System Administration Toolkit: Monitoring a slow system

Martin Brown June 07, 2006

When your UNIX® system runs slow, it is vital that you discover what the problem is as quickly as possible so you can get your system back into the normal operating mode. There are many causes for a slow system, but actually identifying the problem can be exceedingly difficult. In this article, study examples of how to identify and diagnose the cause of your slow running UNIX system to get your machine running properly again.

View more content in this series

About this series

The typical UNIX® administrator has a key range of utilities, tricks, and systems he or she uses regularly to aid in the process of administration. There are key utilities, command-line chains, and scripts that are used to simplify different processes. Some of these tools come with the operating system, but a majority of the tricks come through years of experience and a desire to ease the system administrator's life. The focus of this series is on getting the most from the available tools across a range of different UNIX environments, including methods of simplifying administration in a heterogeneous environment.

Causes of slow systems

There are many different potential causes of a slow system, but they can generally be slotted into the following areas:

- **Too many processes**. Your system is merely running too many applications at the same time, or running a small number of very CPU-intensive operations. Either you are overloading the server, or there is a runaway process that is sapping your system resources.
- **Too much active memory**. If your processes are using a lot of memory, then your system might be swapping a lot of pages in and out to disk and, that means, your system is spending more time swapping the memory than it is actually using it.
- **Hardware fault**. Occasionally you will come across a hardware fault that causes slow downs. A bad network card, hard drive, or memory might result in your system spending a long time waiting for information.

To diagnose the problem, you need to use a number of available tools to examine your UNIX system.

Choosing a connection method

If your machine is running particularly slow, then the first issue will be how you connect to your machine so that you can start the monitoring process. A slow machine might not accept connections through Telnet or over a remote shell protocol, such as ssh.

If you are not already logged in, getting access might be impossible. Instead, consider using the console, either directly or through a separate hardware solution, such as a network or serial based console monitor.

The console is more likely to allow you to log in, because there will already be a login process (which will be replaced with your shell) running. If, once you log in, you are unable to run any processes through your shell, it indicates that your system has run out of process space; a reboot will probably be the only way of returning your system to normal.

To reboot your system, use init or telinit to adjust the run level; run level 6 is usually synonymous with a reboot. Using init/telinit is more likely to reboot the system, as there is only one process involved to force the reboot.

Once the system is back up and running, you need to use some of the techniques in this article to monitor the running state of the system and record the output. If the slow system event happens again, you can then perform a post-mortem debug and examine the reason why your system runs so slowly.

Using uptime

If you suspect that your machine is running slowly, then the first command you should run is `uptime`. `Uptime` reports the current time, the amount of time the machine has been up and running (in other words, the time since the machine has booted), and the current number of users. It then provides three figures that show the load average in the last minute, five minutes, and fifteen minutes. For example:

```
$ uptime
 18:28:54 up 10 days,  8:38,  2 users,  load average: 2.24, 5.34, 3.42
```

In this example, the machine has had a load average of more than two in the last minute, more than five in the last five minutes, and more than three in the last fifteen minutes.

The definition of load average is complex and driven by the status of the processes that are executing. In general, each process that is running, waiting for CPU, or waiting on I/O would add one to the load average. These figures are calculated and then averaged over time.

In a single-CPU system, a load average of over one means that the CPU is under powered for the type of load you are supplying. But because of the multi-process nature of UNIX, it is generally

acceptable for a load average of up to two over the long term (in other words, the fifteen minute figure) before you should be concerned.

In a multi-CPU (or multi-core) system, divide the load average by the number of CPUs. To determine if the machine is over stressed, use the same principles as above.

An alternative way of looking at the figures is to treat them as a percentage; in other words, if the figures above were from a single-CPU system, then the machine would be able to cope with the load if it was 224 percent faster.

Within a multi-CPU system, you should use the number of CPUs plus one to determine the maximum load. For example, a four-CPU system would have a maximum load average of 5.

It is quite common to find a machine that has a load average that is significantly higher than its maximum for a short period. For example, when building or compiling an application or performing a very disk intensive task, your load averages can shoot up. This is why the output includes one, five, and fifteen minute averages, because it helps to smooth out any transient extreme loads.

Any long-term, or unexpected, high values probably signify a problem and need further investigation. If the numbers are low but your system is slow, it might point to a swap space issue.

Using ruptime

If you are managing a large network of systems, then there is a simple way of monitoring the load and usage of all the machines in your network. The ruptime tool collects data that is broadcast by all the machines on the network and collects it into a local file so that the current status of all the machines can be examined.

For example, Listing 1 shows the output from a small network:

Listing 1. Output from a small network

```
$ ruptime
bear          up 10+09:13,     2 users,  load 0.66, 0.68, 0.50
ultra3        up  6+01:16,     1 user,   load 0.00, 0.00, 0.00
atuin       down  4+00:52
```

The last machine has not reported any data for eleven minutes, so it is officially listed as down.

To generate the information, the rwhod daemon (sometimes in.rwhod) needs to be running on each machine in your local network. This daemon broadcasts the information for the local machine and collects the broadcast data from all the other machines.

Because of the way the rwho/ruptime system works, there can be performance issues, especially in very large networks where the number of systems reporting and the network traffic they generate can be considered to be detrimental. In very busy systems, the need to broadcast the data can also mean that the information is never reported, the data can be out of date, or the system could be reported as down when it is just busy.

Tracking large processes

If you suspect that the problem is due to a large or overly busy process, then you should check the output of the ps tool, look for the process size, the percentage of memory, and CPU being utilized. On an SVR4 system (Solaris and AIX®), you can use the following command to get a list of processes (see Listing 2).

Listing 2. Command to get a list of processes

```
$ ps -A -o pcpu,pmem,rss,vsz,comm
%CPU %MEM  RSS  VSZ COMMAND
 0.2  0.0    0    0 fsflush
 0.1  0.2 1464 8288 /usr/lib/ssh/sshd
 0.1  0.1 1032 1320 ps
 0.0  1.0 9536 47608 /usr/openwin/bin/Xsun
 0.0  0.7 6312 10720 dtgreet
 0.0  0.6 6136 9352 /usr/sfw/sbin/snmpd
 0.0  0.4 3208 5720 /usr/lib/fm/fmd/fmd
 0.0  0.3 2808 8512 /usr/lib/ssh/sshd
 0.0  0.3 2800 8504 /usr/lib/ssh/sshd
 0.0  0.3 2768 8512 /usr/lib/ssh/sshd
 0.0  0.3 2368 4056 /usr/sbin/nscd
 0.0  0.2 2096 9176 /usr/dt/bin/dtlogin
...
```

Listing 3 shows how it looks on a BSD-derived system.

Listing 3. Getting a list of processes on a BSD system

```
$ ps -A -o pcpu,pmem,rss,vsz,command|sort -n +3
%CPU %MEM    RSS      VSZ COMMAND
  0.0  0.0    152    27236 nfsd-server
  0.0  0.0    152    27236 nfsd-server
  0.0  0.0    152    27236 nfsd-server
  0.0  0.0    152    27236 nfsd-server
  0.0  0.0    152    27236 nfsd-server
  0.0  0.0    152    27236 nfsd-server
  0.0  0.0    152    27236 nfsd-server
  0.0  0.0    152    27236 nfsd-server
  0.0  0.0    164    27236 nfsd-master
  0.0  0.0    224    27240 /usr/sbin/update
  0.0  0.3   4364    29196 /usr/sbin/securityd
  0.0  0.2   2760    29288 jabberd -c /etc/jabber/jabber.xml -H
 /private/var/jabber/ -U jabber
  0.0  0.0    184    29300 nfsiod -n 4
0.0  0.2   3544    29712 /usr/sbin/configd
  0.0  0.0    500    30628 /usr/sbin/sshd -i
  0.0  0.0    260    30648 /usr/sbin/smbd -D
  0.0  0.0    736    30648 /usr/sbin/smbd -D
  0.0  0.1   1216    30700 /usr/sbin/sshd -i
...
  0.0  0.1   2180    50664 imapd: narcissus.mcslp.pri [192.168.0.110]
mc user.mc
  0.0  0.1   2184    50664 imapd: sulaco.mcslp.pri [192.168.0.101]
mc user.mc
  0.0  0.1   2204    50720 imapd: narcissus.mcslp.pri [192.168.0.110]
 buy user.buy
  0.0  0.1   2264    50720 imapd: sulaco.mcslp.pri [192.168.0.101] buy
 user.buy
  0.0  0.1   2272    50984 imapd: kernel.mcslp.pri [192.168.0.106] slp
 user.slp
  0.0  1.2  18348    54368 servermgrd -x
  0.0  0.2   3200    85920 /usr/sbin/named -f
  0.0  1.1  16820   122240 /usr/libexec/mysqld --basedir=/usr
```

```
 --datadir=/var/mysql --user=mysql --pid-file=/var/mysq
  0.0  0.5   8572   158164 /usr/libexec/slapd -d 0 -h ldap:///
 ldapi://%2Fvar%2Frun%2Fldapi
  0.0  0.0    204   289396 rpc.statd
```

In both cases, the CPU and memory usage percentages have been displayed in the process list so that you can get a good idea of the loading on the system. The 's' and 'stat' columns (for SVR4 and BSD, respectively) show the current status of the process. A large number of running processes (status 'R') indicate that the process is currently running.

By using a combination of the state, CPU, and memory percentage, you should be able to identify if there is a process that is *running away* and over using your system resources.

Using iostat

The iostat tool provides information about the terminal, disk activity, and CPU utilization. You can specify a single numerical argument to set the report interval; a second numerical argument sets the number of reports. For example, Listing 4 shows how to report the statistics every five seconds.

Listing 4. Reporting statistics every five seconds

```
$ iostat 5
   tty        dad1          sd1          nfs1           cpu
 tin tout kps tps serv  kps tps serv  kps tps serv   us sy wt id
   0    7 440  39   14    0   0    3    0   0    0    5 18  0 77
   0   39   2   0    0    0   0    0    0   0    0    0  0  0 100
   0   13   4   3    0    0   0    0    0   0    0    0  0  0 100
   0   13   0   0    0    0   0    0    0   0    0    0  0  0 100
```

The exact information, shown by default, varies from system to system; Listing 4 was from a Solaris system. The example in Listing 5 comes from a BSD environment.

Listing 5. iostat in a BSD environment

```
          disk1           disk0       cpu
  KB/t tps  MB/s   KB/t tps  MB/s  us sy id
 167.67   0  0.02  20.70   5  0.09   6  3 90
  0.00   0  0.00   0.00   0  0.00  15  3 82
  0.00   0  0.00   0.00   0  0.00  16  2 82
  0.00   0  0.00  14.33  24  0.33  18  4 79
  0.00   0  0.00   2.83   1  0.00  23  4 73
```

Dealing with the CPU statistics first, the columns show user (us), system (sy), and idle (id) percentages. The user time shows how much time was spent on user processes. The system time shows the time in system processes (including, when wait time is not shown, the time the system is waiting for I/O). The idle time shows the percentage of time that the CPU(s) were idle.

Disk output shows how busy the individual physical disks (including NFS mounts, if applicable) are, usually in transactions per second and MB or KB transferred per second. Large figures here, especially if combined with high wait/system times, might indicate a disk that is too slow for the system. You can try to spread your application so that it uses different disks you might be able to improve performance.

If the disk is the same as the one used for virtual memory, then it might be a problem with lack of memory and too much swapping.

Using vmstat

You can monitor virtual memory statistics with the vmstat tool. Like iostat, it accepts a numerical interval (see Listing 6).

Listing 6. Monitoring memory statistics with vmstat

```
$ vmstat 5
kthr      memory            page            disk     faults      cpu
 r b w   swap  free  re  mf pi po fr de sr dd s1 -- in   sy
cs us sy id
 0 0 0 2820888 809552 94 525 121 69 50 0 26 16 0  0 297 1342
272  9  4 87
 0 0 0 2824752 778872 2   7  0  0  0  0  0  0  0  0 229   34
109  0  1 99
 0 0 0 2824752 778872 0   0  0  0  0  0  0  2  0  0 233   28
116  0  0 100
 0 0 0 2824752 778872 0   0  0  0  0  0  0  0  0  0 228   26
110  0  0 100
 0 0 0 2824752 778872 0   0  0  0  0  0  0  0  0  0 229   28
111  0  0 100
```

The vmstat tool outputs thread/process information, memory/swap usage, page ins/outs, disk I/O, page faults, and CPU statistics.

The CPU/thread block shows the processes/threads in the run queue (r), blocked processes waiting for I/O resources (b), and those that were swapped. High numbers in the blocked processes column indicates slow disks. High numbers in the swapped column indicate that there are two many processes using too much memory that need to be swapped in and out. Swapping is an expensive process and will significantly degrade your system.

The memory shows the amount of swap currently available and the size of the free list (the number of pages that could be swapped if the RAM were requested). Low swap values indicate that you are running out of swap space, which doesn't necessarily show a problem, as long as you have enough RAM to run the applications. Low free list values might indicate that you have a lot of active RAM in use, which might trigger swap space use if you add more processes to the system.

The page columns show the pages of memory swapped in and out to disk. The key columns are the pi/po (page in/page out), which indicate how many pages have been exchanged. High paging indicates a lack of RAM; a high scan rate (the sr column) shows a potential memory bottleneck.

Using top

The top tool can provide a useful way to monitor a live system and the active processes, loading, and memory statistics. There are many different types of top, some of which are installed by default on some systems and also the latest open source version of the tool. The information provided is like a combination of the uptime, swap space, and ps tools. For example, the following is from a Solaris system running V3.5.1 of the top tool (see Listing 7).

Listing 7. Using top

```
last pid:  9385;  load averages:  7.14,  2.98,  1.21
61 processes:  55 sleeping, 4 running, 1 zombie, 1 on cpu
CPU states:  0.0% idle, 93.8% user,  6.2% kernel,  0.0% iowait,
0.0% swap
Memory: 1024M real, 712M free, 125M swap in use, 2705M swap free

   PID USERNAME LWP PRI NICE  SIZE   RES STATE    TIME    CPU COMMAND
  9313 root       1  22    0   35M   34M run      0:03  8.87% cc1
  9349 root       1  22    0   21M   20M run      0:01  5.47% cc1
  9385 root       1  39    0 4320K 3904K run      0:00  0.38% as
  9384 root       1  29    0 3888K 3424K run      0:00  0.30% as
  9145 root       1  59    0 3736K 2144K cpu      0:00  0.11% top
  9180 root       1  59    0 1808K 1472K sleep    0:00  0.10% make
   486 root       1  59    0   46M 9536K sleep    0:00  0.03% Xsun
   548 root       1  59    0   10M 6360K sleep    0:00  0.03% dtgreet
   553 mc         1  49    0 8288K 1472K sleep    0:01  0.02% sshd
  9345 root       1  49    0 1328K  928K sleep    0:00  0.01% gcc
  9348 root       1  59    0 1328K  928K sleep    0:00  0.01% gcc
  9325 root       1  49    0 1328K  928K sleep    0:00  0.01% gcc
   599 mc         1  59    0 8288K 1488K sleep    0:00  0.00% sshd
  9312 root       1  59    0 1328K  928K sleep    0:00  0.00% gcc
     9 root      16  59    0 9464K 2016K sleep    0:06  0.00%
 svc.configd
```

The top tool shows the CPU usage of individual processes; for example, in the previous sample, you can see that you are compiling a number of files and how much CPU they are using.

You should also keep an eye on the process state: high numbers of running processes might indicate an over-busy system (compare the running processes with the CPU states and load averages for the system). Top itself can be a CPU hog; it is best to run it at a relatively large update interval, to prevent the monitoring having a detrimental effect on the performance of your system. You can specify the update interval in seconds using either `-s` or `-d` command-line option (depending on your platform).

Using SAR

In situations where you cannot monitor the status of your server live but you want to be able to monitor the status of a machine after it has caused a problem, you can use the SAR (system activity reporter) tool. This works by recording information at specific intervals into a global file that can then be post processed to display information about the machine.

Because the recording process continues in the background, it can be used to plot the system performance over time and might help you identify the cause of the problem. Information is generally recorded day by day, for up to a month, at intervals that you specify. Logs are written into /var/log/sa/sa*DD* or /usr/adm/sa/sa*DD*, where *DD* is the day of the month. Enabling SAR is specific to the system, and generally you will need to set up a cron job that automatically runs the collection script (sa1). Another script, sa2, can create a daily report that you can study. For example, the crontab below shows the default system performance statistics recording on a Solaris system:

```
0 * * * 0-6 /usr/lib/sa/sa1
20,40 8-17 * * 1-5 /usr/lib/sa/sa1
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

Once the information has been collected, you can extract data using the `sar` command. The amount of information, which is recorded by the system, can be voluminous, and the level of detail that can be selected and extracted from that data is similarly large. However, you can get a feel for the quantity and quality of the data by using the `-A` command-line argument to SAR, which reports all the currently recorded information.

Listing 8. Output generated with the sar command, using the -A argument

```
11:49:38    %usr    %sys    %wio   %idle
13:20:00       1       1       0      99
13:40:01      19       5       0      76
14:00:00       0       0       0     100
14:20:00       0       0       0     100
14:40:01       0       0       0     100
15:00:00       0       0       0     100
15:20:00       0       0       0     100

Average        3       1       0      96

11:49:38   device        %busy   avque   r+w/s  blks/s  avwait  avserv

...
Average    dad1              1     0.3       5     365    47.3     4.5
           dad1,a            0     0.0       0       4    15.4     8.6
           dad1,b            0     0.0       0       0     0.0    13.8
           dad1,c            0     0.0       0       0     0.0     0.0
           dad1,d            1     0.2       3     143    53.0     3.9
           dad1,e            0     0.0       0      39   117.3     5.9
           dad1,h            0     0.0       1     178    29.0     4.6
           nfs1              0     0.0       0       0     0.0     0.0
           nfs2              0     0.0       0      31     0.5    14.5
           sd1               0     0.0       0       0     0.0     3.3

11:49:38 runq-sz %runocc swpq-sz %swpocc
13:20:00     2.0       2     0.0       0
13:40:01     5.3      15     0.0       0
14:00:00     0.0       0     0.0       0
14:20:00     0.0       0     0.0       0
14:40:01     1.5       0     0.0       0
15:00:00     0.0       0     0.0       0
15:20:00     0.0       0     0.0       0

Average      5.0       2     0.0       0

11:49:38 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
13:20:00       0      11      97       0       1      89       0       0
13:40:01       0     803     100       4     381      99       0       0
14:00:00       0       0     100       0       0      39       0       0
14:20:00       0       0     100       0       0      56       0       0
14:40:01       0       0     100       0       0      61       0       0
15:00:00       0       0     100       0       0      48       0       0
15:20:00       0       0     100       0       0      32       0       0

Average        0     120     100       1      56      99       0       0


11:49:38 swpin/s bswin/s swpot/s bswot/s pswch/s
13:20:00    0.00     0.0    0.00     0.0     305
13:40:01    0.00     0.0    0.00     0.0     223
14:00:00    0.00     0.0    0.00     0.0     111
14:20:00    0.00     0.0    0.00     0.0     112
14:40:01    0.00     0.0    0.00     0.0     112
15:00:00    0.00     0.0    0.00     0.0     114
15:20:00    0.00     0.0    0.00     0.0     114
```

```
Average     0.00     0.0    0.00     0.0     152

11:49:38 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
13:20:00     526      39      26    0.64    0.59   38118   25779
13:40:01    2288     803     320    9.31    6.53  773352 1558934
14:00:00      22       2       2    0.01    0.01     342     186
14:20:00      20       2       2    0.00    0.00     150     128
14:40:01      20       2       2    0.01    0.00     153     128
15:00:00      26       3       3    0.01    0.02     326     167
15:20:00      29       3       3    0.02    0.03     641     272

Average      416     125      52    1.46    1.04  118615  232791

11:49:38  iget/s namei/s dirbk/s
13:20:00       2      31       3
13:40:01      29     385      25
14:00:00       0       1       0
14:20:00       0       0       0
14:40:01       0       0       0
15:00:00       0       1       0
15:20:00       0       2       0

Average        5      61       4

11:49:38 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
13:20:00       0       0      39       0       0       0
13:40:01       1       0     397       0       0       0
14:00:00       0       0       9       0       0       0
14:20:00       0       0       0       0       0       0
14:40:01       0       0       0       0       0       0
15:00:00       0       0      16       0       0       0
15:20:00       0       0      38       0       0       0

Average        0       0      72       0       0       0

11:49:38  proc-sz    ov  inod-sz    ov  file-sz    ov   lock-sz
13:20:00   53/16154    0 1732/69661    0  358/358     0    0/0
13:40:01   54/16154    0 15118/69661    0  358/358     0    0/0
14:00:00   57/16154    0 15120/69661    0  359/359     0    0/0
14:20:00   57/16154    0 15120/69661    0  359/359     0    0/0
14:40:01   57/16154    0 15120/69661    0  359/359     0    0/0
15:00:00   57/16154    0 15121/69661    0  359/359     0    0/0
15:20:00   57/16154    0 15127/69661    0  359/359     0    0/0


11:49:38   msg/s  sema/s
13:20:00    0.00    0.00
13:40:01    0.00    0.00
14:00:00    0.00    0.00
14:20:00    0.00    0.00
14:40:01    0.00    0.00
15:00:00    0.00    0.00
15:20:00    0.00    0.00

Average     0.00    0.00

11:49:38  atch/s  pgin/s ppgin/s  pflt/s  vflt/s slock/s
13:20:00   13.39    3.67    5.05   41.14   77.09    0.00
13:40:01  188.44    9.91   25.61  373.73 1086.42    0.00
14:00:00    0.30    0.05    0.06    0.61    1.59    0.00
14:20:00    0.16    0.00    0.00    0.34    0.76    0.00
14:40:01    0.20    0.00    0.00    0.48    1.01    0.00
15:00:00    0.72    0.01    0.01    0.98    2.37    0.00
15:20:00    0.89    0.02    0.02    1.43    3.47    0.00

Average    29.66    1.90    4.38   60.43  170.40    0.00
```

```
11:49:38  pgout/s ppgout/s pgfree/s pgscan/s %ufs_ipf
13:20:00     0.03     0.06     0.06     0.00     0.00
13:40:01     6.41    19.18    13.84     0.00     0.00
14:00:00     0.00     0.00     0.00     0.00     0.00
14:20:00     0.00     0.00     0.00     0.00     0.00
14:40:01     0.00     0.00     0.00     0.00     0.00
15:00:00     0.00     0.00     0.00     0.00     0.00
15:20:00     0.00     0.00     0.00     0.00     0.00

Average      0.95     2.83     2.05     0.00     0.00

11:49:38 freemem freeswap
13:20:00  109186  5736615
13:40:01   95816  5614822
14:00:00   97408  5649849
14:20:00   97311  5647409
14:40:01   97418  5653711
15:00:00   97338  5648982
15:20:00   97333  5648993

Average    98516  5654784

11:49:38 sml_mem   alloc  fail  lg_mem   alloc  fail  ovsz_alloc  fail
13:20:00 4178176 3572465     0 38477824 32137880     0    14663680     0
13:40:01 16572672 10204085     0 99106816 80782488     0    15310848
    0
14:00:00 16589056 10261693     0 99106816 80797968     0    15343616
  0
14:20:00 16589056 10259613     0 99106816 80736600     0    15343616
    0
14:40:01 16589056 10260061     0 99106816 80820088     0    15343616
 0
15:00:00 16589056 10267477     0 99106816 80902432     0    15343616
    0
15:20:00 16589056 10274757     0 99106816 80864920     0    15343616
   0

Average  14813733 9300022     0 90445528 73863192     0    15241801
  0
```

Where possible, the output above has been trimmed to limit the amount of data shown (not all disk stats are shown, for example). For more information on SAR, check the Related topics section and the manual page for your system.

Summary

While there is not always a direct correlation between a slow UNIX system and the statistical information that you can extract, your first job when discovering a slow system should be to collect as much information as possible. Whether you do this actively (through ps, uptime and other tools) or passively (through SAR or top) depends on the situation. Armed with that information, you should be able to tell whether your UNIX system is slow because it is overworked (CPU over usage), low on physical memory (large amounts of swapping), or there is a problem with a runaway process (high single process CPU time).

Related topics

- System Administration Toolkit: Check out other parts in this series.
- AIX® and UNIX articles: Check out other articles written by Martin Brown.
- Search the AIX and UNIX library by topic:
  - System administration
  - Application development
  - Performance
  - Porting
  - Security
  - Tips
  - Tools and utilities
  - Java technology
  - Linux
  - Open source
- AIX and UNIX: The AIX and UNIX developerWorks zone provides a wealth of information relating to all aspects of AIX systems administration and expanding your UNIX skills.
- New to AIX and UNIX: Visit the New to AIX and UNIX page to learn more about AIX and UNIX.
- AIX 5L™ Wiki: A collaborative environment for technical information related to AIX.
- IBM trial software: Build your next development project with software for download directly from developerWorks.
- developerWorks technical events and webcasts: Stay current with developerWorks technical events and webcasts.
- Podcasts: Tune in and catch up with IBM technical experts.