Monitoring System Resources

Using Linux Commands and Scripts

Asfaw Gedamu

12/03/2024

Monitoring System Resources Using Linux Commands

Description:

This guide outlines the steps for using various Linux commands to monitor system resources and manage Network File System (NFS). It covers:

- Identifying I/O bottlenecks using `find` and `iostat`.

- Monitoring CPU and memory resources using `iostat` and additional commands.
- Starting and stopping NFS services.
- Running commands in a loop using shell constructs.

Prerequisites:

- Basic understanding of Linux commands and navigation.
- Access to a Linux terminal or server with administrative privileges (root access for some tasks).

Target Audience:

- System administrators
- IT support personnel
- Anyone interested in monitoring and managing Linux system resources

Procedures:

1. Identifying I/O Bottlenecks

Find: Use the `find` command to locate specific files or directories that might be causing high disk activity. The `-atime`, `-ctime`, and `-mtime` options help filter by access, modification, or creation time, respectively.

Find all files in the current directory:

```
find . -type f
```

Find all files with a specific extension in the current directory:

```
find . -name "*.txt"
```

Find all files modified within the last 24 hours:

```
find . -mtime -1
```

Find all files larger than 100MB:

```
find . -size +100M
```

Find all files with a specific name and execute a command on them:

```
find . -name "example.txt" -exec cat {} \;
```

Find all files with a specific permission:

```
find . -perm 777
```

Find all files that are empty:

```
find . -empty
```

Find all files that are owned by a specific user:

```
find / -user myusername
```

Find all files that are in a specific group:

```
find / -group mygroupname
```

Find all files that are older than a certain date:

```
find / -type f -newermt "2022-01-01"
```

Find all directories in the current directory:

```
find . -type d
```

Find all files with a specific name and delete them:

```
find . -name "temp*" -delete
```

Find all files with a specific name and move them to a different directory:

```
find . -name "*.log" -exec mv {} /var/log/ \;
```

Find all files that are larger than a certain size and print their size:

```
find . -size +1G -exec ls -lh {} \;
```

Find all files that are older than a certain number of days and print their names:

```
find . -mtime +30 -exec ls {} \;
```

Find all files that have been accessed within the last 7 days and print their names:

```
find . -atime -7 -exec ls {} \;
```

Find all files that match a specific pattern and copy them to a different directory:

```
find . -name "*.txt" -exec cp {} /backup/ \;
```

Find all files that are symbolic links and print their target:

```
find . -type l -exec ls -l {} \;
```

Find all files that have been modified within the last hour and print their names:

```
find . -cmin -60 -exec ls {} \;
```

Find all files that have been modified within the last week and compress them:

```
find . -mtime -7 -exec gzip {} \;
```

Find all files that have been modified within the last 30 days and have a specific string in their content:

```
find . -mtime -30 -exec grep -l "string" {} \;
```

Find all files that have been modified within the last 30 days and have a specific string in their content and delete them:

```
find . -mtime -30 -exec grep -l "string" {} \; -delete
```

Find all files that are larger than a certain size and have a specific string in their content and move them to a different directory:

```
find . -size +1G -exec grep -l "string" {} \; -exec mv {}
/new_folder/ \;
```

Find all files that are smaller than a certain size and have a specific string in their content and print their name and size:

```
find . -size -100M -exec grep -l "string" {} \; -exec ls -lh {}
\;
```

Find all files that have been modified within the last year and have a specific string in their content and compress them with tar:

```
find . -mtime -365 -exec grep -l "string" {} \; -exec tar -czf
{}.tar.gz {} \;
```

Find all files that have been accessed within the last month and have a specific string in their content and print their name and last access time:

```
find . -atime -30 -exec grep -l "string" {} \; -exec stat -c "%n
last accessed on %x" {} \;
```

Find all files that have been modified within the last week and have a specific string in their content and mail them to a specific email address:

```
find . -mtime -7 -exec grep -l "string" {} \; -exec mail -s
"Files with specific string"
email@example.com < {} \;
```

Find all files that have been modified within the last 30 days and have a specific string in their content and run a specific command on them:

```
find . -mtime -30 -exec grep -l "string" {} \; -exec
your_command {} \;
```

Find all files that have been modified within the last year and have a specific string in their content and rename them:

```
find . -mtime -365 -exec grep -l "string" {} \; -exec mv {}
{}_newname \;
```

Find all files that have a specific string in their content and have been modified within the last year, and change their permission:

```
find . -mtime -365 -exec grep -l "string" {} \; -exec chmod 755
{} \;
```

Find all files that have been modified within the last year and have a specific string in their content and also have a specific file extension and print their name and path:

```
find . -mtime -365 -name "*.txt" -exec grep -l "string" {} \; -
exec stat -c "%n path: %h" {} \;
```

Find all files that have a specific string in their content and have been modified within the last year and also have a specific file extension and change their owner:

```
find . -mtime -365 -name "*.txt" -exec grep -l "string" {} \; -
exec chown new_owner {} \;
```

Find all files that have a specific string in their content and have been modified within the last year and also have a specific file extension and change their group:

```
find . -mtime -365 -name "*.txt" -exec grep -l "string" {} \; -
exec chgrp new_group {} \;
```

Find all files that have been modified within the last year and have a specific string in their content and also have a specific file extension and print their name and last access time:

```
find . -mtime -365 -name "*.txt" -exec grep -l "string" {} \; -
exec stat -c "%n last accessed on %x" {} \;
```

Find all files that have been modified within the last year and have a specific string in their content and also have a specific file extension and move them to a different directory:

```
find . -mtime -365 -name "*.txt" -exec grep -l "string" {} \; -
exec mv {} /new_folder/ \;
```

Find all files that have been modified within the last year and have a specific string in their content and also have a specific file extension and compress them with tar:

```
find . -mtime -365 -name "*.txt" -exec grep -l "string" {} \; -
exec tar -czf {}.tar.gz {} \;
```

Find all files that have been modified within the last year and have a specific string in their content and also have a specific file extension and mail them to a specific email address:

```
find . -mtime -365 -name "*.txt" -exec grep -l "string" {} \; -
exec mail -s "Files with specific string" email@example.com < {}
\;
```

Find all files that have a specific string in their content and have been modified within the last year and also have a specific file extension and rename them:

```
find . -mtime -365 -name "*.txt" -exec grep -l "string" {} \; -
exec mv {} {}_newname \;
```

Find all files that have been modified within the last year and have a specific string in their content and also have a specific file extension and change their permission:

```
find . -mtime -365 -name "*.txt" -exec grep -l "string" {} \; -
exec chmod 755 {} \;
```

iostat:

Use the `iostat` command to monitor disk I/O statistics. Analyze metrics like `tps` (transfers per second), `await` (average wait time), and `util` (device utilization) to identify potential bottlenecks.

Example: `iostat -x 2 5` - This shows extended device statistics every 2 seconds for 5 iterations.

Installing Iostat:

Iostat is not installed by default on most Linux systems. However, it can be easily installed using the package manager of your Linux distribution. On Ubuntu and Debian, you can use the following command to install iostat:

```
sudo apt-get install sysstat
```

On Red Hat/Oracle Linux and Fedora, you can use the following command to install iostat:

```
sudo yum install sysstat
```

Using Iostat:

Once iostat is installed, you can use it to monitor I/O statistics of your system. The basic syntax of the iostat command is:

```
iostat [options] [interval [count]]
```

The options specify the type of statistics to display, the interval specifies the time interval between two reports, and the count specifies the number of reports to display.

Identifying I/O Bottlenecks:

To identify I/O bottlenecks using iostat, you need to look at the following statistics:

%util: This is the percentage of time the disk was busy during the interval. A high value indicates a bottleneck.

await: This is the average time for an I/O operation to be completed. A high value indicates a bottleneck.

svctm: This is the average service time for an I/O operation. A high value indicates a bottleneck.

%iowait: This is the percentage of time the CPU spent waiting for I/O operations to complete. A high value indicates a bottleneck.

r/s and w/s: These are the number of read and write operations per second. A high value indicates a bottleneck.

Analyzing the Results:

Once you have collected the statistics, you need to analyze the results to determine if there is an I/O bottleneck. If the %util, await, svctm, and %iowait values are high, it indicates that there is a bottleneck. Also, if the r/s and w/s values are high, it indicates that the disk is busy.

Some common causes of I/O bottlenecks include slow disk I/O, high network traffic, and a high number of I/O operations.

Let's say that you are running a web server on a Linux system and you are experiencing slow performance. You suspect that the slow performance is caused by an I/O bottleneck. To confirm this, you can use iostat to monitor the I/O statistics of the system.

To view the I/O statistics of your system, you can run the following command:

```
iostat
```

This command will display the current I/O statistics of your system. By default, iostat will display the statistics for all the devices and partitions on your system.

You can also specify a specific device or partition to monitor by using the -d or -p option. For example, to monitor the I/O statistics of the sda1 partition, you can use the following command:

```
iostat -p sda1
```

You can also specify the interval and count for the statistics. For example, to view the I/O statistics every 2 seconds for a total of 5 times, you can use the following command:

```
iostat 2 5
```

You can also specify the statistics that you want to display. For example, to display only the device statistics and the CPU statistics, you can use the following command:

```
iostat -d -c
```

Some of the key statistics to look at when identifying I/O bottlenecks include %util, await, svctm, %iowait, r/s, and w/s. For example, to display these statistics you can use the following command:

```
iostat -x -d 2 5
```

There are other useful commands also that can be used to find I/O bottlenecks on a Linux system:

vmstat: vmstat is a command that can be used to monitor virtual memory statistics. It provides information on system activity, including the number of processes, memory usage, and I/O statistics.

```
vmstat -d 2 5
```

This command will display disk statistics every 2 seconds for a total of 5 times.

dstat: dstat is a versatile replacement for vmstat, iostat, netstat and ifstat. It can report resource statistics for the entire system, all individual processors, and specific processes.

```
dstat -d
```

This command will display disk statistics for the entire system.

iotop: iotop is a command that can be used to find the processes that are causing high I/O on a Linux system. It is similar to the top command, but it is specifically designed to monitor I/O activity.

```
iotop -o
```

This command will display the processes that are causing high I/O, sorted by the amount of I/O they are generating.

pidstat: pidstat is a command that can be used to monitor the process-level statistics. It can be used to monitor I/O statistics, memory usage, and other process-level statistics.

```
pidstat -d -p <PID>
```

This command will display disk statistics for a specific process, identified by its PID.

sar: sar is a command that can be used to collect, report and save system activity information. It can be used to monitor various system performance statistics, including I/O statistics, memory usage, and CPU usage.

```
sar -b 2 5
```

This command will display I/O statistics every 2 seconds for a total of 5 times.

2. Monitoring CPU and Memory Resources

- **iostat:** As mentioned earlier, `iostat` can also provide basic CPU utilization information under the `%idle` column.
- **top/htop:** Use `top` or `htop` for real-time monitoring of CPU usage, memory consumption, processes, and overall system load.
- **free/vmstat:** Use `free` to check available and used memory, or `vmstat` for a broader overview of memory usage, paging, and I/O statistics.

To displays information about the CPU architecture, clock speed, and other details about each core in the system.

```
$ cat /proc/cpuinfo
```

To provides a summary of the CPU architecture, including the number of CPUs, cores, and threads, as well as clock speed and other details.

```
$ lscpu
```

To provides real-time information about the system's CPU usage, including the percentage of CPU usage for each process, as well as the total CPU usage.

```
$ top
```

htop is Similar to top but it provides more detailed information and a more user-friendly interface.

```
$ htop
```

To provides detailed statistics about CPU usage, including the average usage for each core, as well as individual statistics for each core.

```
$ mpstat -P ALL
```

To list all running processes and their associated process IDs. You can use the -C option to filter the output by the command name and -o %cpu to sort the output by the CPU usage.

```
$ ps -C firefox -o %cpu
```

To provide detailed statistics about memory usage and CPU usage, including the number of processes in the run queue and the amount of free memory.

```
$ vmstat
```

To provide statistics on input/output operations, including the number of reads and writes, as well as the average response time and the amount of data transferred.

```
$ iostat
```

To provide system activity information, including CPU usage, memory usage, I/O activity, and network activity.

```
$ sar -u
```

To provide detailed statistics about process-level CPU usage, including the average usage for each process and the number of system calls made by each process.

```
$ pidstat -u
```

To show the number of processors or cores in your system.

```
$ nproc
```

To provide information about the system's memory usage, including the amount of free and used memory, as well as the amount of memory used by the system cache and buffers.

```
$ free
```

To show the system uptime, the number of users currently logged in, and the system load averages for the past 1, 5, and 15 minutes.

```
$ uptime
```

To provide detailed information about the system's hardware, including the CPU, memory, storage, and network devices.

```
$ lshw
```

To provides information about the system's GPU, including the GPU model, memory, and driver version, which can be useful if you are running applications that rely heavily on the GPU.

```
$ glxinfo
```

To provide an overview of system performance, including CPU usage, memory usage, disk I/O, and network activity.

```
$ dstat
```

To provide a detailed view of system performance, including CPU usage, memory usage, and disk I/O. It also shows the top processes by CPU and memory usage.

```
$ atop
```

To provide information about disk I/O usage, including the processes that are using the most I/O and the amount of data being transferred.

```
$ iotop
```

To provide a detailed view of a process's memory usage, including the amount of memory used by the process's code, data, and shared libraries.

```
$ pmap pid
```

To find the process ID of a running process by name.

```
$ pgrep processname
```

To provide detailed performance metrics, including CPU, memory, disk I/O, network, and filesystem statistics.

```
$ nmon
```

To provide detailed information about the system's CPU, including the clock speed, temperature, and power usage.

```
$ turbostat
```

3.Mount NFS to a directory

Mount Network File System (NFS) shares.

Syntax

```
mount [options] [NFS server]:[path on NFS server] [mount point]
```

[NFS server]: the hostname or IP address of the NFS server (OR YOU SOURCE SERVER FROM WHERE YOU WANT TO READ DATA)
[path on NFS server]: the path of the NFS share on the NFS server
[mount point]: the directory on the local file system where the NFS share will be mounted

For example, if the NFS server has the IP address 172.168.1.100 and the NFS share is located at /nfs/share, to mount the NFS share on local file system at /mnt, you can use the following command:

```
mount 172.168.1.100:/nfs/share /mnt
```

Note 1: The mount point should be created before you run the mount command.
Note 2: Also, the NFS server should be configured to export the share that you are trying to mount and the client should have the necessary permissions to access the share.

options that can be used with mount command to mount NFS share include :

-o : to specify various options like rw, ro, hard, soft etc
-t : to specify the file system type
-v : to be verbose

For example, to mount the share in read-only mode and with a soft mount option, you can use the following command:

```
mount -o ro,soft 192.168.1.100:/nfs/share1 /mnt
```

For example in our case we use the below command to mount our EBS /utlfile directory from oracle Cloud FSS to have better performance:

```
mount -t nfs -o \rw,bg,hard,timeo=600,nfsvers=3,tcp,nolock
\1.2.3.4:/FileSystem-erp-tst-utlfile /utlfile
```

Sometimes on any server, we have seen the mount command getting hung, and canceling and restarting nfs services helps it. Although not all the time this is the issue and this is not needed.

```
/sbin/service nfs restart
```

3.1. Starting and Stopping NFS

- **Starting NFS:** The specific commands to start NFS services may vary depending on your distribution. Common commands include:
    - systemctl start nfs-server
    - service nfs start
- **Stopping NFS:** Similarly, commands to stop NFS services can vary:
    - systemctl stop nfs-server
    - service nfs stop

To run an NFS server, the portmap service must be running. To verify that portmap is active, type the following command as root:

```
/sbin/service portmap status
```

If the portmap service is running, then the nfs service can be started. To start an NFS server, as root type:

```
/sbin/service nfs start
```

To stop the server, as root type:

```
/sbin/service nfs stop
```

The restart option is a shorthand way of stopping and then starting NFS. This is the most efficient way to make configuration changes take effect after editing the configuration file for NFS.

To restart the server, as root type:

```
/sbin/service nfs restart
```

The condrestart (conditional restart) option only starts nfs if it is currently running. This option is useful for scripts, because it does not start the daemon if it is not running.
To conditionally restart the server, as root type:

```
/sbin/service nfs condrestart
```

To reload the NFS server configuration file without restarting the service, as root type:

```
/sbin/service nfs reload
```

By default, the nfs service does not start automatically at boot time. To configure the NFS to start up at boot time, use an initscript utility, such as /sbin/chkconfig, /sbin/ntsysv, or the Services Configuration Tool program.

It is relatively easy to set up and use, and it provides fast access to shared files. However, it does not provide security features like encryption and authentication.

4. Running Commands in a Loop

- **Shell Loops:** Use looping constructs like `for`, `while`, or `until` to execute commands repeatedly.
    - Example: `for i in {1..5}; do echo "Iteration: $i"; done` - This loop prints "Iteration: 1" to "Iteration: 5".

There are several ways to run a command in a loop in Linux. Some of these with simple examples are listed below. This can prove handy for some DBAs who need to perform simple operation in repetitive way.

4.1. For loop:

You can use the for loop to execute a command a specified number of times.
The basic syntax is:

```
for i in {1..N};
do command;
done
```

Example: run the date command 10 times:

```
for i in {1..10};
do date;
done
```

4.2.While loop:

You can use the while loop to execute a command until a certain condition is met.
The basic syntax is:

```
while [ condition ];
do command;
done
```

Example: run the free -m command every 5 seconds:

```
while true;
do free -m;
sleep 5;
done
```

4.3.Until loop:

You can use the until loop to execute a command until a certain condition is met.
The basic syntax is:

```
until [ condition ];
do command;
done
```

Example: run the df -h command until the available disk space is less than 10%

```
until [[ $(df -h | awk '{print $5}' | grep -v Use | sed
's/%//g') -lt 10 ]];
do df -h;
sleep 5;
done
```

4.4. Watch command:

You can use the watch command to run a command repeatedly and display the output on the screen.
The basic syntax is:

```
watch -n interval command
```

For example, to run the top command every 2 seconds:

```
watch -n 2 top
```

4.5. Cron:

You can use cron to schedule a command to run at a specific time or interval.
The basic syntax is:

```
* * * * * command
- - - - -
| | | | |
| | | | ----- Day of week (0 - 7) (Sunday = both 0 and 7)
| | | ------- Month (1 - 12)
| | --------- Day of month (1 - 31)
| ----------- Hour (0 - 23)
------------- Minute (0 - 59)
```

```
Example: run the df -h command every 30 minutes
```

```
*/30 * * * * df -h >> /var/log/df.log
```

Depending on the task at hand, one of the above methods can be more suitable than the others.

Summary:

This document provides a basic understanding of using `find`, `iostat`, `top/htop`, `free/vmstat`, and shell loops for monitoring system resources and managing NFS services on Linux. Remember, specific commands and configurations might vary depending on your Linux distribution.

Important Notes:

- Always be cautious when running commands with root privileges.
- Consider using tools like `iotop` for a more in-depth analysis of I/O-intensive processes.
- Regularly monitor system resources to identify potential issues and ensure optimal performance.