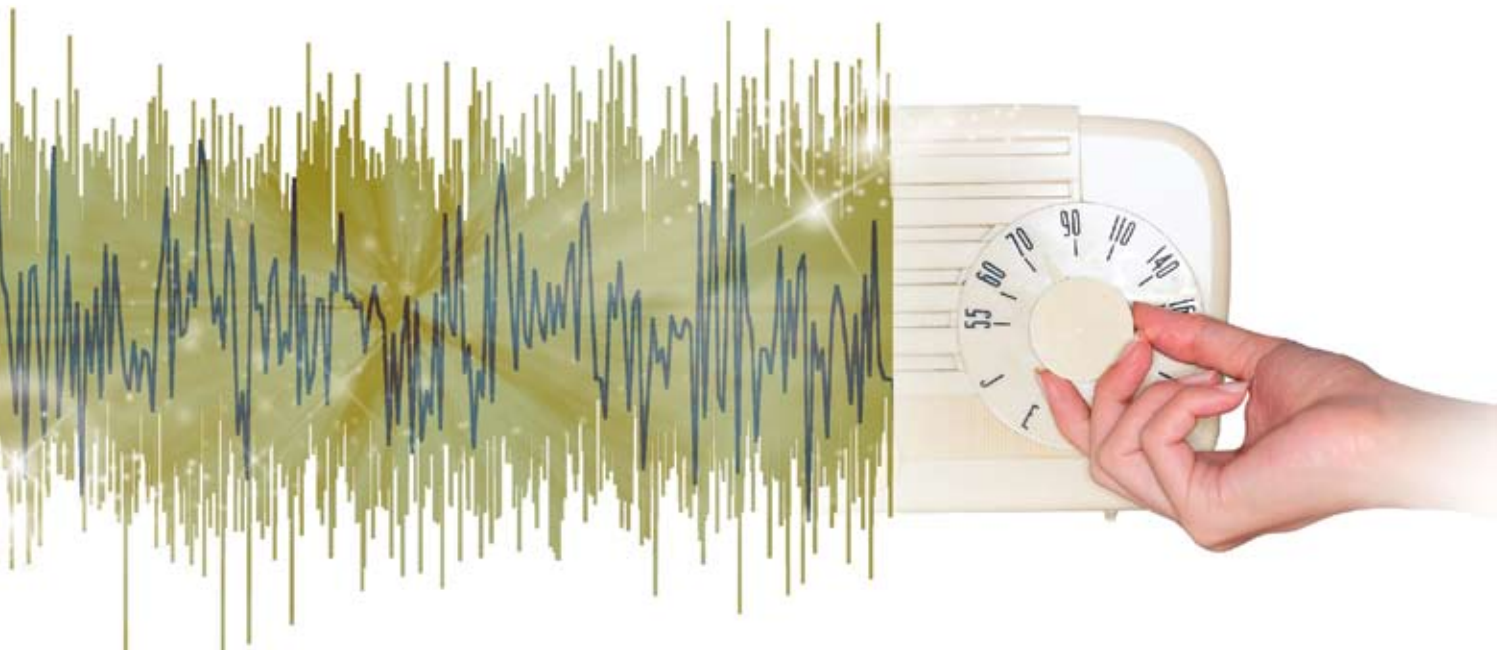


Performance Tuning and Monitoring



The first article in this series introduces a few utilities that we can use to generate system hardware reports and plot the obtained data graphically using gnuplot and MRTG.

We generally use an OS in its native configuration—i.e., without bothering much about how the system will behave or respond in the current load situation. 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 parts — depending upon the road and traffic conditions. Simply put, the objective of this series is to make Linux work more efficiently.

Performance tuning requires that you should have a sound knowledge of computer hardware—how the various computer components talk to each other, in addition

to the various components of the operating system. Collecting relevant data about a problem and analysing it is very important in successful performance tuning.

Let's start with some of the data gathering tools that can be used on Linux machines with kernel 2.6.x. The data gathered by these tools will then be used to analyse the problem, which will eventually lead to the solution.

Start with 'Beater Box'

Never consider performance tuning as some black art, with which you can give certain commands and your machine will start behaving just as you wanted it to. And never directly use all these labs/tools on a

production server. Instead spare yourself a ‘crash test dummy’, also referred to as ‘Beater Box’. Test several or all of the changes on that system to see the effects.

Collecting hardware configuration data

There are several tools that you can use to collect important information about your machine hardware. This is very important in efficient performance tuning.

vmstat

vmstat provides information about processes, memory, paging, block I/O, traps, and CPU activity. It displays either average data or actual samples. You can enable the sampling mode when you additionally provide a sampling frequency and a sampling duration to vmstat.

The following command will display my machine’s virtual memory report after a delay of two seconds, for four times.

```
# vmstat 2 4
```

Take a look at Figure 1 for the output of this command. Note that the first line of this report shows the averages since your last reboot. So there’s no need to panic and you should ignore it.

vmstat displays the following statistics:

1. Process (procs) section
 - r – number of processes waiting for runtime
 - b – number of processes in uninterruptible sleep
2. Memory section
 - swpd – amount of virtual memory used (KB)
 - free – amount of idle memory (KB)
 - buff – amount of memory used as buffers (KB)
 - cache – amount of memory used as cache (KB)
3. Swap section
 - si – amount of memory swapped from the disk (KB

per second)

- so – Amount of memory swapped to the disk (KB per second)

4. IO section

- bi – blocks sent to a block device (blocks/s)
- bo – blocks received from a block device (blocks/s)

5. System section

- in – number of interrupts per second, including the clock
- cs – number of context switches per second

6. CPU section

- us – time spent running non-kernel code (user time, including nice time)
- sy – time spent running kernel code (system time)
- id – time spent idle

dmidecode

The *dmidecode* command reads the system DMI table to display hardware and BIOS information of the system. This command will give you information on the current configuration of your system, as well as the system’s maximum supported configuration. For example, *dmidecode* gives both the current RAM on the system and the maximum RAM supported by the system.

To get information about your motherboard, I can use the following command:

```
# dmidecode -t baseboard
```

Given below is the output of this command on my system:

```
Handle 0x0200, DMI type 2, 9 bytes
Base Board Information
Manufacturer: Dell Inc.
Product Name: 0XD720
```

```
[root@legacy ~]# vmstat 2 4
```

procs		memory				swap		io		system			cpu			
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
1	0	0	1977920	144720	831928	0	0	27	13	246	258	2	1	96	2	0
1	0	0	1977912	144720	831932	0	0	0	0	456	498	0	0	99	0	0
0	0	0	1977912	144720	831932	0	0	0	4	439	497	0	0	99	0	0
0	0	0	1977912	144720	831932	0	0	0	18	443	485	0	0	98	2	0

Figure 1: A typical vmstat output

Version:

Serial Number: .BYNX3C1.

CN486436A14147

Asset Tag:

Handle 0x0200, DMI type 10, 6 bytes

On Board Device Information

Type: Video

Status: Eabled

Description: ATI MOBILITY Radeon X1400

Handle 0x0A01, DMI type 10, 6 bytes

On Board Device Information

Type: Sound

Status: Enabled

Description: Sigmantel 9200

In the same manner, you can get any information about your system. Check the man pages if you are not sure about the options. Running *dmidecode -t* will show you all the options that you can use:

dmidecode option requires an argument — ‘t’

Type number of keyword expected

Valid type keywords are:

```
biod
system
baseboard
chassis
processor
memory
cache
connector
slot
```

By using *dmidecode*, any of these options will give you detailed information about it. For instance, if I want to know about the CPU, I can now easily run: *dmidecode -t processor*

I can also use the *grep* with *dmidecode* to check how much RAM my system will support, as follows:

```
# dmidecode -t memory | grep -i Maximum
```

```
[root@legacy ~]# sar -u 1 10
Linux 2.6.29.6-217.2.3.fc11.i686.PAE (legacy) 08/11/2009

12:51:41 PM    CPU    %user    %nice    %system    %iowait    %steal    %idle
12:51:42 PM    all     13.93     0.00     36.82     0.00     0.00    49.25
12:51:43 PM    all     13.50     0.00     37.00     0.00     0.00    49.50
12:51:44 PM    all     13.50     0.00     37.50     0.00     0.00    49.00
12:51:45 PM    all     13.50     0.00     37.00     0.00     0.00    49.50
12:51:46 PM    all     12.56     0.00     38.19     3.02     0.00    46.23
12:51:47 PM    all     13.00     0.00     38.00     0.00     0.00    49.00
12:51:48 PM    all     13.07     0.00     37.19     0.00     0.00    49.75
12:51:49 PM    all     13.50     0.00     37.00     0.00     0.00    49.50
12:51:50 PM    all     14.43     0.00     36.32     0.00     0.00    49.25
12:51:51 PM    all     12.94     0.00     37.31     2.99     0.00    46.77
Average:      all     13.39     0.00     37.23     0.60     0.00    48.78
```

Figure 2: A sample sar output for CPU activity

```
[root@legacy ~]# sar -r 1 10
Linux 2.6.29.6-217.2.3.fc11.i686.PAE (legacy) 08/11/09

13:00:50    kbmemfree kbmemused  %memused  kbbuffers  kbcached  kswapfree  kswpused  %swpused  kswpcad
13:00:51    1108916    2247092     66.96    126380    1678400    8191992     0     0.00     0
13:00:52    1108860    2247148     66.96    126380    1678400    8191992     0     0.00     0
13:00:53    1108860    2247148     66.96    126380    1678400    8191992     0     0.00     0
13:00:54    1108860    2247148     66.96    126380    1678400    8191992     0     0.00     0
13:00:55    1108860    2247148     66.96    126380    1678400    8191992     0     0.00     0
13:00:56    1108860    2247148     66.96    126380    1678400    8191992     0     0.00     0
13:00:57    1108860    2247148     66.96    126380    1678400    8191992     0     0.00     0
13:00:58    1108860    2247148     66.96    126380    1678400    8191992     0     0.00     0
13:00:59    1108860    2247148     66.96    126380    1678400    8191992     0     0.00     0
13:01:00    1108860    2247148     66.96    126396    1678400    8191992     0     0.00     0
Average:    1108866    2247142     66.96    126386    1678400    8191992     0     0.00     0
```

Figure 3: A sample sar output for memory performance

```
[root@server1 ~]# iostat
Linux 2.6.18-92.el5xen (server1.example.com) 08/12/2009

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.19    0.08   0.20   0.52    0.01   99.00

Device:            tps    Blk_read/s    Blk_wrtn/s    Blk_read    Blk_wrtn
sda                  5.96         536.53         38.50    16207799    1163020
```

Figure 4: A typical iostat output

```
[root@server1 ~]# iostat -p /dev/sda
Linux 2.6.18-92.el5xen (server1.example.com) 08/12/2009

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.19    0.08   0.20   0.52    0.01   99.00

Device:            tps    Blk_read/s    Blk_wrtn/s    Blk_read    Blk_wrtn
sda                  5.98         538.30         38.55    16207751    1160692
sda8                  0.01           0.02           0.00         668         0
sda7                  0.04           0.05           0.01        1459        280
sda6                  0.17           0.39           0.07       11743       2208
sda5                  1.78        29.24         4.09     880469    123280
sda4                  0.00           0.00           0.00         8         0
sda3                  1.33        23.75         6.27     715122    188832
sda2                  8.97       484.76        28.10    14595946    846088
sda1                  0.03           0.06           0.00        1920         4
```

Figure 5: iostat output on the FTP server

Maximum Capacity: 4GB

So this is a very handy tool to know about your system's configuration and capabilities.

sar

The *sar* utility is a part of the *sysstat* package, so make sure you have it installed on your machine. The command collects and reports system activity information like

the disk's I/O transfer rates, paging activity, process-related activities, interrupts, network activity, memory and swap space utilisation, CPU utilisation, kernel activities, TTY statistics, etc. It can optionally also take two arguments—an interval in seconds between reports, and the number of times you want the report. But there are many other options that can be explored.

Figure 2 shows a sample output of my system's CPU activity collected by *sar*. I am using an interval of 1 second and I want the report to be produced 10 times using the command: *sar -u 1 10*

You can see from the figure that my system doesn't have too much of load on the CPU. One thing I want you to configure before using *sar* is to set the time in the 24-hour format, instead of the 12-hour format shown in Figure 2. This is because when we graphically chart all the information for our analysis, a 24-hour format makes more sense. You can configure an alias for *sar* as shown below:

```
# alias sar="LANG=C sar"
```

Now if you run the same command again you will get the time in the 24-hour format. Append the above alias in your *~/.bashrc* file to keep it across reboots.

If you want to know about your memory performance using *sar*, issue the following command:

```
sar -r 1 10.
```

(Figure 3 shows the output on my computer.)

A much better option to analyse the reports is to save them in some file and later, using a graphical plotter like *gnuplot*, plot the statistics. I can run *sar -r 1 10 > ~/mymeminfo* & to collect the report in a file that we can look at later.

iostat

According to its man page, "The *iostat* command is used for monitoring the system input/output device loading by observing the time the devices are active in relation to their average transfer rates. The reports then can be used to change the system configuration to better balance the input/output load between physical disks." The first line generated by *iostat* is the average since the last boot, so it can be ignored.

The simplest way you can use this command is by simply running *iostat* without any arguments (Figure 4 shows the output on my computer). It will show you the boot report of all devices and the CPU, since the last boot.

As you can see in Figure 4 the report also presents me the blocks-read per second (Blk_read/s), the blocks-written per second (Blk_wrtn/s), the total blocks-read (Blk_read) and total blocks-written (Blk_wrtn) since the last boot. This per-second report can be very handy in taking a decision for an upgrade.

I personally find this command very useful in figuring out which partition of my hard disk is under heavy I/O load. So I can use this report for either upgrading my disk or putting that mount on a separate disk. Figure 5 shows the output when I ran this command on my FTP server. (Just look at the output very carefully—it's self-explanatory.)

You can see */dev/sda2* is the partition with maximum blocks-read per second (Blk_read/s) and maximum blocks-read (Blk_read). Why?

I now issue the following command to tell you the reason:

```
# e2label /dev/sda2
/var
```

You can see that we have */var* mounted on this partition since it's an FTP server. So these large numbers are justified. I can use these in the future to mount my */var* on a separate high speed HDD if this number goes even higher.

x86info

x86info is a very useful tool that can be used to display a range of information about the CPUs present in an x86 system. The following is the output on my system:

```
x86info v1.20. Dave Jones 2001-2006
Feedback to <davej@redhat.com>.
```

```
Found 2 CPUs
```

```
-----
CPU #1
Family: 15 Model: 11 Stepping: 2
CPU Model : Athlon 64 CH7-CG
Processor name string: AMD Athlon(tm) 64 X2 Dual Core Processor
5000+
```

```
Feature flags:
fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush mmx fxsr sse sse2 ht sse3 cmpxchg16b

Extended feature flags:
syscall nx mmxext Fast fsave/fxrstor rdtscp lm 3dnowext 3dnow
```

```
lahf/sahf CMP legacy svm ExtApicSpace LockMovCr0 3DNowPrefetch
The physical package has 2 cores
```

```
-----
CPU #2
Family: 15 Model: 11 Stepping: 2
CPU Model : Athlon 64 CH7-CG
Processor name string: AMD Athlon(tm) 64 X2 Dual Core Processor
5000+
```

```
Feature flags:
fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush mmx fxsr sse sse2 ht sse3 cmpxchg16b

Extended feature flags:
syscall nx mmxext Fast fsave/fxrstor rdtscp lm 3dnowext 3dnow lahf/
sahf CMP legacy svm ExtApicSpace LockMovCr0 3DNowPrefetch
The physical package has 2 cores
```

```
-----
WARNING: Detected SMP, but unable to access cpuid driver.
Used Uniprocessor CPU routines. Results inaccurate.
```

This command is very useful in finding the L1/L2 cache of the CPU, which comes in very handy while configuring TLBs, etc. Just try running *x86info -c* to find the cache information of the CPU:

```
x86info v1.20. Dave Jones 2001-2006
Feedback to <davej@redhat.com>.
```

```
Found 2 CPUs
```

```
-----
CPU #1
Family: 15 Model: 11 Stepping: 2
CPU Model : Athlon 64 CH7-CG
Processor name string: AMD Athlon(tm) 64 X2 Dual Core Processor
5000+
```

```
Feature flags:
fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush mmx fxsr sse sse2 ht sse3 cmpxchg16b

Extended feature flags:
syscall nx mmxext Fast fsave/fxrstor rdtscp lm 3dnowext 3dnow lahf/
sahf CMP legacy svm ExtApicSpace LockMovCr0 3DNowPrefetch
Instruction TLB: Fully associative. 32 entries.
Data TLB: Fully associative. 32 entries.
```

```
L1 Data cache:
    Size: 64Kb 2-way associative.
    lines per tag=1 line size=64 bytes.

L1 Instruction cache:
    Size: 64Kb 2-way associative.
    lines per tag=1 line size=64 bytes.

L2 (on CPU) cache:
    Size: 512Kb 8-way associative.
    lines per tag=1 line size=64 bytes.
```

```
The physical package has 2 cores
```

CPU #2

Family: 15 Model: 11 Stepping: 2

CPU Model : Athlon 64 CH7-CG

Processor name string: AMD Athlon(tm) 64 X2 Dual Core Processor
5000+

Feature flags:

fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 ht sse3 cmpxchg16b

Extended feature flags:

syscall nx mmxext Fast fxsave/fxrstor rdtscp lm 3dnowext
3dnow lahf/sahf CMP legacy svm ExtApicSpace LockMovCr0
3DNowPrefetch

Instruction TLB: Fully associative. 32 entries.

Data TLB: Fully associative. 32 entries.

L1 Data cache:

Size: 64Kb 2-way associative.
lines per tag=1 line size=64 bytes.

L1 Instruction cache:

Size: 64Kb 2-way associative.
lines per tag=1 line size=64 bytes.

L2 (on CPU) cache:

Size: 512Kb 8-way associative.
lines per tag=1 line size=64 bytes.

The physical package has 2 cores

WARNING: Detected SMP, but unable to access cpuid driver.

Used Uniprocessor CPU routines. Results inaccurate.

You can see my CPU has an L1 (data cache) of 64 KB, an L1 (instruction cache) of 64 KB and an L2 cache of 512 KB. This information is very useful when tuning your CPU—which we'll do in later sessions.

dumpe2fs

This tool displays very crucial filesystem information like volume, name, state, block size, etc. This information is very useful while tuning RAID and mounting external journals. These topics will be covered in detail in the upcoming sessions.

The following is the output of *dumpe2fs /dev/sda1* which lists the details of my */dev/sda1* file system:

```
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: 515d8a09-85e4-45a7-bc77-3f7336234275
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: has_journal ext_attr resize_inode
dir_index filetype needs_recovery sparse_super
Filesystem flags: signed_directory_hash
Default mount options: user_xattr acl
Filesystem state: clean
```

```
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 51200
Block count: 204800
Reserved block count: 10240
Free blocks: 124082
Free inodes: 51132
First block: 1
Block size: 1024
Fragment size: 1024
Reserved GDT blocks: 256
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 2048
Inode blocks per group: 256
Filesystem created: Sat Jul 25 11:46:19 2009
Last mount time: Tue Aug 18 11:28:43 2009
Last write time: Tue Aug 18 11:28:43 2009
Mount count: 53
Maximum mount count: -1
Last checked: Sat Jul 25 11:46:19 2009
Check interval: 0 (<none>)
Reserved blocks uid: 0 (user root)
Reserved blocks gid: 0 (group root)
First inode: 11
Inode size: 128
Journal inode: 8
Default directory hash: half_md4
Directory Hash Seed: 82ccbbea-44f5-491f-b200-1370050b487f
Journal backup: inode blocks
Journal size: 4113k
<<output truncated>>
```

Graphical reporting of data

Till now we have gathered enough data from our systems. We will now plot this data graphically to analyse it in a better way. The tools that we will use here are *gnuplot* and *mrtg* for graphical reporting.

gnuplot

gnuplot is a plotting tool that we can operate in one of two modes:

1. Interactive mode: Here you issue a command at the *gnuplot* prompt to adjust a graph to your liking.
2. Batch mode: Here you can feed *gnuplot* the commands from a file to make it generate graphs in a batch.

Let's gather some data first using the *sar* command. Then we'll use the *gnuplot* command to visualise the data. Run *sar -r 2 100 > ~/meminfo* & to collect the data and store it in a file named *meminfo*. Remember to set *alias sar="LANG=C sar"* so that you have outputs in the 24-hour clock format. Figure 6 shows the output from the *~/meminfo* file.

Now, use a text editor (for example, *vi*) to open the file and remove any non-numeric lines—which basically means the first three lines of Figure 6 need to be deleted.

Make sure you have *gnuplot* installed. If it is not installed, use your distro's package manager or download it from www.gnuplot.info.

Create a file using a text editor with the following lines:

```
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
```

Save this as *myfree.gplot*. Now run the command: *gnuplot -persist myfree.gplot*. *gnuplot* will display a graphical view of the memory consumption trend as shown in Figure 7.

In the same manner, we can also display the graph for the disk activities, etc.

SNMP and MRTG

MRTG (Multi Router Traffic Grapher) can be a very handy tool for monitoring network traffic passing through each and every interface of your server. So when you finish monitoring your network traffic, you can configure your network as per your performance needs—which we will be discussing later.

Configuring MRTG is fairly simple. As MRTG polls SNMP (**Simple Network Management Protocol**) agents, we should first configure SNMP. Remember that MRTG works with Apache, so make sure you install that as well. Here is an MRTG configuration

Linux 2.6.18-92.el5xen (server1.example.com) 08/13/09									
13:32:59	kbmemfree	kbmemused	%memused	kbbuffers	kbcached	kbswpfree	kbswpused	%swpused	kbswpcad
13:33:01	429904	2485300	85.28	192468	1761860	4096392	140	0.00	140
13:33:03	429904	2485300	85.28	192468	1761864	4096392	140	0.00	140
13:33:05	429904	2485300	85.28	192484	1761856	4096392	140	0.00	140
13:33:07	429904	2485300	85.28	192492	1761864	4096392	140	0.00	140
13:33:09	429128	2485176	85.28	192492	1761864	4096392	140	0.00	140
13:33:11	429128	2485176	85.28	192500	1761864	4096392	140	0.00	140
13:33:13	429128	2485176	85.28	192500	1761864	4096392	140	0.00	140
13:33:15	429128	2485176	85.28	192500	1761864	4096392	140	0.00	140
13:33:17	429128	2485176	85.28	192500	1761864	4096392	140	0.00	140
13:33:19	429252	2485052	85.27	192516	1761864	4096392	140	0.00	140
13:33:21	429252	2485052	85.27	192524	1761856	4096392	140	0.00	140
13:33:23	429252	2485052	85.27	192524	1761864	4096392	140	0.00	140

Figure 6: Memory information output using *sar*

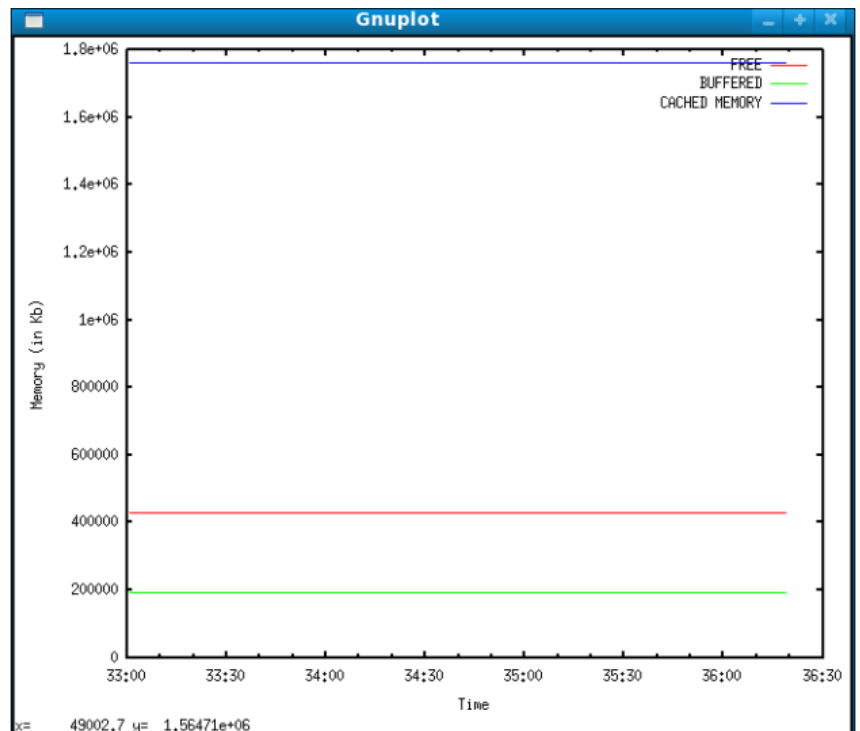


Figure 7: The graphical view of the memory consumption using *gnuplot*

for reference. I am configuring my FTP server that we had used for MRTG.. I am using it intentionally so that we have some real network traffic.

First, install SNMP, MRTG and Apache on the system using your package manager. Now, allow the SNMP through *tcp-wrapper*. Edit */etc/hosts.allow* and write *snmpd: ALL* to allow SNMP.

Use the *snmpconf* tool to create at least one SNMP community. In the first step, choose */etc/snmp/snmpd.conf*. Make sure you are inside the */etc/snmp/* directory before running *snmpconf*:

The following installed configuration files were found:

```
1: /root/.snmp/snmp.conf
```

Would you like me to read them in? Their content will be merged with the output files created by this session.

Valid answer examples: "all", "none", "3", "1,2,5"

Read in which (default = all): 1

In the next step, select *snmpd.conf*. Then select Access Control Setup.

The configuration information which can be put into *snmpd.conf* is divided into sections. Select a configuration section for *snmpd.conf* that you wish to create:

```
1: Access Control Setup
```

```
[root@server1 ~]# for x in $(seq 1 3); do LANG=C LC_ALL=C /usr/bin/mrtg /etc/mrtg/mrtg.cfg --lock-file /var/lock/mrtg/mrtg_1 --conffile /var/lib/mrtg/mrtg_1.ok; done
[root@server1 ~]#
[root@server1 ~]#
[root@server1 ~]# indexmaker /etc/mrtg/mrtg.cfg > /var/www/mrtg/index.html
[root@server1 ~]#
```

Figure 8: Terminal output mrtg configuration commands

2: Extending the Agent

3: Monitor Various Aspects of the Running Host

4: Agent Operating Mode

5: Trap Destinations

6: System Information Setup

Other options: finished

Select section: 1

access community name

Other options: finished, list

Configuring: rocommunity

Description:

a SNMPv1/SNMPv2c read-only access community name

arguments: community [default|hostname|network/bits] [oid]

The community name to add read-only access for: networknuts

In the next option, select “SNMPv1/SNMPv2c read-only access community name”. When prompted for the name, give the community name as ‘networknuts’ (or anything you like).

- Select from:
- 1: a SNMPv3 read-write user
 - 2: a SNMPv3 read-only user
 - 3: a SNMPv1/SNMPv2c read-only access community name
 - 4: a SNMPv1/SNMPv2c read-write

Enter 0.0.0.0/0 to specify that all networks are allowed to use this community. Though you can also specify your own specific IP range, as per your configurations. When prompted for the OID, just press *ENTER* and then type *Finished*. And then in the end, type *Quit*.

Select section: finished

I can create the following types of configuration files for you.
Select the file type you wish to create:
(you can create more than one as you run this program)

- 1: snmpd.conf
- 2: snmptrapd.conf
- 3: snmp.conf

Other options: quit

We had just finished configuring SNMP on our system. Now our job is to point MRTG to this community and display the graphical network statistics.

We have already installed the MRTG and Apache. So what’s left is to configure MRTG to use the ‘networknuts’ community.

Step 1

Edit /etc/httpd/conf/mrtg.conf to allow access from our network.

Allow .example.com

Step 2

Create a configuration file to monitor network traffic on SNMP agents:

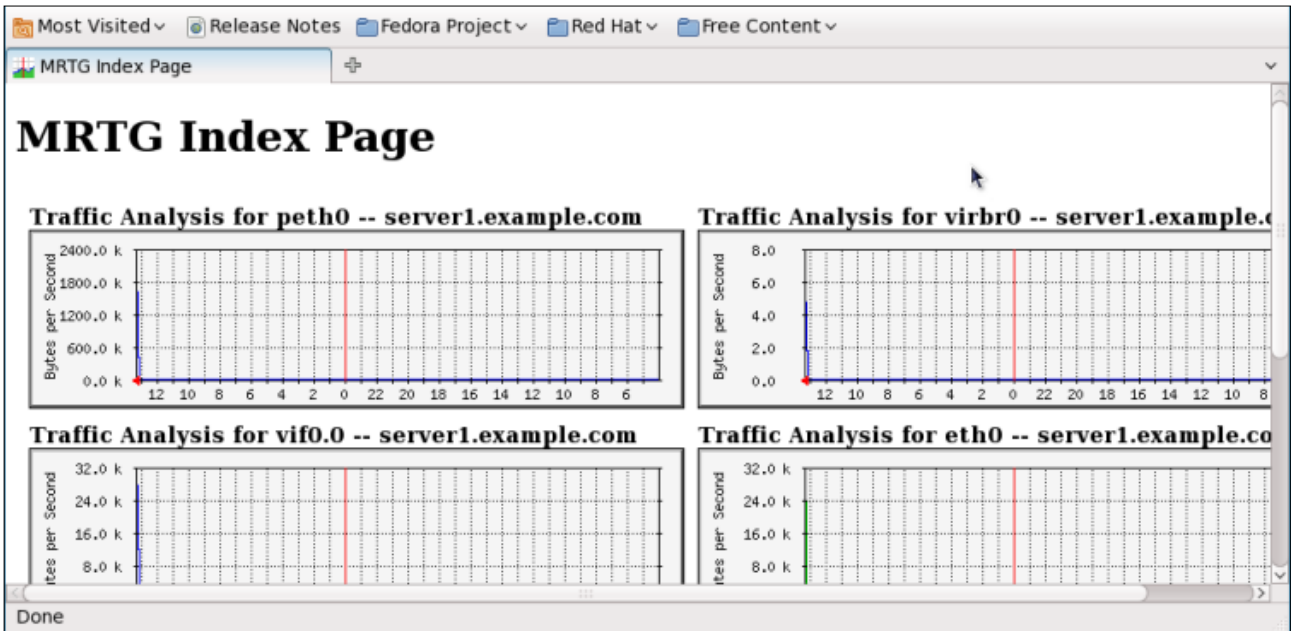


Figure 9: MRTG index page

```
cfgmaker - -ifref=name - -global "workdir:
/var/www/mrtg" \
networknuts@server1.example.com >
/etc/mrtg/mrtg.cfg
```

Step 3

Update MRTG to run after every four minutes by editing `/etc/cron.d/mrtg` as follows:

```
*/4 * * * * root LANG=C LC_ALL=C /usr/
bin/mrtg /etc/mrtg/mrtg.cfg --lock-file
/var/lock/mrtg/mrtg_l --conffile
/var/lib/mrtg/mrtg.ok
```

Step 4

Before closing the above file just copy everything after “root” (that is, starting with `LANG=C`) using the mouse. We will be using *cron* to populate our RRD file.

Step 5

Now execute the command given below:

```
for x in $(seq 1 3); do \
    LANG=C LC_ALL=C /usr/bin/mrtg
/etc/mrtg/mrtg.cfg --lock-file \
/var/lock/mrtg/mrtg_l --
conffile /var/lib/mrtg/mrtg.ok \
done
```

If you get any errors, don't worry and just repeat the same command again.

Step 6

Now run *indexmaker* to create the `/var/www/mrtg/index`.


html file:

```
indexmaker /etc/mrtg/mrtg.cfg > /var/
www/mrtg/index.html
```

Figure 8 shows the output of both the above commands.

Congrats, you have just configured SNMP and MRTG successfully. Time to restart your *snmp* and *http* services. Now open the browser and

view the MRTG output, by pointing your browser to *http://server1.example.com/mrtg*. You should see something similar to Figure 9.

Well, that's it for this session. Next time, we'll look at how to tune the HDD I/O using elevators. We'll also learn about the queuing theory and how to configure a performance RAID. **END** 

By: Alok Srivastava

The author is the founder of Network NUTS and holds MCP, MCSE, MCDBA, MCT, CCNA, CCNP, RHCE and RHCSS certifications. Under his leadership, Network NUTS has been a winner of the “Best Red Hat Training Partner in North India” for the last three years in a row. He has also been a trainer for the Indian Air Force, NIC, LIC, IFFCO, Wartsila India Ltd, the government of Rajasthan, Bajaj Allianz, etc.

Advt
Network Nuts