# 进程/线程信息

|  |  |
| --- | --- |
| 采集项 | 采集方式 |
| top 10进程 | top -bsw -n 1 | grep -A 10 '%%CPU' | tail -n 10 | awk '{print $1,$9,$10,$12}' |

# 进程性能

通常，可以使用ps命令。它能显示当前运行中进程的相关信息，包括进程的PID。Linux

和Unix都支持ps命令，显示所有运行中进程的相关信息。

## ps

ps命令能够提供一份当前进程的快照，如果你想状态可以自动刷新，可以使用top命令。

输入下面的ps命令，显示所有运行中的进程：

ps aux | less

其中：

-A 显示所有的进程

a : 显示终端中包括其他用户的所有进程

x : 显示无终端控制的进程

查看系统中的每个进程

ps –A / ps –e

查看非root运行的进程

ps –U root –u root –N

查看用户test运行的进程

ps –u test

使用ps打印进程数

ps –ejH

ps axjf

获取线程信息

ps –eLf

ps zxms

## top

top命令提供了运行中系统的动态实时视图。

Top的输出信息：

按q退出，h帮助。

将进程快照存储到文件中

top –b –n1 > /tmp/process.log

也可以将结果通过邮件发给自己：

top –b –n1 | mail –s “@”

## pidstat

pidstat是sysstat工具的一个命令，用于监控全部或指定进程的cpu、内存、线程、设备IO等系统资源的占用情况。pidstat首次运行时显示自系统启动开始的各项统计信息，之后运行pidstat将显示自上次运行该命令以后的统计信息。用户可以通过指定统计的次数和时间来获得所需的统计信息。

pidstat [ 选项 ] [ <时间间隔> ] [ <次数> ]

常用的参数：

-u：默认的参数，显示各个进程的cpu使用统计

-r：显示各个进程的内存使用统计

-d：显示各个进程的IO使用情况

-p：指定进程号

-w：显示每个进程的上下文切换情况

-t：显示选择任务的线程的统计信息外的额外信息

-T { TASK | CHILD | ALL }

这个选项指定了pidstat监控的。TASK表示报告独立的task，CHILD关键字表示报告进程下所有线程统计信息。ALL表示报告独立的task和task下面的所有线程。

**注意：**task和子线程的全局的统计信息和pidstat选项无关。这些统计信息不会对应到当前的统计间隔，这些统计信息只有在子线程kill或者完成的时候才会被收集。

-V：版本号

-h：在一行上显示了所有活动，这样其他程序可以容易解析。

-I：在SMP环境，表示任务的CPU使用率/内核数量

-l：显示命令名和所有参数

## pstree

显示进程的树状图。

pstree以树状图显示正在运行的进程。树的根节点为pid或者init。如果制定了用户名，进程树将以用户所拥有的进程作为根节点。

## pgrep

查看进程

使用pgrep命令。能够查出当前正在运行的进程并列出符合条件的进程ID。

例如显示firefox的进程ID：

pgrep firefox

下面的命令将显示进程名为sshd、所有者为root的进程：

pgrep –u root sshd

## htop

htop是一个类似top的交互式进程查看工具，但是可以垂直和水平滚动来查看所有进程和他们的命令行。进程的相关词（killing，renicing）不需要输入PID。

安装htop输入命令：

apt-get install htop

或者

yum install htop

## atop

atop是一个用来查看Linux系统负载的交互式监控工具，它能展现系统级别的关键硬件资源的使用情况，如CPU、内存、硬盘、网络等。

它可以根据进程层级的CPU和内存负载显示哪个进程造成了特定的负载，如果已经安装内核补丁，可以显示每个进程的硬盘和网络负载。

## lsof

lsof，是list open files的简称。它的参数很多，但是我们这里只介绍一些实用的用法(注意有些情况需要root权限执行)。

**查看当前打开的所有文件**

一般来说，直接输入lsof命令产生的结果实在是太多，可能很难找到我们需要的信息。不过借此说明一下一条记录都有哪些信息。

$ lsof(这里选取一条记录显示)

COMMAND   PID     USER   FD    TYPE   DEVICE SIZE/OFF   NODE   NAME

vi     27940       hyb    7u      REG        8,15     16384     137573 /home/hyb/.1.txt.swp

lsof显示的结果，从左往右分别代表：打开该文件的程序名，进程id，用户，文件描述符，文件类型，设备，大小，iNode号，文件名。

我们暂且先关注我们知道的列。这条记录，表明进程id为27940的vi程序，打开了文件描述值为7，且处于读写状态的，在/home/hyb目录下的普通文件(REG regular file).1.txt.swap，当前大小16384字节。

**列出被删除但占用空间的文件**

在生产环境中，我们可能会使用df命令看到磁盘空间占满了，然而实际上又很难找到占满空间的文件，这常常是由于某个大文件被删除了，但是它却被某个进程打开，导致通过普通的方式找不到它的踪迹，最常见的就是日志文件。我们可以通过lsof来发现这样的文件：

$ lsof |grep deleted

Xorg      1131 root  125u      REG                0,5        4      61026 /memfd:xshmfence (deleted)

Xorg      1131 root  126u      REG                0,5        4      62913 /memfd:xshmfence (deleted)

Xorg      1131 root  129u      REG                0,5        4      74609 /memfd:xshmfence (deleted)

可以看到这些被删除的但仍然被打开文件，最后查找出来的时候，会被标记deleted。这个时候就可以根据实际情况分析，到底哪些文件可能过大但是却被删除了，导致空间仍然占满。

**恢复打开但被删除的文件**

前面我们可以找到被删除但是仍然被打开的文件，实际上文件并没有真正的消失，如果是意外被删除的，我们还有手段恢复它。以/var/log/syslog文件为例，我们先删除它(root用户)：

$　rm /var/log/syslog

然后使用lsof查看那个进程打开了该文件：

$ lsof |grep syslog

rs:main    993 1119           syslog    5w      REG               8,10     78419     528470 /var/log/syslog (deleted)

可以找到进程id为993的进程打开了该文件，我们知道每个进程在/proc下都有文件描述符打开的记录：

$ ls -l /proc/993/fd

lr-x------ 1 root   root   64 3月   5 18:30 0 -> /dev/null

l-wx------ 1 root   root   64 3月   5 18:30 1 -> /dev/null

l-wx------ 1 root   root   64 3月   5 18:30 2 -> /dev/null

lrwx------ 1 root   root   64 3月   5 18:30 3 -> socket:[15032]

lr-x------ 1 root   root   64 3月   5 18:30 4 -> /proc/kmsg

l-wx------ 1 root   root   64 3月   5 18:30 5 -> /var/log/syslog (deleted)

l-wx------ 1 root   root   64 3月   5 18:30 6 -> /var/log/auth.log

这里就找到了被删除的syslog文件,文件描述符是５，我们把它重定向出来：

$ cat /proc/993/fd/5 > syslog

$ ls -al /var/log/syslog

-rw-r--r-- 1 root root 78493 3月   5 19:22 /var/log/syslog

这样我们就恢复了syslog文件。

**查看当前文件被哪些进程打开**

Windows下经常遇到要删除某个文件，然后告诉你某个程序正在使用，然而不告诉你具体是哪个程序。我们可以在资源管理器-性能-资源监视器-cpu-关联的句柄处搜索文件，即可找到打开该文件的程序，但是搜索速度感人。

linux就比较容易了，使用lsof命令就可以了，例如要查看当前哪些程序打开了hello.c:

$ lsof hello.c

COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME

tail    28731  hyb    3r   REG   8,15      228 138441 hello.c

但是我们会发现，使用vi打开的hello.c并没有找出来，这是因为vi打开的是一个临时副本。我们换一种方式查找：

$ lsof |grep hello.c

tail   28906  hyb  3r   REG   8,15       228     138441 /home/hyb/workspaces/c/hello.c

vi    28933    hyb   9u   REG    8,15    12288     137573 /home/hyb/workspaces/c/.hello.c.swp

这样我们就找到了两个程序和hello.c文件相关。

这里grep的作用是从所有结果中只列出符合条件的结果。

**查看某个目录文件被打开情况**

$ lsof +D ./

查看当前进程打开了哪些文件

使用方法：lsof -c 进程名

通常用于程序定位问题，例如用于查看当前进程使用了哪些库，打开了哪些文件等等。假设有一个循环打印字符的hello程序：

$ lsof -c hello

COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME

hello   29190  hyb  cwd    DIR   8,15     4096 134538 /home/hyb/workspaces/c

hello   29190  hyb  rtd    DIR   8,10     4096      2 /

hello   29190  hyb  txt    REG   8,15     9816 138314 /home/hyb/workspaces/c/hello

hello   29190  hyb  mem    REG   8,10  1868984 939763 /lib/x86\_64-linux-gnu/libc-2.23.so

hello   29190  hyb  mem    REG   8,10   162632 926913 /lib/x86\_64-linux-gnu/ld-2.23.so

hello   29190  hyb    0u   CHR 136,20      0t0     23 /dev/pts/20

hello   29190  hyb    1u   CHR 136,20      0t0     23 /dev/pts/20

hello   29190  hyb    2u   CHR 136,20      0t0     23 /dev/pts/20

我们可以从中看到，至少它用到了/lib/x86\_64-linux-gnu/libc-2.23.so以及hello文件。

也可以通过进程id查看,可跟多个进程id，使用逗号隔开：

$ lsof -p 29190

COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME

hello   29190  hyb  cwd    DIR   8,15     4096 134538 /home/hyb/workspaces/c

hello   29190  hyb  rtd    DIR   8,10     4096      2 /

hello   29190  hyb  txt    REG   8,15     9816 138314 /home/hyb/workspaces/c/hello

hello   29190  hyb  mem    REG   8,10  1868984 939763 /lib/x86\_64-linux-gnu/libc-2.23.so

hello   29190  hyb  mem    REG   8,10   162632 926913 /lib/x86\_64-linux-gnu/ld-2.23.so

hello   29190  hyb    0u   CHR 136,20      0t0     23 /dev/pts/20

hello   29190  hyb    1u   CHR 136,20      0t0     23 /dev/pts/20

hello   29190  hyb    2u   CHR 136,20      0t0     23 /dev/pts/20

当然这里还有一种方式，就是利用proc文件系统，首先找到hello进程的进程id:

$ ps -ef|grep hello

hyb      29190 27929  0 21:14 pts/20   00:00:00 ./hello 2

hyb      29296 28848  0 21:18 pts/22   00:00:00 grep --color=auto hello

可以看到进程id为29190，查看该进程文件描述记录目录：

$ ls -l /proc/29190/fd

lrwx------ 1 hyb hyb 64 3月   2 21:14 0 -> /dev/pts/20

lrwx------ 1 hyb hyb 64 3月   2 21:14 1 -> /dev/pts/20

lrwx------ 1 hyb hyb 64 3月   2 21:14 2 -> /dev/pts/20

这种方式能够过滤很多信息，因为它只列出了该进程实际打开的，这里它只打开了０,１,２，即标准输入，标准输出和标准错误。

**查看某个用户打开了哪些文件**

linux是一个多用户操作系统，怎么知道其他普通用户打开了哪些文件呢？可使用－ｕ参数

$ lsof -u hyb

**列出除了某个进程或某个用户打开的文件**

实际上和前面使用方法类似，只不过，在进程id前面或者用户名前面加^，例如：

lsof -p ^1     #列出除进程id为１的进程以外打开的文件

lsof -u ^root  #列出除root用户以外打开的文件

# 线程性能

线程是现代操作系统上进行并行执行的一个流行的编程方面的概念，当一个程序内有多个咸亨被拆分出用以执行多个流时，这些线程就会在它们之间共享特定的资源（如内存地址空间、打开的文件），以使拆分开销最小化，并避免大量高成本的IPC（进程间通信）通道。这些功能让线程在并发执行时称为一个高效的机制。

在Linux中，程序中创建的线程（也成为轻量级线程，LWP）会具有和线程的PID相同的“线程组ID”。然后，各个线程会获取其自身的线程ID（TID）。对于Linux内核调度器而言，线程不过是恰好共享特定资源的标准的进程而已。经典的命令行工具，比如ps或者top都可以用来显示线程级别的消息，只是默认情况下显示进程级别的信息。

## ps

ps命令中，-T选项可以开启线程查看。

由进程号pid的进程创建的所有线程

ps –T –p pid

## top

top –H 会列举出所有Linux线程

在top运行时可以通过按-H键将线程查看模式切换为开或者关

要让top输出某个特定进程<pid>并检查该进程内运行的线程状况

top –H –p <pid>

## htop

一个队用户更加友好的方式是，通过htop查看单个进程的线程。它是一个机遇ncurses的交互进程查看器。该程序允许你在树状视图中监控单个独立线程。

要在htop中启用线程查看，并开启htop，然后按F2来进入htop的设置菜单，选择“设置”栏下面的“显示选项”，然后开启“树状视图”和“显示自定义线程名”选项。按F10退出设置。