In the previous chapter, we looked at Oracle performance monitoring tools. However, performance problems often occur outside the Oracle environment at the processor and memory, network, or storage level. It is therefore important to understand the information provided, not only by the Oracle performance monitoring tools, but also by the standard operating system monitoring tools available on Linux. You can use the information provided by these tools to support the findings from Oracle tools to fully diagnose RAC performance.

There are a number of third-party performance monitoring tools that operate in the Linux environment. However, our focus here is on the operating system monitoring tools available by default with Oracle Enterprise Linux that complement the environment available with the Oracle tools. In this category, we cover the default tools available in the base Oracle Enterprise Linux installation—namely, the command line CPU and memory diagnostics with uptime, last, ps, free, ipcs, pmap, lsof, top, vmstat, and strace; and network tools of netstat, ss, and tcpdump. Additionally, if you have installed and configured Oracle Enterprise Linux as detailed in Chapter 6, you will have run the Oracle Validated RPM. One dependency for the latter is the RPM package, sysstat. sysstat includes the following Linux performance monitoring tools: iostat, mpstat, and sar. Consequently, a default Oracle-validated Enterprise Linux environment includes a number of command-line tools that, if mastered, can rapidly and comprehensively give you insight into the system-level performance.

We also provide an overview of additional optional Oracle-provided Linux monitoring tools, as well as information on another open source tool you may wish to investigate. The Oracle tools are called Oracle Cluster Health Monitor and OSWatcher, respectively. The additional open source tool, which is provided by IBM, is called nmon. These tools are easy to install, but provide both an alternative and complementary environment for monitoring Linux environments.

It is important to note that, as is the case with all software the performance monitoring tools covered in this section, the tools just mentioned all require system resources to run, and you should be aware of the level of resources required by each tool. This information should be considered when deciding upon your Linux performance monitoring toolset; therefore, we do not recommend running all of the tools detailed in this section at the same time. Instead, you should select the ones that will work best in your environment.

The uptime and last Commands

uptime is a standard Linux command that reports the amount of time that a system has been running. The following snippet shows you how to use this command:

```
[root@london1 ~]# uptime
15:36:11 up 3 days, 3:50, 4 users, load average: 0.13, 0.14, 0.10
```
uptime provides information on node availability, and it is useful as a command of first resort in diagnosing and troubleshooting node evictions across a RAC cluster. `uptime` also reports the system load over intervals of 1, 5, and 15 minutes.

In a similar vein, the `last` command and its `-x` argument provides a detailed log of system shutdowns and changes in run level, as in this example:

```
[root@london1 ~]# last
root       pts/2    172.17.1.81   Fri Feb  5 09:32 still logged in
root       pts/1    london2.example. Thu Feb  4 16:09 still logged in
root       pts/1    london2.example. Thu Feb  4 16:04 - 16:05 (00:00)
root       pts/0    172.17.1.81   Thu Feb  4 16:00 still logged in
reboot     system boot 2.6.18-164.el5 Thu Feb  4 15:52 (17:53)
root       pts/2    172.17.1.81   Thu Feb  4 15:36 - down (00:13)
root       pts/1    172.17.1.81   Thu Feb  4 13:18 - down (02:31)
root       pts/3    172.17.1.81   Mon Feb  1 14:37 - down (3+01:11)
root       pts/1    172.17.1.81   Mon Feb  1 13:51 - 14:47 (00:55)
root       pts/2    172.17.1.81   Mon Feb  1 13:31 - 14:48 (01:17)
root       pts/1    172.17.1.81   Mon Feb  1 13:30 - 13:32 (00:02)
root       pts/0    172.17.1.81   Mon Feb  1 11:48 - down (3+04:01)
reboot     system boot 2.6.18-164.el5 Mon Feb  1 11:46 (3+04:02)
root       pts/0    172.17.1.81   Fri Jan 29 15:48 - down (00:09)
reboot     system boot 2.6.18-164.el5 Fri Jan 29 11:03 (04:54)
```

The `ps` Command

The `ps` command is one of the most basic, yet essential tools for analyzing performance on a Linux system. At its simplest, `ps` shows a list of processes; if called without arguments, it displays the list of processes running under the current session, as shown here:

```
[root@london1 ~]# ps
       PID  TTY         TIME CMD
      6969 pts/2 00:00:00 bash
      7172 pts/2 00:00:00 ps
```

Fortunately, `ps` can do a lot more than this. For example, it accepts a wealth of arguments to present process listings in almost every conceivable form. The arguments to `ps` can take three forms: standard System V Unix-type options that must be preceded by a dash; BSD-type options that are not preceded by a dash; and GNU long options that are preceded by two dashes. In effect, you may use different combinations of arguments to display similar forms of output. The combination of arguments that you will use most regularly is that of a full listing of all processes that relies on the System V `-ef` arguments. The following shows the first ten lines of output:

```
[root@london1 ~]# ps -ef
   UID   PID  PPID  C   STIME TTY          TIME CMD
root  1    0    0 Feb04 ? 00:00:02 init [3]
root  2    1    0 Feb04 ? 00:00:00 [migration/0]
root  3    1    0 Feb04 ? 00:00:00 [ksoftirqd/0]
root  4    1    0 Feb04 ? 00:00:00 [watchdog/0]
root  5    1    0 Feb04 ? 00:00:00 [migration/1]
root  6    1    0 Feb04 ? 00:00:00 [ksoftirqd/1]
```