Oracle® Linux 7

Monitoring and Tuning the System
Abstract

Oracle® Linux 7: Monitoring and Tuning the System provides information and tasks for monitoring system performance by using the various utilities that are provided in Oracle Linux.

Document generated on: 2020-07-13 (revision: 10386)
# Table of Contents

Preface ...................................................................................................................... v
1 Monitoring the System and Optimizing Performance .............................................. 1
  1.1 About System Performance Problems .................................................................. 1
  1.2 Working With System Performance and Monitoring Utilities ......................... 1
  1.3 Working With the sosreport Utility .................................................................. 2
  1.4 Monitoring Usage of System Resources ............................................................. 3
    1.4.1 Monitoring CPU Usage ............................................................................. 4
    1.4.2 Monitoring Memory Usage ...................................................................... 5
    1.4.3 Monitoring Block I/O Usage ..................................................................... 5
    1.4.4 Monitoring File System Usage .................................................................. 6
    1.4.5 Monitoring Network Usage ...................................................................... 6
  1.5 Using the Graphical System Monitor ................................................................ 6
  1.6 Working With OSWatcher Black Box ................................................................ 6
    1.6.1 Installing OSWbb ..................................................................................... 6
    1.6.2 Running OSWbb ....................................................................................... 7
    1.6.3 Analysing OSWbb Archived Files ............................................................ 8
2 Working With Tuned ............................................................................................... 9
  2.1 About Tuned ...................................................................................................... 9
    2.1.1 About Tuned Profiles .............................................................................. 9
    2.1.2 About the Default Tuned Profiles ............................................................. 11
  2.2 About Static and Dynamic Tuning in Tuned ....................................................... 11
  2.3 Installing and Enabling Tuned From the Command Line ................................... 11
  2.4 Running Tuned in no-daemon Mode .................................................................. 12
  2.5 Administering the Tuned Service and Profiles .................................................. 12
    2.5.1 Listing Tuned Profiles ............................................................................ 12
    2.5.2 Activating a Tuned Profile ...................................................................... 13
    2.5.3 Disabling Tuned ...................................................................................... 13
3 Automating System Tasks ...................................................................................... 15
  3.1 About Automating Tasks .................................................................................. 15
  3.2 Configuring cron Jobs ...................................................................................... 15
  3.3 Configuring anacron Jobs ................................................................................. 17
  3.4 Running One-Time Tasks .................................................................................. 18
    3.4.1 Changing the Behavior of Batch Jobs ...................................................... 18
4 Working With Crash Dumps ................................................................................... 21
  4.1 About Kdump .................................................................................................... 21
    4.1.1 Configuring and Using Kdump .................................................................. 21
    4.1.2 Files Used by Kdump .............................................................................. 23
    4.1.3 Using Kdump with OCFS2 ...................................................................... 23
  4.2 Using the crash Debugger .................................................................................. 23
    4.2.1 Installing the crash Packages ................................................................... 24
    4.2.2 Running crash ......................................................................................... 24
    4.2.3 Kernel Data Structure Analysis Commands .............................................. 26
    4.2.4 System State Commands ....................................................................... 28
    4.2.5 Helper Commands ................................................................................... 31
    4.2.6 Session Control Commands ................................................................... 31
    4.2.7 Guidelines for Examining a Dump File ................................................... 32
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.

Audience

This document is intended for administrators who need to configure and administer Oracle Linux. It is assumed that readers are familiar with web technologies and have a general understanding of using the Linux operating system, including knowledge of how to use a text editor such as *emacs* or *vim*, essential commands such as *cd, chmod, chown, ls, mkdir, mv, ps, pwd*, and *rm*, and using the *man* command to view manual pages.

Document Organization

The document is organized into the following chapters:

- **Chapter 1, Monitoring the System and Optimizing Performance** provides a high-level summary of system monitoring and performance tuning and includes tasks for monitoring the system for optimal performance.
- **Chapter 2, Working With Tuned** describes how to administer the system by using the Tuned feature.
- **Chapter 3, Automating System Tasks** describes how to automate system auditing and logging.
- **Chapter 4, Working With Crash Dumps** describes how to diagnose problems with kernel dumps.

Related Documents

The documentation for this product is available at:

*Oracle® Linux 7 Documentation*

Conventions

The following text conventions are used in this document:

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

Documentation Accessibility

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

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.
Chapter 1 Monitoring the System and Optimizing Performance

Table of Contents

1.1 About System Performance Problems ................................................................. 1
1.2 Working With System Performance and Monitoring Utilities ................................... 1
1.3 Working With the sosreport Utility ..................................................................... 2
1.4 Monitoring Usage of System Resources ............................................................... 3
  1.4.1 Monitoring CPU Usage ............................................................................ 4
  1.4.2 Monitoring Memory Usage ...................................................................... 5
  1.4.3 Monitoring Block I/O Usage .................................................................... 5
  1.4.4 Monitoring File System Usage ............................................................... 6
  1.4.5 Monitoring Network Usage ..................................................................... 6
1.5 Using the Graphical System Monitor ................................................................... 6
1.6 Working With OSWatcher Black Box .................................................................. 6
  1.6.1 Installing OSWbb .................................................................................... 6
  1.6.2 Running OSWbb ..................................................................................... 7
  1.6.3 Analysing OSWbb Archived Files ......................................................... 8

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

1.1 About System Performance Problems

Many performance issues are the result of configuration errors. You can avoid such errors by using a validated configuration that has been pre-tested for the supported software, hardware, storage, drivers, and networking components. A validated configuration incorporates the best practices for Oracle Linux deployment and has undergone real-world testing of the complete stack. Oracle publishes more than 100 validated configurations, which are freely available for download. You should also refer to the release notes for recommendations on setting kernel parameters.

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 associated with HugePages cannot be swapped out, which forces the SGA to remain resident in memory.

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

1.2 Working With System Performance and Monitoring Utilities

Performance issues can be caused by any of a system's components, software or hardware, and by 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 allow you to collect information about system resource usage and errors, and can help you to identify performance problems 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.
Working With the sosreport Utility

The sosreport utility 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.

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

Use the following command to list the available plugins and plugin options.

```
# 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
```
1.4 Monitoring Usage of System Resources

You need to collect and monitor system resources regularly to provide you with a continuous record of a system. Establish a baseline of acceptable measurements under typical operating conditions. You can then use the 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 allows you to plan for future growth and to see how configuration changes might affect future performance.

To run a monitoring command every \textit{interval} seconds in real time and watch its output change, use the \texttt{watch} command. For example, the following command runs the \texttt{mpstat} command once per second:
Monitoring CPU Usage

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

```
# watch -n interval mpstat
```

If installed, the `sar` command records statistics every 10 minutes while the system is running and retains this 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:

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

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

```
# sar -o datafile interval count >/dev/null 2>&1 &
```

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

Oracle OSWatcher Black Box (OSWbb) and OSWbb analyzer (OSWbba) are useful tools for collecting and analysing performance statistics. For more information, see Section 1.6, “Working With OSWatcher Black Box”.

1.4.1 Monitoring CPU Usage

The `uptime`, `mpstat`, `sar`, `dstat`, and `top` utilities allow you to monitor CPU usage. When a 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, this 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 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:
Monitoring Memory Usage

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`).

1.4.2 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.

1.4.3 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` 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. 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.
1.4.4 Monitoring File System Usage

The `sar -v` command reports the number of unused cache entries in the directory cache (`dentrusd`) 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.

1.4.5 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.

1.5 Using the Graphical System Monitor

The GNOME desktop environment includes a graphical system monitor that allows you 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
```

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 `Help > Contents`.

1.6 Working With OSWatcher Black Box

Oracle OSWatcher Black Box (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.

1.6.1 Installing OSWbb

To install OSWbb:

3. Copy the file to the directory where you want to install OSWbb, and run the following command:

```bash
# 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
```

### 1.6.2 Running OSWbb

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

```bash
#./startOSWbb.sh [frequency duration]
```

The optional frequency and duration arguments specifying 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:

```bash
# ./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.

```bash
#./stopOSWbb.sh
```

OSWbb collects data in the following directories under the `oswbb/archive` directory:
<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oswifconfig</td>
<td>Contains output from <code>ifconfig</code>.</td>
</tr>
<tr>
<td>osviolstat</td>
<td>Contains output from <code>iostat</code>.</td>
</tr>
<tr>
<td>oswmeminfo</td>
<td>Contains a listing of the contents of <code>/proc/meminfo</code>.</td>
</tr>
<tr>
<td>oswmpstat</td>
<td>Contains output from <code>mpstat</code>.</td>
</tr>
<tr>
<td>oswnetstat</td>
<td>Contains output from <code>netstat</code>.</td>
</tr>
<tr>
<td>oswprvtnet</td>
<td>If you have enable private network tracing for RAC, contains information about the status of the private networks.</td>
</tr>
<tr>
<td>oswps</td>
<td>Contains output from <code>ps</code>.</td>
</tr>
<tr>
<td>oswslabinfo</td>
<td>Contains a listing of the contents of <code>/proc/slabinfo</code>.</td>
</tr>
<tr>
<td>oswtop</td>
<td>Contains output from <code>top</code>.</td>
</tr>
<tr>
<td>oswvmstat</td>
<td>Contains output from <code>vmstat</code>.</td>
</tr>
</tbody>
</table>

OSWbb stores data in hourly archive files named `system_name_utility_name_timestamp.dat`. Each entry in a file is preceded by a timestamp.

### 1.6.3 Analysing 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 installed Java version 1.4.2 or higher on your system. You can use `yum` to install Java, or you can download a Java RPM for Linux from [http://www.java.com](http://www.java.com).

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

```bash
# 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.

```bash
# java -jar oswbba.jar -i OSWbba_archive -A
```

Chapter 2 Working With Tuned

Table of Contents

2.1 About Tuned ................................................................. 9
  2.1.1 About Tuned Profiles .............................................. 9
  2.1.2 About the Default Tuned Profiles .......................... 11
2.2 About Static and Dynamic Tuning in Tuned .......................... 11
2.3 Installing and Enabling Tuned From the Command Line .............. 11
2.4 Running Tuned in no-daemon Mode ................................ 12
2.5 Administering the Tuned Service and Profiles ....................... 12
  2.5.1 Listing Tuned Profiles ............................................ 12
  2.5.2 Activating a Tuned Profile ...................................... 13
  2.5.3 Disabling Tuned .................................................. 13

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.

2.1 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 Section 2.2, “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.

2.1.1 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.
About Tuned Profiles

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

```
# 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 plug-in instances. Note that these sections are optional. For more information about profile configuration, see the `tuned.conf(5)` manual page.
2.1.2 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.

2.2 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
```

2.3 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:

   ```
   # yum install tuned
   ```

2. Enable and start the `tuned` service:

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

3. Check the active Tuned profile:

   ```
   # tuned-adm active
   Current active profile: balanced
   ```
4. Verify that the Tuned profile is applied to the system:

```
# tuned-adm verify
```

Verification 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:

```
# systemctl start tuned
```

### 2.4 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
```

**Warning**

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. Thus, tuned can no longer automatically roll back any settings files that were changed.

### 2.5 Administering the Tuned Service and 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.

#### 2.5.1 Listing Tuned Profiles

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

```
# 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
- network-throughput - Optimize for streaming network throughput, generally only necessary on older CPUs
- powersave      - Optimize for low power consumption
- throughput-performance - Broadly applicable tuning that provides excellent performance across a variety of 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:

```
# tuned-adm active
```
2.5.2 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:

```
# tuned-adm profile profile-name
```

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

```
# tuned-adm recommend
virtual-guest
```

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

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

2.5.3 Disabling Tuned

To disable tuning temporarily, use the following command:

```
# tuned-adm off
```

Running 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:

```
# systemctl disable --now tuned
```
Chapter 3 Automating System Tasks

Table of Contents

3.1 About Automating Tasks .................................................................................................................. 15
3.2 Configuring cron Jobs ..................................................................................................................... 15
3.3 Configuring anacron Jobs ............................................................................................................... 17
3.4 Running One-Time Tasks ............................................................................................................... 18
  3.4.1 Changing the Behavior of Batch Jobs ....................................................................................... 18

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.

3.1 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 utility, 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 utility 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 utility 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.

3.2 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 usually consists of definitions for the SHELL, PATH, MAILTO, and HOME variables for the
environment in which the jobs run, followed by the job definitions themselves. Comment lines start with a #
character. Job definitions are specified in the following format:

<table>
<thead>
<tr>
<th>minute</th>
<th>hour</th>
<th>day</th>
<th>month</th>
<th>day-of-week</th>
<th>user</th>
<th>command</th>
</tr>
</thead>
</table>

The following is a description of the fields that may be included:
Controlling Access to Running cron Jobs

Controlling Access to Running cron Jobs

minute 0-59.
hour 0-23.
day 1-31.
month 1-12 or jan, feb,..., dec.
day-of-week 0-7 (Sunday is 0 or 7) or sun, mon,..., sat.
user The user to run the command as, or * for the owner of the crontab file.
command The shell script or command to be run.

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

* (asterisk) All valid values for the field.
- (dash) A range of integers, for example, 1-5.
, (comma) A list of values, for example, 0,2,4.
/ (forward slash) A step value, for example, /3 in the hour field means 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

root can add job definition entries to /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 utility. 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.

### 3.3 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
```

```
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.
3.4 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`.

```bash
# 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:

```bash
# 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:

```bash
# 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:

```bash
# 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:

```bash
# atrm 1 2
```

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

3.4.1 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:

```bash
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:

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

```bash
# 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.
Chapter 4 Working With Crash Dumps

Table of Contents

4.1 About Kdump .......................................................... 21
  4.1.1 Configuring and Using Kdump .................................. 21
  4.1.2 Files Used by Kdump ............................................. 23
  4.1.3 Using Kdump with OCFS2 ....................................... 23
4.2 Using the crash Debugger ........................................... 23
  4.2.1 Installing the crash Packages .................................. 24
  4.2.2 Running crash .................................................... 24
  4.2.3 Kernel Data Structure Analysis Commands .................... 26
  4.2.4 System State Commands ....................................... 28
  4.2.5 Helper Commands ............................................... 31
  4.2.6 Session Control Commands ................................... 31
  4.2.7 Guidelines for Examining a Dump File ....................... 32

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 analyse the memory image in a crash dump or for a live system.

4.1 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 which enables a Linux kernel to boot from inside the context of a kernel that is already running without passing through the bootloader stage.

4.1.1 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.
   
   # 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 Allows you to specify the amount of memory to reserve for Kdump. The default setting is 128 MB.
Configuring and Using Kdump

Target Settings
Allows you to 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
Allows to select which 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
Allows you to 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
Files Used by Kdump

4.1.2 Files Used by Kdump

The Kernel Dump Configuration GUI modifies the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/boot/grub2/grub.cfg</td>
<td>Appends the crashkernel option to the kernel line to specify the amount of reserved memory and any offset value.</td>
</tr>
<tr>
<td>/etc/kdump.conf</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

4.1.3 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:

```
# 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.

4.2 Using the crash Debugger

The crash utility 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.
4.2.1 Installing the crash Packages

To use the `crash` utility, 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:

   ```
   # 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:

     ```
     export DLP="https://oss.oracle.com/ol7/debuginfo"
     wget ${DLP}/kernel-uek-debuginfo-`uname -r`.rpm
     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:

     ```
     export DLP="https://oss.oracle.com/ol7/debuginfo"
     wget ${DLP}/kernel-debuginfo-`uname -r`.rpm
     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:

     ```
     export DLP="https://oss.oracle.com/ol7/debuginfo"
     wget ${DLP}/kernel-uek-debuginfo-4.1.12-112.14.15.el7uek.x86_64.rpm
     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:

   ```
   # 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.

   ```
   rpm -Uvh 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` utility requires is installed in `/usr/lib/debug/lib/modules/kernel_version/`.

4.2.2 Running crash

**Warning**

Running the `crash` utility on a live system can cause data corruption or total system failure. Do not use the utility to examine a production system, unless directed to do so by Oracle Support.
To examine the currently running kernel, run the following command:

```bash
# crash
```

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

```bash
# 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:

```bash
# 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:

```bash
# 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
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)
```

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 Section 4.2.3, “Kernel Data Structure Analysis Commands”.

**System state commands**
Examine kernel subsystems on a system-wide or a per-task basis. See Section 4.2.4, “System State Commands”.

**Helper commands**
Perform calculation, translation, and search functions. See Section 4.2.5, “Helper Commands”.

**Session control commands**
Control the crash session. See Section 4.2.6, “Session Control Commands”.

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

### 4.2.3 Kernel Data Structure Analysis Commands

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

- **star**
  - The pointer-to command can be used instead of 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,
    }
    ```
Kernel Data Structure Analysis Commands

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
```

```
crash> sym -l
```

```
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);
}```
System State Commands

4.2.4 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_timeout at ffffffff811297d5
   /usr/src/debug/kernel-2.6.32/linux-2.6.32.x86_64/arch/x86/include/asm/current.h: 14
  #2 [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/kernel/entry_64.S: 488
  RIP: 00007fce20a66243  RSP: 00007fff552c1038  RFLAGS: 00000246
  RAX: 0000000000000017  RBX: ffffffff81011cf2  RCX: ffffffffffffffff
  RDX: 00007fff552c10e0  RSI: 00007fff552c1160  RDI: 000000000000000a
  RBP: 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
...
System State Commands

<table>
<thead>
<tr>
<th>BLKDEV</th>
<th>NAME</th>
<th>GENDISK</th>
<th>OPERATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ramdisk</td>
<td>ffff88007a3de800</td>
<td>brd_fops</td>
</tr>
<tr>
<td>259</td>
<td>blkext</td>
<td>(none)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>loop</td>
<td>ffff880037809800</td>
<td>lo_fops</td>
</tr>
<tr>
<td>8</td>
<td>sd</td>
<td>ffff8800378e9800</td>
<td>sd_fops</td>
</tr>
<tr>
<td>9</td>
<td>md</td>
<td>(none)</td>
<td></td>
</tr>
</tbody>
</table>

...  

```
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 ffff880038f6400 ffff880063d83440 ffff88001c315bc8 SOCK /home/guest/.mozilla/firefox
5 ffff880038f6400 ffff880063d83440 ffff88001c315bc8 SOCK /home/guest/.mozilla/firefox
```

**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  ffff88007a1e440  "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: ffffffff810ae92 <default_shutdown_ena>
enable: ffffffff810afe3 <default_enable>
disable: ffffffff810aeec <default_disable>
ack: ffffffff8102a43d <ack_apic_edge>
mask: ffffffff81029be1 <mask_IO_APIC_irq>
```
System State Commands

**kmem**

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

```
crash> kmem -i

<table>
<thead>
<tr>
<th></th>
<th>PAGES</th>
<th>TOTAL</th>
<th>PERCENTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL MEM</td>
<td>512658</td>
<td>2 GB</td>
<td>----</td>
</tr>
<tr>
<td>FREE</td>
<td>20867</td>
<td>81.5 MB</td>
<td>4% of TOTAL MEM</td>
</tr>
<tr>
<td>USED</td>
<td>491791</td>
<td>1.9 GB</td>
<td>95% of TOTAL MEM</td>
</tr>
<tr>
<td>SHARED</td>
<td>176201</td>
<td>688.3 MB</td>
<td>34% of TOTAL MEM</td>
</tr>
<tr>
<td>BUFFERS</td>
<td>8375</td>
<td>32.7 MB</td>
<td>1% of TOTAL MEM</td>
</tr>
<tr>
<td>CACHED</td>
<td>229933</td>
<td>898.2 MB</td>
<td>44% of TOTAL MEM</td>
</tr>
<tr>
<td>SLAB</td>
<td>39551</td>
<td>154.5 MB</td>
<td>7% of TOTAL MEM</td>
</tr>
<tr>
<td>TOTAL SWAP</td>
<td>1032190</td>
<td>3.9 GB</td>
<td>----</td>
</tr>
<tr>
<td>SWAP USED</td>
<td>2067</td>
<td>8.1 MB</td>
<td>0% of TOTAL SWAP</td>
</tr>
<tr>
<td>SWAP FREE</td>
<td>1030123</td>
<td>3.9 GB</td>
<td>99% of TOTAL SWAP</td>
</tr>
</tbody>
</table>
```

**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

<table>
<thead>
<tr>
<th>PID</th>
<th>PPID</th>
<th>CPU</th>
<th>TASK</th>
<th>ST</th>
<th>%MEM</th>
<th>VSZ</th>
<th>RSS</th>
<th>COMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>2679</td>
<td>2677</td>
<td>0</td>
<td>ffff88007bcc400</td>
<td>IN</td>
<td>4.0</td>
<td>215488</td>
<td>84880</td>
<td>Xorg</td>
</tr>
<tr>
<td>&gt; 13362</td>
<td>11853</td>
<td>0</td>
<td>ffff88007b5a500</td>
<td>RU</td>
<td>6.9</td>
<td>277632</td>
<td>145612</td>
<td>crash</td>
</tr>
<tr>
<td>3685</td>
<td>3683</td>
<td>1</td>
<td>ffff880058714580</td>
<td>IN</td>
<td>0.1</td>
<td>108464</td>
<td>1984</td>
<td>bash</td>
</tr>
<tr>
<td>11853</td>
<td>11845</td>
<td>1</td>
<td>ffff88001c6826c0</td>
<td>IN</td>
<td>0.1</td>
<td>108464</td>
<td>1896</td>
<td>bash</td>
</tr>
</tbody>
</table>
```

**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.
Helper Commands

**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.

### 4.2.5 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.

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

### 4.2.6 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.
4.2.7 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"
```