CC 's Blog

TOP命令

- 使用top命令查看可能会有进程占用率非常高，这个数值是进程内各个线程占用cpu的累加值。

> *关于第三行CPU(s)的理解为:*
> *%us 用户空间占用CPU百分比,当有多个CPU时，则分母为全部CPU计算能力之和。*
> *%sy 内核空间占用CPU百分比,当有多个CPU时，则分母为全部CPU计算能力之和。*
> ***单个进程的CPU的利用率理解为：***
> *当前任务共享从上次屏幕刷新时的CPU时间，以CPU总时间的百分比表示。在一个真实的SMP环境中，如果Irix Mode被设置成off，top将工作在Solaris Mode下，即一个任务的CPU利用率将以CPU的总数分开显示,要切换Irix/Solaris Mode，按I就行。*
> *比如进程占用各CPU使用率总和为10%，则TOP命令显示结果为160%（10%*16核），所以当前图片中Tomcat实际占用各CPU总和使用率为106%/16=6.625%与主机总体负载基本一致。*

```
1.   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    14094 root      15   0  315m  10m 7308 S 891%  2.2   1:49.01
2.  gateway
```

- 使用`top -H p pid`命令可以查看进程内各个线程占用CPU百分比

```
    top中可以看到有107个线程，但是下面9个线程占用CPU很高，下面以线程14086为主，分析其为何high
1.  CPU
2.    PID USER      PR  NI  VIRT  RES  SHR S %CPU MEM    TIME+  COMMAND
3.  14086 root      25   0  922m 914m 538m R  101 10.0  21:35.46 gateway
4.  14087 root      25   0  922m 914m 538m R  101 10.0  10:50.22 gateway
5.  14081 root      25   0  922m 914m 538m S   99 10.0   8:57.36 gateway
```

- 使用命令gstack查看进程中各个线程的函数调用栈
  `gstack pid > gstack.log`，这里的pid是上面查出的

```
#gstack 14094 >
gstack.log
在gstack.log中查找线程ID14086，由于函数栈会暴露函数细节，因此只显示了两个函数桢，线
程ID14086对应线程号是37
Thread 37 (Thread 0x4696ab90 (LWP
14086)):
#0  0x40000410 in __kernel_vsyscall
()
#1  0x40241f33 in poll () from
/lib/i686/nosegneg/libc.so.6
```

- 使用命令gcore可以转存进程映像及内存上下文
  gcore
  pid　　改命令生成core.pid文件

```
#gcore
14094
该命令生成core文件core.14094
```

- 使用strace命令查看系统调用和花费的时间
  strace -T -r -c -p
  pid　　，-c参数显示统计信息，去掉此参数可以查看每个系统调用话费的时间及返回值。

```
% time     seconds  usecs/call     calls    errors
syscall
------ ----------- ----------- --------- --------- ----------------
99.99   22.683879        3385      6702
poll
```

- 用gdb调试core文件，并线程切换到37号(进程号在gstack命令中可以看到)线程

1. gcore和实际的core dump时产生的core文件几乎一样，只是不能用gdb进行某些动态调试
2. (gdb) gdb gateway core.14094
3. (gdb) thread
   37
4. [Switching to thread 37 (Thread 0x4696ab90 (LWP 14086))]#0 0x40000410 in
   __kernel_vsyscall ()
5. (gdb) where
6. #0 0x40000410 in __kernel_vsyscall
   ()
7. #1 0x40241f33 in poll () from
   /lib/i686/nosegneg/libc.so.6
8. 可以根据详细的函数栈进行gdb调试，打印一些变量值，并结合源代码分析为何会poll调用占用很高的CPU。
9. 因为代码涉及到公司产权，顾不在此做详细分析，需要明白的是分析的流程和使用的命令。
0. 流程为：进程ID->线程ID->线程函数调用栈->函数耗时和调用统计->源代码分析

参考文章