Monitoring Linux with Native Tools—Part One

By Robert Andresen

LINUX is gaining interest as a solution across many hardware platforms: x86-based machines, Sun and Apple proprietary hardware and IBM zSeries platforms. But once applications are ported to an open source operating system, what options are available to monitor their performance and availability? The first section of this article covers native Linux solutions for monitoring performance and collecting statistics for capacity planning and will look at tools ranging from real time monitors through those that can build a database of historical system performance. The second section of this article, which will appear in the March/April issue, will conclude the review of tools.

REASONS TO MONITOR

Before Linux can run production applications, however, there will be the requirement to monitor its performance. There are ultimately two different purposes to monitoring any computer system, which basically map onto two different systems management functions. Systems administrators or systems programmers care more about the installation, tuning and troubleshooting of systems under their control. Capacity planners care more about building a performance database to analyze and predict resource consumption over time, looking to predict growth and upgrades required to sustain that growth.

Because of this difference of purpose, these groups will require different tools, though there are definite areas of overlap. Systems management generally requires tools to show what is happening *right now*, whereas capacity planning tends to be more concerned with trending resource usage to recognize growth and future bottlenecks *over time*.

There is overlap, of course, since you cannot do systems management trouble-shooting without understanding acceptable ranges of systems performance metrics. So, where the capacity planning function needs historical metrics to predict future growth bottlenecks, the systems management function needs similar historical data to understand what to look for in the metrics, and which values are an indication of a performance problem.

METRICS TO MEASURE

Ultimately, what the metrics systems administrators care about are much the same as for any computer system. They need to measure use of all physical resources, the usual suspects for x86 and zSeries:

- CPU utilization
- Memory
- Disk devices and controllers
- Network devices

They also need to measure use of system-level resources that may affect performance and capacity:

- Paging
- Swapping
- Inter-process communication constructs

There may be more system-level resources depending on the applications in use, e.g. database locks, but these monitors tend to be part of application support packages.

CPU UTILIZATION:

There are several measurements for CPU utilization, the percentage of available CPU being used, and the use by different states. The four CPU states in Linux are:

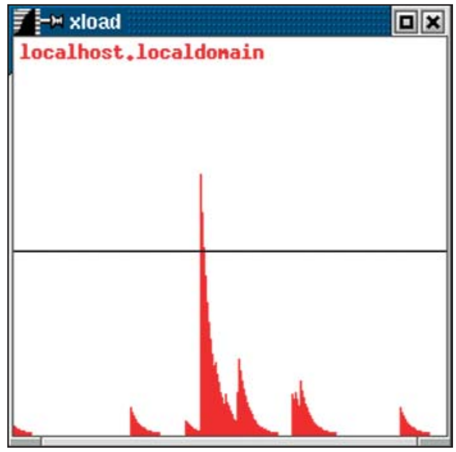
- User: Application use of CPU
- System: CPU used by kernel functions such as I/O or network
- Idle: CPU not being used, available for additional work
- Nice: User CPU use where the process has voluntarily lowered its priority to allow higher priority work to run

FIGURE 1: TOP COMMAND

```
  8:28pm  up 28 min,  2 users,  load average: 0.00, 0.06, 0.16
102 processes: 99 sleeping, 2 running, 1 zombie, 0 stopped
CPU states:  0.7% user,  1.5% system,  0.0% nice, 97.6% idle
Mem:  1031408K av,  577164K used,  454244K free,       0K shrd,    55344K buff
Swap:  514072K av,       0K used,  514072K free                   265008K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 7060 rda       15   0 11876  11M 10476 R     0.5  1.1   0:00 kdeinit
 1483 root      15   0 55744  11M  3064 S     0.3  1.1   0:06 X
 1615 rda       15   0  9864 9860  9048 S     0.3  0.9   0:01 kdeinit
 7093 rda       15   0   996  996   784 R     0.3  0.0   0:00 top
 6957 rda       15   0 50812  49M 40272 S     0.1  4.9   0:06 soffice.bin
    1 root      15   0   480  480   428 S     0.0  0.0   0:04 init
    2 root      15   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root      15   0     0    0     0 SW    0.0  0.0   0:00 kapmd
    4 root      34  19     0    0     0 SWN   0.0  0.0   0:00 ksoftirqd_CPU0
    5 root      15   0     0    0     0 SW    0.0  0.0   0:00 kswapd
    6 root      25   0     0    0     0 SW    0.0  0.0   0:00 bdflush
    7 root      15   0     0    0     0 SW    0.0  0.0   0:00 kupdated
    8 root      25   0     0    0     0 SW    0.0  0.0   0:00 mdrecoveryd
   12 root      15   0     0    0     0 SW    0.0  0.0   0:00 kjournald
   91 root      15   0     0    0     0 SW    0.0  0.0   0:00 khubd
  651 root      16   0   736  736   632 S     0.0  0.0   0:00 dhcpcd
  780 root      15   0   540  540   460 S     0.0  0.0   0:00 syslogd
```

FIGURE 2: XLOAD



Virtualization

Virtualized systems can affect CPU utilization measurements. Linux may not be running on the "bare metal" of the machine. It may be running in a virtual machine managed by VMWare (x86 platforms) or zVM (IBM zSeries platforms). This may cause the CPU percent numbers to be inaccurate. Linux may think it is running 80% of the CPU, but the virtual machine it is in may only be given 10% of the real CPU by the virtual machine manager. In this case the Linux instance would be consuming 8% of the real CPU.

For this reason, it becomes necessary to measure CPU percentages allowed by the virtualization manager, and then prorate the CPU percentages reported in each Linux by this amount. Getting the percentage with which to prorate will depend on the virtualization manager. VMWare can run on the bare metal of the machine, or under another operating system such as Windows or Linux. zVM provides this data and is available in a number of monitoring packages.

Memory

Linux uses both real memory and swap files, similar to virtual memory. Usually on an x86 platform the swap file is defined as twice the amount of real memory. On zSeries it turns out to be a mistake to over-allocate memory to the Linux systems in zVM, as the memory will be used up and cause additional paging at the zVM level. The recommendation is to give each Linux instance as little memory as it can get by on.

If a Linux system needs more real memory than is available it will use its swap file to free up some memory to allow another process' memory to be resident in real. At some point this swapping turns into thrashing, where a process gets swapped in, can't finish what it needs to do and is swapped out again.

Attempting to run X-Windows on a machine with not enough memory is a classic example of this. X-Windows spawns a number of processes that use up all the real memory and get swapped out by Linux. None of these processes get any of the work they intend to do, as Linux is using the entire CPU reading pages in and writing pages out. The goal should be to measure this activity and have enough real resources to match the anticipated application workload.

Disk Devices and Controllers

I/O is a major reason for delay on any platform and operating system. The old saying is that no matter how fast the processor, they all wait at the same speed. I/O operations tend to tie up the device and the controller for the life of the I/O. If other processes want access to the same device they will wait. As I/O increases, it is important to identify high-use file systems and balance them across multiple devices and controllers, if possible.

Network Devices

For most applications, there are increasing degrees of magnitude in delays caused by CPU (microseconds), disk (milliseconds) and network (tenths if not multiple seconds). Therefore, eliminating network bottlenecks may improve performance far more than fixing CPU or disk bottlenecks. Before bottlenecks may be eliminated, they must first be measured and identified. Important metrics include traffic by network device over time, as well as error retransmits and collisions.

TYPES OF TOOLS

Now that you know what to measure, we'll look at four different types of tools used by systems programmers for understanding the performance of the Linux system:

- Real time displays — Automatically refreshing system performance metrics
- Static commands — Displays a snapshot of system performance metrics
- /proc filesystem — Pseudo-filesystem that contains these metrics
- sysstat project — Linux project to display and collect these metrics

(Note: /proc filesystem and sysstat project will be covered in the second part of this article.)

REAL TIME DISPLAYS

top

The first example is the top command. Top is an auto-refreshing list of the processes using the most CPU. By default, it sorts them by CPU use. You may change the sort order, fields displayed and refresh rate either interactively or by configuration. See FIGURE 1.

Important fields to look at include:

Load average: These three numbers show the number of runable processes in the CPU queue over the past minute, five minutes and fifteen minutes. These numbers need to be compared to the number of CPUs available for that Linux instance.

CPU states: Shows the percentage of each of the four possible CPU states.

Mem & Swap: Shows how much real memory and swap space is available, in use and free. Also memory used for buffers, cache and shared memory (one of the interprocess communication options) is shown.

Process information: This covers the process number, owning user, priority, nice value, size (code + data + stack space), RSS (total amount of physical memory), shared memory in use, status (running, sleeping, nice value, swapped), %CPU being used, %Memory being used, CPU time since task has started, and command issued to start the process.

A common debugging use of top is when the CPU spikes, watch the output of top to see which process is causing the spike. That process may be stopped with a kill command.

xload

Xload is a graphical representation of CPU load. You may set the colors and refresh rate. See Figure 2.

It is more of an operational warning than a performance tool.

FIGURE 3: VMSTAT

```
[rda@localhost rda]$ vmstat -n 5
   procs                      memory    swap          io     system         cpu
 r  b  w   swpd   free   buff  cache  si  so    bi    bo   in    cs  us  sy  id
 0  0  0      0 536828  61192 254416   0   0    22    13  155   313   2   1  97
 1  0  0      0 536828  61192 254416   0   0     0     7  279   224   1   0  99
 0  0  0      0 536828  61192 254416   0   0     0     3  279   222   0   0  99
 0  0  0      0 536828  61192 254416   0   0     0     3  278   221   0   1  99
 0  0  0      0 536828  61192 254416   0   0     0    19  349   492   1   1  98
```

FIGURE 4: FREE

```
             total       used       free     shared    buffers     cached
Mem:       1031408     503616     527792          0      69076     242780
-/+ buffers/cache:     191760     839648
Swap:       514072          0     514072
```

FIGURE 5: PS

```
UID        PID  PPID  C STIME TTY          TIME CMD
rda       2315  2313  0 09:04 pts/3    00:00:00 /bin/bash
rda       2838  2315  0 10:28 pts/3    00:00:00 xload -fg red -bg white
rda       2910  2315  0 10:46 pts/3    00:00:00 ps -f
```

FIGURE 6: NETSTAT

```
[root@localhost init.d]# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
172.25.167.0    *               255.255.255.0   U        40 0          0 eth0
127.0.0.0       *               255.0.0.0       U        40 0          0 lo
default         172.25.167.1    0.0.0.0         UG       40 0          0 eth0

[root@localhost init.d]# netstat -i
Kernel Interface table
Iface    MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0    1500   0   30080      0      0      0    16393      0      0      0 BMNR
lo     16436   0   10563      0      0      0    10563      0      0      0 LRU
```

vmstat

Gives information about processes, memory, paging, block IO, traps and CPU activity. Do not confuse this with zVM, no native Linux tools know about zVM. See Figure 3.

The –n 5 parameter says refresh every 5 seconds. The first line produced gives averages since the last reboot. Additional lines give information on a sampling period of length delay. The process and memory reports are instantaneous in either case. The fields mean:

procs: r- number of processes waiting to run
b- number in uninterruptible sleep
w- swapped out, but otherwise runnable
memory: swap, free, buffers, cache
swap: swap ins, swap outs
io: blocks in, blocks out
system: interrupts per second (includes clock), context switches per second
cpu: user, system, idle

vmstat excludes itself from the statistics it presents.

STATIC COMMANDS

The static commands only give a snapshot of what is happening at the time the command was issued, they do not refresh. They may be re-issued if you want to see how the metrics are changing, or they may be invoked by shell scripts and the output stored in files for historical analysis later.

uptime

Uptime shows the same information as the first line of top: how long the system has been running and the load average numbers:

```
8:40pm up 40 min, 2 users, load average: 0.03, 0.06, 0.10
```

free

Displays similar information as the top Mem & Swap sections. See Figure 4.

ps

Displays the running processes according to the authority of the issuer and the parameters used. This is more of a performance diagnostic tool than performance reporting. See Figure 5.

This example (with the –f option) shows: user, process id, parent process id, child count, start time, associated terminal device, CPU time and command.

netstat

Displays network statistics with many different options to choose from. Common parameters are routes, interfaces and statistics. Netstat is also more of a diagnostic tool than performance reporting device. See FIGURE 6.

RX: received, TX: transmitted, OK, Error, Dropped, OVR (unable to transmit).

As Linux has become more stable and feature-rich, more and more shops are using it. The metrics and tools mentioned in this article will help IT gain more out of what they have and how they use it. The second part of this article, which will appear in the March/April issue, will discuss the remaining tools including /proc filesystem and sysstat project.

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.