

Performance Tuning Workshop on Linux

by -
Alok Srivastava

Why we need Performance Tuning?

- We generally use an OS in its native configuration.
- The overall load may vary from time to time, depending on the situation.
- So we need to continually monitor and tune our systems in the same manner like we change our car gears — depending upon the road and traffic conditions.
- Simply put, the objective of this workshop is to make Linux work more efficiently

Collecting Hardware data

- **vmstat** - vmstat provides information about processes, memory, paging, block I/O, traps, and CPU activity. Example – **vmstat 2 4**
- **dmidecode** – reads the system DMI table to display hardware and BIOS information of the system. Gives current configuration of your system, as well as the system's maximum supported configuration. Example – **dmidecode -t memory | grep -i maximum**
- **sar** - Part of the *sysstat* package. It reports disk's I/O transfer rates, paging activity, process activities, IRQ, n/w activity, memory & swap space & CPU utilization etc. It can optionally also take two arguments an interval in seconds between reports, and the number of times you want the report. Example – **sar -r 1 10 > somefilename**

put – alias sar="LANG=C sar" for 24 hours report time.

Collecting Hardware data

- **iostat** - used for monitoring the system input/output device loading by observing the time the devices are active in relation to their average transfer rates. Example – **iostat**
- **x86info** – useful tool that can be used to display a range of information about the CPUs present in an x86 system. Example – **x86info and x86info -c**
- **dumpe2fs** - displays very crucial filesystem information like volume, name, state, block size, etc. This information is very useful while tuning RAID and mounting external journals. Example – **dumpe2fs /dev/sda1**

Graphical Reporting of Data

- **gnuplot** - gnuplot is a plotting tool that we can use for graphical reporting of data and making more sensible reporting. Can be downloaded from www.gnuplot.info
- Create a file vi myfree.gnuplot

```
set xdata time
set timefmt "%H:%M:%S"
set xlabel "Time"
set ylabel "Memory (in Kb)"
plot "meminfo" using 1:2 title "FREE" with lines
replot "meminfo" using 1:5 title "BUFFERED" with lines
replot "meminfo" using 1:6 title "CACHED MEMORY" with lines
```

- Now give the command – **gnuplot -persist myfree.gnuplot**

Tuning Disk I/O using elevators

All modern HDDs (more than 90 per cent of all known brands) use a concept called **ZCAV** (Zonal Constant Angular Velocity).

This takes advantage of the fact that more linear space is available on the outer tracks of the disk platter rather than on the inside tracks. Now since the disk spins at a constant speed, which is also known as **CAV** (Constant Angular Velocity), the read/write I/O speed will be greater at the outer tracks as compared to the inner tracks. You can use a program called **Bonnie++** [www.coker.com.au/bonnie++] to check your hard disk.

Tuning sequential read access

The kernel, when reading a file, always tries to take advantage of sequential disk access. **Read-ahead** is based on the same assumption that if an application is accessing data from Block A, then the chances are more likely that the next Blocks—B, C and D will also need to be read. Therefore, by actually reading ahead and then caching the pages in memory, the kernel improves the I/O and reduces the response time. Turned off automatically whenever a random read request is detected. The read-ahead function is based on two values:

1. The current window—the application reads pages from the buffer cache that is its current window.
2. The 'ahead' window—while the application reads pages from the current window, IO happens on the 'ahead' window.

Tuning sequential read access

To view the current window size, issue

```
cat /sys/block/sda/queue/read_ahead_kb
```

You can tune it to 256 KB, for example, as follows:

```
echo 256 > /sys/block/sda/queue/read_ahead_kb  
cat /sys/block/sda/queue/read_ahead_kb
```

However, note that rebooting the machine will change the default value to 128 KB. Please feel free to use the **/etc/rc.local** file to make your changes permanent.

Tuning disk queue

The I/O subsystem is a series of processes with the responsibility to move blocks of data between the hard disk and the RAM. Generally, every computer task consists of a utility performing either one or both of the following:

Read a block of data off the disk and move it to RAM

Write a new block of data from the RAM to disk

Tuning disk queue

These read/write requests are transformed into 'block device requests' that go into a queue. When adding an entry to the queue, the kernel first tries to enlarge an existing request by merging the new request with one that is already in the queue.

If the request cannot be merged with an existing one, then the new request will be assigned a position in the queue based on several factors including the elevator algorithm. To view & change current queue length:

```
cat /sys/block/sda/queue/nr_requests
```

```
echo 170 > /sys/block/sda/queue/nr_requests
```

```
cat /sys/block/sda/queue/nr_requests
```

Thanks!!