Monitoring Linux with Native Tools: Part Two

By Robert Andresen

In last month's article we discussed native Linux solutions for monitoring performance and collecting statistics for capacity planning. We covered the reasons to monitor Linux performance in order to meet the different needs of system administrators and capacity planners. We also covered several metrics to measure this performance including CPU Utility, memory, disk device and controllers, and new devices, and two types of tools to do so, including Real Time Displays and Static Commands. In the second part of this article, we will cover /proc filesystems and sysstat project and how they help monitor Linux performance.

/PROC FILESYSTEM

The first tool we'll cover is the /proc filesystem. Below is a pseudo-filesystem used to access kernel performance metrics as well as to update some kernel parameters. Scripts or programs may access this data merely by reading the appropriate files.

For example, if we do a directory listing for /proc, we see an entry for every running process, as well as entries for system level metrics. See Figure 1.

FIGURE 1: RUNNING PROCESS AND SYSTEM LEVEL METRICS

```
[root@localhost proc]# pwd
/proc
[root@localhost proc]# ls
1     1282  1333  1538  1591  1667  2262  91           ide          mtrr
1015  1283  1354  1541  1592  1668  2345  945          interrupts   net
1037  1284  1381  1544  1595  1669  2353  948          iomem        partitions
1067  1285  1400  1546  1598  1670  3     apm          ioports      pci
1086  1286  1426  1557  1637  1671  4     bus          irq          scsi
1104  1287  1427  1558  1644  1672  5     cmdline      kcore        self
1160  1288  1428  1568  1646  1989  6     cpuinfo      kmsg         slabinfo
1184  1289  1429  1569  1659  2     650   devices      ksyms        stat
12    1290  1430  1571  1660  2000  7     dma          loadavg      swaps
1217  1291  1431  1572  1661  2013  737   dri          locks        sys
1232  1292  1432  1574  1662  2014  742   driver       mdstat       sysvipc
1255  1293  1439  1579  1663  2015  763   execdomains  meminfo      tty
1278  1300  1441  1580  1664  2016  791   fb           misc         uptime
1280  1301  1442  1587  1665  2024  8     filesystems  modules      version
1281  1315  1459  1589  1666  2259  881   fs           mounts
```

Each number represents a running process, the names are system metrics from the kernel. If we were to look at /proc/stat we would see Figure 2.

FIGURE 2: /PROC/STAT

```
[root@localhost 2]# cat /proc/stat
cpu  12412 70 2968 506115
cpu0 12412 70 2968 506115
page 232980 82619
swap 1 0
intr 749494 521565 8314 0 17 6 2 6 3 1 3 2 96434 72512 0 25308 25321
disk_io: (3,0):(25524,19303,465378,6221,165216) (11,0):(18,18,72,0,0)
ctxt 2121146
btime 1076535165
processes 2475
```

The main page for proc shows all the values by file within /proc. For /proc/stat it shows Figure 3.

As you can see, these files are not user-friendly displays for diagnostic purposes. They are better suited to be accessed by programs or scripts which strip out needed metrics. They are where the other tools discussed in this paper get their metrics.

Figure 4 shows what is available for a running process.

SYSSTAT PROJECT

Thankfully, there is a project in Linux to mine the raw data out of the /proc filesystem and make it available for display as well as for building a historical database. The sysstat project is led by Sébastien Godard from France: Web links to project information are:

http://freshmeat.net/projects/sysstat/
http://perso.wanadoo.fr/sebastien.godard/

The project includes:

- *iostat:* Monitor system input and output device loading by comparing the time the devices are active in relation to their average transfer rates.
- *mpstat:* Monitors CPU activity, aggregate and individual CPU
- *sar:* Collect, save and report system activity metrics

IOSTAT

Iostat generates two reports, first CPU activity, followed by device utilization. See Figure 5.

Again notice the percentages of the four CPU states. The device section shows transfers per second (tps) by device, blocks read or written per second as well as the count of blocks read and written. The –d n c option will cause iostat to display a new report every n seconds for a count of c times.

MPSTAT

Mpstat displays processor utilization, percentage of time for each CPU state and the number of interrupts per second. See Figure 6.

If you want, the –V n c option may specify both an interval and a count to cause mpstat to redisplay every n seconds for a count of c times.

SAR

Sar provides three major functions:

- Create daily performance files with all system metrics
- Display metrics from a current or previous day's file
- Extract data from saved performance files in a format to be loaded into spreadsheets or databases.

File creation is based on the –o option. The data is saved in binary format in files named, by default: /var/log/sa/sadd. See Figure 7.

FIGURE 3: /PROC/STAT

cpu 3357 0 4313 1362393
The number of jiffies (1/100ths of a second) that the system spent in user mode, user mode with low priority (nice), system mode, and the idle task, respectively. The last value should be 100 times the second entry in the uptime pseudo-file.

page 5741 1808
The number of pages the system paged in and the number that were paged out (from disk).

swap 1 0
The number of swap pages that have been brought in and out.

intr 1462898
The number of interrupts received from the system boot.
disk_io: (2,0):(31,30,5764,1,2) (3,0):...
(major,minor):(noinfo, read_io_ops, blks_read, write_io_ops, blks_written)

ctxt 115315
The number of context switches that the system underwent.

btime 769041601
boot time, in seconds since the epoch (January 1, 1970).

processes 86031
Number of forks since boot.

FIGURE 4: RUNNING PROCESS

```
[root@localhost 2]# cat /proc/2/status
Name:    keventd
State:   S (sleeping)
Tgid:    2
Pid:     2
PPid:    1
TracerPid:      0
Uid:     0       0        0        0
Gid:     0       0        0        0
FDSize: 32
Groups:
SigPnd: 0000000000000000
SigBlk: ffffffffffffeffff
SigIgn: 0000000000010000
SigCgt: 0000000000000000
CapInh: 0000000000000000
CapPrm: 00000000ffffffff
CapEff: 00000000fffffeff
```

FIGURE 5: IOSTST REPOA

```
Linux 2.4.18-3 (dhcp64-134-114-41.hhwh.hou.wayport.net)
02/23/2004

avg-cpu:    %user    %nice    %sys    %idle
            10.26    0.00     2.82    86.91

Device:            tps    Blk_read/s    Blk_wrtn/s    Blk_read    Blk_wrtn
dev3-0           22.84        420.82         93.67      312066       69464
dev11-0           0.02          0.10          0.00          72           0
```

FIGURE 6: MPSTAT REPOA

```
Linux 2.4.18-3 (localhost.localdomain)  02/24/2004

10:51:55 AM  CPU    %user  %nice  %system   %idle    intr/s
10:51:55 AM  all     2.26   0.01     0.58   97.15    154.79
```

Notice the default naming convention will keep only a month of data in the binary files. You may override this with the –o option, or extract the data into another format with these parameters:

- *-e hh:mm:ss* Set the ending time of the report
- *-f filename* Extract records from filename
- *-h* When reading data from a file, print its contents in a format that can easily be handled by pattern processing commands like awk
- *-H* When reading data from a file, print its contents in a format that can easily be ingested by a relational database system
- *-i* interval Select data records at seconds as close as possible to the number specified by the interval parameter
- *-s* hh:mm:ss Set the starting time of the data
- *-t* When reading data from a daily data file, indicate that sar should display the timestamps in the original locale time of the data file creator

And what kind of data can be extracted? These parameters show what is saved in the binary files:

- *-b* Report I/O and transfer rate statistics
- *-B* Report paging statistics
- *-c* Report process creation activity
- *-d* Report activity for each block device (kernels 2.4 and later only)
- *-I irq* | SUM | PROC | ALL | XALL

Report statistics for a given interrupt:

- *-n* DEV | EDEV | SOCK | FULL

Report network statistics

- *-q* Report queue length and load averages
- *-r* Report memory and swap space utilization statistics
- *-R* Report memory statistics
- *-u* Report CPU utilization

A few examples of what this looks like: you can see CPU activity starting at 10:00 am. See Figure 8.

Or perhaps you would like to see swap data from February 24th, ending at 9:30 am. See Figure 9.

Hmmm, this is starting to remind me of SMF data from those old extinct IBM mainframes. (Remember them?) The system can write performance metrics to an internal file using a specified interval. You can extract the records you want and load them to a database or even a spreadsheet. See Figure 10 for what the extract output would look like if you chose the –H option for a relational database.

FIGURE 7: BINARY FILE NAMES

```
[root@localhost sa]# pwd
/var/log/sa

[root@localhost sa]# ls -al
total 88
drwxr-xr-x    2 root     root         4096 Feb 25 10:40 .
drwxr-xr-x    9 root     root         4096 Feb 25 10:56 ..
-rw-r--r--    1 root     root        21701 Feb 23 21:20 sa23
-rw-r--r--    1 root     root        42197 Feb 24 15:20 sa24
-rw-r--r--    1 root     root         9989 Feb 25 12:10 sa25
```

FIGURE 8: CPU ACTIVITY

```
[rda@localhost rda]$ sar -s 10:00:00
Linux 2.4.18-3 (localhost.localdomain)  02/24/2004

10:00:00 AM       CPU     %user     %nice   %system     %idle
10:10:00 AM       all      1.11      0.00      0.38     98.51
10:20:00 AM       all      4.18      0.00      0.85     94.97
10:30:00 AM       all      1.39      0.03      0.37     98.21
10:40:00 AM       all      1.61      0.00      0.44     97.95
10:50:00 AM       all      2.14      0.00      0.44     97.43
11:00:00 AM       all      1.87      0.00      0.45     97.68
11:10:00 AM       all      3.52      0.00      0.40     96.09
11:20:00 AM       all      2.77      0.00      0.42     96.80
11:30:00 AM       all      0.58      0.00      0.36     99.05
11:40:00 AM       all      3.99      0.00      0.62     95.39
11:50:00 AM       all      4.00      0.00      1.13     94.86
12:00:00 PM       all      5.05      0.00      0.52     94.43
12:10:00 PM       all      5.64      0.00      0.43     93.94

Average:          all      2.91      0.00      0.52     96.56
```

FIGURE 9: SWAP DATA

```
[root@localhost init.d]# sar -B -f /var/log/sa/sa24 -e 09:30:00
Linux 2.4.18-3 (localhost.localdomain)  02/24/2004

08:20:00 AM  pgpgin/s pgpgout/s  activepg  inadtypg  inaclnpg  inatarpg
08:30:00 AM      0.57      7.60     52313       698     10596     12721
08:40:00 AM      2.48     28.25     58419       976     10717     14022
08:50:00 AM      4.34      9.55     61104       993     10602     14539
09:00:00 AM     35.70      5.78     71026      1005     10616     16529
09:10:00 AM    114.21     21.06     95643      2583     10627     21770
09:20:00 AM      2.09      6.03     95654      2584     10627     21773
09:30:00 AM      2.39      4.63     96261      2584     10774     21923
Average:        26.57     13.04     75774      1632     10651     17611
```

FIGURE 10: EXTRACT OUTPUT

```
[root@localhost init.d]# sar -B -f /var/log/sa/sa24 -H

localhost.localdomain;600;2004-02-24 14:30:00 UTC;0.57;7.60;52313;698;10596;12721
localhost.localdomain;599;2004-02-24 14:40:00 UTC;2.48;28.25;58419;976;10717;14022
localhost.localdomain;600;2004-02-24 14:50:00 UTC;4.34;9.55;61104;993;10602;14539
localhost.localdomain;600;2004-02-24 15:00:00 UTC;35.70;5.78;71026;1005;10616;16529
localhost.localdomain;600;2004-02-24 15:10:00 UTC;114.21;21.06;95643;2583;10627;21770
localhost.localdomain;600;2004-02-24 15:20:00
localhost.localdomain;600;2004-02-24 18:10:00 UTC;0.05;4.58;82698;4861;11113;19734
localhost.localdomain;600;2004-02-24 18:20:00 UTC;0.13;5.04;83208;4862;11086;19831
localhost.localdomain;600;2004-02-24 18:30:00 UTC;0.11;8.31;80067;4812;11224;19220
```

If you save this as a text file, both Excel and Open Office will allow you to specify a semicolon as a field delimiter. See FIGURE 11.

Once you load your data to a spreadsheet or a database, you can generate performance reports and graphs. See FIGURE 12.

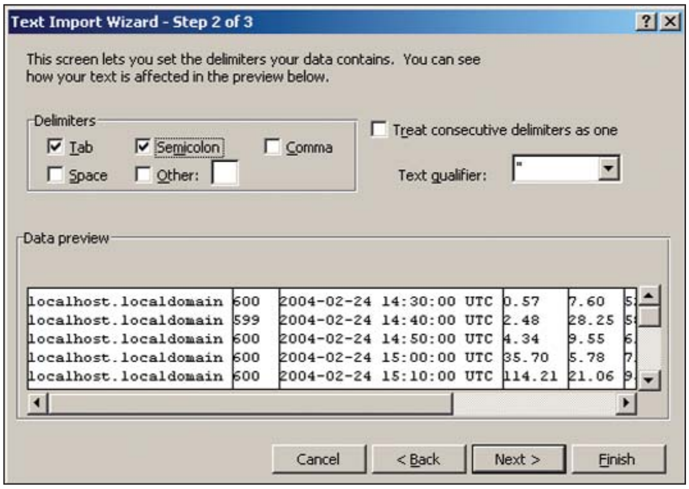
Now you have a tool to track Linux system performance over time and can make capacity planning predictions.

As Linux has become more stable and feature-rich, more and more shops are using it. Whether a company is embracing Linux because its a smaller company or a not-for-profit organization, or leveraging Linux with its server farms, or on the growing IBM zSeries platform, the metrics and tools mentioned in this article will help IT gain more out of what they have and how they use it. From there, IT is ready to begin aligning how they operate their Linux-based systems with the business objectives of their company.

NaSPA member Robert Andresen is a Principal Software Consultant with BMC Software in Chicago. He has been with BMC Software for five years, coming to BMC Software as part of their acquisition of Boole and Babbage. Andresen has been working with Linux since 1995 and is a co-author of the IBM Redbook: Linux on IBM @server zSeries and S/390: System Management. He holds a degree in Mathematics from the Illinois Institute of Technology.

At BMC Software he is focused on the MAINVIEW series of zSeries solution as well as PATROL solutions for Windows, Unix and MQSeries, providing installation and implementation services. His areas of expertise include z/OS, CICS, DB2, MQSeries, Networking and Unix.

FIGURE 11: SELECT SEMICOLON AS FIELD DELIMITER



Text Import Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

☑ Tab ☑ Semicolon ☐ Comma

☐ Space ☐ Other:

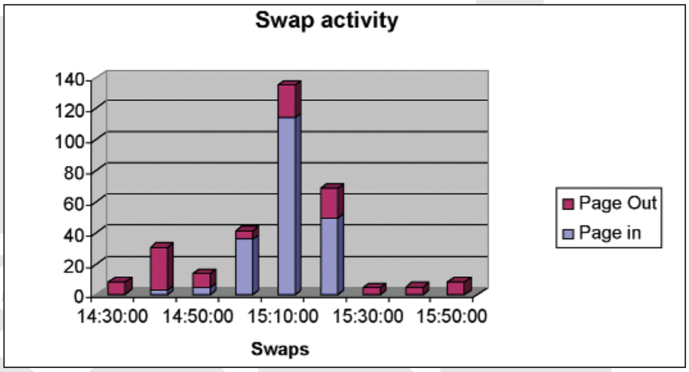
☐ Treat consecutive delimiters as one

Text qualifier: "

Data preview

```
localhost.localdomain 600 2004-02-24 14:30:00 UTC 0.57   7.60  5
localhost.localdomain 599 2004-02-24 14:40:00 UTC 2.48   28.25 5
localhost.localdomain 600 2004-02-24 14:50:00 UTC 4.34   9.55  6.
localhost.localdomain 600 2004-02-24 15:00:00 UTC 35.70  5.78  7.
localhost.localdomain 600 2004-02-24 15:10:00 UTC 114.21 21.06 9.
```

FIGURE 12: PERFORMANCE GRAPHS


Swap activity
14:30:00
14:50:00
15:10:00
15:30:00
15:50:00
Swaps
Page Out
Page in