Chapter 11 System Monitoring - Notes

11.3 Learning Objectives:

- Understand the concept of inventory and gain familiarity w ith available system monitoring tools.
- Understand w here the system stores log files and examine the most important ones.
- Use the `/proc` and `/sys` pseudo-filesystems.
- Use **sar** to father system activity and performance data and create reports that are readable by humans.

11.4 Available Monitoring Tools

Linux distributions comes w ith many standard performance/profiling tools already installed. Many familiar from other UNIX-like operating systems, w hile some developed specifically for Linux.

Most tools make use of mounted **pseudo-filesystems**, especially `/proc` and `/sys`, both w hich have already been discussed w hile examining filesystems/kernel configuration. Will look at both.

Also a number of graphical system monitors that hide many details, but w ill only consider command line tools in course.

Before considering main utilities in detail, can see summary on next few sections, broken dow n by type: note: some utilities have overlapping domains of coverage. Will revisit tables in follow ing chapters that focus on specific topics.

Summary of main process monitoring utility tools:

Process and Load Monitoring Utilities

Utility	Purpose	Package
top	Process activity, dynamically updated	**procps**
uptime	How long system is running and average load	**procps**
ps	Detailed information about processes	**procps**
pstree	Tree of processes and their connections	**psmisc** (or **pstree**)
mpstat	Multiple processor usage	**sysstat**
iostat	CPU utilization and I/O statistics	**sysstat**
sar	Display and collect information about system activity	**sysstat**
numstat	Information about **NUMA** (Non-Uniform Memory Architecture)	**numactl**
strace	Information about all system calls a process makes	**strace**

Memory Monitoring Utilities

Utility	Purpose	Package
free	Brief summary of memory usage	**procps**

vmstat	Detailed virtual memory statistics and block I/O, dynamically updated	**procps**
pmap	Process memory map	**procps**

I/O Monitoring Utilities

Utility	Purpose	Package
iostat	CPU utilization and I/O statistics	**sysstat**
sar	Display and collect information about system activity	**sysstat**
vmstat	Detailed virtual memory statistics and block I/O, dynamically updated	**procps**

Network Monitoring Utilities

Utility	Purpose	Package
netstat	Detailed networking statistics	**netstat**
iptraf	Gather information on network interfaces	**iptraf**
tcpdump	Detailed analysis of network packets and traffic	**tcpdump**
wireshark	Detailed network traffic analysis	**wireshark**

11.5 System Log Files

System log files -> essential for monitoring/troubleshooting. In Linux, messages appear in various files under `/var/log`. Exact names vary with Linux distribution.

Ultimate control of how messages dealt with -> controlled by **syslogd** (usually **rsyslogd** on modern systems) daemon, common to many UNIX-like operating systems. Newer **systemd**-based systems can use **journalctl** instead,but usually retain **syslogd** and cooperate with it.

Important messages sent not only to logging files, but also to system console window. If not running **X**, or are at virtual terminal, will see them directly there as well. In addition, messages will be copied to `/var/log` messages (or to `/var/log/syslog` on Ubuntu), but if running **X**, have to take some steps to view them.

Can view new messages continuously as new lines appear with:

```
$ sudo tail -f /var/log/messages (or /var/log/syslog)
```

or

```
$ dmesg -w
```

which shows only kernel-related messages.

11.6 Important Log Files in /var/log

Besides looking at log messages in terminal window, can see them using graphical interfaces.

On GNOME desktop, can also access messages by clicking on `System -> Administration -> System Log` or `Applications -> System Tools -> Log File` Viewer in your Desktop menus, and other desktops have similar links you can locate.

Some important log files found under `/var/log`:

File	Purpose
`boot.log`	System boot messages
`dmesg`	Kernel messages saved after boot. To see current contents of kernel message buffer, type **dmesg**.
`messages` or `syslog`	All important system messages
`secure`	Security related messages

In order to keep log files from growing without bound, **logrotate** program run periodically, keeps four previous copies (by default) of log files (optionally compressed). Controlled by `/etc/logrotate.conf`.

11.7 The /proc and /sys Pseudo-filesystems

`/proc` and `/sys` pseudo-filesystems contain lot of information about system. Many entries in these directory trees writable, can be used to change system behavior. Most cases, requires **root** user.

Pseudo-filesystems because totally exist in memory. If look at disk partition when system not running, there will be only empty directory which is used as mount point.

Information displayed is gathered only when looked at. No constant/periodic polling to update entries.

11.8 /proc Basics

`/proc` pseudo-filesystem: long history. Has roots in other UNIX operating system variants. Originally developed to display information about **processes** on system, each of which has own subdirectory in `/proc` with all important process characteristics available.

Over time, grew to contain lot of information about system properties, eg. interrupts, memory, networking, etc. in somewhat anarchistic way. Still extensively used, will often refer to it.

11.9 A survey of /proc

What resides in `/proc` pseudo-filesystem:

lsprocubuntu

First, see there is subdirectory for each process on system, whether sleeping, running, scheduled out. Looking at random one:

lsproc3589

Directory full of information about status of process and resources it is using. For example:

procstatus

Other entries give system-wide information. Eg. can see **interrupt** statistics in below screenshot. For each interrupt, see what type it is, how many times it has been handled on each CPU, which devices registered to respond to it. Also get global statistics.

procinterrupts

11.10 /proc/sys

Most tunable system parameters can be found in subdirectory tree rooted at `/proc/sys`:

procsys

Each subdirectory contains information + knobs that can be tuned (with care):

- `abi/`: Contains files with application binary information; rarely used
- `debug/`: Debugging parameters; for now, just some control of exception reporting
- `dev/`: Device parameters, including subdirectories for **cdrom**, **scsi**, **raid**, **parport**
- `fs/`: Filesystem parameters, including quote, files handles used, and maximums, inode and directory information, etc.
- `kernel/`: Kernel parameters. Many important entries here.
- `net/`: Network parameters. Subdirectories for **ipv4**, **netfilter**, etc.
- `vm/`: Virtual memory parameters. Many important entries here.

Viewing/changing parameters can be done with simple commands. Eg. maximum number of threads allowed on system seen by looking at:

```
$ ls -l /proc/sys/kernel/threads-max
$ cat /proc/sys/kernel/threads-max
129498
```

Can then modify value, verify change was effected:

```
$ sudo bash -c 'echo 100000 > /proc/sys/kernel/threads-max'
$ cat /proc/sys/kernel/threads-max
100000
```

Remember from discussion of **sysctl**, same effect accomplished by:

```
$ sudo sysctl kernel.threads-max=100000
```

Viewing value can be done as normal user, changing requires superuser privilege.

11.11 /sys Basics

`/sys` pseudo-filesystem: integral part of **Unified Device Model**. Conceptually, based on **device tree**, one can walk through it and see buses, devices, etc. Also now contains information which may or may not be strictly related to devices, such as kernel modules.

Has more tightly defined structure than `/proc`. Most entries contain only one line of text (although there are exceptions) unlike precursor which has many multi-line entries whose exact contents may change between kernel versions. Thus, interface hopefully more stable.

There are system properties which have display entries in both `/proc` and `/sys`. For compatibility with widely used system utilities, older forms only gradually being whittled down.

11.12 A Survey of /sys

Support for **sysfs** virtual filesystem built into all modern kernels, should be mounted under `/sys`. How ever, unified device model does not require mounting **sysfs** in order to function.

Taking look at 3.18 kernel (w arning; exact layout of this filesystem tends to mutate). Top level directory command yields:

```
$ ls -F /sys
block/ bus/ class/ dev/ devices/ firmware/ fs/ kernel/ module/ power/
```

w hich displays basic device hierarchy. Device model **sysfs** implementation also includes information not strictly related to hardw are.

Netw ork devices examined w ith:

```
$ ls -lF /sys/class/net
```

sysclassnet

Below, can see w hat looking at Ethernet card gives.

Intention w ith **sysfs** to have one text value per line, although not expected to be rigorously enforced.

sysclassnetcard

Underlying device and driver for first netw ork interface can be traced through `device` and (to be seen shortly) `driver` symbolic links. Below show s w hat can be seen w hen looking at directory corresponding to first Ethernet card.

To see full spectrum of information available w ith **sysfs**, w ill just have to examine.

sysclassnetcarddevice

11.13 sar

sar: **S**ystems **A**ctivity **R**eporter. All-purpose tool for gathering system activity + performance data, creating reports readable by humans.

On Linux systems, backend to **sar** is **sadc** (system activity data collector) w hich actually accumulates statistics. Stores information in `/var/log/sa` directory, w ith daily frequency by default, but w hich can be adjusted. Data collection can be started from command line, regular periodic collection usually started as **cron** job stored in `/etc/cron.d/sysstat`.

sar then reads in this data (either from default locations or by use of file specified w ith `-f` option), then produces report.

sar invoked via:

```
$ sar [ options ] [ interval ] [ count ]
```

w here report repeated after interval seconds a total of count times (defaults to 1). With no options, gives report on CPU usage.

sar

List of major **sar** options, or modes, each one of w hich has its ow n sub-options:

sar Options

Option	Meaning
`-A`	Almost all Information
`-b`	I/O and transfer rate statistics (similar to **iostat**)
`-B`	Paging statistics including page faults
`-x`	Block device activity (similar to **iostat -x**)
`-n`	Netw ork statistics
`-P`	Per CPU statistics (as in `sar -P ALL 3`)
`-q`	Queue lengths (run queue, processes, and threads)
`-r`	Sw ap and memory utilization statistics
`-R`	Memory statistics
`-u`	CPU utilization (default)
`-v`	Statistics about inodes and files and files handles
`-w`	Context sw itching statistics
`-W`	Sw apping statistics, pages in and out per second
`-f`	Extract information from specified file, created by the `-o` option
`-o`	Save readings in the file specified, to be read in later w ith `-f` option

For example, below can take look at getting paging statistics, and then I/O and transfer rate statistics.

ksar program -> **java**-based utility for generating nice graphs for **sar** data. Can be dow nloaded from https://sourceforge.net/projects/ksar/.

sario

##

Back to top

Previous Chapter - Table of Contents - Next Chapter