11.2.3 Resource Monitoring Facts

The goal of resource monitoring is to keep track of conditions on network resources and ensure those resources perform optimally. Monitoring will help you to identify problems, identify the source of those problems, identify situations that indicate potential problems, and identify resources that might need to be upgraded or changed. There are many tools available to help you monitor resources and ensure your entire system running smoothly.

This lesson covers the following topics:

- Automatic Bug Reporting Tool (ABRT)
- ABRT tools and commands
- CPU monitoring and configuration
- Memory monitoring and configuration
- Tools to troubleshoot hardware issues

Automatic Bug Reporting Tool (ABRT)

The Automatic Bug Reporting Tool (ABRT) is a tool that can be extremely helpful in troubleshooting a system crash. ABRT is a suite of tools that helps users detect, report, and resolve application crashes. ABRT analyzes the crash and takes steps to resolve the issue.

When a system crash occurs, ABRT:

- Collects data about the system.
- Produces backtrace from the core dump.
- Automatically generates and send reports.
- Creates a bug ticket.
- Identifies and makes available knowledge base articles relevant to the problem.
- Identifies software updates that resolves the issue.

The following table identifies ABRT components and their functions.

Component	Function
abrt	The ABRT utility consists of a daemon and a collection of tools for handling crashes and monitoring logs for errors.
gnome-abrt	GUI application for problem management and reporting.
libreport	A generic library that provides an API for reporting problems to different entities such as email, bugzilla, faf, scp upload, etc. By default, the notifications are sent to root at the local machine. You can use the conf file to change where notifications are sent.

faf	A crash collecting server, also known as the ABRT server. It provides accurate statistics of incoming reports and acts as a proxy in front of issue tracking software, such as bugzilla. It's designed to receive anonymous µReports and to find similar information among them. For known issues, it generates responses with links to faf's problem page, an issue tracker, or knowledge base entry.
satyr	Algorithms for program failure processing, analysis, and reporting. More specifically, satyr: • Generates a description of the failure from various stack traces • Analyzes stack traces of failed processes. • Discovers the thread that caused the failure, in multi-threaded stack traces. • Generates a failure report in a specified format • Sends the report to a remote machine.
retrace server	Provides dump analysis and backtrace generation service over a network using HTTP protocol. It is currently being merged to faf.
µReports	Micro reports, µReports, are small, machine readable reports used for automated reporting. They identify the operating system, versions of software running when the system crashed, information about the call stack at the time of the crash. a stack trace, or multiple stack traces in the case of multi-threaded programs. No private data. is allowed in the report.

To install ABRT and its utilities, run one or both of the following:

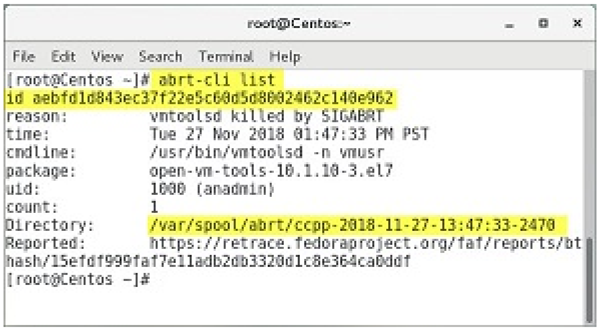
- **yum install abrt-cli**
  This will install the ABRT daemon and tools for use from the command-line.
- **yum install abrt-desktop**
  This will install the ABRT Desktop (ABRT GUI interface).

ⓘ The steps to install ABRT may be different depending on your distribution.

ABRT Tools and Commands

Use the following commands to manage ABRT information:

Command	Function	Examples

<table>
<tr>
<td><b>abrt-cli list</b></td>
<td>Lists all crashes on a machine.
Before this command can be used, you may need to enable the autoreporting feature. To enable autoreporting, run: <b>abrt-auto-reporting enabled</b></td>
<td><b>abrt-cli list</b>

<pre>
[root@Centos ~]# abrt-cli list
id aebfd1d843ec37f22e5c60d5d8002462c140e962
reason: vmtoolsd killed by SIGABRT
time: Tue 27 Nov 2018 01:47:33 PM PST
cmdline: /usr/bin/vmtoolsd -n vmusr
package: open-vm-tools-10.1.10-3.el7
uid: 1000 (anadmin)
count: 1
Directory: /var/spool/abrt/ccpp-2018-11-27-13:47:33-2470
Reported: https://retrace.fedoraproject.org/faf/reports/bt
hash/15efdf999faf7e11adb2db3320d1c8e364ca0ddf
[root@Centos ~]#
</pre>
</td>
</tr>
<tr>
<td><b>abrt-cli list -d <i><ID_OR_PATH></i></b></td>
<td>Displays detailed report data about a particular problem.</td>
<td><b>abrt-cli list -d aebfd1d843ec37f22e5c60d5d8002462c140e962</b>
<b>abrt-cli list -d /var/spool/abrt/<i>ccpp</i></b></td>
</tr>
<tr>
<td><b>abrt-cli report <ID_OR_PATH></b></td>
<td>Reports a problem.</td>
<td><b>abrt-cli report aebfd1d843ec37f22e5c60d5d8002462c140e962</b>
<b>abrt-cli list report /var/spool/abrt/<i>ccpp</i></b></td>
</tr>
<tr>
<td><b>abrt-cli remove <ID_OR_PATH></b></td>
<td>Deletes a problem.</td>
<td><b>abrt-cli remove aebfd1d843ec37f22e5c60d5d8002462c140e962</b>
<b>abrt-cli list remove /var/spool/abrt/<i>ccpp</i></b></td>
</tr>
</table>

CPU Monitoring and Configuration

CPU monitoring involves gathering information such as architecture, vendor name, model, number of cores, and speed of each core. The table below describes some of the Linux CLI commands you can use to monitor your CPU.

Command	Function
/proc/cpuinfo	Displays details about individual CPU cores. Output its contents with **less** or **cat**.
uptime	Displays of the following information. - Current time. - How long the system has been running - How many users are currently logged on - System load averages for the past 1, 5, and 15 minutes. Loads averages is a number that represents the average number of instructions waiting for CPU time.

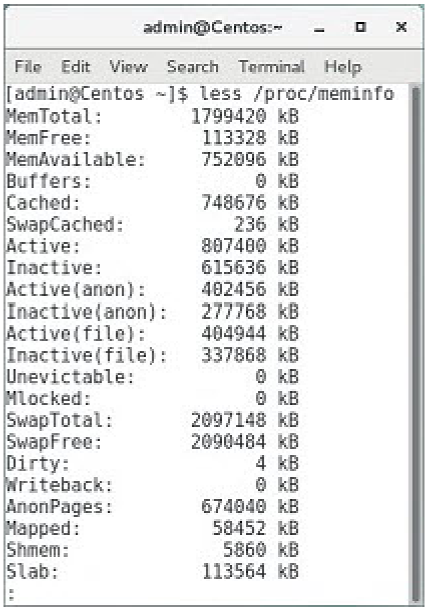
	- A load average of 1 (for a single processor system CPU) means that the CPU is keeping up with the demand or is 100% utilized. - A load average of less than 1 indicates that the CPU is underutilized in other words it has perfect utilization. - Any number greater than 1 indicates that the CPU has been asked to do more than it can do in real time, and some tasks will have to wait for CPU time. In multiprocessor systems, you use the same metrics multiplied by the number of processes. For example, a load average of 2 for a dual processor system indicates that the processors are keeping up with demand.
sar	The **sar** (System Activity Report) command is part of the sysstat package. The **sar** command reports cumulative values in the *count* and *interval* parameters of activity counters in the operating system. Example of information you can use **sar** to view is as follows: - Collective CPU usage - Individual CPU statistics - Memory used and available - Swap space used and available - Overall I/O activities of the system - Individual device I/O activities - Network statistics Be aware of the following about the **sar** command: - Writes information the specified number of times spaced at the specified intervals in seconds. If the *interval* parameter is set to zero, the **sar** command displays the average statistics for the time since the system was started. If the *interval* parameter is specified without the *count* parameter, then reports are generated continuously. The collected data can also be saved in the file specified by the **-o** *filename* flag, in addition to being displayed onto the screen. If *filename* is omitted, sar uses the standard system activity daily data file, the */var/log/sa/sa*dd file, where the dd parameter indicates the current day. By default all the data available from the kernel are saved in the data file. - The **sar** command extracts and writes to standard output records previously saved in a file. This file can be either the one specified by the **-f** flag or, by default, the standard system activity daily data file. - Without the **-P** flag, the **sar** command reports system-wide (global among all processors) statistics, which are calculated as averages for values expressed as percentages, and as sums otherwise. If the **-P** flag is given, the **sar** command reports activity which relates to the specified processor or processors. If **-P ALL** is given, the **sar** command reports statistics for each individual processor and global statistics among all processors. - You can select information about specific system activities using flags. Not specifying any flags selects only CPU activity. Specifying the **-A** flag is equivalent to specifying **-bBdqrRSvwWy -I SUM -I XALL -n ALL -u ALL -P ALL**. - The default version of the **sar** command (CPU utilization report) might be one of the first facilities the user runs to begin system activity investigation, because it

	monitors major system resources. If CPU utilization is near 100 percent (user + nice + system), the workload sampled is CPU-bound. - If multiple samples and multiple reports are desired, it is convenient to specify an output file for the **sar** command. Run the **sar** command as a background process. The syntax for this is: **sar -o datafile interval count >/dev/null 2>&1 &** - All data is captured in binary form and saved to a file (datafile). The data can then be selectively displayed with the **sar** command using the **-f** option. Set the *interval* and *count* parameters to select *count* records at *interval* second intervals. If the *count* parameter is not set, all the records saved in the file will be selected. Collection of data in this manner is useful to characterize system usage over a period of time and determine peak usage hours. - Note: The **sar** command only reports on local activities.
sysctl	This command lets you configure the kernel parameters at runtime. To view all of the available parameters, as root run **sysctl -a**. If needed, you can change one of these parameters which takes effect immediately, and if needed, the change can be written permanently using the **-w** switch. Example: **sysctl -w kernel.sysrq="1"** Keep in mind that there may not be any parameters that directly correlate to CPU usage, but instead, changing some parameters may have a direct impact on the CPU's performance. For example, tuning the network adapter by disabling the TCP timestamps option can result in better CPU utilization. This is really a powerful tool, so use it with care. There are a lot of kernel parameters that if changed could really mess your system up. Be sure to look at the man pages for sysctl before you start working with it, and be informed about any variable, and what it affects before changing the its value.

Memory Monitoring and Configuration

When working with computer memory, consider the following:

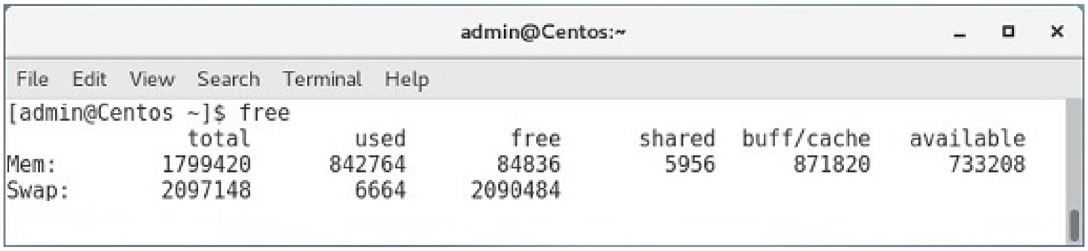
Command	Function
vmstat	vmstat reports information about processes, memory, paging, block IO, traps, disks, and CPU activity. The first report produced gives averages since the last reboot. Additional reports give information on a sampling period of length delay. The process and memory reports are instantaneous in either case.
/proc/meminfo	This file reports statistics about memory usage on the system. It is used by the free command (see below) to report the amount of free and used memory (both physical and swap) on the system as well as the shared memory and buffers used by the kernel. Each line of the file consists of a parameter name, followed by a colon, the value of the parameter, and an option unit of measurement (e.g., "kB").



```
[admin@Centos ~]$ less /proc/meminfo
MemTotal:        1799420 kB
MemFree:          113328 kB
MemAvailable:     752096 kB
Buffers:               0 kB
Cached:           748676 kB
SwapCached:          236 kB
Active:           807400 kB
Inactive:         615636 kB
Active(anon):     402456 kB
Inactive(anon):   277768 kB
Active(file):     404944 kB
Inactive(file):   337868 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:       2097148 kB
SwapFree:        2090484 kB
Dirty:                 4 kB
Writeback:             0 kB
AnonPages:        674040 kB
Mapped:            58452 kB
Shmem:              5860 kB
Slab:             113564 kB
:
```

free

The **free** command displays the total amount of free and used physical and swap memory in your computer, as well as the buffers and caches used by the kernel. The information is gathered by parsing /proc/meminfo.



```
[admin@Centos ~]$ free
              total        used        free      shared  buff/cache   available
Mem:        1799420      842764       84836        5956      871820      733208
Swap:       2097148        6664     2090484
```

Switches for the free command include the following:

Switch	Description
-b **-k** **-m**	The **-b** switch displays the amount of memory in bytes; the **-k** switch (set by default) displays it in kilobytes; the **-m** switch displays it in megabytes.
-t	The **-t** switch displays a line containing the totals.
-o	The **-o** switch disables the display of a "buffer adjusted" line. If the **-o** option is not specified, free subtracts buffer memory from the used memory and adds it to the free memory reported.
-s	The **-s** switch activates continuous polling delay seconds apart. You may actually specify any floating point number for delay, usleep is used for microsecond resolution delay times.
-l	The **-l** switch shows detailed low and high memory statistics.
-v	The -V switch displays version information.

Out of memory killer	When a Linux computer is critically low on memory, the Linux kernel uses the Out Of Memory Killer (OOM Killer) process to review all running processes and kill one or more of them in order to free up system memory and keep the system running. The Out Of Memory Killer works by reviewing all running processes and assigning them a badness score. The process that has the highest score is the one that is killed. The Out Of Memory Killer assigns a badness score based on a number of criteria, such as: • The process and its all of its child processes are using a lot of memory. • The minimum number of processes that need to be killed (ideally one) in order to free up enough memory to resolve the situation. • Root, kernel, and important system processes are given much lower scores. Viewing the system log is the best method used to determine if OOM Killer was that a program was killed. Another method is to run: **dmesg | grep -i "killed process"**
Buffer cache output	Buffers are used by programs with active I/O operations, i.e. data waiting to be written to disk. Cache is the result of completed I/O operations, i.e. buffers that have been flushed or data read from disk to satisfy a request.

Tools to Troubleshoot Hardware Issues

When troubleshooting hardware issues, consider the following:

Tool	Funciton
dmidecode	The dmidecode tool is used for dumping a computer's Desktop Management Interface (DMI) or SMBIOS table contents in a human-readable format. This table contains a description of the system's hardware components, as well as other useful pieces of information such as serial numbers and BIOS revision. Using this table, you can retrieve this information without having to probe for the actual hardware. While this is a good point in terms of report speed and safeness, this also makes the presented information possibly unreliable. While this is a good point in terms of report speed and safeness, this also makes the presented information possibly unreliable.
lshw	The lshw to is used to extract detailed information on the hardware configuration of the machine. It can report exact memory configuration, firmware version, mainboard configuration, CPU version and speed, cache configuration, bus speed, etc. on DMI-capable x86 or IA-64 systems and on some PowerPC machines (PowerMac G4 is known to work). It currently supports DMI (x86 and IA-64 only), OpenFirmware device tree (PowerPC only), PCI/AGP, CPUID (x86), IDE/ATA/ATAPI, PCMCIA (only tested on x86), SCSI and USB. -version Displays the version of lshw and exits.