Oracle Linux 7

Monitoring and Tuning the System

F32306-11
May 2023

Oracle Linux 7 Monitoring and Tuning the System,

F32306-11

Contents

4 Working With Performance Co-Pilot

5 Working With Tuned

6 Automating System Tasks

7 Working With Crash Dumps

Preface

Oracle Linux 7: 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 performance for various system components.

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

This chapter provides information and tasks for monitoring your systems to ensure optimal performance. The various monitoring tools that are provided in Oracle Linux are also described.

About System Performance Problems

Many performance issues are the result of configuration errors, which can be avoided by using a validated configuration that has been pre-tested for the supported software, hardware, storage, drivers, and networking components. A validated configuration incorporates best practices for an Oracle Linux deployment, which includes undergoing real-world testing of the complete stack. Oracle publishes more than 100 validated configurations, which are freely available for download. Refer to the release notes for the Oracle Linux release that you are running for further recommendations on setting kernel parameters.

For example, a typical problem involves out-of-memory errors and generally poor performance when running Oracle Database. The cause of this problem is likely to be that the system is not configured to use the HugePages feature for the System Global Area (SGA). With HugePages, you can set the page size to between 2MB and 256MB, so reducing the total number of pages that the kernel needs to manage. The memory that is associated with HugePages cannot be swapped out, which forces the SGA to remain resident in memory. Problems such as this one can be avoided by using validated configurations and referring to setting recommendations for kernel parameters.

See Oracle Linux 7: Managing Core System Configuration for more information about kernel parameters that affect system performance. You can also review the manual pages for each command with overlapping functionality on your Oracle Linux system.

Working With System Performance and Monitoring Utilities

Performance issues can be caused by any system component, software or hardware, and their interaction. Many performance diagnostics utilities are available for Oracle Linux, including tools that monitor and analyze resource usage by different hardware components and tracing tools for diagnosing performance issues in multiple processes and their threads.

The following utilities enable you to collect information about system resource usage and errors, which can help you to identify performance problems that are caused by overloaded disks, network, memory, or CPUs:

`dmesg`
Displays the contents of the kernel ring buffer, which can contain errors about system resource usage. Provided by the `util-linux-ng` package.

`dstat`
Displays statistics about system resource usage. Provided by the `dstat` package.

free
Displays the amount of free and used memory in the system. Provided by the `procps` package.

iostat
Reports I/O statistics. Provided by the `sysstat` package.

iotop
Monitors disk and swap I/O on a per-process basis. Provided by the `iotop` package.

ip
Reports network interface statistics and errors. Provided by the `iproute` package.

mpstat
Reports processor-related statistics. Provided by the `sysstat` package.

nfsiostat
Reports I/O statistics for NFS mounts. Provided by the `nfs-utils` package.

sar
Reports information about system activity. Provided by the `sysstat` package.

ss
Reports network interface statistics. Provided by the `iproute` package.

top
Provides a dynamic real-time view of the tasks that are running on a system. Provided by the `procps` package.

uptime
Displays the system load averages for the past 1, 5, and 15 minutes. Provided by the `procps` package.

vmstat
Reports virtual memory statistics. Provided by the `procps` package.

Monitoring the Usage of System Resources

You need to collect and monitor system resources regularly so that you are provided with a continuous record of a system's performance. First, establish a baseline of acceptable measurements under typical operating conditions. You can then use that baseline as a reference point to make it easier to identify memory shortages, spikes in resource usage, and other problems when they occur. Monitoring system performance also enables you to plan for future growth and determine how configuration changes might affect future performance.

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, run the `mpstat` command once per second with the following command:

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

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

```
sudo mpstat seconds
```

If it is installed, the `sar` command records statistics every 10 minutes while the system is running and retains that information for every day of the current month. The following command displays all the statistics that `sar` recorded for day `DD` of the current month:

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

To run the `sar` command as a background process and collect data in a file that you can display later by using the `-f` option:

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

In the previous command, *count* is the number of samples to record.

OSWbb and OSWbb analyzer (OSWbba) are useful tools for collecting and analyzing performance statistics. For more information, see Working With OSWatcher Black Box.

Monitoring CPU Usage

The `uptime`, `mpstat`, `sar`, `dstat`, and `top` utilities enable you to monitor CPU usage. When your system's CPU cores are all occupied executing the code of processes, other processes must wait until a CPU core becomes free or the scheduler switches a CPU core to run their code. If too many processes are queued too often, then that can represent a bottleneck in the performance of the system.

The commands `mpstat -P ALL` and `sar -u -P ALL` display CPU usage statistics for each CPU core and is averaged across all CPU cores.

The `%idle` value shows the percentage of time that a CPU was not running system code or process code. If the value of `%idle` is near 0% most of the time on all CPU cores, the system is CPU-bound for the workload that it is running. The percentage of time spent running system code (`%system` or `%sys`) should not usually exceed 30%, especially if `%idle` is close to 0%.

The system load average represents the number of processes that are running on CPU cores, waiting to run, or waiting for disk I/O activity to complete averaged over a period of time. On a busy system, the load average reported by `uptime` or `sar -q` should usually be not greater than two times the number of CPU cores over periods as long as 5 or 15 minutes. If the load average exceeds four times the number of CPU cores for long periods, the system is overloaded.

In addition to load averages (`ldavg-*`), the `sar -q` command reports the number of processes currently waiting to run (the *run-queue size*, `runq-sz`) and the total number of processes (`plist_sz`). The value of `runq-sz` also provides an indication of CPU saturation.

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

A combination of sustained large load average or large run queue size and low `%idle` can indicate that the system has insufficient CPU capacity for the workload. When CPU usage is high, use a command such as `dstat` or `top` to determine which processes are most likely to be responsible. For example, the following `dstat` command shows which processes are using CPUs, memory, and block I/O most intensively:

```
sudo dstat --top-cpu --top-mem --top-bio
```

The `top` command provides a real-time display of CPU activity. By default, `top` lists the most CPU-intensive processes on the system. In its upper section, `top` displays general information including the load averages over the past 1, 5 and 15 minutes, the number of running and sleeping processes (tasks), and total CPU and memory usage. In its lower section, `top` 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. Type `f` to select which fields `top` displays, `o` to change the order of the fields, or `O` to change the sort field. For example, entering `On` sorts the list on the percentage memory usage field (`%MEM`).

Monitoring Memory Usage

The `sar -r` command reports memory utilization statistics, including `%memused`, which is the percentage of physical memory in use.

`sar -B` reports memory paging statistics, including `pgscank/s`, which is the number of memory pages scanned by the `kswapd` daemon per second, and `pgscand/s`, which is the number of memory pages scanned directly per second.

`sar -W` reports swapping statistics, including `pswpin/s` and `pswpout/s`, which are the numbers of pages per second swapped in and out per second.

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

Once a system runs out of real or physical memory and starts using swap space, its performance deteriorates dramatically. If you run out of swap space, your programs or the entire operating system are likely to crash. If `free` or `top` indicate that little swap space remains available, this is also an indication you are 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.

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.

`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` command 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. Use the left and right arrow keys to change the sort field, and press `A` to toggle the I/O units between bytes per second and total number of bytes, or `O` to toggle 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`).

`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.

The `ss -s` command displays summary statistics for each protocol.

Using the Graphical System Monitor

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

To display the System Monitor, use the following command:

```
sudo gnome-system-monitor
```

The **Resources** tab displays:

- 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 **Contents** from the Help menu.

2
Working With the sosreport Command

This chapter describes how to install and run the `sosreport` command and configure its associated modules to collect system configuration and log information about your Oracle Linux systems.

About sosreport

The `sosreport` command collects information about a system such as hardware configuration, software configuration, and operational state. You can also use `sosreport` to enable diagnostics and analytical functions. To assist in troubleshooting a problem, `sosreport` records the information in a compressed file that you can send to a support representative.

Installing sosreport

If the `sos` package is not already installed on your system, use `yum` to install it:

```
sudo yum install sos
```

Use the `sosreport -l` command to list the available plugins and plugin options:

```
sudo sosreport -l

The following plugins are currently enabled:

acpid            acpid related information
anaconda         Anaconda / Installation information
.
.
.
The following plugins are currently disabled:

amd                Amd automounter information
cluster            cluster suite and GFS related information
.
.
.
The following plugin options are available:
apache.log             off gathers all apache logs
auditd.syslogsize      15 max size (MiB) to collect per syslog file
.
.
.
```

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

Running sosreport

You can run `sosreport` to record information about a problem area and specify options to tailor the report it generates as follows:

```
sudo sosreport [options ...]
```

For example, to record only information about Apache and Tomcat, and to gather all the Apache logs, use the following command:

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

sosreport (version 2.2)
.
.
.
Press ENTER to continue, or CTRL-C to quit.
```

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

```
sudo sosreport -a -k rpm.rpmva=off
```

Note that this process can take a considerable amount of time, but once it has completed, press Enter and then provide any additional information that is required:

```
Please enter your first initial and last name [email_address]: AName
Please enter the case number that you are generating this report for: case#

  Running plugins. Please wait ...

  Completed [55/55] ...
Creating compressed archive...

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

The md5sum is: checksum

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

`sosreport` saves the report as an `xz`-compressed `tar` file in `/tmp`.

Optionally, to obfuscate sensitive information, you can run the `soscleaner` command on the compressed archive generated from the `sosreport` command.

The cleaned report obfuscates the following details:

- IPv4 addresses and networks (network topologies are retained)
- MAC addresses
- Host names
- User names

> **Note:**
>
> Reports processed with the `soscleaner` command obfuscate certain details that may be needed for advanced troubleshooting, such as networking information.

If the `soscleaner` package is not already installed on your system, use `yum` to install it from the `ol7_addons` repository:

```
sudo yum install soscleaner
```

To generate a cleaned report, run the `soscleaner` command on the compressed archive generated from the `sosreport` command in the `/tmp` directory:

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

Press Enter to proceed. After the `soscleaner` command completes, a new `xz`-compressed `tar` file with `-obfuscated` in the file name is created the `/tmp` directory.

For more information, see the `sosreport(1)` and `soscleaner(1)` manual pages.

Reviewing Information Gathered by sosreport

The `sosreport` 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 `sosreport` 3.9. To verify the modules you have installed on your system, read Installing sosreport.
>
> Disabling modules prevents the `sosreport` command from collecting certain details that may 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/*`

Module	Information Type	Included Files
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	
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

Module	Information Type	Included Files
`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/clocksource/*/current_clocksource • /sys/fs/pstore • /var/log/dmesg

Module	Information Type	Included Files
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
ssh	SSH configuration	• /etc/ssh/ssh_config • /etc/ssh/sshd_config

Module	Information Type	Included Files
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/yum.log

3
Working With OSWatcher Black Box

This chapter describes how to install and configure Oracle OSWatcher Black Box (OSWbb) to collect operating system and network performance metrics about your Oracle Linux systems.

About OSWbb

OSWbb collects and archives operating system 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. Log in to My Oracle Support (MOS) at https://support.oracle.com.
2. Download OSWatcher from the link listed by Doc ID 301137.1 at https://support.oracle.com/epmos/faces/DocumentDisplay?id=301137.1.
3. Copy the file to the directory where you want to install OSWbb, and run the following command:

   ```
   sudo tar xvf oswbbVERS.tar
   ```

   VERS represents the version number of OSWatcher, for example 730 for OSWatcher 7.30.

   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:

   ```
   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 should collect data and the number of hours for which OSWbb should run. 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
...
```

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 following directories under the `oswbb/archive` directory:

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

Directory	Description
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 named *system_name_command_name_timestamp*.dat. Each entry in a file is preceded by a timestamp.

Analyzing OSWbb Archived Files

From release v4.0.0, you can use the OSWbb analyzer (OSWbba) to provide information on system slowdowns, system hangs and other performance problems, and also to graph data collected from iostat, netstat, and vmstat. OSWbba requires that you have Java version 1.4.2 or a later version installed on your system. You can use yum to install Java; or you can download a Java RPM for Linux from http://www.java.com.

Use the following command to run OSWbba from the oswbb directory:

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

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 per second, writes per second, service time for I/O requests, and percentage utilization 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), which are available from My Oracle Support (MOS) at https://support.oracle.com.

4
Working With Performance Co-Pilot

This chapter describes how to install and configure Performance Co-Pilot (PCP) so that you can analyze system performance across one or more Oracle Linux installations.

About PCP

PCP collects operating system 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 `ol7_latest` and `ol7_addons` yum repositories on your system.

   For more information, see Oracle Linux 7: Managing Software.

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

   ```
   sudo yum 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 7: 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 once every minute to once every 5 seconds:

```
...
# 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

All of the archives that the `pmlogger` service generates are stored in the `/var/log/pcp/pmlogger/`*`hostname`* directory. If you navigate to that directory in a terminal, you can run a number of commands to review the performance metrics that have been collected.

For more information about those commands and their parameters, see their respective manual pages.

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

To review the data for specific performance metrics within a specified timespan, use the `pmdumptext` command. For example, to review resource usage metrics for CPU load, memory utilisation 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
```

You can also use the `pmstat` to review 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
```

You can also compare the metrics between two time periods by using 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
```

5
Working With Tuned

This chapter describes the Tuned monitoring tool and Tuned profiles . The chapter also includes tasks for using Tuned to optimize performance on your Oracle Linux systems.

About Tuned

The Tuned system tuning tool is used to monitor the system to optimize its performance under certain conditions. The Tuned tool uses the `udev` device manager to monitor connected devices, enabling both static and dynamic tuning of your system's settings. Note that dynamic tuning is turned off by default in this release. To enable dynamic tuning, see About Static and Dynamic Tuning in Tuned.

Tuned uses several predefined profiles to tune your system. The profiles that are provided are designed for particular use cases and fall into one of the following two categories: power-saving profiles and performance-boosting profiles. Performance-boosting profiles address low latency and high throughput for storage and the network and virtualization host performance.

Based on the product that is currently in use, a default profile is automatically set. You can use the `tuned-adm recommend` command to determine which profile is recommended for a particular product. Note that if no recommendation is available, the `balanced` profile is set.

You can modify the rules that are defined for each profile, as well as customize how a specific device is tuned by using a specific profile. In addition, you can configure Tuned so that any changes in device usage triggers an adjustment in the current settings so that the performance of active devices is improved and power consumption for inactive devices is reduced.

About Tuned Profiles

The following Tuned profiles are typically installed by default in Oracle Linux 7 or can be installed from a separate package:

- `balanced` (default profile): Is a power-saving profile. This profile provides a balance between performance and power consumption. The profile uses auto-scaling and auto-tuning when possible. A possible drawback is increased latency.
- `powersave`: Is a profile that provides maximum power saving performance. The profile can minimize actual power consumption by throttling performance.

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

- `throughput-performance` (default profile): Is a server profile that is optimized for high throughput. The profile disables power-savings mechanisms and enables `sysctl` settings to improve the throughput performance of the disk and network IO.

- `latency-performance`: Is a server profile that is optimized for low latency. The profile disables power-savings mechanisms and enables `sysctl` settings to improve latency.
- `network-latency`: Is a profile that 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`: A profile for throughput network tuning. It is based on the `throughput-performance` profile. In addition, this profile increases kernel network buffers.
- `virtual-guest` (default profile): Is a profile that is designed for virtual guests and is based on the `throughput-performance` profile. This profile decreases virtual memory swappiness and increases disk readahead values.
- `virtual-host`: Is a profile that is designed for virtual hosts and is based on the `throughput-performance` profile. This profile decreases virtual memory swappiness, increases disk readahead values, and enables a more aggressive value of dirty pages writeback.
- `desktop`: Is a profile that is optimized for desktop environments and is based on the `balanced` profile. In addition, this profile enables scheduler autogroups for better response of interactive applications.

> **Note:**
>
> You can install additional profiles to better match your system configuration and intended use case. For example, if you are using a real-time kernel with Oracle Linux, you can use a real-time profile. Most of these optional packages can be installed from the `ol7_optional_latest` channel.
>
> Note that real-time profiles have no effect on kernels that are not compiled with real-time support enabled.
>
> To list all of the profiles that are currently available for installation, use the following command
>
> ```
> sudo yum list tuned-profiles*
> ```

Tuned profiles that are installed on the system by default are stored in the `/usr/lib/tuned` and `/etc/tuned` directories. Distribution-specific profiles are stored in the `/usr/lib/tuned` directory. Note that each profile has its own directory. Each profile directory consists of a main configuration file, `tuned.conf`, as well as other optional files.

If you want to use a custom profile, copy the profile directory to the `/etc/tuned` directory, which is where custom profiles are stored. In the event there are two profiles with the same name, the custom profile that is located in `/etc/tuned/` is used.

The `tuned.conf` file can contain one `[main]` section and additional sections for configuring plugin instances. Note that these sections are optional. For more information about profile configuration, see the `tuned.conf(5)` manual page.

About the Default Tuned Profiles

A default Tuned profile is automatically selected when you install Oracle Linux. The default profile that is selected is based on the given environment and the performance goals to be achieved in that particular use case. The following default profiles are provided:

- `throughput-performance`: Is a profile that is used in an environment where compute nodes are running Oracle Linux. This profile achieves the best throughput performance.
- `virtual-guest`: Is a profile that is used in an environment where virtual machines are running Oracle Linux. This profile achieves the best performance. If you are not interested in the best performance, you can change the profile to either the `balanced` or `powersave` profile.
- `balanced`: Is a profile that is used for other use cases. This profile achieves balanced performance and power consumption.

About Static and Dynamic Tuning in Tuned

Static tuning applies settings that you have defined in the configuration files for `sysctl`, `sysfs`, and other system configuration tools throughout the operating system.

You can configure the `tuned` service to monitor the activity of system components and dynamically tuned system settings, based on information that the service collects about the system and its current running state.

Dynamic tuning can be particularly useful in situations where you need the load on devices like the CPU, hard drives, and network adapters to consume as little power as possible when idle, but require high throughput and low latency when under a high load.

You enable dynamic tuning by setting the correct value in the `/etc/tuned/tuned-main.conf` settings file, for example:

```
dynamic_tuning = 1
```

You must then set the time interval in seconds for `tuned` to analyze the current system state in the same configuration file so that it can dynamically tune the system, based on the collected results, for example:

```
update_interval = 10
```

Installing and Enabling Tuned From the Command Line

The following procedure describes how to install and enable Tuned, install Tuned profiles, and preset a default Tuned profile for your Oracle Linux systems.

1. If the `tuned` package is not already installed, install it:

   ```
   sudo yum install tuned
   ```

2. Enable and start the `tuned` service:

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

3. Check the active Tuned profile:

   ```
   sudo tuned-adm active
   ```

```
Current active profile: balanced
```

4. Verify that the Tuned profile is applied to the system:

```
sudo tuned-adm verify

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

If a message indicating the current system settings do not match is displayed, try restarting the `tuned` service:

```
sudo systemctl start tuned
```

Running Tuned in no-daemon Mode

Running `tuned` in `no-daemon` mode does not require any resident memory. However, note that when running the service in this mode, `tuned` does not perform any dynamic tuning. While in `no-daemon` mode, `tuned` only applies the settings and then exits.

To run `tuned` in `no-daemon` mode, you must set the following value in the `/etc/tuned/tuned-main.conf` settings file:

```
daemon = 0
```

> **NOT_SUPPORTED:**
>
> Take note that if you decide to run `tuned` in `no-daemon` mode, be aware that some functions do not work without running the daemon. In particular, `tuned` no longer supports D-Bus services or the hot-plug kernel subsystem. Consequently `tuned` can no longer automatically roll back any settings files that were changed.

Administering the Tuned Service and Tuned Profiles

You administer Tuned by using the `tuned-adm` command. The following tasks describe how to administer Tuned profiles and the `tuned` service on your Oracle Linux systems.

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

Listing Tuned Profiles

To list all of the available Tuned profiles on a system:

```
sudo tuned-adm list

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

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

The current active profile is also displayed with this output.

To display just the currently active profile:

```
sudo tuned-adm active

Current active profile: balanced
```

Activating a Tuned Profile

> **Note:**
>
> To activate a Tuned profile, the `tuned` service must be running on your system.

Use the following command to activate a specific selected Tuned profile:

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

To have Tuned recommend the profile that is most suitable for your system, use the `tuned-adm recommend` command:

```
sudo tuned-adm recommend

virtual-guest
```

To activate a combination of multiple profiles, use the following command syntax:

```
sudo tuned-adm profile profile1 profile2
```

Disabling Tuned

To disable tuning temporarily, use the following command:

```
sudo tuned-adm off
```

The previous command disables any tuning settings until you restart the `tuned` service. When you restart the service, all of the previous tuning settings are re-applied.

You can disable tuning on a more permanent basis by stopping and disabling the `tuned` service as follows:

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

6
Automating System Tasks

This chapter describes how to configure the system to run tasks automatically within a specific period of time, at a specified time and date, or when the system is lightly loaded.

About Automating Tasks

You can use automated tasks to perform periodic backups, monitor the system, run custom scripts, as well as other administrative tasks.

The `cron` and `anacron` utilities enable you to schedule the execution of recurring tasks, referred to as *jobs*, according to a combination of the following: time, day of the month, month, day of the week, and week. With the `cron` command, you can schedule jobs to run as often as every minute. If the system is down when a job is scheduled, `cron` does not run the job when the system restarts.

The `anacron` command you to schedule a system job to run only once per day. However, if a scheduled job has not been run, that job runs when the system restarts. The `anacron` command is mainly intended for use on laptop computers.

You do not usually need to run `cron` and `anacron` directly. The `crond` daemon executes scheduled tasks on behalf of `cron` and it starts `anacron` once every hour. `crond` looks in `/etc/crontab` or in files in `/etc/cron.d` for system `cron` job definitions, and `/var/spool/cron` for `cron` job definitions belonging to users. `crond` checks each job definition to see whether it should run in the current minute. If a job is scheduled for execution, `crond` runs it as the owner of the job definition file or, for system `cron` jobs, the user specified in the job definition (if any).

`crond` runs the `0anacron` script in the `/etc/cron.hourly` directory as `root` once per hour according to the schedule in `/etc/cron.d/0hourly`. If `anacron` is not already running and the system is connected to mains and not battery power, `crond` starts `anacron`.

`anacron` runs the scripts in the `/etc/cron.daily`, `/etc/cron.weekly`, and `/etc/cron.monthly` directories as `root` once per day, week or month, according to the job definitions that are scheduled in `/etc/anacrontab`.

Configuring cron Jobs

System `cron` jobs are defined in `crontab`-format files in `/etc/crontab` or in files in `/etc/cron.d`. A `crontab` file typically consists of definitions for the `SHELL`, `PATH`, `MAILTO`, and `HOME` variables for the environment in which the jobs are run, followed by the job definitions themselves. Comment lines start with a `#` character; job definitions are specified by using the following format:

```
minute  hour  day  month  day-of-week  user  command
```

Each of the fields that you can specify are defined 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 can be `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 commas (`,`), for example, `0,2,4`.

/
Specify a step value by using the forward 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
```

The `root` user can add job definition entries to the `/etc/crontab`, or add `crontab`-format files to the `/etc/cron.d` directory.

> **Note:**
>
> If you add an executable job script to the `/etc/cron.hourly` directory, `crond` runs the script once every hour. Your script should check that it is not already running.

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

Controlling Access to Running cron Jobs

If permitted, users other than `root` can configure `cron` tasks by using the `crontab` command. All user-defined `crontab`-format files are stored in the `/var/spool/cron` directory with the same name as the users that created them.

`root` can use the `/etc/cron.allow` and `/etc/cron.deny` files to restrict access to `cron`. `crontab` checks the access control files each time that a user tries to add or delete a `cron` job. If `/etc/cron.allow` exists, only users listed in it are allowed to use `cron`, and `/etc/cron.deny` is ignored. If `/etc/cron.allow` does not exist, users listed in `/etc/cron.deny` are not allowed to use `cron`. If neither file exists, only `root` can use `cron`. The format of both `/etc/cron.allow` and `/etc/cron.deny` is one user name on each line.

To create or edit a `crontab` file as a user, log in as that user and type the command `crontab -e`, which opens your `crontab` file in the `vi` editor (or the editor specified by the `EDITOR` or `VISUAL` environment variables). The file has the same format as `/etc/crontab` except that the user field is omitted. When you save changes to the file, these are written to the file `/var/spool/cron/`*`username`* . To list the contents of your `crontab` file, use the `crontab -l` command. To delete your `crontab` file, use the `crontab -r` command.

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

Configuring anacron Jobs

System `anacron` jobs are defined in `/etc/anacrontab`, which 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.

Job definitions are specified in the following format:

```
period  delay  job-id  command
```

The following is a description of the fields that may be included:

period
Frequency of job execution specified in days or as `@daily`, `@weekly`, or `@monthly` for once per day, week, or month.

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.

The following entries are taken from the default `/etc/anacrontab` file:

```
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
```

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 anywhere between 03:11 and 03:50 every day if the system is running, or after the system is booted and the time is less than 22:00. The `run-parts` script sequentially executes every program within the directory specified as its argument.

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

Scripts in `/etc/cron.monthly` run once per week 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

You can use the `at` command to schedule a one-time task to run at a specified time, or the `batch` command to schedule a one-time task to run when the system load average drops below 0.8. The `atd` service must be running to use `at` or `batch`.

```
sudo systemctl is-active atd

active
```

`at` takes a time as its argument and reads the commands to be run from the standard input. For example, run the commands in the file `atjob` in 20 minutes time:

```
sudo at now + 20 minutes < ./atjob

job 1 at 2013-03-19 11:25
```

The `atq` command shows the `at` jobs that are queued to run:

```
sudo atq

1    2013-03-19 11:25 a root
```

The `batch` command also reads command from the standard input, but it does not run until the system load average drops below 0.8. For example:

```
sudo batch < batchjob

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 1 2
```

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

Changing the Behavior of Batch Jobs

The load average of a system, as displayed by the `uptime` and `w` commands, represents the average number of processes that are queued to run on the CPUs or CPU cores over a given time period. Typically, a system might not 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 below 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 above which `atd` will not 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.

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

1. Edit the `atd` configuration file, `/etc/sysconfig/atd`, uncomment the line that defines the `OPTS` variable, and edit the line to specify the new load-average limit and minimum interval time, for example:

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

   This example sets the minimum interval to 100 seconds and the load-average limit to 3.

2. Restart the `atd` service:

   ```
   sudo systemctl restart atd
   ```

3. Verify that the `atd` daemon is running with the new minimum interval and load-average limit:

   ```
   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.

7
Working With Crash Dumps

This chapter describes how to configure a system to create a memory image in the event of a system crash, and how to use the `crash` debugger to analyze the memory image in a crash dump or for a live system.

About Kdump

Kdump is the Linux kernel crash-dump mechanism. Oracle recommends that you enable the Kdump feature. In the event of a system crash, Kdump creates a memory image (`vmcore`) that can help in determining the cause of the crash. Enabling Kdump requires you to reserve a portion of system memory for exclusive use by Kdump. This memory is unavailable for other uses.

Kdump uses `kexec` to boot into a second kernel whenever the system crashes. `kexec` is a fast-boot mechanism that enables a Linux kernel to boot from inside the context of a kernel that is already running without passing through the bootloader stage.

Configuring and Using Kdump

During installation, you are given the option of enabling Kdump and specifying the amount of memory to reserve for it. If you prefer, you can enable kdump at a later time as described in this section.

If the `kexec-tools` and `system-config-kdump` packages are not already installed on your system, use `yum` to install them.

To enable Kdump by using the Kernel Dump Configuration GUI.

1. Enter the following command:

   ```
   sudo system-config-kdump
   ```

   The Kernel Dump Configuration GUI starts. If Kdump is currently disabled, the green **Enable** button is selectable and the **Disable** button is greyed out.

2. Click **Enable** to enable Kdump.
3. You can select the following settings tags to adjust the configuration of Kdump.

   Basic Settings
   Specify the amount of memory to reserve for Kdump. The default setting is 128 MB.

   Target Settings
   Specify the target location for the `vmcore` dump file on a locally accessible file system, to a raw disk device, or to a remote directory using NFS or SSH over IPv4. The default location is `/var/crash`.

   You cannot save a dump file on an eCryptfs file system, on remote directories that are NFS mounted on the `rootfs` file system, or on remote directories that access require the

use of IPv6, SMB, CIFS, FCoE, wireless NICs, multipathed storage, or iSCSI over software initiators to access them.

Filtering Settings

Specify the type of data to include in or exclude from the dump file. Selecting or deselecting the options alters the value of the argument that Kdump specifies to the `-d` option of the core collector program, `makedumpfile`.

Expert Settings

Choose which kernel to use, edit the command line options that are passed to the kernel and the core collector program, choose the default action if the dump fails, and modify the options to the core collector program, `makedumpfile`.

The Unbreakable Enterprise Kernel supports the use of the `crashkernel=auto` setting for UEK Release 3 Quarterly Update 1 and later. If you use the `crashkernel=auto` setting, the output of the `dmesg` command shows `crashkernel=XM@0M`, which is normal. The setting actually reserves 128 MB plus 64 MB for each terabyte of physical memory.

> **Note:**
>
> You cannot configure `crashkernel=auto` for Xen or for the UEK prior to UEK Release 3 Quarterly Update 1. Only standard settings such as `crashkernel=128M@48M` are supported. For systems with more than 128 GB of memory, the recommended setting is `crashkernel=512M@64M`.

You can select one of five default actions should the dump fail:

mount rootfs and run /sbin/init
Mount the root file system and run `init`. The `/etc/init.d/kdump` script attempts to save the dump to `/var/crash`, which requires a large amount of memory to be reserved.

reboot
Reboot the system, losing the `vmcore`. This is the default action.

shell
Enter a shell session inside the `initramfs` so that you can attempt to record the core. To reboot the system, exit the shell.

halt
Halt the system.

poweroff
Power down the system.

Click **Help** for more information on these settings.

4. Click **Apply** to save your changes. The GUI displays a popup message to remind you that you must reboot the system for the changes to take effect.
5. Click **OK** to dismiss the popup messages.
6. Select **File > Quit**.

7. Reboot the system at a suitable time.

Files Used by Kdump

The Kernel Dump Configuration GUI modifies the following files:

File	Description
`/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 where 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 supported parameters.

If 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.

Using Kdump with OCFS2

By default, a fenced node in an OCFS2 cluster restarts instead of panicking so that it can quickly rejoin the cluster. If the reason for the restart is not apparent, you can change the node's behavior so that it panics and generates a `vmcore` for analysis.

To configure a node to panic when it next fences, run the following command on the node after the cluster starts:

```
sudo echo panic > /sys/kernel/config/cluster/cluster_name/fence_method
```

In the previous command, *cluster_name* is the name of the cluster. To set the value after each reboot of the system, add this line to `/etc/rc.local`. To restore the default behavior, set the value of `fence_method` to `reset` instead of `panic` and remove the line from `/etc/rc.local`.

For more information, see the Oracle Cluster File System Version 2 chapter in Oracle Linux 7: Managing File Systems .

Using the crash Debugger

The `crash` command enables you to analyze the state of an Oracle Linux system while it is running; or, the state of a core dump resulting from a kernel crash. The `crash` is merged with the GNU Debugger `gdb` to provide source code debugging capabilities.

Installing the crash Packages

To use the `crash` command, you must install the `crash` package and the appropriate `debuginfo` and `debuginfo-common` packages.

To install the required packages:

1. Install the latest version of the `crash` package:

   ```
   sudo yum install crash
   ```

2. Download the appropriate `debuginfo` and `debuginfo-common` packages for the `vmcore` or kernel that you want to examine from https://oss.oracle.com/ol7/debuginfo/:

   - If you want to examine the Unbreakable Enterprise Kernel that is running on the system, you would use command similar to the following to download the packages:

     ```
     sudo export DLP="https://oss.oracle.com/ol7/debuginfo"

     sudo wget ${DLP}/kernel-uek-debuginfo-`uname -r`.rpm

     sudo wget ${DLP}/kernel-uek-debuginfo-common-`uname -r`.rpm
     ```

   - If you want to examine the Red Hat Compatible Kernel (RHCK) that is running on the system, you would use commands similar to the following to download the packages:

     ```
     sudo export DLP="https://oss.oracle.com/ol7/debuginfo"

     sudo wget ${DLP}/kernel-debuginfo-`uname -r`.rpm

     sudo wget ${DLP}/kernel-debuginfo-common-`uname -r`.rpm
     ```

   - If you want to examine a `vmcore` file that relates to kernel that is different than the currently running kernel, download the appropriate `debuginfo` and `debuginfo-common` packages for the kernel that produced the `vmcore`, for example:

     ```
     sudo export DLP="https://oss.oracle.com/ol7/debuginfo"

     sudo wget ${DLP}/kernel-uek-debuginfo-4.1.12-112.14.15.el7uek.x86_64.rpm

     sudo wget ${DLP}/kernel-uek-debuginfo-
     common-4.1.12-112.14.15.el7uek.x86_64.rpm
     ```

     > **Note:**
     >
     > If the `vmcore` file was produced by Kdump, you can use the following command to determine the version:
     >
     > ```
     > sudo crash --osrelease /var/tmp/vmcore/2013-0211-2358.45-
     > host03.28.core
     >
     > 2.6.39-200.24.1.el6uek.x86_64
     > ```

3. Install the `debuginfo` and `debuginfo-common` packages.

   ```
   sudo rpm -Uhv kernel-uek-debuginfo-4.1.12-112.14.15.el7uek.x86_64.rpm \
     kernel-uek-debuginfo-common-4.1.12-112.14.15.el7uek.x86_64.rpm
   ```

   The `vmlinux` kernel object file, also known as the *namelist* file, that the `crash` command requires is installed in `/usr/lib/debug/lib/modules/`*`kernel_version`*`/`.

Running crash

> **NOT_SUPPORTED:**
>
> Running the `crash` command on a live system can cause data corruption or total system failure. Do not use the command to examine a production system, unless directed to do so by Oracle Support.

To examine the currently running kernel, run the following command:

```
sudo crash
```

To determine the version of the kernel that produced a `vmcore` file:

```
sudo crash --osrelease /var/tmp/vmcore/2013-0211-2358.45-host03.28.core

2.6.39-200.24.1.el6uek.x86_64
```

To examine a `vmcore` file, specify the path to the file as an argument, for example:

```
sudo crash /var/tmp/vmcore/2013-0211-2358.45-host03.28.core
```

> **Note:**
>
> The appropriate `vmlinux` file must exist in `/usr/lib/debug/lib/modules/`*`kernel_version`*`/`.

If the `vmlinux` file is located elsewhere, you will need to specify its path in the command first, followed by path to the `vmcore` file, for example:

```
sudo crash /var/tmp/namelist/vmlinux-host03.28 /var/tmp/vmcore/2013-0211-2358.45-
host03.28.core
```

For example, the following `crash` output is from a `vmcore` file that was dumped after a system panic:

```
      KERNEL: /usr/lib/debug/lib/modules/2.6.39-200.24.1.el6uek.x86_64/vmlinux
    DUMPFILE: /var/tmp/vmcore/2013-0211-2358.45-host03.28.core
        CPUS: 2
        DATE: Fri Feb 11 16:55:41 2013
      UPTIME: 04:24;54
LOAD AVERAGE: 0.00, 0.01, 0.05
       TASKS: 84
    NODENAME: host03.mydom.com
     RELEASE: 2.6.39-200.24.1.el6uek.x86_64
     VERSION: #1 SMP Sat Jun 23 02:39:07 EDT 2012
     MACHINE: x86_64  (2992 MHz)
      MEMORY: 2 GB
       PANIC: "Oops: 0002" (check log for details)
         PID: 1696
     COMMAND: "insmod"
        TASK: c74de000
```

```
        CPU: 0
      STATE: TASK_RUNNING (PANIC)

crash>
```

In the previous example, the output includes the following information:

- Number of CPUs
- Load average over the last 1 minute, 5 minutes, and 15 minutes,
- Number of tasks running
- Amount of memory,
- Panic string
- Command that was executing at the time the dump was created

In the example, an attempt that was made by `insmod` to install a module resulted in an *oops* violation.

At the crash> prompt, you can type `help` or `?` to display the available `crash` commands. Type `help` *`command`* to display more information for a specified command.

The `crash` commands can be grouped into the following groups, according to purpose:

Kernel Data Structure Analysis Commands
Display kernel text and data structures. See Kernel Data Structure Analysis Commands .

System state commands
Examine kernel subsystems on a system-wide or a per-task basis. See System State Commands.

Helper commands
Perform calculation, translation, and search functions. See Helper Commands

Session control commands
Control the `crash` session. See Session Control Commands

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

Kernel Data Structure Analysis Commands

The following `crash` commands takes advantage of `gdb` integration to display the following kernel data structures symbolically:

*****
The *pointer-to* command can be used instead `struct` or `union`. The `gdb` module calls the appropriate function, for example:

```
crash> *buffer_head
```

```
struct buffer_head {
    long unsigned int b_state;
    struct buffer_head *b_this_page;
    struct page *b_page;
    sector_t b_blocknr;
    size_t b_size;
    char *b_data;
    struct block_device *b_bdev;
    bh_end_io_t *b_end_io;
    void *b_private;
    struct list_head b_assoc_buffers;
    struct address_space *b_assoc_map;
    atomic_t b_count;
}
SIZE: 104
```

dis

Disassembles source code instructions of a complete kernel function, from a specified address for a specified number of instructions, or from the beginning of a function up to a specified address, for example:

```
crash> dis fixup_irqs


0xffffffff81014486 <fixup_irqs>:        push   %rbp
0xffffffff81014487 <fixup_irqs+1>:      mov    %rsp,%rbp
0xffffffff8101448a <fixup_irqs+4>:      push   %r15
0xffffffff8101448c <fixup_irqs+6>:      push   %r14
0xffffffff8101448e <fixup_irqs+8>:      push   %r13
0xffffffff81014490 <fixup_irqs+10>:     push   %r12
0xffffffff81014492 <fixup_irqs+12>:     push   %rbx
0xffffffff81014493 <fixup_irqs+13>:     sub    $0x18,%rsp
0xffffffff81014497 <fixup_irqs+17>:     nopl   0x0(%rax,%rax,1)
...
```

p

Displays the contents of a kernel variable, for example:

```
crash> p init_mm


init_mm = $5 = {
  mmap = 0x0,
  mm_rb = {
    rb_node = 0x0
  },
  mmap_cache = 0x0,
  get_unmapped_area = 0,
  unmap_area = 0,
  mmap_base = 0,
  task_size = 0,
  cached_hole_size = 0,
  free_area_cache = 0,
  pgd = 0xffffffff81001000,
...
```

struct

Displays either a structure definition, or a formatted display of the contents of a structure at a specified address, for example:

```
crash> struct cpu


struct cpu {
    int node_id;
    int hotpluggable;
    struct sys_device sysdev;
}
SIZE: 88
```

sym
Translates a kernel symbol name to a kernel virtual address and section, or a kernel virtual address to a symbol name and section. You can also query (`-q`) the symbol list for all symbols containing a specified string or list (`-l`) all kernel symbols, for example:

```
crash> sym jiffies


ffffffff81b45880 (A) jiffies

crash> sym -q runstate


c590 (d) per_cpu__runstate
c5c0 (d) per_cpu__runstate_snapshot
ffffffff8100e563 (T) xen_setup_runstate_info

crash> sym -l


0 (D) __per_cpu_start
0 (D) per_cpu__irq_stack_union
4000 (D) per_cpu__gdt_page
5000 (d) per_cpu__exception_stacks
b000 (d) per_cpu__idt_desc
b010 (d) per_cpu__xen_cr0_value
b018 (D) per_cpu__xen_vcpu
b020 (D) per_cpu__xen_vcpu_info
b060 (d) per_cpu__mc_buffer
c570 (D) per_cpu__xen_mc_irq_flags
c578 (D) per_cpu__xen_cr3
c580 (D) per_cpu__xen_current_cr3
c590 (d) per_cpu__runstate
c5c0 (d) per_cpu__runstate_snapshot
...
```

union
Similar to the `struct` command, displaying kernel data types that are defined as unions instead of structures.

whatis
Displays the definition of structures, unions, typedefs or text or data symbols, for example:

```
crash> whatis linux_binfmt
```

```
struct linux_binfmt {
    struct list_head lh;
    struct module *module;
    int (*load_binary)(struct linux_binprm *, struct pt_regs *);
    int (*load_shlib)(struct file *);
    int (*core_dump)(long int, struct pt_regs *, struct file *, long unsigned int);
    long unsigned int min_coredump;
    int hasvdso;
}
SIZE: 64
```

System State Commands

The following commands display kernel subsystems, on a system-wide or per-task basis:

bt
Displays a kernel stack trace of the current context or of a specified PID or task. In the case of a dump that followed a kernel panic, the command traces the functions that were called leading up to the panic. For example:

```
crash> bt


PID: 10651  TASK: d1347000  CPU: 1   COMMAND: "insmod"
 #0 [d1547e44] die at c010785a
 #1 [d1547e54] do_invalid_op at c0107b2c
 #2 [d1547f0c] error_code (via invalid_op) at c01073dc
...
```

You can use the `-l` option to display the line number of the source file that corresponds to each function call in a stack trace.

```
crash> bt -l 1


PID: 1      TASK: ffff88007d032040  CPU: 1   COMMAND: "init"
 #0 [ffff88007d035878] schedule at ffffffff8144fdd4
    /usr/src/debug/kernel-2.6.32/linux-2.6.32.x86_64/kernel/sched.c: 3091
 #1 [ffff88007d035950] schedule_hrtimeout_range at ffffffff814508e4
    /usr/src/debug/kernel-2.6.32/linux-2.6.32.x86_64/arch/x86/include/asm/current.h:
14
 #2 [ffff88007d0359f0] poll_schedule_timeout at ffffffff811297d5
    /usr/src/debug/kernel-2.6.32/linux-2.6.32.x86_64/arch/x86/include/asm/current.h:
14
 #3 [ffff88007d035a10] do_select at ffffffff81129d72
    /usr/src/debug/kernel-2.6.32/linux-2.6.32.x86_64/fs/select.c: 500
 #4 [ffff88007d035d80] core_sys_select at ffffffff8112a04c
    /usr/src/debug/kernel-2.6.32/linux-2.6.32.x86_64/fs/select.c: 575
 #5 [ffff88007d035f10] sys_select at ffffffff8112a326
    /usr/src/debug/kernel-2.6.32/linux-2.6.32.x86_64/fs/select.c: 615
 #6 [ffff88007d035f80] system_call_fastpath at ffffffff81011cf2
    /usr/src/debug////////kernel-2.6.32/linux-2.6.32.x86_64/arch/x86/kernel/
entry_64.S:
    488
    RIP: 00007fce20a66243  RSP: 00007fff552c1038  RFLAGS: 00000246
    RAX: 0000000000000017  RBX: ffffffff81011cf2  RCX: ffffffffffffffff
    RDX: 00007fff552c10e0  RSI: 00007fff552c1160  RDI: 000000000000000a
    RBP: 0000000000000000   R8: 0000000000000000   R9: 0000000000000200
```

```
    R10: 00007fff552c1060  R11: 0000000000000246  R12: 00007fff552c1160
    R13: 00007fff552c10e0  R14: 00007fff552c1060  R15: 00007fff552c121f
    ORIG_RAX: 0000000000000017  CS: 0033  SS: 002b
```

`bt` is probably the most useful `crash` command. It has a large number of options that you can use to examine a kernel stack trace. For more information, enter `help bt`.

`dev`
Displays character and block device data. The `-d` and `-i` options display disk I/O statistics and I/O port usage. For example:

```
crash> dev


CHRDEV    NAME                 CDEV        OPERATIONS
   1      mem             ffff88007d2a66c0  memory_fops
   4      /dev/vc/0       ffffffff821f6e30  console_fops
   4      tty             ffff88007a395008  tty_fops
   4      ttyS            ffff88007a3d3808  tty_fops
   5      /dev/tty        ffffffff821f48c0  tty_fops
...
BLKDEV    NAME                 GENDISK      OPERATIONS
   1      ramdisk         ffff88007a3de800  brd_fops
 259      blkext               (none)
   7      loop            ffff880037809800  lo_fops
   8      sd              ffff8800378e9800  sd_fops
   9      md                   (none)
...

crash> dev -d


MAJOR GENDISK            NAME       REQUEST QUEUE      TOTAL ASYNC  SYNC   DRV
    8 0xffff8800378e9800 sda        0xffff880037b513e0    10     0    10     0
   11 0xffff880037cde400 sr0        0xffff880037b50b10     0     0     0     0
  253 0xffff880037902c00 dm-0       0xffff88003705b420     0     0     0     0
  253 0xffff880037d5f000 dm-1       0xffff88003705ab50     0     0     0     0

crash> dev -i


    RESOURCE        RANGE    NAME
ffffffff81a9e1e0  0000-ffff  PCI IO
ffffffff81a96e30  0000-001f  dma1
ffffffff81a96e68  0020-0021  pic1
ffffffff81a96ea0  0040-0043  timer0
ffffffff81a96ed8  0050-0053  timer1
ffffffff81a96f10  0060-0060  keyboard
...
```

`files`
Displays information about files that are open in the current context or in the context of a specific PID or task. For example:

```
crash> files 12916
```

```
PID: 12916  TASK: ffff8800276a2480  CPU: 0   COMMAND: "firefox"
ROOT: /    CWD: /home/guest
 FD       FILE            DENTRY           INODE       TYPE PATH
  0 ffff88001c57ab00 ffff88007ac399c0 ffff8800378b1b68 CHR  /null
  1 ffff88007b315cc0 ffff88006046f800 ffff8800604464f0 REG  /home/guest/.xsession-
errors
  2 ffff88007b315cc0 ffff88006046f800 ffff8800604464f0 REG  /home/guest/.xsession-
errors
  3 ffff88001c571a40 ffff88001d605980 ffff88001be45cd0 REG  /home/guest/.mozilla/
firefox
  4 ffff88003faa7300 ffff880063d83440 ffff88001c315bc8 SOCK
  5 ffff88003f8f6a40 ffff88007b41f080 ffff88007aef0a48 FIFO
...
```

fuser
Displays the tasks that reference a specified file name or inode address as the current root directory, current working directory, open file descriptor, or that memory map the file. For example:

```
crash> fuser /home/guest


 PID         TASK        COMM             USAGE
 2990  ffff88007a2a8440  "gnome-session"  cwd
 3116  ffff8800372e6380  "gnome-session"  cwd
 3142  ffff88007c54e540  "metacity"       cwd
 3147  ffff88007aa1e440  "gnome-panel"    cwd
 3162  ffff88007a2d04c0  "nautilus"       cwd
 3185  ffff88007c00a140  "bluetooth-appl  cwd
...
```

irq
Displays interrupt request queue data. For example:

```
crash> irq 0


    IRQ: 0
 STATUS: 400000 ()
HANDLER: ffffffff81b3da30            <ioapic_chip>
         typename: ffffffff815cdaef  "IO-APIC"
          startup: ffffffff8102a513  <startup_ioapic_irq>
         shutdown: ffffffff810aef92  <default_shutdown>
           enable: ffffffff810aefe3  <default_enable>
          disable: ffffffff810aeecc  <default_disable>
              ack: ffffffff8102a43d  <ack_apic_edge>
             mask: ffffffff81029be1  <mask_IO_APIC_irq>
...
```

kmem
Displays the state of the kernel memory subsystems. For example:

```
crash> kmem -i


              PAGES        TOTAL      PERCENTAGE
 TOTAL MEM   512658         2 GB         ----
      FREE    20867      81.5 MB    4% of TOTAL MEM
      USED   491791       1.9 GB   95% of TOTAL MEM
```

```
    SHARED   176201     688.3 MB   34% of TOTAL MEM
   BUFFERS     8375      32.7 MB    1% of TOTAL MEM
    CACHED   229933     898.2 MB   44% of TOTAL MEM
      SLAB    39551     154.5 MB    7% of TOTAL MEM

TOTAL SWAP  1032190       3.9 GB         ----
 SWAP USED     2067       8.1 MB    0% of TOTAL SWAP
 SWAP FREE  1030123       3.9 GB   99% of TOTAL SWAP
```

`kmem` has a large number of options. For more information, enter `help kmem`.

log
Displays the kernel message buffer in chronological order. This is the same data that `dmesg` displays but the output can include messages that never made it to `syslog` or disk.

mach
Displays machine-specific information such as the `cpuinfo` structure and the physical memory map.

mod
Displays information about the currently installed kernel modules. The `-s` and `-S` options load debug data (if available) from the specified module object files to enable symbolic debugging.

mount
Displays information about currently mounted file systems.

net
Displays network-related information.

ps
Displays information about processes. For example:

```
crash> ps Xorg crash bash


   PID    PPID  CPU       TASK        ST  %MEM     VSZ    RSS  COMM
   2679   2677   0  ffff88007cbcc400  IN   4.0  215488  84880  Xorg
> 13362  11853   0  ffff88007b25a500  RU   6.9  277632 145612  crash
   3685   3683   1  ffff880058714580  IN   0.1  108464   1984  bash
  11853  11845   1  ffff88001c6826c0  IN   0.1  108464   1896  bash
```

pte
Translates a page table entry (PTE) to the physical page address and page bit settings. If the PTE refers to a swap location, the command displays the swap device and offset.

runq
Displays the list of tasks that are on the run queue of each CPU.

sig
Displays signal-handling information for the current context or for a specified PID or task.

swap
Displays information about the configured swap devices.

task
Displays the contents of the `task_struct` for the current context or for a specified PID or task.

timer
Displays the entries in the timer queue in chronological order.

vm
Displays the virtual memory data, including the addresses of `mm_struct` and the page directory, resident set size, and total virtual memory size for the current context or for a specified PID or task.

vtop
Translates a user or kernel virtual address to a physical address. The command also displays the PTE translation, `vm_area_struct` data for user virtual addresses, `mem_map page` data for a physical page, and the swap location or file location if the page is not mapped.

waitq
Displays tasks that are blocked on a specified wait queue.

Helper Commands

The following commands perform calculation, translation, and search functions:

ascii
Translates a hexadecimal value to ASCII. With no argument, the command displays an ASCII chart.

btop
Translates a hexadecimal address to a page number.

eval
Evaluates an expression and displays the result in hexadecimal, decimal, octal, and binary. For example:

```
crash> eval 4g / 0x100


hexadecimal: 1000000  (16MB)
    decimal: 16777216
      octal: 100000000
     binary: 0000000000000000000000000000000000000001000000000000000000000000
```

list
Displays the contents of a linked list of data objects, typically structures, starting at a specified address.

ptob
Translates a page number to its physical address (byte value).

ptov
Translates a physical address to a kernel virtual address.

search
Searches for a specified value in a specified range of user virtual memory, kernel virtual memory, or physical memory.

rd
Displays a selected range of user virtual memory, kernel virtual memory, or physical memory using the specified format.

wr
Writes a value to a memory location specified by symbol or address.

> **NOT_SUPPORTED:**
>
> To avoid data loss or data corruption, take great care when using the `wr` command.

Session Control Commands

The following commands control the `crash` session:

alias
Defines an alias for a command. With no argument, the command displays the current list of aliases.

exit, q, or quit
Ends the `crash` session.

extend
Loads or unloads the specified `crash` extension shared object libraries.

foreach
Executes the `bt`, `files`, `net`, `task`, `set`, `sig`, `vm`, or `vtop` command on multiple tasks.

gdb
Passes any arguments to the GNU Debugger for processing.

repeat
Repeats a command indefinitely until you type `Ctrl-C`. This command is only useful when you use `crash` to examine a live system.

set
Sets the context to a specified PID or task. With no argument, the command displays the current context.

Guidelines for Examining a Dump File

The steps for debugging a memory dump from a kernel crash vary widely, according to the problem.

The following are guidelines for some basic investigations that you can try:

- Use `bt` to trace the functions that led to the kernel panic.
- Use `bt -a` to trace the active task on each CPU. There is often a relationship between the panicking task on one CPU and the running tasks on the other CPUs. If the listed command is `cpu_idle` or `swapper`, no task was running on a CPU.

- Use `bt -l` to display the line number of the source files corresponding to each function call in the stack trace.
- Use `kmem -i` to obtain a summary of memory and swap usage. Look for a `SLAB` value greater than 500 MB and a `SWAP USED` value greater than 0%.
- Use `ps | grep UN` to check for processes in the `TASK_UNINTERRUPTIBLE` state (*D state*), usually because they are waiting on I/O. Such processes contribute to the load average and cannot be killed.
- Use `files` to display the files that a process had open.

You can shell indirection operators to save output from a command, to a file for later analysis, or to pipe the output through commands such as `grep`, as shown in the following example:

```
crash> foreach files > files.txt


crash> foreach bt | grep bash


PID: 3685   TASK: ffff880058714580  CPU: 1   COMMAND: "bash"
PID: 11853  TASK: ffff88001c6826c0  CPU: 0   COMMAND: "bash"
```