Oracle Linux 8

Monitoring and Tuning the System

F24025-18
January 2024

Oracle Linux 8 Monitoring and Tuning the System,

F24025-18

Contents

Preface

Oracle Linux 9: Monitoring and Tuning the System describes the various utilities, features, and services that you can use to monitor system performance, detect performance issues, and improve the performance of various system components.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Monitoring the System and Optimizing Performance

Performance issues can be caused by several system components, including software or hardware, as well as any related interactions. Many performance diagnostics utilities are available in Oracle Linux and include tools that monitor and analyze the resource usage of different hardware components, as well as tracing tools for diagnosing performance issues in multiple processes and related threads.

Many performance issues are the result of configuration errors. You can avoid these errors by using a validated configuration that has been pretested for the enabled software, hardware, storage, drivers, and networking components. A validated configuration incorporates best practices for an Oracle Linux deployment and has undergone real-world testing of the complete stack. Oracle publishes many validated configurations, which are freely available for download. Refer to the release notes for the release that you are running for additional recommendations on kernel parameter settings.

System Performance and Monitoring Utilities

Different Oracle Linux utilities are available that enable you to collect information about system resource usage and errors, and help you to identify performance problems that are caused by overloaded disks, network, memory, or CPUs.

The following packages of the system performance and monitoring utilities are installed by default. If they are not already available, you can install the packages as required.

- `util-linux` provides `dmesg` that displays the contents of the kernel ring buffer, which can contain errors about system resource usage.
- `procps-ng` provides these utilities:
  - `free`: Displays the amount of free and used memory in the system.
  - `top`: Provides a dynamic real-time view of the tasks that are running on a system.
  - `uptime`: Displays the system load averages for the past 1, 5, and 15 minutes.
  - `vmstat`: Reports virtual memory statistics.
- `iproute` provides these utilities:
  - `ip`: Reports network interface statistics and errors.
  - `ss`: Reports network interface statistics.

> **Tip:**
>
> To verify if a utility is available in the system, check if the utility's package is installed. For example, for the `dmesg` utility, you can type:
>
> ```
> dnf info util-linux
>
>
> Installed Packages
> Name         : util-linux
> Version      : version-number
> ...
> ```

You can install the following additional packages to take advantage of additional utilities.

- `sysstat` provides these utilities:
  - `iostat`: Reports I/O statistics.
  - `mpstat`: Reports processor-related statistics.
  - `sar`: Reports information about system activity.
- `iotop` provides `iotop` that monitors disk and swap I/O on a per-process basis.
- `nfs-utils` provides `nfsiostat` that reports I/O statistics for NFS mounts

Many of these utilities provide overlapping functionality. For more information, see the individual manual page for the utility.

For a hands-on tutorial and associated video content on many of these utilities, see Monitor system resources on Oracle Linux.

Monitoring the Usage of System Resources

You can monitor system performance by collecting information about system resources. For better assessment, first, establish a baseline of acceptable measurements under typical operating conditions. You can then use that baseline as a reference point so that you can identify more easily memory shortages, spikes in resource usage, and other problems when they occur. You can also use performance monitoring to plan for future growth and determine how configuration changes might affect future performance.

For more information about monitoring the use of resources in the system, see also Working With OSWatcher Black Box and Working With Performance Co-Pilot.

Monitoring Usage in Real Time

To run a monitoring command for a set number of seconds in real time and watch the output change, use the `watch` command. For example, you can repeatedly run the `mpstat` command every second by using the following command:

```
sudo watch -n 1 mpstat
```

That command generates a single-line output that changes information every second, for example:

```
hh:mm:ss  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal
%guest  %gnice   %idle
hh:mm:ss  all    1.44    0.02    0.80    0.01    0.07    0.05    0.06
0.00    0.00   97.56
```

Alternatively, many of the commands enable you to specify the sampling interval in seconds, for example:

```
sudo mpstat 1
```

The command displays the same information as the previous command, except that the information is in a running list where a new line of information is generated every second, for example:

```
hh:mm:ss  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal
%guest  %gnice   %idle
hh:mm:ss  all    0.00    0.00    0.00    0.00    0.50    0.50    0.00
0.00    0.00   99.01
hh:mm:ss  all    0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00    0.00  100.00
...
```

To stop real time monitoring, press Ctrl-c.

Monitoring Usage Through Logs

The `sar` utility records statistics every 10 minutes while the system is running and retains that information for every day of the current month. The information is stored in `/var/log/sa/saDD`.

To display the contents of the `sar` log of a specific day of the current month, type:

```
sudo sar -A -f /var/log/sa/saDD
```

You can also use the `sar` command to create a customized log that contains a record of specific information that you want to monitor. Use the following syntax:

```
sudo sar -o datafile seconds count >/dev/null 2>&1 &
```

In the previous command, *datafile* is the full path to the customized log where you want to store the information, while *count* represents the number of samples to record. With this command, the `sar` process runs in the background and collects the data.

To display the results of this monitoring, you would type:

```
sudo sar -A -f datafile
```

Monitoring CPU Usage

When all of the CPU cores are occupied with executing system processes, other processes must wait until a CPU core becomes free or when the scheduler switches a CPU core to run its code. Too many processes that are queued too often can represent a system performance bottleneck.

The following are some of the commands for monitoring CPU usage. For the different options and arguments that you can use with these commands, refer to their respective manual pages.

- `uptime`
- `mpstat`
- `sar`
- `top`

The following examples show how you can use these commands to obtain statistics on your system's memory usage:

- Review CPU usage statistics for each CPU core and averaged the data across all of the CPU cores.

  - `mpstat -P ALL`

    ```
    05:10:01 PM  CPU    %usr   %nice    %sys   %iowait    %irq
    %soft  %steal  %guest  %gnice   %idle
    05:10:01 PM  all    0.76    0.02    0.70      0.02    0.08
    0.05    0.06    0.00    0.00   98.32
    05:10:01 PM    0    0.75    0.01    0.68      0.00    0.09
    0.03    0.07    0.00    0.00   98.37
    05:10:01 PM    1    0.76    0.03    0.72      0.03    0.06
    0.06    0.06    0.00    0.00   98.27
    ```

  - `sar -u -P ALL`

    ```
    03:00:01 PM  CPU  %user   %nice   %system   %iowait  %steal
    %idle
    03:10:01 PM  all  8.30    0.00    2.20      0.22     0.10
    89.18
    03:10:01 PM    0  8.22    0.00    2.64      0.31     0.09
    88.74
    03:10:01 PM    1  8.39    0.00    1.77      0.13     0.10
    89.61
    ...
    ```

    The output of these commands include data under the `%idle` heading, which show the percentage of time that a CPU is not running system or process code. If the percentage is near 0% most of the time on all CPU cores, then the system is CPU-bound for the workload that is running.

We recommend that the percentage of time to run system code, which is reported under the `%system` or `%sys` heading, not exceed 30%, especially if `%idle` is close to 0%.

- Review information about system load average.

  – `uptime`

  ```
  21:25:34 up  6:28,  1 user,  load average: 0.02, 0.10, 0.04
  ```

  – `sar -q`

  ```
  14:57:55     LINUX RESTART    (2 CPU)

  03:00:01 PM   runq-sz  plist-sz   ldavg-1   ldavg-5  ldavg-15
  blocked
  03:10:01 PM         0       214      0.19      0.30      0.22
  0
  03:20:01 PM         0       212      0.00      0.05      0.10
  0
  ...
  Average:      runq-sz  plist-sz   ldavg-1   ldavg-5  ldavg-15
  blocked
  Average:            1       212      0.12      0.08      0.05
  0
  ```

  The system load average consists of the combination of the number of processes that are currently running on CPU cores, waiting to run, and waiting for disk I/O activity to complete, all of which are then averaged over time.

  The `uptime` command shows the information in a single line, while the `sar` syntax displays the information in columns under `ldavg-*` headings. Additionally, the `sar` syntax also shows the number of processes currently waiting to run as well as the total number of processes, which are reported under the `runq-sz` and `plist_sz` headings.

  On a busy system, we recommend that the load average typically not be greater than two times the number of CPU cores over a period of 5 or 15 minutes. If the load average exceeds four times the number of CPU cores for long periods, then the system is overloaded.

  For a better assessment of the system load, determine the system's average load under normal loads where users and applications do not experience problems with system responsiveness. Then, look for deviations from this benchmark over time. A dramatic rise in the load average can indicate a serious performance problem.

- Review a real-time listing of CPU activity.

  `top`

  ```
  top - 22:13:34 up  7:16,  1 user,  load average: 0.00, 0.02, 0.01
  Tasks: 149 total,   1 running, 148 sleeping,   0 stopped,   0 zombie
  %Cpu(s):  0.2 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.2 hi,  0.0 si,
  0.0 st
  MiB Mem :  14705.5 total,  11738.9 free,    753.2 used,   2213.4 buff/
  cache
  ```

```
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used.  13588.9
avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM
TIME+ COMMAND
  34781 root      20   0   11384   7100   1700 S   1.0   0.0
0:00.03 pidstat
   2014 root      20   0   26216   3568   3056 S   0.7   0.0
0:04.78 OSWatcher
   1481 root      20   0  202628  38328  36792 S   0.3   0.3
0:02.94 sssd_nss
   ...
```

By default, `top` lists the most CPU-intensive processes on the system. The output's upper section displays general information including the load averages over the past 1, 5 and 15 minutes, the number of running and sleeping processes or tasks, and total CPU and memory usage.

The second table displays a list of processes, including the process ID number (PID), the process owner, CPU usage, memory usage, running time, and the command name. By default, the list is sorted by CPU usage, with the top consumer of CPU listed first.

To stop the `top` process, press Ctrl-c.

All the commands can be used together to provide you a picture of the system's CPU usage. For example, sustained large load average or large run queue size and low `%idle` percentages can indicate that the system has insufficient CPU capacity for the workload. When CPU usage is high, the `top` command can identify which processes are likely responsible.

Monitoring Memory Usage

The following are useful utilities to monitor memory usage:

Monitoring Memory Usage With the sar Command

To monitor memory usage, use the `sar` command with available options depending on the information you want to obtain. For a list of options, type `sar --help`.

- `sar -r`: Reports memory usage statistics, such as free (`kbmemfree`), available (`kbavail`), and used (`kbmemused`) memory. The report also include `%memused`, which is the percentage of physical memory in use.
- `sar -B`: Reports memory paging statistics, such as page in (`pgpgin/s`) and page out (`pgpgout/s`), page faults and major faults, and so on. The report also includes `pgscank/s`, which is the number of memory pages scanned by the `kswapd` daemon each second, and `pgscand/s`, which is the number of memory pages scanned directly each second.
- `sar -W`: Reports swapping statistics, including `pswpin/s` and `pswpout/s`, which are the numbers of pages each second swapped in and out each second.

If `%memused` is near 100% and the scan rate is continuously over 200 pages each second, the system has a memory shortage.

When a system runs out of real or physical memory and starts using swap space, system performance deteriorates dramatically. If you run out of swap space, some programs or even the entire OS are likely to malfunction. If the `free` or `top` commands indicate that little swap space remains available, then the system is running low on memory.

The output from the `dmesg` command might include notification of any problems with physical memory that were detected at boot time.

Using the Adaptive Memory Management Daemon

To manage memory usage, you can use the Adaptive Memory Management daemon, which is available beginning with UEK R6. This daemon is a user space service that monitors free memory on an Oracle Linux system and predicts memory fragmentation and usage. It can also automatically reclaim memory if the system memory becomes too fragmented or is at risk of being filled to capacity.

If the system memory becomes highly fragmented, `adaptivemmd` triggers the kernel to compact memory so that fragmented space can be reclaimed before it is reallocated. If the system is likely to exhaust the available memory, watermarks are adjusted and this can trigger the kernel to free up new pages in memory. Adaptive Memory Management is available in Unbreakable Enterprise Kernel Release 6 and later.

To use this utility, do the following:

1. Install the `adaptivemm` package.

   ```
   sudo dnf install -y adaptivemm
   ```

2. Start the daemon service.

   ```
   sudo systemctl enable --now adaptivemmd
   ```

To see the different options that you can use with the `adaptivemmd` command, type:

```
sudo adaptivemmd -h
```

You can change the configuration options in `/etc/sysconfig/adaptivemmd`.

For more information see the `adaptivemmd(8)` manual page.

Monitoring Block I/O Usage

The `iostat` command monitors the loading of block I/O devices by observing the time that the devices are active relative to the average data transfer rates. You can use this information to adjust the system configuration to balance the I/O loading across disks and host adapters. The following is a sample of the command output:

```
iostat


avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.69    0.05    0.77    0.00    0.03   98.46

Device             tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               1.05         2.32        25.19     611410    6640659
```

```
dm-0              0.62         2.07        20.22      545716     5329660
dm-1              0.70         0.02         4.97        4632     1308788
```

`iostat -x` reports extended statistics about block I/O activity at one second intervals, including `%util`, which is the percentage of CPU time spent handling I/O requests to a device, and `avgqu-sz`, which is the average queue length of I/O requests that were issued to that device. If `%util` approaches 100% or `avgqu-sz` is greater than 1, device saturation is occurring.

You can also use the `sar -d` command to report on block I/O activity, including values for `%util` and `avgqu-sz`.

The `iotop` utility can help you identify which processes are responsible for excessive disk I/O. `iotop` has a similar user interface to `top`. In its upper section, `iotop` displays the total disk input and output usage in bytes per second. In its lower section, `iotop` displays I/O information for each process, including disk input output usage in bytes per second, the percentage of time spent swapping in pages from disk or waiting on I/O, and the command name. The following is a sample command output:

```
sudo iotop
```

```
Total DISK READ :    0.00 B/s | Total DISK WRITE :       0.00 B/s
Actual DISK READ:    0.00 B/s | Actual DISK WRITE:       0.00 B/s
    TID  PRIO  USER     DISK READ DISK WRITE>
COMMAND
      1 be/4 root        0.00 B/s    0.00 B/s systemd --switched-root
--system --deserialize 16
      2 be/4 root        0.00 B/s    0.00 B/s [kthreadd]
...
```

While you review the output, use the arrow keys to change the sort field, and press `A` to switch the I/O units between bytes each second and total number of bytes, or `O` to switch between displaying all processes or only those processes that are performing I/O.

Monitoring File System Usage

The `sar -v` command reports the number of unused cache entries in the directory cache (`dentunusd`) and the numbers of in-use file handles (`file-nr`), `inode` handlers (`inode-nr`), and pseudo terminals (`pty-nr`).

```
sar -v
```

```
12:00:01 AM dentunusd   file-nr  inode-nr    pty-nr
12:10:33 AM     80101      2944     73074         0
12:20:33 AM     79788      2944     72654         0
```

`nfsiostat` reports I/O statistics for each NFS file system that is mounted. If this command is not available install the `nfs-utils` package.

Monitoring Network Usage

The `ip -s link` command displays network statistics and errors for all network devices, including the numbers of bytes transmitted (`TX`) and received (`RX`). The `dropped` and `overrun` fields provide an indicator of network interface saturation, for example:

```
ip -s link
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    RX: bytes  packets  errors  dropped overrun mcast
    240        4        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    240        4        0       0       0       0
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:60:95:d5 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    258187485  671730   0       0       0       17296
    TX: bytes  packets  errors  dropped carrier collsns
    13227598   130827   0       0       0       0
```

The `ss -s` command displays summary statistics for each protocol, for example:

```
ss -s
```

```
Total: 193
TCP:   9 (estab 2, closed 0, orphaned 0, timewait 0)
```

Using the Graphical System Monitor

The GNOME desktop environment includes a graphical system monitor that you can use to display information about the system configuration, running processes, resource usage, and file systems.

To display the System Monitor, use the following command:

```
gnome-system-monitor
```

Selecting the **Resources** tab displays the following information:

- CPU usage history in graphical form and the current CPU usage as a percentage.
- Memory and swap usage history in graphical form and the current memory and swap usage.
- Network usage history in graphical form, the current network usage for reception and transmission, and the total amount of data received and transmitted.

To display the System Monitor Manual, press `F1` or select Help, then select Contents.

2
Working With the sos Command

The `sos` command collects information about a system such as hardware configuration, software configuration, and operational state. You can also use the `sos report` command to enable diagnostics and analytical functions on the current system.

The generated report is useful in cases where you are being assisted by Oracle support in troubleshooting a problem in the system. The support representative can use the report to obtain an accurate picture of the system, its resources, all the applications and processes that exist in the system, and all other data that can help determine the causes of the issues you are encountering.

The `sos` utility requires the installation of the `sos` package. To install the package, type:

```
sudo dnf install sos
```

Running the sos Command

To obtain a list of options and arguments that you can use with the `sos` utility, type:

```
sos report -h


optional arguments:
  -h, --help            show this help message and exit

Global Options:
  --batch               Do not prompt interactively
  --config-file CONFIG_FILE
                        specify alternate configuration file
...
```

Creating the SOS Report

To collect all diagnostic and configuration information from the system and its installed applications, run the following command:

```
sudo sos report


sosreport (sosreport version)
...
The generated archive may contain data considered sensitive and its
content should be reviewed by the originating organization before being
passed to any third party.
...
Press ENTER to continue, or CTRL-C to quit.
```

Every time you issue the `sos` utility, the utility always prompts you whether to continue or to quit. If you press Enter to continue, you can use an optional prompt to specify a case ID for the report.

```
Optionally, please enter the case id that you are generating this
report for []:
```

If you are generating the report as related to a specific troubleshooting case, you can enter the case ID at this prompt.

After you have provided information as prompted, the command proceeds to generate the report, which can take a considerable time to complete. At the end of the process, the screen displays a message similar to the following:

```
Your sosreport has been generated and saved in:
        /var/tmp/sosreport-hostname-case#-datestamp-ID.tar.xz

 Size   20.62MiB
 Owner  root
 sha256
428f7b4118acd2d349bb022946877d853aa0eefbb4d340af3839810dc634b8b7

Please send this file to your support representative.
```

The report is generated as an `xa`-compressed `tar` file in the `/var/tmp` directory. In the report's file name, the *ID* is randomly created by the utility.

> **Important:**
>
> As previously indicated, the report can be useful in cases where you engage Oracle support to diagnose and troubleshoot issues that you have observed in the system. However, the report contains sensitive information specific to your company. Ensure that you review the contents of the report and identify sensitive information before sending the report to any third-party.

Hiding Sensitive Information in an SOS Report

To secure sensitive information before sending the report externally, you can use the `clean` functionality of the `sos` utility. This functionality attempts to obfuscate any information in the report that is generally considered to be potentially sensitive, such as the following information:

- IPv4 addresses and networks (network topologies are retained)
- MAC addresses
- Host names
- Usernames
- Any words or phrases that you specify with the `--keyword` option

To use the `sos clean` utility on a generated report, type the following command and follow the prompts that are displayed:

```
sudo sos clean /var/tmp/sosreport-hostname-case#-datestamp-ID.tar.xz


...
Users should review any resulting data and/or archives generated or
processed by
this utility for remaining sensitive content before being passed to a third
party.

Press ENTER to continue, or CTRL-C to quit.
```

At the end of the process, the screen displays a message similar to the following:

```
Successfully obfuscated 1 report(s)

A mapping of obfuscated elements is available at
    /var/tmp/sosreport-host0-2022-08-08-qxbegcn-private_map

The obfuscated archive is available at
    /var/tmp/sosreport-host0-2022-08-08-qxbegcn-obfuscated.tar.xz

    Size    3.62MiB
    Owner    root

Please send the obfuscated archive to your support representative and keep
the mapping file private
```

The resulting report that has been scrubbed of sensitive information is also stored in `/var/tmp`. However, the file name itself is revised. The hostname is generic, and importantly, `obfuscated` is added to the file name so you can readily identify the clean version of the report.

Caution:

Consider the following about the `sos clean` utility:

- The `clean` functionality is a best-effort method to identify and then mask sensitive information. However, `sos clean` does not guarantee that the coverage of the masking process is complete in a specific system.
- Reports that are processed with the `sos clean` command obfuscate certain details which a third-party such as a support representative might need to provide better assistance when troubleshooting problems.
- You must always audit archives and reports that are generated by the `sos` utility before sending any of these files externally.

To automatically clean any `sos` report that you create, use the following command syntax when generating a report:

```
sodo sos report --clean
```

For more information, see the `sos-report(1)` and `sos-clean(1)` manual pages. See also https://github.com/sosreport/sos/wiki.

Additional Sample Usages of the sos Command

The `sos report` command can also be used with other options. For example, to only list available plugins and plugin options in the report, type:

```
sudo sos report -l
```

The plugins that are displayed by the command are grouped according to the following sections:

- All enabled plugins
- All disabled plugins
- Available options for all of the plugins
- Available plugin options

See the `sos-report(1)` manual page for information about how to enable or disable plugins and how to set values for plugin options.

You can also obtain only information specific to a problem area and specify options to tailor the report that is generated. For example, to record only information about Apache and Tomcat and to gather all of the Apache logs, type:

```
sudo sos report -o apache,tomcat -k apache.log=on
```

To enable all of the Boolean options for all of the loaded plugins (excluding the `rpm.rpmva` plugin) and verify all packages:

```
sudo sos report -a -k rpm.rpmva=off
```

For more information, see the `sos-report(1)` and `sos-clean(1)` manual pages. See also https://github.com/sosreport/sos/wiki.

Reviewing Information Gathered by sosreport

The `sos` command is automatically configured to collect hardware information, system configuration files and log data; but, you can enable and disable modules to suit your own data protection needs.

> **Note:**
>
> The module information that is provided in this table relates to `sos` 3.9. To verify the modules you have installed, run the `sos report` command. The output includes the version of the `sos` utility that you are currently running.
>
> Disabling modules prevents the `sos` command from collecting certain details that might be needed for advanced troubleshooting, such as networking information.

Module	Information Type	Included Files
`anaconda`	Installation log files	• `/root/install.log` • `/root/install.log.syslog` • `/var/log/anaconda` • `/var/log/anaconda.*`
`auditd`	Audit log files	• `/etc/audit/auditd.conf` • `/etc/audit/audit.rules` • `/var/log/audit/*`
`boot`	System boot process details	• `/etc/milo.conf` • `/etc/silo.conf` • `/boot/efi/efi/redhat/elilo.conf` • `/etc/yaboot.conf` • `/boot/yaboot.conf`
`cron`	Root user `cron` commands	• `/etc/cron*` • `/etc/crontab` • `/var/log/cron` • `/var/spool/cron`
`cups`	Printer log files	• `/etc/cups/*.conf` • `/etc/cups/*.types` • `/etc/cups/lpoptions` • `/etc/cups/ppd/*.ppd` • `/var/log/cups/*`
`date`	Context data	• `/etc/localtime`
`devicemapper`	Hardware details	

Module	Information Type	Included Files
filesys	List of all files in use	• /proc/fs/* • /proc/mounts • /proc/filesystems • /proc/self/mounts • /proc/self/mountinfo • /proc/self/mountstats • /proc/[0-9]*/mountinfo • /etc/mtab • /etc/fstab
grub2	Kernel and system start-up configuration	• /boot/efi/EFI/*/grub.cfg • /boot/grub2/grub.cfg • /boot/grub2/grubenv • /boot/grub/grub.cfg • /boot/loader/entries • /etc/default/grub • /etc/grub2.cfg • /etc/grub.d/*
hardware	Hardware details	• /proc/interrupts • /proc/irq • /proc/dma • /proc/devices • /proc/rtc • /var/log/mcelog • /sys/class/dmi/id/* • /sys/class/drm/*/edid
host	Host identification	• /etc/sos.conf • /etc/hostid

Module	Information Type	Included Files
kernel	System log files	• /etc/conf.modules • /etc/modules.conf • /etc/modprobe.conf • /etc/modprobe.d • /etc/sysctl.conf • /etc/sysctl.d • /lib/modules/*/modules.dep • /lib/sysctl.d • /proc/cmdline • /proc/driver • /proc/kallsyms • /proc/lock* • /proc/buddyinfo • /proc/misc • /proc/modules • /proc/slabinfo • /proc/softirqs • /proc/sys/kernel/random/boot_id • /proc/sys/kernel/tainted • /proc/timer* • /proc/zoneinfo • /sys/firmware/acpi/* • /sys/kernel/debug/tracing/* • /sys/kernel/livepatch/* • /sys/module/*/parameters • /sys/module/*/initstate • /sys/module/*/refcnt • /sys/module/*/taint • /sys/module/*/version • /sys/devices/system/clocksource/*/available_clocksource • /sys/devices/system/

Module	Information Type	Included Files
		clocksource/*/current_clocksource • /sys/fs/pstore • /var/log/dmesg
libraries	List of shared libraries	• /etc/ld.so.conf • /etc/ld.so.conf.d/*
logs	System log files	• /etc/syslog.conf • /etc/rsyslog.conf • /etc/rsyslog.d • /run/log/journal/* • /var/log/auth.log • /var/log/auth.log.1 • /var/log/auth.log.2* • /var/log/boot.log • /var/log/dist-upgrade • /var/log/installer • /var/log/journal/* • /var/log/kern.log • /var/log/kern.log.1 • /var/log/kern.log.2* • /var/log/messages* • /var/log/secure* • /var/log/syslog • /var/log/syslog.1 • /var/log/syslog.2* • /var/log/udev • /var/log/unattended-upgrades
lvm2	Hardware details	
memory	Hardware details	• /proc/pci • /proc/meminfo • /proc/vmstat • /proc/swaps • /proc/slabinfo • /proc/pagetypeinfo • /proc/vmallocinfo • /sys/kernel/mm/ksm • /sys/kernel/mm/transparent_hugepage/enabled

Module	Information Type	Included Files
networking	Network identification	• /etc/dnsmasq* • /etc/host* • /etc/inetd.conf • /etc/iproute2 • /etc/network* • /etc/nftables • /etc/nftables.conf • /etc/nsswitch.conf • /etc/resolv.conf • /etc/sysconfig/nftables.conf • /etc/xinetd.conf • /etc/xinetd.d • /etc/yp.conf • /proc/net/* • /sys/class/net/*/device/numa_node • /sys/class/net/*/flags • /sys/class/net/*/statistics/*
pam	Login security settings	• /etc/pam.d/* • /etc/security
pci	Hardware details	• /proc/bus/pci • /proc/iomem • /proc/ioports
process	List of all running processes and process details	• /proc/sched_debug • /proc/stat • /proc/[0-9]*/smaps
processor	Hardware details	• /proc/cpuinfo • /sys/class/cpuid • /sys/devices/system/cpu
rpm	Installed software packages	• /var/lib/rpm/* • /var/log/rpmpkgs
sar	Resource and usage data	• /var/log/sa/*
selinux	Security settings	• /etc/sestatus.conf • /etc/selinux • /var/lib/selinux
services	All defined system services	• /etc/inittab • /etc/rc.d/* • /etc/rc.local

Module	Information Type	Included Files
ssh	SSH configuration	• /etc/ssh/ssh_config • /etc/ssh/sshd_config
x11	GUI logs for the X Window System	• /etc/X11/* • /var/log/Xorg.*.log • /var/log/Xorg.*.log.old • /var/log/XFree86.*.log • /var/log/XFree86.*.log.old
yum	Installed software packages	• /etc/pki/consumer/cert.pem • /etc/pki/entitlement/*.pem • /etc/pki/product/*.pem • /etc/yum/* • /etc/yum.repos.d/* • /etc/yum/pluginconf.d/* • /var/log/dnf.log

3
Working With OSWatcher Black Box

Oracle OSWatcher Black Box (OSWbb) collects and archives OS and network metrics that you can use to diagnose performance issues. OSWbb operates as a set of background processes on the server and gathers data on a regular basis, invoking such UNIX utilities as `vmstat`, `mpstat`, `netstat`, `iostat`, and `top`.

OSWbb is particularly useful for Oracle RAC (Real Application Clusters) and Oracle Grid Infrastructure configurations. The RAC-DDT (Diagnostic Data Tool) script file includes OSWbb, but does not install it by default.

Installing OSWbb

To install OSWbb:

1. Sign in to My Oracle Support (MOS) at https://support.oracle.com.
2. Download OSWatcher from the link that is listed by Doc ID 301137.1 at https://support.oracle.com/epmos/faces/DocumentDisplay?id=301137.1.
3. Copy the file to the directory that you want to install OSWbb, then run the following command:

   ```
   tar xvf oswbbVERS.tar
   ```

   In the previous command, *VERS* represents the version number of OSWatcher, for example 832 for OSWatcher 8.32.

   Extracting the tar file creates a directory named `oswbb`, which contains all the directories and files that are associated with OSWbb, including the `startOSWbb.sh` script.
4. To enable the collection of `iostat` information for NFS volumes, edit the `OSWatcher.sh` script in the `oswbb` directory, and set the value of `nfs_collect` to `1` as follows:

   ```
   nfs_collect=1
   ```

Running OSWbb

To start OSWbb, run the `startOSWbb.sh` script from the `oswbb` directory.

```
sudo ./startOSWbb.sh [frequency duration]
```

The optional frequency and duration arguments specify how often in seconds OSWbb collects data and the number of hours for which OSWbb runs. The default values are 30

seconds and 48 hours. The following example starts OSWbb recording data at intervals of 60 seconds, and has it record data for 12 hours:

```
sudo ./startOSWbb.sh 60 12


...
Testing for discovery of OS Utilities...
VMSTAT found on your system.
IOSTAT found on your system.
MPSTAT found on your system.
IFCONFIG found on your system.
NETSTAT found on your system.
TOP found on your system.

Testing for discovery of OS CPU COUNT
oswbb is looking for the CPU COUNT on your system
CPU COUNT will be used by oswbba to automatically look for cpu problems

CPU COUNT found on your system.
CPU COUNT = 4

Discovery completed.

Starting OSWatcher Black Box v7.3.0  on date and time
With SnapshotInterval = 60
With ArchiveInterval = 12
...
Data is stored in directory: OSWbba_archive

Starting Data Collection...

oswbb heartbeat: date and time
oswbb heartbeat: date and time + 60 seconds
...
```

In the previous output, *OSWbba_archive* is the path of the archive directory that contains the OSWbb log files.

To stop OSWbb prematurely, run the `stopOSWbb.sh` script from the `oswbb` directory:

```
sudo ./stopOSWbb.sh
```

OSWbb collects data in the directories that are under the `oswbb/archive` directory, which are described in the following table.

Directory	Description
`oswifconfig`	Contains output from `ifconfig`.
`oswiostat`	Contains output from `iostat`.
`oswmeminfo`	Contains a listing of the contents of `/proc/meminfo`.

Directory	Description
oswmpstat	Contains output from mpstat.
oswnetstat	Contains output from netstat.
oswprvtnet	If you have enable private network tracing for RAC, contains information about the status of the private networks.
oswps	Contains output from ps.
oswslabinfo	Contains a listing of the contents of /proc/slabinfo.
oswtop	Contains output from top.
oswvmstat	Contains output from vmstat.

OSWbb stores data in hourly archive files, which are named *system_name_utility_name_timestamp*.dat. Each entry in a file is preceded by a timestamp.

Analyzing OSWbb Archived Files

You can use the OSWbb analyzer (OSWbba) to provide information about system slowdowns, system delays, and other performance problems. You can also use OSWbba to graph data that is collected from the iostat, netstat, and vmstat utilities. OSWbba requires that you have Java version 1.4.2 or a later version installed on the system.

You can download a Java RPM for Linux by visiting http://www.java.com, or you can install Java by using the dnf command:

```
sudo dnf install java-1.8.0-jdk
```

Run OSWbba from the oswbb directory as follows:

```
sudo java -jar oswbba.jar -i OSWbba_archive
```

In the previous command, *OSWbba_archive* is the path of the archive directory that contains the OSWbb log files.

You can use OSWbba to display the following types of performance graph:

- Process run, wait and block queues.
- CPU time spent running in system, user, and idle mode.
- Context switches and interrupts.
- Free memory and available swap.
- Reads each second, writes each second, service time for I/O requests, and percentage usage of bandwidth for a specified block device.

You can also use OSWbba to save the analysis to a report file, which reports instances of system slowdown, spikes in run queue length, or memory shortage, describes probable causes, and offers suggestions of how to improve performance.

```
sudo java -jar oswbba.jar -i OSWbba_archive -A
```

For more information about OSWbb and OSWbba, refer to the OSWatcher Black Box User Guide (Article ID 301137.1) and the OSWatcher Black Box Analyzer User Guide (Article ID 461053.1) on My Oracle Support (MOS) at https://support.oracle.com.

4
Working With Performance Co-Pilot

Performance Co-Pilot (PCP) collects OS and network metrics that you can use to diagnose performance issues. PCP provides a monitor host that you can use to send requests for metrics and logs to a pair of collector host services that are installed on each Oracle Linux system that you monitor.

Installing PCP

1. Enable the `ol8_appstream` and `ol8_addons` yum repositories on the system.

   For more information, see Oracle Linux: Managing Software on Oracle Linux.

2. Install the `pcp-oracle-conf`, `pcp`, `pcp-system-tools`, and `pcp-gui` packages by using the `dnf` command:

   ```
   sudo dnf install pcp-oracle-conf pcp-system-tools pcp-gui
   ```

3. Enable and start the Performance Metrics Collector Daemon (`pmcd`) and Performance Metrics Logger (`pmlogger`) collector host services:

   ```
   sudo systemctl enable --now pmcd pmlogger
   ```

Stopping PCP

To temporarily halt data collection, stop the Performance Metrics Collector Daemon (`pmcd`) and Performance Metrics Logger (`pmlogger`) collector host services:

```
sudo systemctl stop pmcd pmlogger
```

To halt data collection for an indefinite period and ensure that that they don't start again automatically when the system boots, fully disable them:

```
sudo systemctl disable --now pmcd pmlogger
```

For more information about masking and unmasking services to prevent scripts from restarting disabled system services, see Oracle Linux 8: Managing Core System Configuration.

Reviewing Information Gathered by PCP

If the `pcp-oracle-conf` package is installed then the only metrics collected by the `pmlogger` service are those listed in the `/var/lib/pcp/config/pmlogger/config.ora` configuration file.

If PCP has been installed without the `pcp-oracle-conf` package, review the `/var/lib/pcp/config/pmlogger/config.default` configuration file instead.

You can modify the frequency with which those metrics are collected in the same configuration file. For example, to increase the frequency from each minute to every 5 seconds, revise the file as follows:

```
...
# It is safe to make additions from here on ...
#

log mandatory on every 5 seconds {
  filesys.free
  filesys.used
  ...
}
```

All of the archives that the `pmlogger` service generates are stored in the `/var/log/pcp/pmlogger/`*`hostname`* directory. For more information, see the `pmlogconf(1)` manual page.

To verify the PCP configuration at the time that `pmlogger` collected specific performance metrics, use the `pcp` command:

```
sudo pcp -a 20220321.0.xz
```

Using PCP Monitor Host to Analyze Performance Metrics

Different commands are available for viewing data about the system's performance. The commands you use depend on whether you want to view the data in real time or from the logs. The following sections provide sample commands to display performance information.

For more information about the commands and parameters to use to view performance metrics, see their respective manual pages. See also https://pcp.io/docs/guide.html for additional examples of how to display the system's different performance metrics.

Review Live Performance Metrics in Real Time

To monitor all the outgoing metrics from the `eth0` network interface in real time, use the `pmrep` command:

```
sudo pmrep -i eth0 -v network.interface.out
```

To monitor live hard drive operations for each partition with a two second interval, use the `pmval` command:

```
sudo pmval -t 2sec -f 3 disk.partitions.write
```

Review Recorded Performance Metrics

All of the archives that the `pmlogger` service generates are stored in the `/var/log/pcp/pmlogger/`*`hostname`* directory. Navigate to this directory to review the archived performance metrics.

To review the data for specific performance metrics within a specified time span, use the `pmdumptext` command. For example, to review resource usage metrics for CPU load, memory usage and disk write operations between 13:00 and 14:00 on a specific date:

```
sudo pmdumptext -Xlimu -t 10m -S @13:00 -T @14:00 'kernel.all.load[1]'
'mem.util.used' 'disk.partitions.write' -a 20220321.0.xz
```

The `pmstat` command can provide system performance metrics in a format similar to that produced by the `sar` command. For example, to review performance metrics averaged over 10 minute interval between 09:00 and 10:00 on a specific date:

```
sudo pmstat -t 10m -S @09:00 -T @10:00 -a 20220321.0.xz
```

To compare the metrics between two time periods, use the `pmdiff` command. For example, to compare the metrics between 02:00 and 03:00 on one day to the metrics between 09:00 and 10:00 on a different day:

```
sudo pmdiff -S @02:00 -T @03:00 -B @09:00 -E @10:00 20220321.0.xz
20220320.0.xz
```

Review Details About Recorded Performance Metrics

To review detailed information about a specific metric, use the `pminfo` command. For example, to review details about free memory:

```
sudo pminfo -df mem.freemem -a 20220321.0.xz
```

Validate System Status When Performance Metrics Were Captured

To verify the host, timezone and time period that an archive containing performance metrics contains, use the `pmdumplog` command:

```
sudo pmdumplog -L 20220321.0.xz
```

To review a list of every enabled performance metric, use the `pminfo` command:

```
sudo pminfo -a 20220321.0.xz
```

Running dstat With Performance Co-Pilot

The `dstat` utility in previous Oracle Linux releases is no longer being actively developed. Instead, it is implemented with Performance Co-Pilot (PCP) to perform basically the same functionalities to diagnose system performance.

This tool is immediately available with the installation of the PCP packages as described in Installing PCP.

To obtain information on the usage of the new `dstat` utility, type:

```
pcp dstat -h


Usage: pcp-dstat [-afv] [options...] [delay [count]]
Versatile tool for generating system resource statistics

Dstat options:
  -c, --cpu             enable cpu stats
     -C 0,3,total          include cpu0, cpu3 and total
  -d, --disk            enable disk stats
     -D total,sda          include sda and total
...
```

By default, running the command without any other options shows you statistics about CPU, disk, network, page, and system use.

```
pcp dstat


You did not select any stats, using -cdngy by default.
----total-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai stl| read  writ| recv  send|  in   out | int   csw
  0   0 100   0   0|   0     0 | 198B  719B|   0     0 | 156   254
  0   0 100   0   0|   0    12k|  66B  302B|   0     0 | 160   264
  0   0  99   0   0|   0     0 | 132B  384B|   0     0 | 136   219
...
```

As with the previous iteration of the tool, `pcp dstat` generates a running list of metrics or statistics in real time. To stop the process, type Ctrl-c.

You can narrow the information output by using different options that are available for the command. For example, to display the statistics only of CPU 1, you would type:

```
pcp dstat -c -C 1,total


-----cpu1-usage---------total-usage----
usr sys idl wai stl:usr sys idl wai stl
  0   0 100   0   0:  0   0 100   0   0
  1   0 100   0   0:  0   0  99   0   0
```

```
  0   0 100   0   0:  0   0 100   0   0
...
```

Similarly, to display only network statistics of a specific interface, such as ens3, and including totals, you would type:

```
pcp dstat -n -N ens3,total
```

```
--net/ens3---net/total-
 recv  send: recv  send
  66B  350B:  66B  350B
  66B  190B:  66B  190B
  66B  198B:  66B  198B
  66B  198B:  66B  198B
...
```

To store any statistics that are being gathered into a file for later review, include the `-o` *outputfile* option in the command.

For example, to collect network statistics and save the information in a log, type:

```
pcp dstat -n -f -o /tmp/netstat-log
```

Use the `-f` option to review full information.

```
--net/ens3-----net/lo--
 recv  send: recv  send
  66B  358B:   0     0
  66B  174B:   0     0
  66B  190B:   0     0
 341B  419B:   0     0
  66B  190B:   0     0
  66B  190B:   0     0
  66B  190B:   0     0 ^C
```

Thereafter, to view the log, type:

```
cat /tmp/netstat-log
```

```
...
"Host:","hostname",,,,"User:","user"
"Cmdline:","pcp-dstat -n -f -o /tmp/netstat-log",,,,"Date:","date"
"net/ens3",,"net/lo",
"net/ens3:recv","net/ens3:send","net/lo:recv","net/lo:send"
65.934,357.641,0,0
66.000,173.999,0,0
66.000,190.001,0,0
340.992,418.991,0,0
66.001,190.004,0,0
```

```
66,190,0,0
66.000,189.999,0,0
```

For more information about `pcp dstat`, see the `pdp-dstat(1)` manual page.

5
Working With TuneD

The TuneD utility is a tool for monitoring a system so you can optimize its performance under certain conditions. The utility consists of the following main components:

- TuneD profiles
- The `tune-adm` command

The basic TuneD feature is provided by the `tuned` package. Depending on where Oracle Linux is running, this package might be automatically installed with the OS, such as on an Oracle Linux instance on Oracle Cloud Infrastructure.

To verify whether TuneD is already in the system, type:

```
dnf list tuned


Installed Packages
tuned.noarch     version
...
```

If the feature is not in the system, do the following:

1. Install the `tuned` package.

   ```
   sudo dnf install tuned
   ```

2. Enable the `tuned` service.

   ```
   sudo systemctl enable --now tuned
   ```

To list other TuneD packages that are available but not installed in the system, type:

```
sudo dnf list tuned-profiles*
```

If you require other TuneD packages, you would need to install these separately.

About TuneD Profiles

TuneD performs system optimization by working with *TuneD profiles*. TuneD profiles consist of predefined sets of optimization rules that apply tuning tasks under different use cases.

After the appropriate package is installed, you can list the available profiles with the following command:

```
tuned-adm list
```

The output of this command depends on where Oracle Linux is running. On physical and virtual systems, the available profiles might resemble the following:

```
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
- hpc-compute                 - Optimize for HPC compute workloads
...
```

On an Oracle Linux instance that is running in Oracle Cloud Infrastructure, the list would include additional profiles:

```
...
- oci-busy-polling            - Enable Busy Polling conditionally in
OCI
- oci-cpu-power               - Set processor power management
parameters in OCI
- oci-nic                     - Increase combined channels to 16 on
NICs with bnxt_en driver on BM shapes in OCI
- oci-rps-xps                 - Enable RPS/XPS conditionally in OCI
...
```

The following are selected profiles that TuneD uses to optimize the system. The list is not exhaustive.

- `balanced`: Provides a balance between performance and power consumption. The profile uses automatic scaling and automatic tuning when possible. A possible drawback is increased latency.
- `powersave`: Provides maximum power saving performance. The profile can minimize actual power consumption by throttling performance.

  > **Note:**
  >
  > In some instances, the `balanced` profile is more efficient than the `powersave` profile and therefore, a better choice.

- `throughput-performance`: Disables power-savings mechanisms and enables `sysctl` settings to improve the throughput performance of the disk and network IO.
- `latency-performance`: Optimized for low latency by disabling power-savings mechanisms and enabling `sysctl` settings to improve latency.
- `network-latency`: Provides low latency network tuning and is based on the `latency-performance` profile. In addition, this profile disables transparent huge pages and NUMA balancing and tunes several network-related `sysctl` settings.
- `network-throughput`: Used for optimizing throughput network tuning, based on the `throughput-performance` profile. In addition, this profile increases kernel network buffers.

- `virtual-guest`: Designed for virtual guests and is based on the `throughput-performance` profile. This profile decreases virtual memory `swappiness` values and increases disk `readahead` values.
- `virtual-host`: Designed for virtual hosts and is based on the `throughput-performance` profile. This profile decreases virtual memory `swappiness` values, increases disk `readahead` values, and sets a more aggressive value for dirty pages `writeback`.
- `desktop`: Optimized for desktop environments and is based on the `balanced` profile. In addition, this profile sets scheduler `autogroups` for better response of interactive applications.

About Configuration Files for TuneD Profiles

Profiles are automatically stored in the following locations:

- `/usr/lib/tuned` contains the predefined profiles.
- `/etc/tuned` contains custom profiles in addition to other files that are installed by default.

TuneD profiles direct how optimization is performed on the system's resources and components. The profiles are predefined and fall under one of two general categories: *power-saving profiles* and *performance-boosting profiles*.

For example, the `/usr/lib/tuned/desktop` profile optimizes desktop-related resources, while `/usr/lib/tuned/powersave` optimizes the system's power consumption.

Profiles operate by using optimization rules that have been set for the profiles. Each profile's rules are contained in a corresponding `tuned.conf` file. Thus, for the `desktop` profile, the rules are defined in `/usr/lib/tuned/desktop/tuned.conf`, while the rules for the `powersave` profile are defined in `/usr/lib/tuned/powersave/tuned.conf`.

As an example, the following shows the contents of the configuration file for the `desktop` profile:

```
less /usr/lib/tuned/desktop/tuned.conf


#
# tuned configuration
#

[main]
summary=Optimize for the desktop use-case
include=balanced

[sysctl]
kernel.sched_autogroup_enabled=1
```

In these configuration files, you can modify the rules for that profile or customize how profiles optimize specific devices. In addition, you can configure TuneD so that any changes in device usage triggers an adjustment in the current settings. By adjusting the configuration file definitions, you can customize and improve optimization in your specific systems.

To create a custom profile, copy the `/usr/lib/tuned/profile` directory to the `/etc/tuned` directory. Then modify the profile's `tuned.conf`.

For more information about profile configuration, see the `tuned.conf(5)` manual page.

Using the tuned-adm Command

To avail of the features in the TuneD utility, use the `tuned-adm` command. The following tasks describe how to administer TuneD profiles and the `tuned` service in Oracle Linux.

For more information, see the `tuned-adm(8)` and `tuned(8)` manual pages.

Listing Available and Active Profiles in the System

To list the profiles in the system, type:

```
sudo tuned-adm list
```

The number of profiles that is displayed depends on whether you are using an Oracle Linux system or an Oracle Linux instance in Oracle Cloud Infrastructure.

The output always ends by displaying the profile that is currently in use. On an Oracle Linux instance in Oracle Cloud Infrastructure, for example, the final line would be similar to the following:

```
...
Current active profile: oci-rps-xps oci-busy-polling oci-cpu-power oci-
nic
```

On a physical system or a virtual machine, the default active profile might be different.

To directly query which profile is active in the system or instance without listing all available profiles, type:

```
sudo tuned-adm active
```

Verifying That the System is Optimized According to the Profile

You might want to verify that the system has been optimized to match the settings as defined in the active profile. Follow these steps:

1. Run the `verify` subcommand.

   ```
   sudo tuned-adm verify
   ```

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

2. Optionally, examine the contents of the TuneD log for more detailed information.

```
less /var/log/tuned/tuned.log
```

Querying for a Recommended Profile

Provided that the `recommend` parameter is enabled in `/etc/tuned/tuned-main.conf`, TuneD can suggest an optimization profile that is more suitable, for example:

```
sudo tuned-adm recommend


balanced
```

Selecting and Activating Profiles

When `tuned` is enabled on a system or instance, a default TuneD profile becomes automatically active. However, you can select a different profile and activate it so that system optimization is performed based on that profile. Use the following syntax:

```
sudo tuned-adm profile profile1 [profile2 profile3 ...]
```

The following is a suggested method of changing an active profile:

1. Check the current active profile.

```
sudo tuned-adm active


balanced
```

2. Optionally, list the available profiles in the system.

```
sudo tuned-adm list
```

3. Select a different profile to activate.

```
sudo tuned-adm profile powersave
```

4. Verify the change.

```
sudo tuned-adm active


powersave
```

> **Caution:**
>
> TuneD might activate multiple profiles. Where profile rules conflict, TuneD applies the settings of the last profile that you specified in the command.
>
> When profiles are merged, TuneD can't determine whether the combination is logical or not. Consequently, parameters in those profiles that are related to the performance of a common component or resource risk being tuned in opposite ways. Merging profiles does not always guarantee that better optimization is obtained.
>
> Oracle Linux instances on Oracle Cloud Infrastructure are an exception, where multiple `oci-*` profiles are automatically activated when TuneD is installed.

Disabling TuneD

To disable TuneD temporarily, switch the TuneD service off. Switching the service off removes active profiles and effectively stops all tuning operations in the system. To resume tuning, activate a profile. The following steps illustrate how to halt system tuning.

1. Optionally, note down the current active profile.

   ```
   sudo tuned-adm active
   ```

2. Switch TuneD off.

   ```
   sudo tuned-adm off
   ```

3. Verify the status of system tuning.

   ```
   sudo tuned-adm active


   No current active profile.
   ```

To resume system tuning, do the following:

1. Determine what profile to reactivate.
   - To identify and then use the default profile that has been predefined for the current system environment or instance, type:

     ```
     sudo tuned-adm recommend
     ```

   - To select a profile other than the default profile, display the available profiles. Type:

     ```
     sudo tuned-adm list
     ```

   - To reuse the same profile that was active before TuneD was switched off, consult notes you have taken based on the previous steps to switch off the TuneD service.

2. Activate the selected profile.

```
sudo tuned-adm profile profile
```

To disable system tuning permanently, stop the `tuned` service.

```
sudo systemctl disable --now tuned
```

Applying Global Settings to the TuneD Utility

The `/etc/tuned/tuned-main.conf` regulates the entire `tuned` service through defined global settings. The following is an extract of the file:

```
# Global tuned configuration file.

# Whether to use daemon. Without daemon it just applies tuning. It is
# not recommended, because many functions don't work without daemon,
# e.g. there will be no D-Bus, no rollback of settings, no hotplug,
# no dynamic tuning, ...
daemon = 1

# Dynamicaly tune devices, if disabled only static tuning will be used.
dynamic_tuning = 0

# How long to sleep before checking for events (in seconds)
# higher number means lower overhead but longer response time.
sleep_interval = 1

# Update interval for dynamic tunings (in seconds).
# It must be multiply of the sleep_interval.
update_interval = 10

# Recommend functionality, if disabled "recommend" command will be not
# available in CLI, daemon will not parse recommend.conf but will return
# one hardcoded profile (by default "balanced").
recommend_command = 1
...
```

The following are sample cases for which you can configure parameters in `/etc/tuned/tuned-main.conf`:

- Selecting between static or dynamic

  With the `dynamic_tuning` parameter, you can choose whether system tuning is static or dynamic.

  By default, static tuning is operative in the system. Static tuning applies settings that have been predefined for `sysctl` and `sysfs` commands, or those that are set for configuration tools at the moment these tools are activated. Thereafter, no additional tuning is performed.

  Dynamic tuning instead is performed continuously. TuneD monitors the system at intervals throughout the system's up time. Based on the information gathered at a specific interval, TuneD optimizes the system. The interval at which TuneD monitors and

optimizes components is determined by the value of the `uptime_interval`, which by default is set to 10 seconds.

- Selecting between daemon or no-daemon mode

  With the `daemon` parameter, you can set the mode for system tuning.

  By default, the functionalities of TuneD are active if the daemon is running. If TuneD is switched to run with the daemon disabled, then TuneD applies the profile settings and then exits. This mode is not recommended because some TuneD functionalities, such as compatibility with DBus, hotplugging, and rollback of settings, are absent if the daemon is disabled.

6
Automating System Tasks

You can automate tasks to perform periodic backups, monitor the system, run custom scripts, as well as other administrative tasks. In Oracle Linux, the two utilities that are used for job scheduling are `cron` and `anacron`. Both tools enable you to automate the running of tasks, also referred to as *jobs*, but slightly differ in how the tasks are run. Both utilities automatically run through their respective daemons, and so, you do not need to run these utilities manually.

You can also use `systemd` timer unit files for scheduling tasks. All of the utilities described in this document for task automation can work in combination.

Working With cron

With `cron`, you can schedule jobs to run as often as every minute. System `cron` jobs are defined in the `cron` table called `/etc/crontab` or in files in the `/etc/cron.d` directory. User defined jobs are stored in the `/var/spool/cron` directory and are named after their users, for example, `/var/spool/cron/jsmith`. The `crond` daemon, which uses the `cron` utility to run scheduled jobs, looks in these locations to determine which jobs need to be run.

If the daemon identifies a job that was configured to run in the current minute, then the `crond` daemon runs that job as the owner of the job definition. If the job is a system `cron` job, then the daemon runs the job as the user that is specified in the job definition, provided that the user is defined.

If the system is down when a job is scheduled to run, then when the system is restarted, the daemon bypasses that job until the job's next scheduled run.

For a tutorial on how to work with the `cron` utility, see Use the Crontab Utility to Schedule Tasks on Oracle Linux.

About the cron Table Fields

The contents of `/etc/crontab` typically consist of definitions for the `SHELL`, `PATH`, `MAILTO`, and `HOME` variables for the environment in which the jobs run. These definitions are then followed by the job definitions themselves. Comment lines start with a `#` character.

A `/etc/crontab` file without any configured job appears as follows:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
```

```
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

Job definitions consist of information that you specify in the appropriate fields as follows:

minute
Specify a value of 0-59.

hour
Specify a value of 0-23.

day
Specify a value of 1-31.

month
Specify a value of 1-12 or `jan`, `feb`,..., `dec`.

day-of-week
Specify a value of 0-7 (Sunday is 0 or 7) or `sun`, `mon`,...,`sat`.

user
Specify the user running the command; or, you can specify an asterisk (`*`), which indicates the owner of the `crontab` file.

command
Specify the shell script or command to be run.

For the *minute* through *day-of week* fields, you can use the following special characters:

*****
Specify an asterisk (`*`) for all of the valid values for the field.

-
Specify a dash (`-`) to indicate a range of integers, for example, `1-5`.

,
Specify a list of values, separated by commands (`,`), for example, `0,2,4`.

/
Specify a step value by using the slash (`/`), for example, `/3` in the *hour* field. This entry is interpreted as every three hours.

For example, the following entry would run a command every five minutes on weekdays:

```
0-59/5  *  *  *  1-5  *  command
```

Run a command at one minute past midnight on the first day of the months April, June, September, and November:

```
1  0  1  4,6,9,11  *  *  command
```

> **Note:**
>
> If you add an executable job script to the `/etc/cron.hourly` directory, `crond` runs the script every hour.

All jobs in the `/etc/crontab` file are run as root.

For more information, see the `crontab(5)` manual page.

Creating a cron Job

Any user can create a `cron` job, but the location of the job definition depends on the user's privileges.

- An administrator who signs in as `root` creates jobs that are stored in `/etc/crontab`. Jobs in `/etc/crontab` are run as root, unless the job definition specifies a different user.
- A user with administrator privileges can create `cron` jobs. The jobs are stored in `/etc/crontab.d/` and named after the administrator's username.
- A regular can create jobs which are stored in `/etc/crontab.d/`. The job file is based after the user's name.

To create or edit a `crontab` file as a user, such as `jsmith`, do the following:

1. Sign in to the system as that user, for example `jsmith`.
2. Edit `crontab` with a text editor.

   ```
   crontab -e
   ```

   If you want to use a specific text editor to create or edit a `cron` job, use the following syntax:

   ```
   env EDITOR=text-editor crontab -e
   ```

3. When the editor opens, create the cron job by using the format as described in About the cron Table Fields.

   Suppose that you want to define a backup job that you want to run every 15 minutes. Further, you created a script `mybackup.sh` for this task in the user home directory. You would then create the schedule as follows:

   ```
   15 * * * * /home/jsmith/mybackup.sh
   ```

4. Save the file and exit.

   The file is saved as `/var/spool/cron/jsmith`.

5. View and verify the contents of the new cron job.

```
crontab -l


15 * * * * /home/jsmith/mybackup.sh
```

To delete the user `crontab` file, type:

```
crontab -r
```

For more information, see the `crontab(5)` manual page.

Controlling Access to Running cron Jobs

The following files, each of which contains usernames, manage access control for running cron jobs:

- `/etc/cron.allow` contains the list of users who are permitted to run `cron` jobs.
- `/etc/cron.deny` contains the list of users who are not permitted to run `cron` jobs.

If both files exist, then `/etc/cron.allow` takes precedence. Users in this list are permitted to run `cron` jobs, and `/etc/cron.deny` is ignored. If only `/etc/cron.deny` exists, then only those users that are not on this list can run `cron` jobs.

If none of these two files exists, then only `root` can run `cron` jobs.

Configuring anacron Jobs

The `anacron` utility differs with the `cron` utility in that `anacron` limits the number of times a scheduled job can be run to only daily. The `anacron` utility is mainly intended for use on laptop computers.

If `anacron` is not already running and the system is connected to mains and not battery power, `crond` starts `anacron`.

The `crond` daemon runs the `/etc/cron.hourly/0anacron` script as `root` each hour according to the schedule in `/etc/cron.d/0hourly`, which has the following job definition:

```
# Run the hourly jobs
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
01 * * * * root run-parts /etc/cron.hourly
```

Then, based on the configuration settings in `/etc/anacrontab`, the `0anacron` script processes the contents in the `/etc/cron.daily`, `/etc/cron.weekly`, and `/etc/cron.monthly` directories.

If a scheduled job has not been run because of system downtime, then that job runs when the system restarts.

System `anacron` jobs are defined in `/etc/anacrontab` as follows:

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days  delay in minutes  job-identifier   command
1               5                 cron.daily       nice run-parts /etc/
cron.daily
7               25                cron.weekly      nice run-parts /etc/
cron.weekly
@monthly        45                cron.monthly     nice run-parts /etc/
cron.monthly
```

The top of the file contains definitions for the `SHELL`, `PATH`, `MAILTO`, `RANDOM_DELAY`, and `START_HOURS_RANGE` variables for the environment in which the jobs run, followed by the job definitions themselves. Comment lines start with a `#` character.

`RANDOM_DELAY` is the maximum number of random time in minutes that `anacron` adds to the *delay* parameter for a job. The default minimum delay is 6 minutes. The random offset is intended to prevent `anacron` overloading the system with too many jobs at the same time.

`START_HOURS_RANGE` is the time range of hours during the day when anacron can run scheduled jobs.

The bottom part of the file contains job definitions. Each job consists of entries that are spread across 4 columns under the following headings:

period
Frequency of job execution specified in days or as `@daily`, `@weekly`, or `@monthly` for daily, weekly, or monthly.

delay
Number of minutes to wait before running a job.

job-id
Unique name for the job in log files.

command
The shell script or command to be run.

By default, `anacron` runs jobs between 03:00 and 22:00 and randomly delays jobs by between 11 and 50 minutes. The job scripts in `/etc/cron.daily` run between 03:11 and 03:50 every day if the system is running, or after the system is booted and the time is earlier than 22:00. The `run-parts` script sequentially runs every program within the directory specified as its argument.

Scripts in `/etc/cron.weekly` run weekly with a delay offset of between 31 and 70 minutes.

Scripts in `/etc/cron.monthly` run monthly with a delay offset of between 51 and 90 minutes.

For more information, see the `anacron(8)` and `anacrontab(5)` manual pages.

Running One-Time Tasks

Through the `atd` service, you can use certain commands to schedule one-time tasks.

This section describes the use of the `at` and `batch` commands for this purpose. Before you can use these commands, ensure that the `at` service is running.

```
sudo systemctl is-active atd
```

- To schedule a task to run one time only at a specified tine, use the `at` command.

  Suppose that you have defined a job in `~/atjob`. To schedule that job in 20 minutes time, you would type:

  ```
  at now + 20 minutes < ~/atjob
  ```

  ```
  job 1 at 2021-03-19 11:25
  ```

- To schedule a batch job to run when the system load average is light, use the `batch` command.

  Suppose that you have defined a batch job in `~/batchjob`. To schedule this job to run provided that the system load average is less than 0.8, you would type:

  ```
  sudo batch < batchjob
  ```

  ```
  job 2 at 2013-03-19 11:31
  ```

  > **Note:**
  >
  > The system load average threshold under which you can schedule user-defined batch jobs to run is 0.8, by default. However, that value can vary. See Changing the Behavior of Batch Jobs.

- To list all the scheduled one-time jobs that are in queue, type:

  ```
  sudo atq
  ```

  ```
  job 1 at 2021-03-19 11:25
  job 2 at 2013-03-19 11:31
  ```

- To cancel one or more queued jobs, specify their job numbers to the `atrm` command, for example:

  ```
  sudo atrm 2
  ```

For more information, see the `at(1)` manual page.

Changing the Behavior of Batch Jobs

The system load average represents the average number of processes that are queued to run on the CPUs or CPU cores over time. Typically, a system might not be considered overloaded until the load average exceeds 0.8 times the number of CPUs or CPU cores. On such systems, you would usually want `atd` to be able to run batch jobs when the load average drops to less than the number of CPUs or CPU cores, rather than the default limit of 0.8. For example, on a system with 4 CPU cores, you could set the load-average limit over which `atd` can't run batch jobs to 3.2.

If you know that a batch job typically takes more than a minute to run, you can also change the minimum interval that `atd` waits between starting batch jobs. The default minimum interval is 60 seconds.

For more information about monitoring CPU usage and to display the system load average, see Monitoring CPU Usage.

To change the load-average limit and minimum interval time for batch jobs:

1. Open the `/etc/sysconfig/atd` configuration file with a text editor.
2. Uncomment the line that defines the `OPTS` variable.
3. Provide new values for the load average limit and the minimum interval time to the OPTS variable, for example:

   ```
   OPTS="-b 100 -l 3"
   ```

   This example sets the minimum interval to 100 seconds and the load-average limit to 3.
4. Restart the `atd` service:

   ```
   sudo systemctl restart atd
   ```

5. Verify that the `atd` daemon is running with the new minimum interval and load-average limit, for example:

   ```
   sudo systemctl status atd
   ```

   ```
   atd.service - Job spooling tools
      Loaded: loaded (/usr/lib/systemd/system/atd.service; enabled)
      Active: active (running) since Mon 2014-04-28 15:37:04 BST; 2min 53s
   ago
    Main PID: 6731 (atd)
      CGroup: /system.slice/atd.service
              └─6731 /usr/sbin/atd -f -b 100 -l 3

   Apr 28 15:37:04 localhost.localdomain systemd[1]: Started Job spooling
   tools.
   ```

For more information, see the `systemctl(1)` and `atd(8)` manual pages.

Working With Systemd Timers

Timer unit files are a type of `systemd` file that the `systemctl` utility uses to schedule tasks, similar to the `cron` utility that uses `crontab` and other `cron` jobs for the same purpose.

Typically, packages that use specific services to function in the system include their own `systemd` timer unit files. Thus, when these packages are installed with Oracle Linux, the timer unit files are automatically included. You can display with the timer files in the system with the following command:

```
systemctl list-unit-files --type=timer
```

> **Note:**
>
> The list of timer files might differ depending on where Oracle Linux is running, such as in an instance in Oracle Cloud Infrastructure, a physical system, and so on.

Each timer unit file contains parameter settings that manage the schedule of a task. For example, the schedule for running `dnf-makecache.service` is set in the `dnf-makecache.timer` file. The file contains the following settings:

```
systemctl cat dnf-makecache.timer


# /usr/lib/systemd/system/dnf-makecache.timer
[Unit]
Description=dnf makecache --timer
ConditionKernelCommandLine=!rd.live.image
# See comment in dnf-makecache.service
ConditionPathExists=!/run/ostree-booted
Wants=network-online.target

[Timer]
OnBootSec=10min
OnUnitInactiveSec=1h
RandomizedDelaySec=60m
Unit=dnf-makecache.service

[Install]
WantedBy=timers.target
```

The schedule information is specified under the `[Timer]` section. In the sample configuration, the `dnf-makecache.service` service is set to automatically run 10 minutes after the system is booted. The service then goes into idle mode for an hour, as specified by the `OnUnitInactiveSec` parameter. At the end of the hour, the service runs again. This cycle continues every hour indefinitely.

The `RandomizedDelaySec` setting provides a value limit for how much a run can be delayed beyond its schedule. In the example, the service is permitted to run one minute later than its schedule at the very latest. This parameter is useful for preventing too many jobs that start at the same time on a specified schedule, which would otherwise risk overloading the resources.

`OnCalendar` is another useful parameter for task scheduling. Suppose that the parameter is set as follows:

```
OnCalendar=*:00/10
```

The `*:00` indicates every hour at the top of the hour, while the `/10` setting indicates 10 minutes. Therefore, the job is set to run hourly, at ten minutes past the top of the hour.

For a complete list of `systemd` timer unit file parameters for scheduling a job, see the `systemd.timer(5)` manual pages.

For a tutorial on how to use `systemd` in Oracle Linux, including how to configure `systemd` timer unit files, see Use systemd on Oracle Linux.

7

Configuring and Using Auditing

Auditing collects data at the kernel level that you can then analyze to identify unauthorized activity. Auditing collects data in greater detail than system logging does. The process of examining audit trails to locate events of interest can be significantly challenging. Therefore, consider automating this process.

Some of the definitions in the audit configuration file, `/etc/audit/auditd.conf`, include the following:

- Data retention policy
- Maximum size of the audit volume
- Action to take if the capacity of the audit volume is exceeded
- Locations of local and remote audit trail volumes

The default audit trail volume is `/var/log/audit/audit.log`. See the `auditd.conf(5)` manual page for more information.

By default, auditing captures specific events such as system logins, modifications to accounts, and `sudo` actions. You can configure auditing to capture detailed system call activity or modifications to certain files. The kernel audit daemon (`auditd`) records the events that you configure, including the event type, a timestamp, the associated user ID, and success or failure of the system call.

The entries in the audit rules file, `/etc/audit/audit.rules`, determine which events are audited. Each rule is a command line option that is passed to the `auditctl` command. Configure this file to match organization's security policy.

The following are examples of rules that you might set in the `/etc/audit/audit.rules` file:

- Record all unsuccessful exits from `open` and `truncate` system calls for files and store the information in the `/etc` directory hierarchy.

  ```
  -a exit,always -S open -S truncate -F /etc -F success=0
  ```

- Record all files opened by a user with UID 10.

  ```
  -a exit,always -S open -F uid=10
  ```

- Record all files that have been revised or whose attributes were changed by any user who originally signed in with a UID of 500 or greater.

  ```
  -a exit,always -S open -F auid>=500 -F perm=wa
  ```

- Record requests for write or for file attribute change access. Store the records in the `/etc/sudoers` file and tag such a record with the string `sudoers-change`.

  ```
  -w /etc/sudoers -p wa -k sudoers-change
  ```

- Record requests for write and for file attribute change access and store records in the `/etc` directory hierarchy.

  ```
  -w /etc/ -p wa
  ```

- Require a reboot after changing the audit configuration.

  ```
  -e 2
  ```

> **Note:**
>
> We recommend that you define rules to reboot at the end of the `/etc/audit/audit.rules` file.

For more examples of audit rules, see also the `auditctl(8)` and `audit.rules(7)` manual pages.

Stringent auditing requirements generate large amounts of audit data and can impose a significant performance overhead. Some site security policies stipulate that a system must shut down if events can't be recorded because the audit volumes have exceeded their capacity. Generally, we recommend that you direct audit data to separate file systems in rotation to prevent overspill and to facilitate backups.

If you tag audit records, then searching an audit volume with the `ausearch` command becomes easier by referring to those tags. For example, to examine records that are tagged with the string `sudoers-change`, you would type:

```
sudo ausearch -k sudoers-change
```

The `aureport` command generates summaries of audit data. For example, the following command generates a report that shows every sign-in event from 1 second after midnight on the previous day until the current time:

```
sudo aureport -l -i -ts yesterday -te now
```

You can set up `cron` jobs or `systemd` timers that run `aureport` periodically to generate reports of interest. See Use the Crontab Utility to Schedule Tasks on Oracle Linux and Use systemd on Oracle Linux.

See the `ausearch(8)` and `aureport(8)` manual pages for more information.

For a hands-on tutorial on using the auditing tools on Oracle Linux, see Audit Oracle Linux with Auditd.

Working With System Log files

The log files contain messages about the system, kernel, services, and applications. The `journald` logging daemon, which is part of `systemd`, records system messages in nonpersistent journal files in memory. The `journald` daemon forwards messages to the system logging daemon, `rsyslog`.

Files in `/run` are volatile. Thus, the log data is lost after a reboot unless you create the directory `/var/log/journal`. You can use the `journalctl` command to query the journal logs.

For more information, see the `journalctl(1)` and `systemd-journald.service(8)` manual pages.

For a hands-on tutorial introducing system logging tools, see System Logging on Oracle Linux.

About Logging Configuration (/etc/rsyslog.conf)

The `/etc/rsyslog.conf` file is the configuration file for the `rsyslogd` daemon. The file contains global directives, module directives, and rules for the daemon or service. By default, `rsyslog` processes and archives only `syslog` messages. However, if required, you can configure `rsyslog` to archive any other messages that `journald` forwards, including kernel, boot, `initrd`, `stdout`, and `stderr` messages.

To obtain the latest information about `rsyslog`, refer to https://www.rsyslog.com/doc/. Alternatively, you can install the `rsyslog-doc` package locally.

```
sudo dnf install -y rsyslog-doc
```

The documentation is installed in `/usr/share/doc/rsyslog/html`.

> **Important:**
>
> The format to configure parameters in `/etc/rsyslog.conf` has changed. The following formats are supported which enables backward compatibility with previous configuration:
>
> - Basic or `sysklogd` format, which has been used since the beginning of system logging.
> - Legacy format, where directives are defined on their own specific lines in the file, with each directive being preceded by the dollar (`$`) sign, such as `$MainMsgQueueSize`.
> - Advanced format, which uses the `RainerScript` scripting language for configuring `rsyslog`.
>
> For more information about these formats, see the relevant sections in https://www.rsyslog.com/doc/.

The `/etc/rsyslog.conf` file is divided into the following main sections:

Modules
Modules contain configuration parameters for processing messages. The processed or transformed messages can then be transmitted to various targets as required. Modules are classified into different categories, such as output, input, parser, library, and so on. For a complete list of these module classes, see the appropriate section in https://www.rsyslog.com/doc/. For a list of the modules, see the `rsyslog.conf(5)` manual page.

Modules enable different `rsyslog` functionalities to become operative, provided that those modules are loaded. Modules are loaded through the `module load` directive as follows:

```
module(load="module-name")
```

> **Note:**
>
> The directive uses the advanced format for loading a module and replaces the `$ModLoad module-name` legacy format.

Global directives
Global directives specify configuration options that apply to the `rsyslogd` daemon. A directive might specify the location of auxiliary files. A directive can also be a `module(load" ")` statement that applies global settings, such as the timestamp format to use for all messages, as shown in the following example:

```
module(load="builtin:omfile" Template=RSYSLOG_TraditionalFileFormat")
```

Because the module applies to all messages, the directive is specified under the Global Directives section.

Rules
Rules or rule sets determine how logged messages are managed.
A rule consists of two fields: a selector field and an action field. The two fields are separated by one more spaces or tabs.

- The selector field has two parts, separated by a period: a facility keyword and a priority keyword. Facility keywords include `auth`, `authpriv`, `cron`, `daemon`, `kern`, and so on. Priority keywords include `debug`, `info`, `notice`, `warning`, and so on. Thus, `kern.*` selects kernel messages of all priority levels, while `kern.emerg` selects emergency kernel messages only.

  For a list of both facility and priority selectors, see the `rsyslog.conf(5)` manual page.
- The action field typically indicates to which log file the message content is written. For example, the following rule indicates that `cron` messages are to be logged in `/var/log/cron`:

  ```
  cron.*   /var/log/cron
  ```

You can customize `rsyslog` configuration in two ways:

- Directly change the `/etc/rsyslog.conf` file itself.
- Create a configuration file and store it in `/etc/rsyslog.d`. You might choose this option to prevent your custom configurations from being overwritten in case system packages are updated.

Some changes are simple to implement on the `/etc/rsyslog.conf` file, such as changing the log for a specific selector. For example, to change the log for `cron` messages to `cron_new`, you would enter the following:

```
cron.*     /var/log/cron_new
```

You then restart the `rsyslog` service for the change to take effect.

Other changes might require additional parameter definitions and steps.

Suppose that you want to create a rule that would use TCP to forward messages to another server where the messages are logged. The following steps show you how to implement this sample rule.

1. Create a separate file, for example, `/etc/rsyslog.d/forwarding`, where you would set the parameters for TCP forwarding, similar to the following:

   ```
   *.* action(type="omfwd"
         queue.filename="fwdRule1"
         queue.maxdiskspace="1g"
         queue.saveOnShutdown="on"
         queue.type="linkedlist"
         action.resumeRetryCount="-1"
         target="remote-host.com" port="30514" protocol="tcp"
        )
   ```

   queue.filename
   Prefix to be attached to the backup files. The prefixed backup files are created in the location as specified by the `workDir` global directive, for example, `global(workDirectory="/var/log")`.

   queue.maxdiskspace
   Space limit for log files.

   queue.saveOnShutdown
   Saves data in memory if `rsyslog` shuts down.

   queue.type
   Enables a LinkedList in-memory queue.

   action.resumeRetryCount
   A setting of `-1` means to retry indefinitely if the host is unavailable.

   target
   Can be a host name or an IP address.

   Based on the sample configuration, `rsyslog` forwards messages to the remote server `remote-host.com`. The `rsyslog` service also keeps the message in memory in case the remote server is unavailable. If `rsyslog` shuts down or has exhausted allotted memory, then `rsyslog` creates files on disk with the appropriate prefix to the file names.

2. Open the `/etc/rsyslog.conf` and verify the following:

- Ensure that the module for TCP syslog reception is loaded. Verify that the comment marks are removed from the following lines:

  ```
  module(load="imtcp")
  input(type="imtcp" port="514")
  ```

- Ensure that the global directive to include `/etc/rsyslog.d` files in `rsyslog` configuration is enabled. Verify that the following line is not commented out:

  ```
  include(file="/etc/rsyslog.d/*.conf" mode="optional")
  ```

3. Save the file and exit.
4. Restart the `rsyslog` service.

   ```
   systemctl restart rsyslog
   ```

To manage the rotation and archival of the correct logs, edit `/etc/logrotate.d/syslog` so that it references each of the log files that are defined in the `RULES` section of `/etc/rsyslog.conf`. You can configure how often the logs are rotated and how many past copies of the logs are archived by editing `/etc/logrotate.conf`.

For more information, see the `logrotate(8)`, `logwatch(8)`, `rsyslogd(8)` and `rsyslog.conf(5)` manual pages.

Configuring Logwatch

Logwatch is a monitoring system that you can configure to report on areas of interest in the system logs. After you install the `logwatch` package, the `/etc/cron.daily/0logwatch` script runs every night and sends an email report to `root`. You can set local configuration options in `/etc/logwatch/conf/logwatch.conf` that override the main configuration file `/usr/share/logwatch/default.conf/logwatch.conf`, including the following:

- Log files to monitor, including log files that are stored for other hosts.
- Names of services to monitor, or to be excluded from monitoring.
- Level of detail to report.
- User to be sent an emailed report.

As best practice, configure Logwatch on your log server to monitor the logs for suspicious messages, and disable Logwatch on log clients. However, if you do use Logwatch, disable high precision timestamps by adding the following entry to the GLOBAL DIRECTIVES section of `/etc/rsyslog.conf` on each system:

```
module(load="builtin:omfile" Template=RSYSLOG_TraditionalFileFormat")
```

You can also run `logwatch` directly from the command line.

For more information, see the `logwatch(8)` manual page.

Using Process Accounting

The `psacct` package implements the process accounting service in addition to the following utilities that you can use to monitor process activities:

`ac`
Displays connection times in hours for a user as recorded in the `wtmp` file (by default, `/var/log/wtmp`).

`accton`
Turns on process accounting to the specified file. If you do not specify a file name argument, process accounting is stopped. The default system accounting file is `/var/account/pacct`.

`lastcomm`
Displays information about previously rund commands as recorded in the system accounting file.

`sa`
Summarizes information about previously rund commands as recorded in the system accounting file.

> **Note:**
>
> As for any logging activity, ensure that the file system has enough space to store the system accounting and `wtmp` files. Monitor the size of the files and truncate them as needed.

For more information, see the `ac(1)`, `accton(8)`, `lastcomm(1)`, and `sa(8)` manual pages.

8
Working With Kernel Dumps

The Kdump feature provides a kernel crash information dumping mechanism in Oracle Linux. The `kdump` service saves the contents of the system's memory for later analysis. The second kernel resides in a reserved part of the system memory.

Kdump uses the `kexec` system call to boot into the second kernel, called a *capture kernel*, without the need to reboot the system, and then captures the contents of the stopped kernel's memory as a crash dump (`vmcore`) and saves it. The `vmcore` crash dump can help with determining the cause of the malfunction.

Enabling the Kdump feature is highly recommended because a crash dump might be the only information source that is available if a system failure occurs. Kdump is vital in many mission-critical environments.

Before enabling Kdump, ensure that the system meets all of the memory requirements for using Kdump. To capture a kernel crash dump and save it for further analysis, reserve part of the system's memory permanently for that purpose. When you do so, that part of the system's memory is no longer be available to the main kernel.

The following table lists the minimum amount of reserved memory that is required to use Kdump, based on the system's architecture and the amount of available memory.

Table 8-1 Kdump Memory Requirements

Architecture	Available Memory	Minimum Reserved Memory
`x86_64`	1 GB to 64 GB	160 MB of RAM
	64 GB to 1 TB	256 MB of RAM
	1 TB and more	512 MB of RAM
Arm (aarch64)	2 TB and more	512 MB of RAM

For information about configuring Kdump by using the Cockpit web console, see Oracle Linux: Using the Cockpit Web Console

> **Note:**
>
> Kdump can also be used for troubleshooting in a cluster setup that uses the OCFS2 file system. For more information, see *Configuring the Behavior of Fenced Nodes With Kdump* in Oracle Linux 8: Managing Shared File Systems.

Installing Kdump

During an Oracle Linux interactive installation with the graphical installer, you have the option to enable Kdump and specify how much system memory is reserved for Kdump. The installer

screen is titled **Kdump** and is available from the main Installation Summary screen of the installer.

If you don't enable Kdump at installation time, or it is not enabled by default during an installation, as in the case of a custom kickstart installation, you can install and enable the feature by using the command line.

Before you install and configure Kdump by using the command line, ensure that the system meets all of the necessary memory specifications. For details, see Table 8-1.

1. Install the `kdump` package:

   ```
   sudo dnf install kexec-tools
   ```

2. With the proper administrative privileges, edit the `/etc/default/grub` file and set the `crashkernel=` option to the required value.

   For example, you would reserve 64 MB of memory as follows:

   ```
   crashkernel=64M
   ```

   You can also set the amount of reserved memory as a variable by using the following syntax: `crashkernel=`*`range1`*`:`*`size1`*`,`*`range2`*`:`*`size2`*.

   For example, you might set the memory as a variable as follows:

   ```
   crashkernel=512M-2G:64M,2G-:128M
   ```

3. (Optional) Consider defining an offset value for the reserved memory.

   Because the `crashkernel` reservation occurs very early, some systems require that you reserve memory with a certain fixed offset. When a fixed offset is specified, the reserved memory begins at that point. For example, you would reserve 128 MB of memory, starting at 16 MB as follows:

   ```
   crashkernel=128M@16M
   ```

   If no offset value is set, Kdump offsets reserved memory automatically.

4. Refresh the grub configuration to apply changes:

   ```
   sudo grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

5. Reboot the system and finish configuring Kdump.

   For instructions, see Configuring the Kdump Output Location.

6. Enable the `kdump` service:

   ```
   sudo systemctl enable --now kdump.service
   ```

Configuring Kdump

When you install and configure Kdump, the following files are modified:

- `/boot/grub2/grub.cfg`: Appends the `crashkernel` option to the kernel line to specify the amount of reserved memory and any offset value.
- `/etc/kdump.conf`: Sets the location in which the dump file can be written, the filtering level for the `makedumpfile` command, and the default behavior to take if the dump fails. See the comments in the file for information about the enabled parameters.

When you edit these files, you must reboot the system for the changes to take effect.

For more information, see the `kdump.conf(5)` manual page.

Configuring the Default Kdump Failure State

By default, if `kdump` fails to send its result to the configured outlook locations, it reboots the server. This action deletes any data that has been collected for the dump. To prevent this outcome, you can uncomment and change the `default` value in the `/etc/kdump.conf` file as follows:

```
default dump_to_rootfs
```

The `dump_to_rootfs` option attempts to save the result to a local directory, which can be particularly useful if a network share is unreachable. You can use `shell` instead to copy the data manually from the command line.

> **Note:**
>
> The `poweroff`, `restart`, and `halt`options are also valid for the default `kdump` failure state. However, performing these actions causes you to lose the collected data if those actions are performed.

Configuring the Kdump Output Location

After installing Kdump, you can define the location in which the resulting output is saved. For Oracle Linux, Kdump files are stored in the `/var/crash` directory by default.

To save the result to other locations, such as NFS mounts, externally mounted drives, and remote file servers, edit the `/etc/kdump.conf` file and remove the `#` comment character at the beginning of each line that you want to enable.

For example, to add a new directory location, prefix it with the `path` keyword:

```
path /usr/local/cores
```

Use `raw` to output directly to a specific device in the `/dev` directory. You can also manually specify the output file system for a particular device by using its label, name or UUID, for example:

```
ext4 UUID=5b065be6-9ce0-4154-8bf3-b7c4c7dc7365
```

Kdump files can also be transferred over a secure shell connection, as shown in the following example:

```
ssh user@example.com
sshkey /root/.ssh/mykey
```

You can also set the Kdump files to be exported to a compatible network share:

```
nfs example.com:/output
```

When you have finished configuring the output location for Kdump, enable the `kdump` service.

```
sudo systemctl enable --now kdump.service
```

Analyzing Kdump Output

You can use the `crash` utility to analyze the contents of `kdump` core dumps in a shell prompt, which is useful when troubleshooting problems.

1. Install the `crash` package:

   ```
   sudo dnf install crash
   ```

2. Identify the running kernel, for example:

   ```
   uname -r
   ```

   The output of the previous command is similar to the following:

   ```
   4.18.0-80.el8.x86_64
   ```

3. Provide the location of the kernel `debuginfo` module and the location of the core dump as parameters to the `crash` utility, for example:

   ```
   sudo crash /usr/lib/debug/lib/modules/4.18.0-80.el8.x86_64/vmlinux \
    /var/crash/127.0.0.1-2019-10-28-12:38:25/vmcore
   ```

   In the previous command, `4.18.0-80.el8.x86_64` is the running kernel and `127.0.0.1-2019-10-28-12:38:25` represents the *ipaddress-timestamp*.

4. Inside the `crash` shell, use the `help log` command for information about how to use the `log` command.

   You can also use the `bt`, `ps`, `vm`, and `files` commands to get more information about the core dump.

5. When you have finished analyzing the core dump, exit the shell.

For more detailed information about using the `crash` utility, see the `crash(8)` manual page.

Alternatively, you can use `drgn` to analyze core dumps. For more information, see Working With the drgn Kernel Debugging Utility.

Using Early Kdump

New in Oracle Linux 8, early Kdump loads the crash kernel and `initramfs` early enough to capture `vmcore` information for early malfunctions.

Because the `kdump` service starts too late, early malfunctions don't trigger the `kdump` kernel to boot, which prevents the capture of diagnostic information. To address that problem, you can enable early Kdump by adding a `dracut` module so that the crash kernel and `initramfs` are loaded as early as possible.

> **Note:**
>
> The following limitations apply for early Kdump:
>
> - The feature does not support `Fadump`.
> - Early Kdump becomes active the moment the system's `initramfs` begins to be processed. Any malfunction that occurs before that moment isn't captured even if early Kdump is enabled.

For more information about configuring early Kdump, see the step-by-step instructions in the `/usr/share/doc/kexec-tools/early-kdump-howto.txt` file.

9
Working With Core Dumps

Core dumps contain crash information for userspace applications and services running on Oracle Linux. They can be generated on demand by using a debugger, or the `systemd-coredump` service can be configured to generate them automatically in the event of a process stopping prematurely.

Core dumps contain a log summary of the crash event that typically includes the process ID, owner, termination signal, and a stack trace. For more information, see the `systemd-coredump(8)` manual pages.

The `coredumpctl` command can be used to review core dumps that have been written to the system journal or saved as a file. For more information, see the `coredumpctl(1)` manual pages.

To learn more about Systemd, the daemon that initializes the system and manages running services in Oracle Linux, see Oracle Linux 8: Managing Core System Configuration.

Enabling Core Dumps

Core dumps aren't enabled by default, so you must configure Systemd to generate them.

1. Create the `/etc/systemd/system.conf.d/10-enable-coredumps.conf` configuration file and add the following content:

   ```
   DumpCore=yes
   DefaultLimitCORE=infinity
   ```

2. Restart the `systemd` daemon to apply the change without restarting Oracle Linux:

   ```
   sudo systemctl daemon-reload
   ```

Configuring Core Dumps

To adjust the scope of the data captured in Systemd core dumps and define where Systemd stores them, change the `/etc/systemd/coredump.conf` configuration file. For more information, see the `coredump.conf(5)` manual pages.

Before running the `coredumpctl` command, remove any core dump size limits that apply to the current shell session:

```
sudo ulimit -c unlimited
```

For more information about the `ulimit` command, see the `ulimit(1)` manual pages.

Analyzing Core Dumps

- Use the `coredumpctl` command to list the core dumps that are available on the system:

  ```
  coredumpctl list
  ```

- To review more information about the core dumps stored for a particular application, specify the executable as an option:

  ```
  coredumpctl list executable-path
  ```

- To review all the core dumps that are stored for a failed process on the system, specify the process ID instead:

  ```
  coredumpctl list process-id
  ```

Exporting Core Dumps

1. To export the core dump for bug reporting purposes, specify the process ID and output file when you run the `coredumpctl dump` command:

   ```
   coredumpctl dump process-id -o output-file
   ```

2. Optionally, you can export an SOS report with extra information about the system. For more information, see Working With the sos Command.
3. On the same system or a different one, install the `gdb` package and then step through a core dump with the GNU Debugger by using the `coredumpctl debug` command:

   ```
   sudo dnf install gdb
   ```

   ```
   coredumpctl debug process-id
   ```

For more information about the `coredumpctl` command, see the `coredumpctl(1)` manual pages.

10

Working With the drgn Kernel Debugging Utility

Drgn is a tool and a programming library that can be used to extract debug information from both the live kernel of the running machine, and memory crash dumps from halted systems (`vmcore`).

To configure an Oracle Linux system to generate `vmcore` crash dumps, follow the instructions in Working With Kernel Dumps.

Drgn can be used as part of a root cause analysis to provide extra metrics that aren't already exposed through existing dashboards and interfaces.

For more information, see https://drgn.readthedocs.io/.

(Optional) Installing DebugInfo Packages

You can optionally install `*-debuginfo` packages to add extra debugging symbols in generated core dumps. They're intended for development purposes only, so we recommend that you only install them in development environments.

1. Enable the Oracle Linux 8 `debuginfo` repository by creating the `/etc/yum.repos.d/debuginfo.repo` file with root privileges and the following contents:

   ```
   [debuginfo]
   name=Oracle Linux 8 Debuginfo Packages
   baseurl=https://oss.oracle.com/ol8/debuginfo/
   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
   gpgcheck=1
   enabled=1
   ```

2. Use the package manager to search for DebugInfo packages to install:

   ```
   dnf search *-debuginfo
   ```

Installing drgn

1. If you're running Oracle Linux with the Unbreakable Enterprise Kernel (UEK), install the `kernel-uek-debuginfo` package by using the `dnf` command:

   ```
   sudo dnf install -y kernel-uek-debuginfo-$(uname -r)
   ```

   If you're running Oracle Linux with the Red Hat Compatible Kernel (RHCK), install the `kernel-debuginfo` package instead:

   ```
   sudo dnf install -y kernel-debuginfo-$(uname -r)
   ```

2. Enable the `ol8_addons` repository:

   ```
   sudo dnf config-manager --enable ol8_addons
   ```

   For more information, see Oracle Linux: Managing Software on Oracle Linux.

3. Install the `drgn` package:

   ```
   sudo dnf install -y drgn
   ```

Using the drgn Command

To debug the running kernel, use the `drgn` command to analyze the contents of the `/proc/kcore` dump file:

```
sudo drgn
```

To debug a running kernel or `vmcore` crash dump, specify the dump file by using the `-c` option. Optionally, specify the `vmlinux` and module symbols by also using the `-s` option:

```
sudo drgn -c path/to/dumpfile -s path/to/vmlinux
```

For example, to debug `/proc/kcore` for a live kernel and specify kernel drivers, run the following command:

```
sudo drgn  -c /proc/kcore -s /usr/lib/debug/lib/modules/$(uname -r)/
vmlinux \
   -s /lib/modules/$(uname -r)/kernel/net/netfilter/xt_comment.ko.xz
```

To perform the same operation on a `vmcore` crash dump file:

```
sudo drgn -c /var/crash/127.0.0.1-2023-06-02-09:33:07/vmcore \
 -s /usr/lib/debug/lib/modules/5.15.0-101.103.2.1.el8uek.x86_64/
vmlinux \
 -s /lib/modules/5.15.0-101.103.2.1.el8uek.x86_64/kernel/net/netfilter/
xt_comment.ko.xz
```

For more information about how to use the `drgn` command, use the `-h` option:

```
sudo drgn -h
```

Using the drgn Library With Python

Unlike the `crash` utility, Drgn wasn't originally designed to be a standalone kernel debugging tool. Drgn is a Python programming library that exposes debugging information for scripting and review purposes.

Before you can start using drgn with Python scripts, ensure that Python is correctly installed on the system. For more information, see Oracle Linux 8: Installing and Managing Python.

The `prog` array variable contains the information about the kernel that you are debugging. For example, to return the data collected for `slab_caches`, run the following statements in the `drgn` shell:

```
prog["slab_caches"]


(struct list_head){
        .next = (struct list_head *)0xffff8b831d972260,
        .prev = (struct list_head *)0xffff8b8007c02060,
}
```

Standard python structures can also be used to iterate through debug information:

```
slab_caches = prog["slab_caches"]


slab_caches.next


*(struct list_head *)0xffff8b831d972260 = {
        .next = (struct list_head *)0xffff8b831d972460,
        .prev = (struct list_head *)slab_caches+0x0 = 0xffffffff836e3da0,
}
```

For more information about the drgn API and script syntax, see https://drgn.readthedocs.io/.