peak memory consumed by Redis (in bytes)]  
Data Type: 32-bit unsigned int  InDom: 24.0 0x6000000  
Semantics: instant  Units: count  
inst [0 or "localhost:6379"] value 572234920

To view the peak memory usage, see the `redis.used_memory_peak` metric.

Additional resources

- `pmdaredis(1), pmproxy(1), and pminfo(1)` man pages
- `PCP deployment architectures`
CHAPTER 5. LOGGING PERFORMANCE DATA WITH PMLOGGER

With the PCP tool you can log the performance metric values and replay them later. This allows you to perform a retrospective performance analysis.

Using the `pmlogger` tool, you can:

- Create the archived logs of selected metrics on the system
- Specify which metrics are recorded on the system and how often

5.1. MODIFYING THE PMLOGGER CONFIGURATION FILE WITH PMLOGCONF

When the `pmlogger` service is running, PCP logs a default set of metrics on the host.

Use the `pmlogconf` utility to check the default configuration. If the `pmlogger` configuration file does not exist, `pmlogconf` creates it with a default metric values.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.

Procedure

1. Create or modify the `pmlogger` configuration file:

   ```bash
   # pmlogconf -r /var/lib/pcp/config/pmlogger/config.default
   ```

2. Follow `pmlogconf` prompts to enable or disable groups of related performance metrics and to control the logging interval for each enabled group.

Additional resources

- `pmlogconf(1)` and `pmlogger(1)` man pages
- Tools distributed with PCP
- System services distributed with PCP

5.2. EDITING THE PMLOGGER CONFIGURATION FILE MANUALLY

To create a tailored logging configuration with specific metrics and given intervals, edit the `pmlogger` configuration file manually. The default `pmlogger` configuration file is `/var/lib/pcp/config/pmlogger/config.default`. The configuration file specifies which metrics are logged by the primary logging instance.

In manual configuration, you can:

- Record metrics which are not listed in the automatic configuration.
- Choose custom logging frequencies.
Add PMDA with the application metrics.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.

Procedure

- Open and edit the `/var/lib/pcp/config/pmlogger/config.default` file to add specific metrics:

```plaintext
# It is safe to make additions from here on ... 
#
log mandatory on every 5 seconds {
    xfs.write
    xfs.write_bytes
    xfs.read
    xfs.read_bytes
}
log mandatory on every 10 seconds {
    xfs.allocs
    xfs.block_map
    xfs.transactions
    xfs.log
}
[access]
disallow * : all;
allow localhost : enquire;
```

Additional resources

- `pmlogger(1)` man page
- Tools distributed with PCP
- System services distributed with PCP

**5.3. ENABLING THE PMLOGGER SERVICE**

The `pmlogger` service must be started and enabled to log the metric values on the local machine.

This procedure describes how to enable the `pmlogger` service.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.

Procedure

- Start and enable the `pmlogger` service:
# systemctl start pmlogger
# systemctl enable pmlogger

Verification steps

- Verify if the **pmlogger** service is enabled:

  ```bash
  # pcp
  
  Performance Co-Pilot configuration on workstation:
  
  platform: Linux workstation 4.18.0-80.el8.x86_64 #1 SMP Wed Mar 13 12:02:46 UTC 2019
  x86_64
  hardware: 12 cpus, 2 disks, 1 node, 36023MB RAM
  timezone: CEST-2
  services: pmcd
  pmda: root pmcd proc xfs linux mmv kvm jbd2
  pmlogger: primary logger: /var/log/pcp/pmlogger/workstation/20190827.15.54
  ```

Additional resources

- **pmlogger(1)** man page
- Tools distributed with PCP
- System services distributed with PCP
- `/var/lib/pcp/config/pmlogger/config.default` file

### 5.4. SETTING UP A CLIENT SYSTEM FOR METRICS COLLECTION

This procedure describes how to set up a client system so that a central server can collect metrics from clients running PCP.

**Prerequisites**

- PCP is installed. For more information, see [Installing and enabling PCP](#).

**Procedure**

1. Install the **pcp-system-tools** package:

   ```bash
   # dnf install pcp-system-tools
   ```

2. Configure an IP address for **pmcd**:

   ```bash
   # echo "-i 192.168.4.62" >>/etc/pcp/pmcd/pmcd.options
   ```

   Replace `192.168.4.62` with the IP address, the client should listen on.

   By default, **pmcd** is listening on the localhost.
3. Configure the firewall to add the public zone permanently:

```bash
# firewall-cmd --permanent --zone=public --add-port=44321/tcp
success

# firewall-cmd --reload
success
```

4. Set an SELinux boolean:

```bash
# setsebool -P pcp_bind_all_unreserved_ports on
```

5. Enable the **pmcd** and **pmlogger** services:

```bash
# systemctl enable pmcd pmlogger
# systemctl restart pmcd pmlogger
```

### Verification steps

- Verify if the **pmcd** is correctly listening on the configured IP address:

```bash
# ss -tlp | grep 44321
LISTEN 0 5 127.0.0.1:44321 0.0.0.0:* users:("pmcd",pid=151595,fd=6))
LISTEN 0 5 192.168.4.62:44321 0.0.0.0:* users:("pmcd",pid=151595,fd=0))
LISTEN 0 5 0.0.0.0:44321 0.0.0.0:* users:("pmcd",pid=151595,fd=7))
```

### Additional resources

- **pmlogger(1)**, **firewall-cmd(1)**, **ss(8)**, and **setsebool(8)** man pages
- Tools distributed with PCP
- System services distributed with PCP
- `/var/lib/pcp/config/pmlogger/config.default` file

### 5.5. SETTING UP A CENTRAL SERVER TO COLLECT DATA

This procedure describes how to create a central server to collect metrics from clients running PCP.

#### Prerequisites

- PCP is installed. For more information, see **Installing and enabling PCP**.
- Client is configured for metrics collection. For more information, see **Setting up a client system for metrics collection**.

#### Procedure

1. Install the **pcp-system-tools** package:

```bash
# dnf install pcp-system-tools
```
2. Create the `/etc/pcp/pmlogger/control.d/remote` file with the following content:

```plaintext
# DO NOT REMOVE OR EDIT THE FOLLOWING LINE
$version=1.1

192.168.4.13 n n PCP_ARCHIVE_DIR/rhel7u4a -r -T24h10m -c config.rhel7u4a
192.168.4.14 n n PCP_ARCHIVE_DIR/rhel6u10a -r -T24h10m -c config.rhel6u10a
192.168.4.62 n n PCP_ARCHIVE_DIR/rhel8u1a -r -T24h10m -c config.rhel8u1a
192.168.4.69 n n PCP_ARCHIVE_DIR/rhel9u3a -r -T24h10m -c config.rhel9u3a

Replace 192.168.4.13, 192.168.4.14, 192.168.4.62 and 192.168.4.69 with the client IP addresses.
```

3. Enable the `pmcd` and `pmlogger` services:

```plaintext
# systemctl enable pmcd pmlogger
# systemctl restart pmcd pmlogger
```

**Verification steps**

- Ensure that you can access the latest archive file from each directory:

```plaintext
# for i in /var/log/pcp/pmlogger/rhel*/./*.0; do pmdumplog -L $i; done
```

Log Label (Log Format Version 2)
Performance metrics from host rhel6u10a.local
  ending     Mon Nov 25 22:06:04.874 2019
Archive timezone: JST-9
PID for pmlogger: 24002

Log Label (Log Format Version 2)
Performance metrics from host rhel7u4a
  commencing Tue Nov 26 06:49:24.954 2019
  ending     Tue Nov 26 07:06:24.979 2019
Archive timezone: CET-1
PID for pmlogger: 10941
[..]

The archive files from the `/var/log/pcp/pmlogger/` directory can be used for further analysis and graphing.

**Additional resources**

- `pmlogger(1)` man page
- **Tools distributed with PCP**
- **System services distributed with PCP**
- `/var/lib/pcp/config/pmlogger/config.default` file

## 5.6. REPLAYING THE PCP LOG ARCHIVES WITH PMREP

After recording the metric data, you can replay the PCP log archives. To export the logs to text files and import them into spreadsheets, use PCP utilities such as `pcp2csv`, `pcp2xml`, `pmrep` or `pmlogsummary`. 
Using the `pmrep` tool, you can:

- View the log files
- Parse the selected PCP log archive and export the values into an ASCII table
- Extract the entire archive log or only select metric values from the log by specifying individual metrics on the command line

**Prerequisites**

- PCP is installed. For more information, see [Installing and enabling PCP](#).
- The `pmlogger` service is enabled. For more information, see [Enabling the pmlogger service](#).
- Install the `pcp-system-tools` package:

  ```shell
  # dnf install pcp-gui
  ```

**Procedure**

- Display the data on the metric:

  ```shell
  $ pmrep --start @3:00am --archive 20211128 --interval 5seconds --samples 10 --output csv
  disk.dev.write
  Time,"disk.dev.write-sda","disk.dev.write-sdb"
  2021-11-28 03:00:00,,
  2021-11-28 03:00:05,4.000,5.200
  2021-11-28 03:00:10,1.600,7.600
  2021-11-28 03:00:15,0.800,7.100
  2021-11-28 03:00:20,16.600,8.400
  2021-11-28 03:00:25,21.400,7.200
  2021-11-28 03:00:30,21.200,6.800
  2021-11-28 03:00:35,21.000,27.600
  2021-11-28 03:00:40,12.400,33.800
  2021-11-28 03:00:45,9.800,20.600
  ```

  The mentioned example displays the data on the `disk.dev.write` metric collected in an archive at a 5 second interval in comma-separated-value format.

  **NOTE**

  Replace `20211128` in this example with a filename containing the `pmlogger` archive you want to display data for.

**Additional resources**

- `pmlogger(1)`, `pmrep(1)`, and `pmlogsummary(1)` man pages
- Tools distributed with PCP
- System services distributed with PCP
CHAPTER 6. MONITORING PERFORMANCE WITH PERFORMANCE CO-PILOT

Performance Co-Pilot (PCP) is a suite of tools, services, and libraries for monitoring, visualizing, storing, and analyzing system-level performance measurements.

As a system administrator, you can monitor the system’s performance using the the PCP application in Red Hat Enterprise Linux 9.

6.1. MONITORING POSTFIX WITH PMDA-POSTFIX

This procedure describes how to monitor performance metrics of the postfix mail server with pmda-postfix. It helps to check how many emails are received per second.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.
- The pmlogger service is enabled. For more information, see Enabling the pmlogger service.

Procedure

1. Install the following packages:
   a. Install the pcp-system-tools:
      
      # dnf install pcp-system-tools
   
   b. Install the pmda-postfix package to monitor postfix:
      
      # dnf install pcp-pmda-postfix postfix
   
   c. Install the logging daemon:
      
      # dnf install rsyslog
   
   d. Install the mail client for testing:
      
      # dnf install mutt

2. Enable the postfix and rsyslog services:

   # systemctl enable postfix rsyslog
   # systemctl restart postfix rsyslog

3. Enable the SELinux boolean, so that pmda-postfix can access the required log files:

   # setsebool -P pcp_read_generic_logs=on

4. Install the PMDA:

   # cd /var/lib/pcp/pmdas/postfix/
# ./Install

Updating the Performance Metrics Name Space (PMNS) ...
Terminate PMDA if already installed ...
Updating the PMCD control file, and notifying PMCD ...
Waiting for pmcd to terminate ...
Starting pmcd ...
Check postfix metrics have appeared ... 7 metrics and 58 values

Verification steps

- Verify the **pmda-postfix** operation:

  ```
echo testmail | mutt root
  ```

- Verify the available metrics:

  ```
  # pminfo postfix
  postfix.received
  postfix.sent
  postfix.queues.incoming
  postfix.queues.maildrop
  postfix.queues.hold
  postfix.queues.deferred
  postfix.queues.active
  ```

Additional resources

- `rsyslogd(8)`, `postfix(1)`, and `setsebool(8)` man pages

- Tools distributed with PCP

- System services distributed with PCP

- `/var/lib/pcp/config/pmlogger/config.default` file

### 6.2. VISUALLY TRACING PCP LOG ARCHIVES WITH THE PCP CHARTS APPLICATION

After recording metric data, you can replay the PCP log archives as graphs. The metrics are sourced from one or more live hosts with alternative options to use metric data from PCP log archives as a source of historical data. To customize the **PCP Charts** application interface to display the data from the performance metrics, you can use line plot, bar graphs, or utilization graphs.

Using the **PCP Charts** application, you can:

- Replay the data in the **PCP Charts** application application and use graphs to visualize the retrospective data alongside live data of the system.

- Plot performance metric values into graphs.

- Display multiple charts simultaneously.
Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.
- Logged performance data with the pmlogger. For more information, see Logging performance data with pmlogger.
- Install the pcp-gui package:
  
  # dnf install pcp-gui

Procedure

1. Launch the PCP Charts application from the command line:

  # pmchart

Figure 6.1. PCP Charts application

The pmtime server settings are located at the bottom. The start and pause button allows you to control:

- The interval in which PCP polls the metric data
- The date and time for the metrics of historical data

2. Click File and then New Chart to select metric from both the local machine and remote machines by specifying their host name or address. Advanced configuration options include the ability to manually set the axis values for the chart, and to manually choose the color of the plots.

3. Record the views created in the PCP Charts application:

   Following are the options to take images or record the views created in the PCP Charts application:

   - Click File and then Export to save an image of the current view.
• Click **Record** and then **Start** to start a recording. Click **Record** and then **Stop** to stop the recording. After stopping the recording, the recorded metrics are archived to be viewed later.

4. Optional: In the **PCP Charts** application, the main configuration file, known as the **view**, allows the metadata associated with one or more charts to be saved. This metadata describes all chart aspects, including the metrics used and the chart columns. Save the custom **view** configuration by clicking **File** and then **Save View**, and load the **view** configuration later.

The following example of the **PCP Charts** application view configuration file describes a stacking chart graph showing the total number of bytes read and written to the given XFS file system **loop1**:

```plaintext
#kmchart
version 1

chart title "Filesystem Throughput /loop1" style stacking antialiasing off
plot legend "Read rate" metric xfs.read_bytes instance "loop1"
plot legend "Write rate" metric xfs.write_bytes instance "loop1"
```

**Additional resources**

- **pmchart(1)** and **pmtime(1)** man pages
- **Tools distributed with PCP**

**6.3. COLLECTING DATA FROM SQL SERVER USING PCP**

The SQL Server agent is available in Performance Co-Pilot (PCP), which helps you to monitor and analyze database performance issues.

This procedure describes how to collect data for Microsoft SQL Server via **pcp** on your system.

**Prerequisites**

- You have installed Microsoft SQL Server for Red Hat Enterprise Linux and established a 'trusted' connection to an SQL server.
- You have installed the Microsoft ODBC driver for SQL Server for Red Hat Enterprise Linux.

**Procedure**

1. Install PCP:

```
# dnf install pcp-zeroconf
```

2. Install packages required for the **pyodbc** driver:

```
# dnf install python3-pyodbc
```

3. Install the **mssql** agent:

   a. Install the Microsoft SQL Server domain agent for PCP:

```
# dnf install pcp-pmda-mssql
```
b. Edit the /etc/pcp/mssql/mssql.conf file to configure the SQL server account’s username and password for the mssql agent. Ensure that the account you configure has access rights to performance data.

username: user_name
password: user_password

Replace user_name with the SQL Server account and user_password with the SQL Server user password for this account.

4. Install the agent:

```
# cd /var/lib/pcp/pmdas/mssql
# ./Install
Updating the Performance Metrics Name Space (PMNS) ... 
Terminate PMDA if already installed ... 
Updating the PMCD control file, and notifying PMCD ... 
Check mssql metrics have appeared ... 168 metrics and 598 values 
[...]
```

Verification steps

- Using the pcp command, verify if the SQL Server PMDA (mssql) is loaded and running:

```
$ pcp
Performance Co-Pilot configuration on rhel.local:

platform: Linux rhel.local 4.18.0-167.el8.x86_64 #1 SMP Sun Dec 15 01:24:23 UTC 2019 x86_64
hardware: 2 cpus, 1 disk, 1 node, 2770MB RAM
timezone: PDT+7
services: pmcd pmproxy
  pmcd: Version 5.0.2-1, 12 agents, 4 clients
  pmda: root pmcd proc pmproxy xfs linux nfsclient mmv kvm mssql jbd2 dm
  pmlogger: primary logger: /var/log/pcp/pmlogger/rhel.local/20200326.16.31
  pmie: primary engine: /var/log/pcp/pmie/rhel.local/pmie.log

- View the complete list of metrics that PCP can collect from the SQL Server:

```
# pminfo mssql
```

- After viewing the list of metrics, you can report the rate of transactions. For example, to report on the overall transaction count per second, over a five second time window:

```
# pmval -t 1 -T 5 mssql.databases.transactions
```

- View the graphical chart of these metrics on your system by using the pmchart command. For more information, see Visually tracing PCP log archives with the PCP Charts application.

Additional resources

- pcp(1), pminfo(1), pmval(1), pmchart(1), and pmdamssql(1) man pages
6.4. GENERATING PCP ARCHIVES FROM SADC ARCHIVES

You can use the **sadf** tool provided by the **sysstat** package to generate PCP archives from native **sadc** archives.

**Prerequisites**

- A **sadc** archive has been created:

  ```
  # /usr/lib64/sa/sadc 1 5 -
  ```

  In this example, **sadc** is sampling system data 1 time in a 5 second interval. The outfile is specified as `-` which results in **sadc** writing the data to the standard system activity daily data file. This file is named saDD and is located in the `/var/log/sa` directory by default.

**Procedure**

- Generate a PCP archive from a **sadc** archive:

  ```
  # sadf -l -O pcparchive=/tmp/recording -2
  ```

  In this example, using the `-2` option results in **sadf** generating a PCP archive from a **sadc** archive recorded 2 days ago.

**Verification steps**

You can use PCP commands to inspect and analyze the PCP archive generated from a **sadc** archive as you would a native PCP archive. For example:

- To show a list of metrics in the PCP archive generated from an **sadc** archive archive, run:

  ```
  $ pminfo --archive /tmp/recording
  Disk.dev.avactive
  Disk.dev.read
  Disk.dev.write
  Disk.dev.blkread
  [...] 
  ```

- To show the timespace of the archive and hostname of the PCP archive, run:

  ```
  $ pmdumplog --label /tmp/recording
  Log Label (Log Format Version 2)
  Performance metrics from host shard
  commencing Tue Jul 20 00:10:30.642477 2021
  ending Wed Jul 21 00:10:30.222176 2021
  ```

- To plot performance metrics values into graphs, run:

  ```
  $ pmchart --archive /tmp/recording
  ```
CHAPTER 7. PERFORMANCE ANALYSIS OF XFS WITH PCP

The XFS PMDA ships as part of the `pcp` package and is enabled by default during the installation. It is used to gather performance metric data of XFS file systems in Performance Co-Pilot (PCP).

This section describes how to analyze XFS file system's performance using PCP.

7.1. INSTALLING XFS PMDA MANUALLY

If the XFS PMDA is not listed in the `pcp` configuration output, install the PMDA agent manually.

This procedure describes how to manually install the PMDA agent.

Prerequisites

- PCP is installed. For more information, see Installing and enabling PCP.

Procedure

1. Navigate to the xfs directory:
   
   ```
   # cd /var/lib/pcp/pmdas/xfs/
   ```

Verification steps

- Verify that the `pmcd` process is running on the host and the XFS PMDA is listed as enabled in the configuration:

  ```
  # pcp
  ```

Performance Co-Pilot configuration on workstation:

- platform: Linux workstation 4.18.0-80.el8.x86_64 #1 SMP Wed Mar 13 12:02:46 UTC 2019 x86_64
- hardware: 12 cpus, 2 disks, 1 node, 36023MB RAM
- timezone: CEST-2
- services: pmcd
- pmcd: Version 4.3.0-1, 8 agents
- pmda: root pmcd proc xfs linux mmv kvm jbd2

Additional resources

- `pmcd(1)` man page
- Tools distributed with PCP

7.2. EXAMINING XFS PERFORMANCE METRICS WITH PMINFO

PCP enables XFS PMDA to allow the reporting of certain XFS metrics per each of the mounted XFS file systems. This makes it easier to pinpoint specific mounted file system issues and evaluate performance.

The `pminfo` command provides per-device XFS metrics for each mounted XFS file system.
This procedure displays a list of all available metrics provided by the XFS PMDA.

**Prerequisites**

- PCP is installed. For more information, see [Installing and enabling PCP](#).

**Procedure**

- Display the list of all available metrics provided by the XFS PMDA:
  
  ```
  # pminfo xfs
  ```

- Display information for the individual metrics. The following examples examine specific XFS read and write metrics using the `pminfo` tool:
  
  - Display a short description of the `xfs.write_bytes` metric:
    
    ```
    # pminfo --oneline xfs.write_bytes
    xfs.write_bytes [number of bytes written in XFS file system write operations]
    ```

  - Display a long description of the `xfs.read_bytes` metric:
    
    ```
    # pminfo --helptext xfs.read_bytes
    xfs.read_bytes
    Help:
    This is the number of bytes read via read(2) system calls to files in XFS file systems. It can be used in conjunction with the read_calls count to calculate the average size of the read operations to file in XFS file systems.
    ```

  - Obtain the current performance value of the `xfs.read_bytes` metric:
    
    ```
    # pminfo --fetch xfs.read_bytes
    xfs.read_bytes
    value 4891346238
    ```

  - Obtain per-device XFS metrics with `pminfo`:
    
    ```
    # pminfo --fetch --oneline xfs.perdev.read xfs.perdev.write
    xfs.perdev.read [number of XFS file system read operations]
    inst [0 or "loop1"] value 0
    inst [0 or "loop2"] value 0
    xfs.perdev.write [number of XFS file system write operations]
    inst [0 or "loop1"] value 86
    inst [0 or "loop2"] value 0
    ```

**Additional resources**
7.3. RESETTING XFS PERFORMANCE METRICS WITH PMSTORE

With PCP, you can modify the values of certain metrics, especially if the metric acts as a control variable, such as the `xfs.control.reset` metric. To modify a metric value, use the `pmstore` tool.

This procedure describes how to reset XFS metrics using the `pmstore` tool.

**Prerequisites**

- PCP is installed. For more information, see [Installing and enabling PCP](#).

**Procedure**

1. Display the value of a metric:
   ```bash
   $ pminfo -f xfs.write
   xfs.write
   value 325262
   ```

2. Reset all the XFS metrics:
   ```bash
   # pmstore xfs.control.reset 1
   xfs.control.reset old value=0 new value=1
   ```

**Verification steps**

- View the information after resetting the metric:
  ```bash
  $ pminfo --fetch xfs.write
  xfs.write
  value 0
  ```

**Additional resources**

- `pmstore(1)` and `pminfo(1)` man pages
- Tools distributed with PCP
- PCP metric groups for XFS

7.4. PCP METRIC GROUPS FOR XFS

The following table describes the available PCP metric groups for XFS.
Table 7.1. Metric groups for XFS

<table>
<thead>
<tr>
<th>Metric Group</th>
<th>Metrics provided</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>xfs.*</code></td>
<td>General XFS metrics including the read and write operation counts, read and write byte counts. Along with counters for the number of times inodes are flushed, clustered and number of failure to cluster.</td>
</tr>
<tr>
<td><code>xfs.allocs.*</code></td>
<td>Range of metrics regarding the allocation of objects in the file system, these include number of extent and block creations/frees. Allocation tree lookup and compares along with extend record creation and deletion from the btree.</td>
</tr>
<tr>
<td><code>xfs.alloc_btree.*</code></td>
<td></td>
</tr>
<tr>
<td><code>xfs.block_map.*</code></td>
<td>Metrics include the number of block map read/write and block deletions, extent list operations for insertion, deletions and lookups. Also operations counters for compares, lookups, insertions and deletion operations from the blockmap.</td>
</tr>
<tr>
<td><code>xfs.bmap_btree.*</code></td>
<td></td>
</tr>
<tr>
<td><code>xfs.dir_ops.*</code></td>
<td>Counters for directory operations on XFS file systems for creation, entry deletions, count of “getdent” operations.</td>
</tr>
<tr>
<td><code>xfs.transactions.*</code></td>
<td>Counters for the number of meta-data transactions, these include the count for the number of synchronous and asynchronous transactions along with the number of empty transactions.</td>
</tr>
<tr>
<td><code>xfs.inode_ops.*</code></td>
<td>Counters for the number of times that the operating system looked for an XFS inode in the inode cache with different outcomes. These count cache hits, cache misses, and so on.</td>
</tr>
<tr>
<td><code>xfs.log.*</code></td>
<td>Counters for the number of log buffer writes over XFS file systems includes the number of blocks written to disk. Metrics also for the number of log flushes and pinning.</td>
</tr>
<tr>
<td><code>xfs.log_tail.*</code></td>
<td></td>
</tr>
<tr>
<td><code>xfs.xstrat.*</code></td>
<td>Counts for the number of bytes of file data flushed out by the XFS flush daeomon along with counters for number of buffers flushed to contiguous and non-contiguous space on disk.</td>
</tr>
<tr>
<td><code>xfs.attr.*</code></td>
<td>Counts for the number of attribute get, set, remove and list operations over all XFS file systems.</td>
</tr>
<tr>
<td><code>xfs.quota.*</code></td>
<td>Metrics for quota operation over XFS file systems, these include counters for number of quota reclaims, quota cache misses, cache hits and quota data reclaims.</td>
</tr>
</tbody>
</table>
### 7.5. PER-DEVICE PCP METRIC GROUPS FOR XFS

The following table describes the available per-device PCP metric group for XFS.

**Table 7.2. Per-device PCP metric groups for XFS**

<table>
<thead>
<tr>
<th>Metric Group</th>
<th>Metrics provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>xfs.perdev.*</td>
<td>General XFS metrics including the read and write operation counts, read and write byte counts. Along with counters for the number of times inodes are flushed, clustered and number of failure to cluster.</td>
</tr>
<tr>
<td>xfs.perdev.allocs.*</td>
<td>Range of metrics regarding the allocation of objects in the file system, these include number of extent and block creations/frees. Allocation tree lookup and compares along with extend record creation and deletion from the btree.</td>
</tr>
<tr>
<td>xfs.perdev.alloc_btree.*</td>
<td></td>
</tr>
<tr>
<td>xfs.perdev.block_map.*</td>
<td>Metrics include the number of block map read/write and block deletions, extent list operations for insertion, deletions and lookups. Also operations counters for compares, lookups, insertions and deletion operations from the blockmap.</td>
</tr>
<tr>
<td>xfs.perdev.bmap_btree.*</td>
<td></td>
</tr>
<tr>
<td>xfs.perdev.dir_ops.*</td>
<td>Counters for directory operations of XFS file systems for creation, entry deletions, count of &quot;getdent&quot; operations.</td>
</tr>
<tr>
<td>xfs.perdev.transactions.*</td>
<td>Counters for the number of meta-data transactions, these include the count for the number of synchronous and asynchronous transactions along with the number of empty transactions.</td>
</tr>
<tr>
<td>xfs.perdev.inode_ops.*</td>
<td>Counters for the number of times that the operating system looked for an XFS inode in the inode cache with different outcomes. These count cache hits, cache misses, and so on.</td>
</tr>
<tr>
<td>Metric Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>xfs.perdev.log.*</td>
<td>Counters for the number of log buffer writes over XFS filesystems includes the number of blocks written to disk. Metrics also for the number of log flushes and pinning.</td>
</tr>
<tr>
<td>xfs.perdev.log_tail.*</td>
<td></td>
</tr>
<tr>
<td>xfs.perdev.xstrat.*</td>
<td>Counts for the number of bytes of file data flushed out by the XFS flush daemon along with counters for number of buffers flushed to contiguous and non-contiguous space on disk.</td>
</tr>
<tr>
<td>xfs.perdev.attr.*</td>
<td>Counts for the number of attribute get, set, remove and list operations over all XFS file systems.</td>
</tr>
<tr>
<td>xfs.perdev.quota.*</td>
<td>Metrics for quota operation over XFS file systems, these include counters for number of quota reclaims, quota cache misses, cache hits and quota data reclaims.</td>
</tr>
<tr>
<td>xfs.perdev.buffer.*</td>
<td>Range of metrics regarding XFS buffer objects. Counters include the number of requested buffer calls, successful buffer locks, waited buffer locks, miss_locks, miss_retries and buffer hits when looking up pages.</td>
</tr>
<tr>
<td>xfs.perdev.btrees.*</td>
<td>Metrics regarding the operations of the XFS btrees.</td>
</tr>
</tbody>
</table>
CHAPTER 8. SETTING UP GRAPHICAL REPRESENTATION OF PCP METRICS

Using a combination of `pcp`, `grafana`, `pcp redis`, `pcp bpftrace`, and `pcp vector` provides graphs, based on the live data or data collected by Performance Co-Pilot (PCP).

This section describes how to set up and access the graphical representation of PCP metrics.

8.1. SETTING UP PCP WITH PCP-ZEROCONF

This procedure describes how to set up PCP on a system with the `pcp-zeroconf` package. Once the `pcp-zeroconf` package is installed, the system records the default set of metrics into archived files.

Procedure

- Install the `pcp-zeroconf` package:
  
  ```bash
  # dnf install pcp-zeroconf
  ```

Verification steps

- Ensure that the `pmlogger` service is active, and starts archiving the metrics:
  
  ```bash
  # pcp | grep pmlogger
  pmlogger: primary logger: /var/log/pcp/pmlogger/localhost.localdomain/20200401.00.12
  ```

Additional resources

- `pmlogger` man page
- Monitoring performance with Performance Co-Pilot

8.2. SETTING UP A GRAFANA-SERVER

Grafana generates graphs that are accessible from a browser. The `grafana-server` is a back-end server for the Grafana dashboard. It listens, by default, on all interfaces, and provides web services accessed through the web browser. The `grafana-pcp` plugin interacts with the `pmproxy` protocol in the backend.

This procedure describes how to set up a `grafana-server`.

Prerequisites

- PCP is configured. For more information, see Setting up PCP with `pcp-zeroconf`.

Procedure

1. Install the following packages:
   
   ```bash
   # dnf install grafana grafana-pcp
   ```

2. Restart and enable the following service:
# systemctl restart grafana-server
# systemctl enable grafana-server

3. Open the server’s firewall for network traffic to the Grafana service.

   # firewall-cmd --permanent --add-service=grafana
   success

   # firewall-cmd --reload
   success

Verification steps

- Ensure that the `grafana-server` is listening and responding to requests:

   # ss -ntlp | grep 3000
   LISTEN 0 128 *:3000 *:* users:(("grafana-server",pid=19522,fd=7))

- Ensure that the `grafana-pcp` plugin is installed:

   # grafana-cli plugins ls | grep performancecopilot-pcp-app
   performancecopilot-pcp-app @ 3.1.0

Additional resources

- `pmproxy(1)` and `grafana-server` man pages

8.3. ACCESSING THE GRAFANA WEB UI

This procedure describes how to access the Grafana web interface.

Using the Grafana web interface, you can:

- add PCP Redis, PCP bpftrace, and PCP Vector data sources
- create dashboard
- view an overview of any useful metrics
- create alerts in PCP Redis

Prerequisites

1. PCP is configured. For more information, see Setting up PCP with pcp-zeroconf.

2. The `grafana-server` is configured. For more information, see Setting up a grafana-server.

Procedure

1. On the client system, open a browser and access the `grafana-server` on port `3000`, using `http://192.0.2.0:3000` link.
   Replace `192.0.2.0` with your machine IP.
2. For the first login, enter admin in both the Email or username and Password field. Grafana prompts to set a New password to create a secured account. If you want to set it later, click Skip.

3. From the menu, hover over the Configuration icon and then click Plugins.

4. In the Plugins tab, type performance co-pilot in the Search by name or type text box and then click Performance Co-Pilot (PCP) plugin.

5. In the Plugins / Performance Co-Pilot pane, click Enable.

6. Click Grafana icon. The Grafana Home page is displayed.

**Figure 8.1. Home Dashboard**

![Home Dashboard](image)

**NOTE**

The top corner of the screen has a similar icon, but it controls the general Dashboard settings.

7. In the Grafana Home page, click Add your first data source to add PCP Redis, PCP bpftrace, and PCP Vector data sources. For more information on adding data source, see:

   - To add pcp redis data source, view default dashboard, create a panel, and an alert rule, see Creating panels and alert in PCP Redis data source.

   - To add pcp bpftrace data source and view the default dashboard, see Viewing the PCP bpftrace System Analysis dashboard.

   - To add pcp vector data source, view the default dashboard, and to view the vector checklist, see Viewing the PCP Vector Checklist.

8. Optional: From the menu, hover over the admin profile icon to change the Preferences including Edit Profile, Change Password, or to Sign out.
8.4. CONFIGURING PCP REDIS

This section provides information for configuring PCP Redis data source.

Use the PCP Redis data source to:

- View data archives
- Query time series using pmseries language
- Analyze data across multiple hosts

Prerequisites

1. PCP is configured. For more information, see Setting up PCP with pcp-zeroconf.
2. The grafana-server is configured. For more information, see Setting up a grafana-server.

Procedure

1. Install the redis package:

   ```
   # dnf install redis
   ```

2. Start and enable the following services:

   ```
   # systemctl start pmproxy redis
   # systemctl enable pmproxy redis
   ```

3. Mail transfer agent, for example, sendmail or postfix is installed and configured.

4. Ensure that the allow_loading_unsigned_plugins parameter is set to PCP Redis database in the grafana.ini file:

   ```
   # vi /etc/grafana/grafana.ini
   allow_loading_unsigned_plugins = pcp-redis-datasource
   ```

5. Restart the grafana-server:

   ```
   # systemctl restart grafana-server
   ```

Verification steps

- Ensure that the pmproxy and redis are working:

  ```
  # pmseries disk.dev.read
  2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
  ```
This command does not return any data if the `redis` package is not installed.

Additional resources
- `pmseries(1)` man page

8.5. CREATING PANELS AND ALERT IN PCP REDIS DATA SOURCE

After adding the PCP Redis data source, you can view the dashboard with an overview of useful metrics, add a query to visualize the load graph, and create alerts that help you to view the system issues after they occur.

Prerequisites

1. The PCP Redis is configured. For more information, see Configuring PCP Redis.
2. The `grafana-server` is accessible. For more information, see Accessing the Grafana web UI.

Procedure

1. Log into the Grafana web UI.
2. In the Grafana Home page, click Add your first data source.
3. In the Add data source pane, type redis in the Filter by name or type text box and then click PCP Redis.
4. In the Data Sources / PCP Redis pane, perform the following:
   a. Add `http://localhost:44322` in the URL field and then click Save & Test.
   b. Click Dashboards tab -> Import -> PCP Redis: Host Overview to see a dashboard with an overview of any useful metrics.

Figure 8.2. PCP Redis: Host Overview

5. Add a new panel:
a. From the menu, hover over the Create icon → Dashboard → Add new panel icon to add a panel.

b. In the Query tab, select the PCP Redis from the query list instead of the selected default option and in the text field of A, enter metric, for example, kernel.all.load to visualize the kernel load graph.

c. Optional: Add Panel title and Description, and update other options from the Settings.

d. Click Save to apply changes and save the dashboard. Add Dashboard name.

e. Click Apply to apply changes and go back to the dashboard.

Figure 8.3. PCP Redis query panel

6. Create an alert rule:

a. In the PCP Redis query panel click Alert and then click Create Alert.

b. Edit the Name, Evaluate query, and For fields from the Rule, and specify the Conditions for your alert.

c. Click Save to apply changes and save the dashboard. Click Apply to apply changes and go back to the dashboard.
d. Optional: In the same panel, scroll down and click Delete icon to delete the created rule.

e. Optional: From the menu, click Alerting icon to view the created alert rules with different alert statuses, to edit the alert rule, or to pause the existing rule from the Alert Rules tab.
   To add a notification channel for the created alert rule to receive an alert notification from Grafana, see Adding notification channels for alerts.

8.6. ADDING NOTIFICATION CHANNELS FOR ALERTS

By adding notification channels, you can receive an alert notification from Grafana whenever the alert rule conditions are met and the system needs further monitoring.

You can receive these alerts after selecting any one type from the supported list of notifiers, which includes DingDing, Discord, Email, Google Hangouts Chat, HipChat, Kafka REST Proxy, LINE, Microsoft Teams, OpsGenie, PagerDuty, Prometheus Alertmanager, Pushover, Sensu, Slack, Telegram, Threema Gateway, VictorOps, and webhook.

Prerequisites

1. The grafana-server is accessible. For more information, see Accessing the Grafana web UI.

2. An alert rule is created. For more information, see Creating panels and alert in PCP Redis data source.

3. Configure SMTP and add a valid sender’s email address in the grafana/grafana.ini file:

   ```
   # vi /etc/grafana/grafana.ini
   [smtp]
   enabled = true
   from_address = abc@gmail.com
   
   Replace abc@gmail.com by a valid email address.
   ```
1. From the menu, hover over the Alerting icon → click Notification channels → Add channel.

2. In the Add notification channel details pane, perform the following:
   a. Enter your name in the Name text box
   b. Select the communication Type, for example, Email and enter the email address. You can add multiple email addresses using the ; separator.
   c. Optional: Configure Optional Email settings and Notification settings.

3. Click Save.

4. Select a notification channel in the alert rule:
   a. From the menu, hover over the Alerting icon and then click Alert rules.
   b. From the Alert Rules tab, click the created alert rule.
   c. On the Notifications tab, select your notification channel name from the Send to option, and then add an alert message.
   d. Click Apply.

Additional resources

- Upstream Grafana documentation for alert notifications

8.7. SETTING UP AUTHENTICATION BETWEEN PCP COMPONENTS

You can setup authentication using the scram-sha-256 authentication mechanism, which is supported by PCP through the Simple Authentication Security Layer (SASL) framework.

Procedure

1. Install the sasl framework for the scram-sha-256 authentication mechanism:
   ```bash
   # dnf install cyrus-sasl-scram cyrus-sasl-lib
   ```

2. Specify the supported authentication mechanism and the user database path in the pmcd.conf file:
   ```bash
   # vi /etc/sasl2/pmcd.conf
   mech_list: scram-sha-256
   sasldb_path: /etc/pcp/passwd.db
   ```

3. Create a new user:
   ```bash
   # useradd -r metrics
   ```
Replace *metrics* by your user name.

4. Add the created user in the user database:

   ```
   # saslpasswd2 -a pmcd metrics
   
   Password:
   Again (for verification):
   
   To add the created user, you are required to enter the *metrics* account password.
   ```

5. Set the permissions of the user database:

   ```
   # chown root:pcp /etc/pcp/passwd.db
   # chmod 640 /etc/pcp/passwd.db
   ```

6. Restart the `pmcd` service:

   ```
   # systemctl restart pmcd
   ```

**Verification steps**

- Verify the sasl configuration:

   ```
   # pminfo -f -h "pcp://127.0.0.1?username=metrics" disk.dev.read
   Password:
   disk.dev.read
   inst [0 or "sda"] value 19540
   ```

**Additional resources**

- `saslauthd(8)`, `pminfo(1)`, and `sha256` man pages
- How can I setup authentication between PCP components, like PMDAs and pmcd in RHEL 8.2?

**8.8. INSTALLING PCP BPFTRACE**

Install the PCP `bpftrace` agent to introspect a system and to gather metrics from the kernel and user-space tracepoints.

The `bpftrace` agent uses bpftrace scripts to gather the metrics. The `bpftrace` scripts use the enhanced Berkeley Packet Filter (eBPF).

This procedure describes how to install a `pcp bpftrace`.

**Prerequisites**

1. PCP is configured. For more information, see Setting up PCP with pcp-zeroconf.

2. The `grafana-server` is configured. For more information, see Setting up a grafana-server.

3. The `scram-sha-256` authentication mechanism is configured. For more information, see Setting up authentication between PCP components.
Procedure

1. Install the `pcp-pmda-bpftrace` package:

   ```
   # dnf install pcp-pmda-bpftrace
   ```

2. Edit the `bpftrace.conf` file and add the user that you have created in the `setting-up-authentication-between-pcp-components`:

   ```
   # vi /var/lib/pcp/pmdas/bpftrace/bpftrace.conf
   
   [dynamic_scripts]
   enabled = true
   auth_enabled = true
   allowed_users = root, metrics
   
   Replace metrics by your user name.
   ```

3. Install `bpftrace` PMDA:

   ```
   # cd /var/lib/pcp/pmdas/bpftrace/
   # ./Install
   
   Updating the Performance Metrics Name Space (PMNS) ...
   Terminate PMDA if already installed ...
   Updating the PMCD control file, and notifying PMCD ...
   Check bpftrace metrics have appeared ... 7 metrics and 6 values
   ```

   The `pmda-bpftrace` is now installed, and can only be used after authenticating your user. For more information, see Viewing the PCP bpftrace System Analysis dashboard.

Additional resources

- `pmdabpftrace(1)` and `bpftrace` man pages

8.9. VIEWING THE PCP BPFTRACE SYSTEM ANALYSIS DASHBOARD

Using the PCP bpftrace data source, you can access the live data from sources which are not available as normal data from the `pmlogger` or archives

In the PCP bpftrace data source, you can view the dashboard with an overview of useful metrics.

Prerequisites

1. The PCP bpftrace is installed. For more information, see Installing PCP bpftrace.

2. The `grafana-server` is accessible. For more information, see Accessing the Grafana web UI.

Procedure

1. Log into the Grafana web UI.

2. In the Grafana Home page, click Add your first data source
3. In the Add data source pane, type bpftrace in the Filter by name or type text box and then click PCP bpftrace.

4. In the Data Sources / PCP bpftrace pane, perform the following:
   b. Toggle the Basic Auth option and add the created user credentials in the User and Password field.
   c. Click Save & Test.

   Figure 8.5. Adding PCP bpftrace in the data source

   d. Click Dashboards tab → Import → PCP bpftrace: System Analysis to see a dashboard with an overview of any useful metrics.

   Figure 8.6. PCP bpftrace: System Analysis

8.10. INSTALLING PCP VECTOR

This procedure describes how to install a pcp vector.
Prerequisites

1. PCP is configured. For more information, see Setting up PCP with pcp-zeroconf.
2. The grafana-server is configured. For more information, see Setting up a grafana-server.

Procedure

1. Install the pcp-pmda-bcc package:
   ```bash
   # dnf install pcp-pmda-bcc
   ```
2. Install the bcc PMDA:
   ```bash
   # cd /var/lib/pcp/pmdas/bcc
   # ./Install
   [Wed Apr  1 00:27:48] pmdabcc(22341) Info: Initializing, currently in 'notready' state.
   [Wed Apr  1 00:27:48] pmdabcc(22341) Info: Enabled modules:
   [Wed Apr  1 00:27:48] pmdabcc(22341) Info: ['biolatency', 'sysfork', ...
   Updating the Performance Metrics Name Space (PMNS) ...
   Terminate PMDA if already installed ...
   Updating the PMCD control file, and notifying PMCD ...
   Check bcc metrics have appeared ... 1 warnings, 1 metrics and 0 values
   ```

Additional resources

- pmdabcc(1) man page

8.11. VIEWING THE PCP VECTOR CHECKLIST

The PCP Vector data source displays live metrics and uses the pcp metrics. It analyzes data for individual hosts.

After adding the PCP Vector data source, you can view the dashboard with an overview of useful metrics and view the related troubleshooting or reference links in the checklist.

Prerequisites

1. The PCP Vector is installed. For more information, see Installing PCP Vector.
2. The grafana-server is accessible. For more information, see Accessing the Grafana web UI.

Procedure

1. Log into the Grafana web UI.
2. In the Grafana Home page, click Add your first data source
3. In the Add data source pane, type vector in the Filter by name or type text box and then click PCP Vector.
4. In the Data Sources / PCP Vector pane, perform the following:
   a. Add http://localhost:44322 in the URL field and then click Save & Test.
b. Click **Dashboards tab → Import → PCP Vector: Host Overview** to see a dashboard with an overview of any useful metrics.

**Figure 8.7. PCP Vector: Host Overview**

![PCP Vector: Host Overview dashboard](image)

5. From the menu, hover over the **Performance Co-Pilot** plugin and then click **PCP Vector Checklist**. In the PCP checklist, click **help** or **warning** icon to view the related troubleshooting or reference links.

**Figure 8.8. Performance Co-Pilot / PCP Vector Checklist**

![PCP Vector Checklist](image)

### 8.12. TROUBLESHOOTING GRAFANA ISSUES

This section describes how to troubleshoot Grafana issues, such as, Grafana does not display any data, the dashboard is black, or similar issues.

**Procedure**

- Verify that the **pmlogger** service is up and running by executing the following command:
$ systemctl status pmlogger

- Verify if files were created or modified to the disk by executing the following command:

```shell
$ ls /var/log/pcp/pmlogger/$(hostname)/ -rlt
```

```
total 4024
-rw-r--r--. 1 pcp pcp   45996 Oct 13  2019 20191013.20.07.meta.xz
-rw-r--r--. 1 pcp pcp     412 Oct 13  2019 20191013.20.07.index
-rw-r--r--. 1 pcp pcp   32188 Oct 13  2019 20191013.20.07.0.xz
-rw-r--r--. 1 pcp pcp   44756 Oct 13  2019 20191013.20.30-00.meta.xz
[..]
```

- Verify that the pmproxy service is running by executing the following command:

```shell
$ systemctl status pmproxy
```

- Verify that pmproxy is running, time series support is enabled, and a connection to Redis is established by viewing the /var/log/pcp/pmproxy/pmproxy.log file and ensure that it contains the following text:

```
pmproxy(1716) Info: Redis slots, command keys, schema version setup
```

Here, 1716 is the PID of pmproxy, which will be different for every invocation of pmproxy.

- Verify if the Redis database contains any keys by executing the following command:

```shell
$ redis-cli dbsize
(integer) 34837
```

- Verify if any PCP metrics are in the Redis database and pmproxy is able to access them by executing the following commands:

```shell
$ pmseries disk.dev.read
2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df

$ pmseries "disk.dev.read[count:10]"
2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
70e83e88d4e1857a3a31605c6d1333755f2dd17c
 [Mon Jul 26 12:21:00.087401000 2021] 117758
70e83e88d4e1857a3a31605c6d1333755f2dd17c
70e83e88d4e1857a3a31605c6d1333755f2dd17c
[..]

$ redis-cli --scan --pattern "**(pmseries 'disk.dev.read')"*
```

```
pcp:metric.name:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:values:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:desc:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:labelvalue:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:instances:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
pcp:labelflags:series:2eb3e58d8f1e231361fb15cf1aa26fe534b4d9df
```
Verify if there are any errors in the Grafana logs by executing the following command:

```
$ journalctl -e -u grafana-server
-- Logs begin at Mon 2021-07-26 11:55:10 IST, end at Mon 2021-07-26 12:30:15 IST. --
Jul 26 11:55:17 localhost.localdomain systemd[1]: Starting Grafana instance...
  lvl=info msg="Starting Grafana" logger=server version=7.3.6
  lvl=info msg="Config loaded from" logger=settings file=/usr/s
  lvl=info msg="Loading data from" logger=data file=/etc/g
[...]
```
CHAPTER 9. SETTING THE DISK SCHEDULER

The disk scheduler is responsible for ordering the I/O requests submitted to a storage device.

You can configure the scheduler in several different ways:

- Set the scheduler using TuneD, as described in Setting the disk scheduler using TuneD
- Set the scheduler using udev, as described in Setting the disk scheduler using udev rules
- Temporarily change the scheduler on a running system, as described in Temporarily setting a scheduler for a specific disk

NOTE

In Red Hat Enterprise Linux 9, block devices support only multi-queue scheduling. This enables the block layer performance to scale well with fast solid-state drives (SSDs) and multi-core systems.

The traditional, single-queue schedulers, which were available in Red Hat Enterprise Linux 7 and earlier versions, have been removed.

9.1. AVAILABLE DISK SCHEDULERS

The following multi-queue disk schedulers are supported in Red Hat Enterprise Linux 9:

**none**

Implements a first-in first-out (FIFO) scheduling algorithm. It merges requests at the generic block layer through a simple last-hit cache.

**mq-deadline**

Attempts to provide a guaranteed latency for requests from the point at which requests reach the scheduler.

The **mq-deadline** scheduler sorts queued I/O requests into a read or write batch and then schedules them for execution in increasing logical block addressing (LBA) order. By default, read batches take precedence over write batches, because applications are more likely to block on read I/O operations. After **mq-deadline** processes a batch, it checks how long write operations have been starved of processor time and schedules the next read or write batch as appropriate.

This scheduler is suitable for most use cases, but particularly those in which the write operations are mostly asynchronous.

**bfq**

Targets desktop systems and interactive tasks.

The **bfq** scheduler ensures that a single application is never using all of the bandwidth. In effect, the storage device is always as responsive as if it was idle. In its default configuration, **bfq** focuses on delivering the lowest latency rather than achieving the maximum throughput.

**bfq** is based on **cfq** code. It does not grant the disk to each process for a fixed time slice but assigns a **budget** measured in number of sectors to the process.

This scheduler is suitable while copying large files and the system does not become unresponsive in this case.
kyber

The scheduler tunes itself to achieve a latency goal by calculating the latencies of every I/O request submitted to the block I/O layer. You can configure the target latencies for read, in the case of cache-misses, and synchronous write requests. This scheduler is suitable for fast devices, for example NVMe, SSD, or other low latency devices.

9.2. DIFFERENT DISK SCHEDULERS FOR DIFFERENT USE CASES

Depending on the task that your system performs, the following disk schedulers are recommended as a baseline prior to any analysis and tuning tasks:

Table 9.1. Disk schedulers for different use cases

<table>
<thead>
<tr>
<th>Use case</th>
<th>Disk scheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional HDD with a SCSI interface</td>
<td>Use <strong>mq-deadline</strong> or <strong>bfq</strong>.</td>
</tr>
<tr>
<td>High-performance SSD or a CPU-bound system with fast storage</td>
<td>Use <strong>none</strong>, especially when running enterprise applications. Alternatively, use <strong>kyber</strong>.</td>
</tr>
<tr>
<td>Desktop or interactive tasks</td>
<td>Use <strong>bfq</strong>.</td>
</tr>
<tr>
<td>Virtual guest</td>
<td>Use <strong>mq-deadline</strong>. With a host bus adapter (HBA) driver that is multi-queue capable, use <strong>none</strong>.</td>
</tr>
</tbody>
</table>

9.3. THE DEFAULT DISK SCHEDULER

Block devices use the default disk scheduler unless you specify another scheduler.

**NOTE**

For **non-volatile Memory Express (NVMe)** block devices specifically, the default scheduler is **none** and Red Hat recommends not changing this.

The kernel selects a default disk scheduler based on the type of device. The automatically selected scheduler is typically the optimal setting. If you require a different scheduler, Red Hat recommends to use **udev** rules or the **TuneD** application to configure it. Match the selected devices and switch the scheduler only for those devices.

9.4. DETERMINING THE ACTIVE DISK SCHEDULER

This procedure determines which disk scheduler is currently active on a given block device.

**Procedure**

- Read the content of the `/sys/block/device/queue/scheduler` file:

```bash
# cat /sys/block/device/queue/scheduler
[mq-deadline] kyber bfq none
```
In the file name, replace device with the block device name, for example sdc.

The active scheduler is listed in square brackets ([]).

9.5. SETTING THE DISK SCHEDULER USING TUNED

This procedure creates and enables a TuneD profile that sets a given disk scheduler for selected block devices. The setting persists across system reboots.

In the following commands and configuration, replace:

- device with the name of the block device, for example sdf
- selected-scheduler with the disk scheduler that you want to set for the device, for example bfq

Prerequisites

- The tuned service is installed and enabled. For details, see Installing and enabling TuneD.

Procedure

1. Optional: Select an existing TuneD profile on which your profile will be based. For a list of available profiles, see TuneD profiles distributed with RHEL.
   To see which profile is currently active, use:

   ```
   $ tuned-adm active
   ```

2. Create a new directory to hold your TuneD profile:

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

3. Find the system unique identifier of the selected block device:

   ```
   $ udevadm info --query=property --name=/dev/device | grep -E '(WWN|SERIAL)'
   ID_WWN=0x5002538d00000000_
   ID_SERIAL=Generic-_SD_MMC_20120501030900000-0:0
   ID_SERIAL_SHORT=20120501030900000
   ```

   **NOTE**
   
   The command in the this example will return all values identified as a World Wide Name (WWN) or serial number associated with the specified block device. Although it is preferred to use a WWN, the WWN is not always available for a given device and any values returned by the example command are acceptable to use as the device system unique ID.

4. Create the /etc/tuned/my-profile/tuned.conf configuration file. In the file, set the following options:
   a. Optional: Include an existing profile:
b. Set the selected disk scheduler for the device that matches the WWN identifier:

```plaintext
[main]
include=existing-profile

[disk]
devices_udev_regex=IDNAME=device system unique id
elevator=selected-scheduler
```

Here:

- Replace `IDNAME` with the name of the identifier being used (for example, `ID_WWN`).
- Replace `device system unique id` with the value of the chosen identifier (for example, `0x5002538d00000000`).

To match multiple devices in the `devices_udev_regex` option, enclose the identifiers in parentheses and separate them with vertical bars:

```plaintext
devices_udev_regex=(ID_WWN=0x5002538d00000000) | (ID_WWN=0x1234567800000000)
```

5. Enable your profile:

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

**Verification steps**

1. Verify that the TuneD profile is active and applied:

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

```plaintext
$ tuned-adm verify
Verification succeeded, current system settings match the preset profile.
See tuned log file (`/var/log/tuned/tuned.log`) for details.
```

2. Read the contents of the `/sys/block/device/queue/scheduler` file:

```plaintext
# cat /sys/block/device/queue/scheduler
[mq-deadline] kyber bfq none
```

In the file name, replace `device` with the block device name, for example `sdc`.

The active scheduler is listed in square brackets (`[]`).

**Additional resources**

- Customizing TuneD profiles.
9.6. SETTING THE DISK SCHEDULER USING UDEV RULES

This procedure sets a given disk scheduler for specific block devices using udev rules. The setting persists across system reboots.

In the following commands and configuration, replace:

- `device` with the name of the block device, for example `sdf`
- `selected-scheduler` with the disk scheduler that you want to set for the device, for example `bfq`

**Procedure**

1. Find the system unique identifier of the block device:

   ```bash
   $ udevadm info --name=/dev/device | grep -E '(WWN|SERIAL)'
   E: ID_WWN=0x5002538d00000000
   E: ID_SERIAL=Generic-_SD_MMC_20120501030900000-0:0
   E: ID_SERIAL_SHORT=20120501030900000
   ```

   **NOTE**
   
   The command in this example will return all values identified as a World Wide Name (WWN) or serial number associated with the specified block device. Although it is preferred to use a WWN, the WWN is not always available for a given device and any values returned by the example command are acceptable to use as the device system unique ID.

2. Configure the udev rule. Create the `/etc/udev/rules.d/99-scheduler.rules` file with the following content:

   ```bash
   ACTION=="add|change", SUBSYSTEM=="block", ENV{IDNAME}"device system unique id",
   ATTR{queue/scheduler}="selected-scheduler"
   ```

   Here:

   - Replace `IDNAME` with the name of the identifier being used (for example, `ID_WWN`).
   - Replace `device system unique id` with the value of the chosen identifier (for example, `0x5002538d00000000`).

3. Reload udev rules:

   ```bash
   # udevadm control --reload-rules
   ```

4. Apply the scheduler configuration:

   ```bash
   # udevadm trigger --type=devices --action=change
   ```

**Verification steps**

- Verify the active scheduler:
9.7. TEMPORARILY SETTING A SCHEDULER FOR A SPECIFIC DISK

This procedure sets a given disk scheduler for specific block devices. The setting does not persist across system reboots.

**Procedure**

- Write the name of the selected scheduler to the `/sys/block/device/queue/scheduler` file:

  ```bash
  # echo selected-scheduler > /sys/block/device/queue/scheduler
  ```

  In the file name, replace `device` with the block device name, for example `sdc`.

**Verification steps**

- Verify that the scheduler is active on the device:

  ```bash
  # cat /sys/block/device/queue/scheduler
  ```
CHAPTER 10. 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

10.1. 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 support 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

10.2. 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.

**10.3. 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`. 
As a system administrator, you can use the PowerTOP tool to analyze and manage power consumption.

**11.1. 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 9 uses version 2.x of PowerTOP.

**11.2. USING POWERTOP**

**Prerequisites**

- To be able to use PowerTOP, make sure that the powertop package has been installed on your system:
  
  ```bash
  # dnf install powertop
  ```

**11.2.1. Starting PowerTOP**

**Procedure**

- To run PowerTOP, use the following command:
  
  ```bash
  # powertop
  ```

**IMPORTANT**

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

**11.2.2. Calibrating PowerTOP**

**Procedure**

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

11.2.3. 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
  ```

11.2.4. Additional resources

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

11.3. 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
- WakeUp

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

11.3.1. 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
Wakesups per second. The number of wakesups per second indicates how efficiently the services or the devices and drivers of the kernel are performing. Less wakesups 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 wakesups per second, GPU operations per second, and virtual file system operations per second

11.3.2. 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.

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

11.3.4. 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.

11.3.5. The WakeUp tab
The WakeUp tab displays the device wakeup settings available for users to change as and when required.

Use the up and down keys to move through the available settings, and the enter key to enable or disable a setting.
11.4. WHY POWERTOP DOES NOT DISPLAY FREQUENCY STATS VALUES IN SOME INSTANCES

While using the Intel P-State driver, PowerTOP only displays values in the Frequency Stats tab if the driver is in passive mode. But, even in this case, the values may be incomplete.

In total, there are three possible modes of the Intel P-State driver:

- Active mode with Hardware P-States (HWP)
- Active mode without HWP
- Passive mode

Switching to the ACPI CPUfreq driver results in complete information being displayed by PowerTOP. However, it is recommended to keep your system on the default settings.

To see what driver is loaded and in what mode, run:

```
# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_driver
```

- `intel_pstate` is returned if the Intel P-State driver is loaded and in active mode.
- `intel_cpufreq` is returned if the Intel P-State driver is loaded and in passive mode.
- `acpi-cpufreq` is returned if the ACPI CPUfreq driver is loaded.

While using the Intel P-State driver, add the following argument to the kernel boot command line to force the driver to run in passive mode:

```
intel_pstate=passive
```
To disable the Intel P-State driver and use, instead, the ACPI CPUfreq driver, add the following argument to the kernel boot command line:

```
intel_pstate=disable
```

### 11.5. 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.

### 11.6. OPTIMIZING POWER CONSUMPTION

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

#### 11.6.1. 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
  ```

#### 11.6.2. 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.

#### 11.6.3. Optimizing power consumption using the powertop2tuned utility
Prerequisites

- The `powertop2tuned` utility is installed on the system:
  
  ```bash
  # dnf install tuned-utils
  ```

Procedure

1. Create a custom profile:

  ```bash
  # powertop2tuned new_profile_name
  ```

2. Activate the new profile:

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

Additional information

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

  ```bash
  $ powertop2tuned --help
  ```

11.6.4. 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.
CHAPTER 12. GETTING STARTED WITH PERF

As a system administrator, you can use the `perf` tool to collect and analyze performance data of your system.

12.1. INTRODUCTION TO PERF

The `perf` user-space tool interfaces with the kernel-based subsystem _Performance Counters for Linux_ (PCL). `perf` is a powerful tool that uses the Performance Monitoring Unit (PMU) to measure, record, and monitor a variety of hardware and software events. `perf` also supports tracepoints, kprobes, and uprobes.

12.2. INSTALLING PERF

This procedure installs the `perf` user-space tool.

Procedure

- Install the `perf` tool:
  
  ```bash
  # dnf install perf
  ```

12.3. COMMON PERF COMMANDS

This section provides an overview of commonly used `perf` commands.

Commonly used `perf` commands

**perf stat**

This command provides overall statistics for common performance events, including instructions executed and clock cycles consumed. Options allow for selection of events other than the default measurement events.

**perf record**

This command records performance data into a file, `perf.data`, which can be later analyzed using the `perf report` command.

**perf report**

This command reads and displays the performance data from the `perf.data` file created by `perf record`.

**perf list**

This command lists the events available on a particular machine. These events will vary based on performance monitoring hardware and software configuration of the system.

**perf top**

This command performs a similar function to the `top` utility. It generates and displays a performance counter profile in realtime.

**perf trace**

This command performs a similar function to the `strace` tool. It monitors the system calls used by a specified thread or process and all signals received by that application.

**perf help**
This command displays a complete list of `perf` commands.

Additional resources

- Add the `--help` option to a subcommand to open the man page.
CHAPTER 13. CONFIGURING AN OPERATING SYSTEM TO OPTIMIZE CPU UTILIZATION

This section describes how to configure the operating system to optimize CPU utilization across their workloads.

13.1. TOOLS FOR MONITORING AND DIAGNOSING PROCESSOR ISSUES

The following are the tools available in Red Hat Enterprise Linux 9 to monitor and diagnose processor-related performance issues:

- **turbostat** tool prints counter results at specified intervals to help administrators identify unexpected behavior in servers, such as excessive power usage, failure to enter deep sleep states, or system management interrupts (SMIs) being created unnecessarily.

- **numactl** utility provides a number of options to manage processor and memory affinity. The **numactl** package includes the **libnuma** library which offers a simple programming interface to the NUMA policy supported by the kernel, and can be used for more fine-grained tuning than the **numactl** application.

- **numastat** tool displays per-NUMA node memory statistics for the operating system and its processes, and shows administrators whether the process memory is spread throughout a system or is centralized on specific nodes. This tool is provided by the **numactl** package.

- **numad** is an automatic NUMA affinity management daemon. It monitors NUMA topology and resource usage within a system in order to dynamically improve NUMA resource allocation and management.

- **/proc/interrupts** file displays the interrupt request (IRQ) number, the number of similar interrupt requests handled by each processor in the system, the type of interrupt sent, and a comma-separated list of devices that respond to the listed interrupt request.

- **pqos** utility is available in the **intel-cmt-cat** package. It monitors CPU cache and memory bandwidth on recent Intel processors. It monitors:
  - The instructions per cycle (IPC).
  - The count of last level cache MISSES.
  - The size in kilobytes that the program executing in a given CPU occupies in the LLC.
  - The bandwidth to local memory (MBL).
  - The bandwidth to remote memory (MBR).

- **x86_energy_perf_policy** tool allows administrators to define the relative importance of performance and energy efficiency. This information can then be used to influence processors that support this feature when they select options that trade off between performance and energy efficiency.

- **taskset** tool is provided by the **util-linux** package. It allows administrators to retrieve and set the processor affinity of a running process, or launch a process with a specified processor affinity.

Additional resources

Red Hat Enterprise Linux 9.0 Monitoring and managing system status and performance
Additional resources

- `turbostat(8)`, `numactl(8)`, `numastat(8)`, `numa(7)`, `numad(8)`, `pqos(8)`, `x86_energy_perf_policy(8)`, and `taskset(1)` man pages

### 13.2. TYPES OF SYSTEM TOPOLOGY

In modern computing, the idea of a CPU is a misleading one, as most modern systems have multiple processors. The topology of the system is the way these processors are connected to each other and to other system resources. This can affect system and application performance, and the tuning considerations for a system.

The following are the two primary types of topology used in modern computing:

#### Symmetric Multi-Processor (SMP) topology

SMP topology allows all processors to access memory in the same amount of time. However, because shared and equal memory access inherently forces serialized memory accesses from all the CPUs, SMP system scaling constraints are now generally viewed as unacceptable. For this reason, practically all modern server systems are NUMA machines.

#### Non-Uniform Memory Access (NUMA) topology

NUMA topology was developed more recently than SMP topology. In a NUMA system, multiple processors are physically grouped on a socket. Each socket has a dedicated area of memory and processors that have local access to that memory, these are referred to collectively as a node. Processors on the same node have high speed access to that node’s memory bank, and slower access to memory banks not on their node.

Therefore, there is a performance penalty when accessing non-local memory. Thus, performance sensitive applications on a system with NUMA topology should access memory that is on the same node as the processor executing the application, and should avoid accessing remote memory wherever possible.

Multi-threaded applications that are sensitive to performance may benefit from being configured to execute on a specific NUMA node rather than a specific processor. Whether this is suitable depends on your system and the requirements of your application. If multiple application threads access the same cached data, then configuring those threads to execute on the same processor may be suitable. However, if multiple threads that access and cache different data execute on the same processor, each thread may evict cached data accessed by a previous thread. This means that each thread ‘misses’ the cache and wastes execution time fetching data from memory and replacing it in the cache. Use the `perf` tool to check for an excessive number of cache misses.

### 13.2.1. Displaying system topologies

There are a number of commands that help understand the topology of a system. This procedure describes how to determine the system topology.

**Procedure**

- To display an overview of your system topology:

```bash
$ numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 4 8 12 16 20 24 28 32 36
node 0 size: 65415 MB
node 0 free: 43971 MB
[...]```
To gather the information about the CPU architecture, such as the number of CPUs, threads, cores, sockets, and NUMA nodes:

```
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                40
On-line CPU(s) list:   0-39
Thread(s) per core:    1
Core(s) per socket:    10
Socket(s):             4
NUMA node(s):          4
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 47
Model name:            Intel(R) Xeon(R) CPU E7- 4870 @ 2.40GHz
Stepping:              2
CPU MHz:               2394.204
BogoMIPS:              4787.85
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0,4,8,12,16,20,24,28,32,36
NUMA node1 CPU(s):     2,6,10,14,18,22,26,30,34,38
NUMA node2 CPU(s):     1,5,9,13,17,21,25,29,33,37
NUMA node3 CPU(s):     3,7,11,15,19,23,27,31,35,39
```

To view a graphical representation of your system:

```
# dnf install hwloc-gui
# lstopo
```
Figure 13.1. The lstopo output

- To view the detailed textual output:

```
# dnf install hwloc
# lstopo-no-graphics
Machine (15GB)
Package L#0 + L3 L#0 (8192KB)
  L2 L#0 (256KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0
    PU L#0 (P#0)
    PU L#1 (P#4)
  HostBridge L#0
PCI 8086:5917
  GPU L#0 "renderD128"
  GPU L#1 "controlD64"
  GPU L#2 "card0"
PCIBridge
  PCI 8086:24fd
    Net L#3 "wlp61s0"
PCIBridge
  PCI 8086:f1a6
PCI 8086:15d7
  Net L#4 "enp0s31f6"
```

Additional resources

- `numactl(8)`, `lscpu(1)`, and `lstopo(1)` man pages

13.3. CONFIGURING KERNEL TICK TIME
By default, Red Hat Enterprise Linux 9 uses a tickless kernel, which does not interrupt idle CPUs in order to reduce power usage and allow new processors to take advantage of deep sleep states.

Red Hat Enterprise Linux 9 also offers a dynamic tickless option, which is useful for latency-sensitive workloads, such as high performance computing or real-time computing. By default, the dynamic tickless option is disabled. Red Hat recommends using the `cpu-partitioning` TuneD profile to enable the dynamic tickless option for cores specified as `isolated_cores`.

This procedure describes how to manually persistently enable dynamic tickless behavior.

**Procedure**

1. To enable dynamic tickless behavior in certain cores, specify those cores on the kernel command line with the `nohz_full` parameter. On a 16 core system, append this parameter on the `GRUB_CMDLINE_LINUX` option in the `/etc/default/grub` file:

   ```
   nohz_full=1-15
   ```

   This enables dynamic tickless behavior on cores 1 through 15, moving all timekeeping to the only unspecified core (core 0).

2. To persistently enable the dynamic tickless behavior, regenerate the GRUB2 configuration using the edited default file. On systems with BIOS firmware, execute the following command:

   ```
   # grub2-mkconfig -o /etc/grub2.cfg
   ```

   On systems with UEFI firmware, execute the following command:

   ```
   # grub2-mkconfig -o /etc/grub2-efi.cfg
   ```

3. When the system boots, manually move the `rcu` threads to the non-latency-sensitive core, in this case core 0:

   ```
   # for i in `pgrep rcu[^c]` ; do taskset -pc 0 $i ; done
   ```

4. Optional: Use the `isolcpus` parameter on the kernel command line to isolate certain cores from user-space tasks.

5. Optional: Set the CPU affinity for the kernel’s `write-back bdi-flush` threads to the housekeeping core:

   ```
   echo 1 > /sys/bus/workqueue/devices/writeback/cpumask
   ```

**Verification steps**

- Once the system is rebooted, verify if `dynticks` are enabled:

  ```
  # journalctl -xe | grep dynticks
  ```

- Verify that the dynamic tickless configuration is working correctly:

  ```
  # perf stat -C 1 -e irq_vectors:local_timer_entry taskset -c 1 sleep 3
  ```
This command measures ticks on CPU 1 while telling CPU 1 to sleep for 3 seconds.

- The default kernel timer configuration shows around 3100 ticks on a regular CPU:

```
# perf stat -C 0 -e irq_vectors:local_timer_entry taskset -c 0 sleep 3
```

Performance counter stats for 'CPU(s) 0':

```
3,107  irq_vectors:local_timer_entry
3.001342790 seconds time elapsed
```

- With the dynamic tickless kernel configured, you should see around 4 ticks instead:

```
# perf stat -C 1 -e irq_vectors:local_timer_entry taskset -c 1 sleep 3
```

Performance counter stats for 'CPU(s) 1':

```
4  irq_vectors:local_timer_entry
3.001544078 seconds time elapsed
```

Additional resources

- `perf(1)` and `cpuset(7)` man pages
- All about nohz_full kernel parameter Red Hat Knowledgebase article
- How to verify the list of "isolated" and "nohz_full" CPU information from sysfs? Red Hat Knowledgebase article

### 13.4. OVERVIEW OF AN INTERRUPT REQUEST

An interrupt request or IRQ is a signal for immediate attention sent from a piece of hardware to a processor. Each device in a system is assigned one or more IRQ numbers which allow it to send unique interrupts. When interrupts are enabled, a processor that receives an interrupt request immediately pauses execution of the current application thread in order to address the interrupt request.

Because interrupt halts normal operation, high interrupt rates can severely degrade system performance. It is possible to reduce the amount of time taken by interrupts by configuring interrupt affinity or by sending a number of lower priority interrupts in a batch (coalescing a number of interrupts).

Interrupt requests have an associated affinity property, `smp_affinity`, which defines the processors that handle the interrupt request. To improve application performance, assign interrupt affinity and process affinity to the same processor, or processors on the same core. This allows the specified interrupt and application threads to share cache lines.

On systems that support interrupt steering, modifying the `smp_affinity` property of an interrupt request sets up the hardware so that the decision to service an interrupt with a particular processor is made at the hardware level with no intervention from the kernel.

### 13.4.1. Balancing interrupts manually
If your BIOS exports its NUMA topology, the `irqbalance` service can automatically serve interrupt requests on the node that is local to the hardware requesting service.

**Procedure**

1. Check which devices correspond to the interrupt requests that you want to configure.

2. Find the hardware specification for your platform. Check if the chipset on your system supports distributing interrupts.
   - a. If it does, you can configure interrupt delivery as described in the following steps. Additionally, check which algorithm your chipset uses to balance interrupts. Some BIOSes have options to configure interrupt delivery.
   - b. If it does not, your chipset always routes all interrupts to a single, static CPU. You cannot configure which CPU is used.

3. Check which Advanced Programmable Interrupt Controller (APIC) mode is in use on your system:
   
   ```
   $ journalctl --dmesg | grep APIC
   ```

   Here,

   - If your system uses a mode other than `flat`, you can see a line similar to `Setting APIC routing to physical flat`.
   - If you can see no such message, your system uses `flat` mode.
     If your system uses `x2apic` mode, you can disable it by adding the `nox2apic` option to the kernel command line in the bootloader configuration.

     Only non-physical flat mode (`flat`) supports distributing interrupts to multiple CPUs. This mode is available only for systems that have up to 8 CPUs.

4. Calculate the `smp_affinity mask`. For more information on how to calculate the `smp_affinity mask`, see Setting the `smp_affinity mask`.

**Additional resources**

- `journalctl(1)` and `taskset(1)` man pages

**13.4.2. Setting the smp_affinity mask**

The `smp_affinity` value is stored as a hexadecimal bit mask representing all processors in the system. Each bit configures a different CPU. The least significant bit is CPU 0.

The default value of the mask is `f`, which means that an interrupt request can be handled on any processor in the system. Setting this value to `1` means that only processor 0 can handle the interrupt.

**Procedure**

1. In binary, use the value 1 for CPUs that handle the interrupts. For example, to set CPU 0 and CPU 7 to handle interrupts, use `0000000010000001` as the binary code:

   **Table 13.1. Binary Bits for CPUs**
| CPU | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Binary | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

2. Convert the binary code to hexadecimal:
   For example, to convert the binary code using Python:

   ```python
   >>> hex(int('0000000010000001', 2))
   '0x81'
   ```

   On systems with more than 32 processors, you must delimit the `smp_affinity` values for discrete 32 bit groups. For example, if you want only the first 32 processors of a 64 processor system to service an interrupt request, use `0xffffffff,00000000`.

3. The interrupt affinity value for a particular interrupt request is stored in the associated `/proc/irq/irq_number/smp_affinity` file. Set the `smp_affinity` mask in this file:

   ```bash
   # echo mask > /proc/irq/irq_number/smp_affinity
   ```

Additional resources

- `journalctl(1)`, `irqbalance(1)`, and `taskset(1)` man pages
CHAPTER 14. TUNING SCHEDULING POLICY

In Red Hat Enterprise Linux, the smallest unit of process execution is called a thread. The system scheduler determines which processor runs a thread, and for how long the thread runs. However, because the scheduler’s primary concern is to keep the system busy, it may not schedule threads optimally for application performance.

For example, say an application on a NUMA system is running on Node A when a processor on Node B becomes available. To keep the processor on Node B busy, the scheduler moves one of the application’s threads to Node B. However, the application thread still requires access to memory on Node A. But, this memory will take longer to access because the thread is now running on Node B and Node A memory is no longer local to the thread. Thus, it may take longer for the thread to finish running on Node B than it would have taken to wait for a processor on Node A to become available, and then to execute the thread on the original node with local memory access.

14.1. CATEGORIES OF SCHEDULING POLICIES

Performance sensitive applications often benefit from the designer or administrator determining where threads are run. The Linux scheduler implements a number of scheduling policies which determine where and for how long a thread runs.

The following are the two major categories of scheduling policies:

Normal policies

Normal threads are used for tasks of normal priority.

Realtime policies

Realtime policies are used for time-sensitive tasks that must complete without interruptions. Realtime threads are not subject to time slicing. This means the thread runs until they block, exit, voluntarily yield, or are preempted by a higher priority thread.

The lowest priority realtime thread is scheduled before any thread with a normal policy. For more information, see Static priority scheduling with SCHED_FIFO and Round robin priority scheduling with SCHED_RR.

Additional resources

- sched(7), sched_setaffinity(2), sched_getaffinity(2), sched_setscheduler(2), and sched_getscheduler(2) man pages

14.2. STATIC PRIORITY SCHEDULING WITH SCHED_FIFO

The SCHED_FIFO, also called static priority scheduling, is a realtime policy that defines a fixed priority for each thread. This policy allows administrators to improve event response time and reduce latency. It is recommended to not execute this policy for an extended period of time for time sensitive tasks.

When SCHED_FIFO is in use, the scheduler scans the list of all the SCHED_FIFO threads in order of priority and schedules the highest priority thread that is ready to run. The priority level of a SCHED_FIFO thread can be any integer from 1 to 99, where 99 is treated as the highest priority. Red Hat recommends starting with a lower number and increasing priority only when you identify latency issues.
WARNING

Because realtime threads are not subject to time slicing, Red Hat does not recommend setting a priority as 99. This keeps your process at the same priority level as migration and watchdog threads; if your thread goes into a computational loop and these threads are blocked, they will not be able to run. Systems with a single processor will eventually hang in this situation.

Administrators can limit `SCHED_FIFO` bandwidth to prevent realtime application programmers from initiating realtime tasks that monopolize the processor.

The following are some of the parameters used in this policy:

```
/proc/sys/kernel/sched_rt_period_us
```
This parameter defines the time period, in microseconds, that is considered to be one hundred percent of the processor bandwidth. The default value is 1000000 µs, or 1 second.

```
/proc/sys/kernel/sched_rt_runtime_us
```
This parameter defines the time period, in microseconds, that is devoted to running real-time threads. The default value is 950000 µs, or 0.95 seconds.

### 14.3. ROUND ROBIN PRIORITY SCHEDULING WITH SCHED_RR

The `SCHED_RR` is a round-robin variant of the `SCHED_FIFO`. This policy is useful when multiple threads need to run at the same priority level.

Like `SCHED_FIFO`, `SCHED_RR` is a realtime policy that defines a fixed priority for each thread. The scheduler scans the list of all `SCHED_RR` threads in order of priority and schedules the highest priority thread that is ready to run. However, unlike `SCHED_FIFO`, threads that have the same priority are scheduled in a round-robin style within a certain time slice.

You can set the value of this time slice in milliseconds with the `sched_rr_timeslice_ms` kernel parameter in the `/proc/sys/kernel/sched_rr_timeslice_ms` file. The lowest value is 1 millisecond.

### 14.4. NORMAL SCHEDULING WITH SCHED_OTHER

The `SCHED_OTHER` is the default scheduling policy in Red Hat Enterprise Linux 9. This policy uses the Completely Fair Scheduler (CFS) to allow fair processor access to all threads scheduled with this policy. This policy is most useful when there are a large number of threads or when data throughput is a priority, as it allows more efficient scheduling of threads over time.

When this policy is in use, the scheduler creates a dynamic priority list based partly on the niceness value of each process thread. Administrators can change the niceness value of a process, but cannot change the scheduler’s dynamic priority list directly.

### 14.5. SETTING SCHEDULER POLICIES

Check and adjust scheduler policies and priorities by using the `chrt` command line tool. It can start new processes with the desired properties, or change the properties of a running process. It can also be used for setting the policy at runtime.
Procedure

1. View the process ID (PID) of the active processes:

   ```
   # ps
   ```

   Use the `--pid` or `-p` option with the `ps` command to view the details of the particular PID.

2. Check the scheduling policy, PID, and priority of a particular process:

   ```
   # chrt -p
   pid 468's current scheduling policy: SCHED_FIFO
   pid 468's current scheduling priority: 85

   # chrt -p 476
   pid 476's current scheduling policy: SCHED_OTHER
   pid 476's current scheduling priority: 0
   ```

   Here, 468 and 476 are PID of a process.

3. Set the scheduling policy of a process:

   a. For example, to set the process with PID 1000 to SCHED_FIFO, with a priority of 50:

      ```
      # chrt -f -p 50 1000
      ```

   b. For example, to set the process with PID 1000 to SCHED_OTHER, with a priority of 0:

      ```
      # chrt -o -p 0 1000
      ```

   c. For example, to set the process with PID 1000 to SCHED_RR, with a priority of 10:

      ```
      # chrt -r -p 10 1000
      ```

   d. To start a new application with a particular policy and priority, specify the name of the application:

      ```
      # chrt -f 36 /bin/my-app
      ```

Additional resources

- `chrt(1)` man page
- Policy Options for the chrt command
- Changing the priority of services during the boot process

14.6. POLICY OPTIONS FOR THE CHRT COMMAND

Using the `chrt` command, you can view and set the scheduling policy of a process.

The following table describes the appropriate policy options, which can be used to set the scheduling policy of a process.
Table 14.1. Policy Options for the chrt Command

<table>
<thead>
<tr>
<th>Short option</th>
<th>Long option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-f</td>
<td>--fifo</td>
<td>Set schedule to SCHED_FIFO</td>
</tr>
<tr>
<td>-o</td>
<td>--other</td>
<td>Set schedule to SCHED_OTHER</td>
</tr>
<tr>
<td>-r</td>
<td>--rr</td>
<td>Set schedule to SCHED_RR</td>
</tr>
</tbody>
</table>

14.7. CHANGING THE PRIORITY OF SERVICES DURING THE BOOT PROCESS

Using the systemd service, it is possible to set up real-time priorities for services launched during the boot process. The unit configuration directives are used to change the priority of a service during the boot process.

The boot process priority change is done by using the following directives in the service section:

**CPUSchedulingPolicy=**
- Sets the CPU scheduling policy for executed processes. It is used to set other, fifo, and rr policies.

**CPUSchedulingPriority=**
- Sets the CPU scheduling priority for executed processes. The available priority range depends on the selected CPU scheduling policy. For real-time scheduling policies, an integer between 1 (lowest priority) and 99 (highest priority) can be used.

The following procedure describes how to change the priority of a service, during the boot process, using the mcelog service.

**Prerequisites**

1. Install the tuned package:

   ```shell
   # dnf install tuned
   ```

2. Enable and start the tuned service:

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

**Procedure**

1. View the scheduling priorities of running threads:

   ```shell
   # tuna --show_threads
   ```

<table>
<thead>
<tr>
<th>thread</th>
<th>ctxt_switches</th>
<th>pid SCHED_rtpri affinity voluntary nonvoluntary</th>
<th>cmd</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTHER</td>
<td>0xff 3181</td>
<td>0xff</td>
<td>systemd</td>
</tr>
<tr>
<td>OTHER</td>
<td>0xff 254</td>
<td>0xff</td>
<td>kthread</td>
</tr>
<tr>
<td>OTHER</td>
<td>0xff 2</td>
<td>0xff</td>
<td>rcu_gp</td>
</tr>
<tr>
<td>OTHER</td>
<td>0xff 2</td>
<td>0xff</td>
<td>rcu_par_gp</td>
</tr>
</tbody>
</table>

121
2. Create a supplementary `mcelog` service configuration directory file and insert the policy name and priority in this file:

   ```
   # cat <<-EOF > /etc/systemd/system/mcelog.system.d/priority.conf
   >
   > [SERVICE]
   > CPUSchedulingPolicy=_fifo_
   > CPUSchedulingPriority=_20_
   > EOF
   ```

3. Reload the `systemd` scripts configuration:

   ```
   # systemctl daemon-reload
   ```

4. Restart the `mcelog` service:

   ```
   # systemctl restart mcelog
   ```

**Verification steps**

- Display the `mcelog` priority set by `systemd` issue:

   ```
   # tuna -t mcelog -P
   thread      ctxt_switches
   pid SCHED_ rtpri affinity voluntary nonvoluntary             cmd
   826     FIFO    20  0,1,2,3        13            0          mcelog
   ```

**Additional resources**

- `systemd(1)` and `tuna(8)` man pages
- Description of the priority range

**14.8. PRIORITY MAP**

Priorities are defined in groups, with some groups dedicated to certain kernel functions. For real-time scheduling policies, an integer between 1 (lowest priority) and 99 (highest priority) can be used.

The following table describes the priority range, which can be used while setting the scheduling policy of a process.

Table 14.2. Description of the priority range
<table>
<thead>
<tr>
<th>Priority</th>
<th>Threads</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Low priority kernel threads</td>
<td>This priority is usually reserved for the tasks that need to be just above SCHED_OTHER.</td>
</tr>
<tr>
<td>2 - 49</td>
<td>Available for use</td>
<td>The range used for typical application priorities.</td>
</tr>
<tr>
<td>50</td>
<td>Default hard-IRQ value</td>
<td></td>
</tr>
<tr>
<td>51 - 98</td>
<td>High priority threads</td>
<td>Use this range for threads that execute periodically and must have quick response times. Do not use this range for CPU-bound threads as you will starve interrupts.</td>
</tr>
<tr>
<td>99</td>
<td>Watchdogs and migration</td>
<td>System threads that must run at the highest priority.</td>
</tr>
</tbody>
</table>

14.9. TUNED CPU-PARTITIONING PROFILE

For tuning Red Hat Enterprise Linux 9 for latency-sensitive workloads, Red Hat recommends to use the cpu-partitioning TuneD profile.

Prior to Red Hat Enterprise Linux 9, the low-latency Red Hat documentation described the numerous low-level steps needed to achieve low-latency tuning. In Red Hat Enterprise Linux 9, you can perform low-latency tuning more efficiently by using the cpu-partitioning TuneD profile. This profile is easily customizable according to the requirements for individual low-latency applications.

The following figure is an example to demonstrate how to use the cpu-partitioning profile. This example uses the CPU and node layout.

**Figure 14.1. Figure cpu-partitioning**

You can configure the cpu-partitioning profile in the `/etc/tuned/cpu-partitioning-variables.conf` file using the following configuration options:

**Isolated CPUs with load balancing**
In the cpu-partitioning figure, the blocks numbered from 4 to 23, are the default isolated CPUs. The kernel scheduler’s process load balancing is enabled on these CPUs. It is designed for low-latency processes with multiple threads that need the kernel scheduler load balancing.

You can configure the cpu-partitioning profile in the `/etc/tuned/cpu-partitioning-variables.conf` file using the `isolated_cores=cpu-list` option, which lists CPUs to isolate that will use the kernel scheduler load balancing.

The list of isolated CPUs is comma-separated or you can specify a range using a dash, such as `3-5`. This option is mandatory. Any CPU missing from this list is automatically considered a housekeeping CPU.

**Isolated CPUs without load balancing**

In the cpu-partitioning figure, the blocks numbered 2 and 3, are the isolated CPUs that do not provide any additional kernel scheduler process load balancing.

You can configure the cpu-partitioning profile in the `/etc/tuned/cpu-partitioning-variables.conf` file using the `no_balance_cores=cpu-list` option, which lists CPUs to isolate that will not use the kernel scheduler load balancing.

Specifying the `no_balance_cores` option is optional, however any CPUs in this list must be a subset of the CPUs listed in the `isolated_cores` list.

Application threads using these CPUs need to be pinned individually to each CPU.

**Housekeeping CPUs**

Any CPU not isolated in the `cpu-partitioning-variables.conf` file is automatically considered a housekeeping CPU. On the housekeeping CPUs, all services, daemons, user processes, movable kernel threads, interrupt handlers, and kernel timers are permitted to execute.

**Additional resources**

- `tuned-profiles-cpu-partitioning(7)` man page

**14.10. USING THE TUNED CPU-PARTITIONING PROFILE FOR LOW-LATENCY TUNING**

This procedure describes how to tune a system for low-latency using the TuneD’s `cpu-partitioning` profile. It uses the example of a low-latency application that can use `cpu-partitioning` and the CPU layout as mentioned in the `cpu-partitioning` figure.

The application in this case uses:

- One dedicated reader thread that reads data from the network will be pinned to CPU 2.
- A large number of threads that process this network data will be pinned to CPUs 4-23.
- A dedicated writer thread that writes the processed data to the network will be pinned to CPU 3.

**Prerequisites**

- You have installed the `cpu-partitioning` TuneD profile by using the `dnf install tuned-profiles-cpu-partitioning` command as root.
Procedure

1. Edit /etc/tuned/cpu-partitioning-variables.conf file and add the following information:

   # Isolated CPUs with the kernel's scheduler load balancing:
   isolated_cores=2-23
   # Isolated CPUs without the kernel's scheduler load balancing:
   no_balance_cores=2,3

2. Set the cpu-partitioning TuneD profile:

   # tuned-adm profile cpu-partitioning

3. Reboot
   After rebooting, the system is tuned for low-latency, according to the isolation in the cpu-partitioning figure. The application can use taskset to pin the reader and writer threads to CPUs 2 and 3, and the remaining application threads on CPUs 4-23.

Additional resources

- tuned-profiles-cpu-partitioning(7) man page

14.11. CUSTOMIZING THE CPU-PARTITIONING TUNED PROFILE

You can extend the TuneD profile to make additional tuning changes.

For example, the cpu-partitioning profile sets the CPUs to use cstate=1. In order to use the cpu-partitioning profile but to additionally change the CPU cstate from cstate1 to cstate0, the following procedure describes a new TuneD profile named my_profile, which inherits the cpu-partitioning profile and then sets C state-0.

Procedure

1. Create the /etc/tuned/my_profile directory:

   # mkdir /etc/tuned/my_profile

2. Create a tuned.conf file in this directory, and add the following content:

   # vi /etc/tuned/my_profile/tuned.conf
   [main]
   summary=Customized tuning on top of cpu-partitioning
   include=cpu-partitioning
   [cpu]
   force_latency=cstate.id:0|1

3. Use the new profile:

   # tuned-adm profile my_profile
NOTE

In the shared example, a reboot is not required. However, if the changes in the `my_profile` profile require a reboot to take effect, then reboot your machine.

Additional resources

- `tuned-profiles-cpu-partitioning(7)` man page
CHAPTER 15. CONFIGURING RESOURCE MANAGEMENT USING CGROUPS VERSION 2 WITH SYSTEMD

The core of systemd is service management and supervision. systemd ensures that the right services start at the right time and in the correct order during the boot process. When the services are running, they have to run smoothly to use the underlying hardware platform optimally. Therefore, systemd also provides capabilities to define resource management policies and to tune various options, which can improve the performance of the service.

15.1. PREREQUISITES

- Basic knowledge of the Linux cgroup subsystem.

15.2. INTRODUCTION TO RESOURCE DISTRIBUTION MODELS

For resource management, systemd uses the cgroups v2 interface.

To modify the distribution of system resources, you can apply one or more of the following resource distribution models:

Weights

The resource is distributed by adding up the weights of all sub-groups and giving each sub-group the fraction matching its ratio against the sum.

For example, if you have 10 cgroups, each with Weight of value 100, the sum is 1000 and each cgroup receives one tenth of the resource.

Weight is usually used to distribute stateless resources. The CPUWeight= option is an implementation of this resource distribution model.

Limits

A cgroup can consume up to the configured amount of the resource, but you can also overcommit resources. Therefore, the sum of sub-group limits can exceed the limit of the parent cgroup. The MemoryMax= option is an implementation of this resource distribution model.

Protections

A protected amount of a resource can be set up for a cgroup. If the resource usage is below the protection boundary, the kernel will try not to penalize this cgroup in favor of other cgroups that compete for the same resource. An overcommit is also allowed. The MemoryLow= option is an implementation of this resource distribution model.

Allocations

Exclusive allocations of an absolute amount of a finite resource. An overcommit is not allowed. An example of this resource type in Linux is the real-time budget.

Additional resources

- Managing CPU with systemd
- Allocating memory resources using systemd
- Configuring I/O bandwidth using systemd
15.3. ALLOCATING CPU RESOURCES USING SYSTEMD

On a system managed by systemd, each system service is started in its cgroup. By enabling the support for the CPU cgroup controller, the system uses the service-aware distribution of CPU resources instead of the per-process distribution. In the service-aware distribution, each service receives approximately the same amount of CPU time relative to all other services running on the system, regardless of the number of processes that comprise the service.

If a specific service requires more CPU resources, you can grant them by changing the CPU time allocation policy for the service.

**Procedure**

To set a CPU time allocation policy option when using systemd:

1. Check the assigned values of the CPU time allocation policy option in the service of your choice:

   ```
   $ systemctl show --property <CPU time allocation policy option> <service name>
   ```

2. Set the required value of the CPU time allocation policy option as a root:

   ```
   # systemctl set-property <service name> <CPU time allocation policy option>=<value>
   ```

   **NOTE**
   The cgroup properties are applied immediately after they are set. Therefore, the service does not need to be restarted.

The cgroup properties are applied immediately after they are set. Therefore, the service does not need to be restarted.

**Verification steps**

- To verify whether you successfully changed the required value of the CPU time allocation policy option for your service, run the following command:

  ```
  $ systemctl show --property <CPU time allocation policy option> <service name>
  ```

**Additional resources**

- CPU time allocation policy options for systemd
- Introduction to resource distribution models

15.4. CPU TIME ALLOCATION POLICY OPTIONS FOR SYSTEMD

The most frequently used CPU time allocation policy options include:

**CPUWeight=**

Assigns higher priority to a particular service over all other services. You can select a value from the interval 1 – 10,000. The default value is 100.

For example, to give `httpd.service` twice as much CPU as to all other services, set the value to `CPUWeight=200`. 
Note that CPUWeight= is applied only in cases when the operating system is overloaded.

CPUQuota=
Assigns the absolute CPU time quota to a service. The value of this option specifies the maximum percentage of CPU time that a service will receive relative to the total CPU time available, for example CPUQuota=30%.
Note that CPUQuota= represents the limit value for particular resource distribution models described in Introduction to resource distribution models.

For more information on CPUQuota=, see the systemd.resource-control(5) man page.

Additional resources
- Introduction to resource distribution models
- Allocating CPU resources using systemd

15.5. ALLOCATING MEMORY RESOURCES USING SYSTEMD

This section describes how to use any of the memory configuration options (MemoryMin, MemoryLow, MemoryHigh, MemoryMax, MemorySwapMax) to allocate memory resources using systemd.

Procedure
To set a memory allocation configuration option when using systemd:

1. Check the assigned values of the memory allocation configuration option in the service of your choice:
   
   ```
   $ systemctl show --property <memory allocation configuration option> <service name>
   ```

2. Set the required value of the memory allocation configuration option as a root:
   
   ```
   # systemctl set-property <service name> <memory allocation configuration option>=<value>
   ```

   **NOTE**

   The cgroup properties are applied immediately after they are set. Therefore, the service does not need to be restarted.

Verification steps
- To verify whether you successfully changed the required value of the memory allocation configuration option for your service, run the following command:
  
  ```
  $ systemctl show --property <memory allocation configuration option> <service name>
  ```

Additional resources
- Memory allocation configuration options for systemd
- Introduction to resource distribution models
15.6. MEMORY ALLOCATION CONFIGURATION OPTIONS FOR SYSTEMD

You can use the following options when using systemd to configure system memory allocation

**MemoryMin**

Hard memory protection. If the memory usage is below the limit, the cgroup memory will not be reclaimed.

**MemoryLow**

Soft memory protection. If the memory usage is below the limit, the cgroup memory can be reclaimed only if no memory is reclaimed from unprotected cgroups.

**MemoryHigh**

Memory throttle limit. If the memory usage goes above the limit, the processes in the cgroup are throttled and put under a heavy reclaim pressure.

**MemoryMax**

Absolute limit for the memory usage. You can use the kilo (K), mega (M), giga (G), tera (T) suffixes, for example `MemoryMax=1G`.

**MemorySwapMax**

Hard limit on the swap usage.

**NOTE**

When you exhaust your memory limit, the Out-of-memory (OOM) killer will stop the running service. To prevent this, lower the `OOMScoreAdjust=` value to increase the memory tolerance.

Additional resources

- Allocating memory resources using systemd
- Introduction to resource distribution models

15.7. CONFIGURING I/O BANDWIDTH USING SYSTEMD

To improve the performance of a specific service in RHEL 9, you can allocate I/O bandwidth resources to that service using systemd.

To do so, you can use the following I/O configuration options:

- IOWeight
- IODeviceWeight
- IOReadBandwidthMax
- IOWriteBandwidthMax
- IOReadIOPSMax
- IOWriteIOPSMax

Procedure
To set an I/O bandwidth configuration option using systemd:

1. Check the assigned values of the I/O bandwidth configuration option in the service of your choice:

   ```
   $ systemctl show --property <I/O bandwidth configuration option> <service name>
   ```

2. Set the required value of the I/O bandwidth configuration option as a root:

   ```
   # systemctl set-property <service name> <I/O bandwidth configuration option>=<value>
   ```

The cgroup properties are applied immediately after they are set. Therefore, the service does not need to be restarted.

**Verification steps**

- To verify whether you successfully changed the required value of the I/O bandwidth configuration option for your service, run the following command:

  ```
  $ systemctl show --property <I/O bandwidth configuration option> <service name>
  ```

**Additional resources**

- I/O bandwidth configuration options for systemd
- Introduction to resource distribution models

### 15.8. I/O BANDWIDTH CONFIGURATION OPTIONS FOR SYSTEMD

To manage the block layer I/O policies with systemd, the following configuration options are available:

**IOWeight**

Sets the default I/O weight. The weight value is used as a basis for the calculation of how much of the real I/O bandwidth the service receives in relation to the other services.

**IODeviceWeight**

Sets the I/O weight for a specific block device.

For example,

```
IODeviceWeight=/dev/disk/by-id/dm-name-rhel-root 200
```

**IOLReadBandwidthMax, IOWriteBandwidthMax**

Sets the absolute bandwidth per device or a mount point.

For example,

```
IOWriteBandwidth=/var/log 5M
```

**NOTE**

Systemd handles the file-system-to-device translation automatically.

**IOLReadIOPSMax, IOWriteIOPSMax**

A similar option to the previous one: sets the absolute bandwidth in Input/Output Operations Per Second (IOPS).
NOTE

Weight-based options are supported only if the block device is using the CFQ I/O scheduler. No option is supported if the device uses the Multi-Queue Block I/O queuing mechanism.

Additional resources

- Configuring I/O bandwidth using systemd
- Introduction to resource distribution models
CHAPTER 16. CONFIGURING HUGE PAGES

Physical memory is managed in fixed-size chunks called pages. On the x86_64 architecture, supported by Red Hat Enterprise Linux 9, the default size of a memory page is 4 KB. This default page size has proved to be suitable for general-purpose operating systems, such as Red Hat Enterprise Linux, which supports many different kinds of workloads.

However, specific applications can benefit from using larger page sizes in certain cases. For example, an application that works with a large and relatively fixed data set of hundreds of megabytes or even dozens of gigabytes can have performance issues when using 4 KB pages. Such data sets can require a huge amount of 4 KB pages, which can lead to overhead in the operating system and the CPU.

This section provides information about huge pages available in RHEL 9 and how you can configure them.

16.1. AVAILABLE HUGE PAGE FEATURES

With Red Hat Enterprise Linux 9, you can use huge pages for applications that work with big data sets, and improve the performance of such applications.

The following are the huge page methods, which are supported in RHEL 9:

**HugeTLB pages**

HugeTLB pages are also called static huge pages. There are two ways of reserving HugeTLB pages:

- **At boot time:** It increases the possibility of success because the memory has not yet been significantly fragmented. However, on NUMA machines, the number of pages is automatically split among the NUMA nodes. For more information on parameters that influence HugeTLB page behavior at boot time, see Parameters for reserving HugeTLB pages at boot time and how to use these parameters to configure HugeTLB pages at boot time, see Configuring HugeTLB at boot time.

- **At run time:** It allows you to reserve the huge pages per NUMA node. If the run-time reservation is done as early as possible in the boot process, the probability of memory fragmentation is lower. For more information on parameters that influence HugeTLB page behavior at run time, see Parameters for reserving HugeTLB pages at run time and how to use these parameters to configure HugeTLB pages at run time, see Configuring HugeTLB at run time.

**Transparent HugePages (THP)**

With THP, the kernel automatically assigns huge pages to processes, and therefore there is no need to manually reserve the static huge pages. The following are the two modes of operation in THP:

- **system-wide:** Here, the kernel tries to assign huge pages to a process whenever it is possible to allocate the huge pages and the process is using a large contiguous virtual memory area.

- **per-process:** Here, the kernel only assigns huge pages to the memory areas of individual processes which you can specify using the `madvise()` system call.

**NOTE**

The THP feature only supports 2 MB pages.
For information on enabling and disabling THP, see Enabling transparent hugepages and Disabling transparent hugepages.

16.2. PARAMETERS FOR RESERVING HUGETLB PAGES AT BOOT TIME

Use the following parameters to influence HugeTLB page behavior at boot time.

For more information on how to use these parameters to configure HugeTLB pages at boot time, see Configuring HugeTLB at boot time.

Table 16.1. Parameters used to configure HugeTLB pages at boot time

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>hugepages</td>
<td>Defines the number of persistent huge pages configured in the kernel at boot time.</td>
<td>The default value is 0.</td>
</tr>
<tr>
<td></td>
<td>In a NUMA system, huge pages, that have this parameter defined, are divided equally between nodes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can assign huge pages to specific nodes at runtime by changing the value of the nodes in the /sys/devices/system/node/node_id/hugepages/hugepages-size/nr_hugepages file.</td>
<td></td>
</tr>
<tr>
<td>hugepagesz</td>
<td>Defines the size of persistent huge pages configured in the kernel at boot time.</td>
<td>Valid values are 2 MB and 1 GB. The default value is 2 MB.</td>
</tr>
<tr>
<td>default_hugepagesz</td>
<td>Defines the default size of persistent huge pages configured in the kernel at boot time.</td>
<td>Valid values are 2 MB and 1 GB. The default value is 2 MB.</td>
</tr>
</tbody>
</table>

16.3. CONFIGURING HUGETLB AT BOOT TIME

The page size, which the HugeTLB subsystem supports, depends on the architecture. The x86_64 architecture supports 2 MB huge pages and 1 GB gigantic pages.

This procedure describes how to reserve a 1 GB page at boot time.

Procedure

1. Create a HugeTLB pool for 1 GB pages by appending the following line to the kernel command-line options in the /etc/default/grub file as root:
default_hugepagesz=1G hugepagesz=1G

2. Regenerate the GRUB2 configuration using the edited default file:
   a. If your system uses BIOS firmware, execute the following command:
   ```
   # grub2-mkconfig -o /boot/grub2/grub.cfg
   ```
   b. If your system uses UEFI framework, execute the following command:
   ```
   # grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
   ```

3. Create a new file called `hugetlb-gigantic-pages.service` in the `/usr/lib/systemd/system/` directory and add 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=/usr/lib/systemd/hugetlb-reserve-pages.sh

   [Install]
   WantedBy=sysinit.target
   ```

4. Create a new file called `hugetlb-reserve-pages.sh` in the `/usr/lib/systemd/` directory and add the following content:
   While adding the following content, replace `number_of_pages` with the number of 1GB pages you want to reserve, and `node` with the name of the node on which to reserve these pages.

   ```
   #!/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 number_of_pages node
   ```

For example, to reserve two 1 GB pages on `node0` and one 1GB page on `node1`, replace the `number_of_pages` with 2 for `node0` and 1 for `node1`: 
resv_pages 2 node0
resv_pages 1 node1

5. Create an executable script:

```bash
# chmod +x /usr/lib/systemd/hugetlb-reserve-pages.sh
```

6. Enable early boot reservation:

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

**NOTE**

- You can try reserving more 1GB pages at runtime by writing to `nr_hugepages` at any time. However, such reservations can fail due to memory fragmentation. The most reliable way to reserve 1 GB pages is by using this `hugetlb-reserve-pages.sh` script, which runs early during boot.

- Reserving static huge pages can effectively reduce the amount of memory available to the system, and prevents it from properly utilizing its full memory capacity. Although a properly sized pool of reserved huge pages can be beneficial to applications that utilize it, an oversized or unused pool of reserved huge pages will eventually be detrimental to overall system performance. When setting a reserved huge page pool, ensure that the system can properly utilize its full memory capacity.

Additional resources

- `systemd.service(5)` man page
- `/usr/share/doc/kernel-doc-kernel_version/Documentation/vm/hugetlbpage.txt` file

### 16.4. PARAMETERS FOR RESERVING HUGETLB PAGES AT RUN TIME

Use the following parameters to influence HugeTLB page behavior at run time.

For more information on how to use these parameters to configure HugeTLB pages at run time, see [Configuring HugeTLB at run time](#).

**Table 16.2. Parameters used to configure HugeTLB pages at run time**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>File name</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nr_hugepages</code></td>
<td>Defines the number of huge pages of a specified size assigned to a specified NUMA node.</td>
<td><code>/sys/devices/system/node/node_id/hugepages/hugepages-size/nr_hugepages</code></td>
</tr>
</tbody>
</table>
nr_overcommit_hugepages

Defines the maximum number of additional huge pages that can be created and used by the system through overcommitting memory. Writing any non-zero value into this file indicates that the system obtains that number of huge pages from the kernel's normal page pool if the persistent huge page pool is exhausted. As these surplus huge pages become unused, they are then freed and returned to the kernel's normal page pool.

Parameter Description File name
nr_overcommit_hugepages Defines the maximum number of additional huge pages that can be created and used by the system through overcommitting memory. Writing any non-zero value into this file indicates that the system obtains that number of huge pages from the kernel's normal page pool if the persistent huge page pool is exhausted. As these surplus huge pages become unused, they are then freed and returned to the kernel's normal page pool. /proc/sys/vm/nr_overcommit_hugepages

16.5. CONFIGURING HUGETLB AT RUN TIME

This procedure describes how to add 20 2048 kB huge pages to node2.

To reserve pages based on your requirements, replace:
- 20 with the number of huge pages you wish to reserve,
- 2048kB with the size of the huge pages,
- node2 with the node on which you wish to reserve the pages.

Procedure

1. Display the memory statistics:

```bash
# numastat -cm | egrep 'Node|Huge'
Node 0 Node 1 Node 2 Node 3  Total add
AnonHugePages         0      2      0      8     10
HugePages_Total       0      0      0      0      0
HugePages_Free        0      0      0      0      0
HugePages_Surp        0      0      0      0      0
```

2. Add the number of huge pages of a specified size to the node:

```bash
# echo 20 > /sys/devices/system/node/node2/hugepages/hugepages-2048kB/nr_hugepages
```

Verification steps

- Ensure that the number of huge pages are added:

```bash
# numastat -cm | egrep 'Node|Huge'
Node 0 Node 1 Node 2 Node 3  Total
AnonHugePages         0      2      0      8     10
```
HugePages_Total 0 0 40 0 40
HugePages_Free 0 0 40 0 40
HugePages_Surp 0 0 0 0

Additional resources
- numastat(8) man page

16.6. ENABLING TRANSPARENT HUGEPAGES

THP is enabled by default in Red Hat Enterprise Linux 9. However, you can enable or disable THP. This procedure describes how to enable THP.

Procedure
1. Check the current status of THP:
   
   # cat /sys/kernel/mm/transparent_hugepage/enabled

2. Enable THP:

   # echo always > /sys/kernel/mm/transparent_hugepage/enabled

3. To prevent applications from allocating more memory resources than necessary, disable the system-wide transparent huge pages and only enable them for the applications that explicitly request it through the madvise:

   # echo madvise > /sys/kernel/mm/transparent_hugepage/enabled

   **NOTE**

   Sometimes, providing low latency to short-lived allocations has higher priority than immediately achieving the best performance with long-lived allocations. In such cases, you can disable direct compaction while leaving THP enabled.

   Direct compaction is a synchronous memory compaction during the huge page allocation. Disabling direct compaction provides no guarantee of saving memory, but can decrease the risk of higher latencies during frequent page faults. Note that if the workload benefits significantly from THP, the performance decreases. Disable direct compaction:

   # echo madvise > /sys/kernel/mm/transparent_hugepage/defrag

Additional resources
- madvise(2) man page
- Disabling transparent hugepages.

16.7. DISABLING TRANSPARENT HUGEPAGES

THP is enabled by default in Red Hat Enterprise Linux 9. However, you can enable or disable THP.
This procedure describes how to disable THP.

Procedure

1. Check the current status of THP:

   # cat /sys/kernel/mm/transparent_hugepage/enabled

2. Disable THP:

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

16.8. IMPACT OF PAGE SIZE ON TRANSLATION LOOKASIDE BUFFER SIZE

Reading address mappings from the page table is time-consuming and resource-expensive, so CPUs are built with a cache for recently-used addresses, called the Translation Lookaside Buffer (TLB). However, the default TLB can only cache a certain number of address mappings.

If a requested address mapping is not in the TLB, called a TLB miss, the system still needs to read the page table to determine the physical to virtual address mapping. Because of the relationship between application memory requirements and the size of pages used to cache address mappings, applications with large memory requirements are more likely to suffer performance degradation from TLB misses than applications with minimal memory requirements. It is therefore important to avoid TLB misses wherever possible.

Both HugeTLB and Transparent Huge Page features allow applications to use pages larger than \(4\ \text{KB}\). This allows addresses stored in the TLB to reference more memory, which reduces TLB misses and improves application performance.