1.首先是用top命令确定 有问题的pid

然后用top -H -p pid命令查看进程内各个线程占用的CPU百分比  
>top -H -p 14094

或者使用  > ps H -e -o pid,tid,pcpu,cmd --sort=pcpu |grep xxx

2.接下来，我们用gdb来attach目标进程

gdb

gdb>attach 2907

在gdb中，列出线程状态：

http://www.cppblog.com/Images/OutliningIndicators/None.gif(gdb) info threads http://www.cppblog.com/Images/OutliningIndicators/None.gif

9 Thread 47056948181264 (LWP 11066)  0x00002acc4a3dec91 in nanosleep () from /lib/libc.so.6 http://www.cppblog.com/Images/OutliningIndicators/None.gif

8 Thread 47056956573968 (LWP 11067)  0x00002acc4a406fc2 in select () from /lib/libc.so.6 http://www.cppblog.com/Images/OutliningIndicators/None.gif

7 Thread 47056964966672 (LWP 11068)  0x00002acc4a3dec91 in nanosleep () from /lib/libc.so.6 http://www.cppblog.com/Images/OutliningIndicators/None.gif

 6 Thread 47056973359376 (LWP 11069)  0x00002acc4a3dec91 in nanosleep () from /lib/libc.so.6 http://www.cppblog.com/Images/OutliningIndicators/None.gif

5 Thread 47056981752080 (LWP 11070)  0x00002acc4a3dec91 in nanosleep () from /lib/libc.so.6 http://www.cppblog.com/Images/OutliningIndicators/None.gif

4 Thread 47056990144784 (LWP 11071)  0x00002acc4a40e63c in recvfrom () from /lib/libc.so.6 http://www.cppblog.com/Images/OutliningIndicators/None.gif

3 Thread 47057194060048 (LWP 11072)  0x00002acc4a406fc2 in select () from /lib/libc.so.6 http://www.cppblog.com/Images/OutliningIndicators/None.gif

2 Thread 47057226893584 (LWP 11073)  CSendFile::SendFile (this=0x2acc5d4aff40, pathname=@0x2acc5d4afee0) http://www.cppblog.com/Images/OutliningIndicators/None.gif    at ../src/csendfile.cpp:101 http://www.cppblog.com/Images/OutliningIndicators/None.gif

1 Thread 47056939784832 (LWP 11065)  0x00002acc4a3dec91 in nanosleep () from /lib/libc.so.6 http://www.cppblog.com/Images/OutliningIndicators/None.gif(gdb)

gdb已经列出了各线程正在执行的函数，我们需要更多信息，记住11073对应的行首标号，这是gdb为线程分配的id，这里为2，然后执行切换：

(gdb) thread 2 http://www.cppblog.com/Images/OutliningIndicators/None.gif

[Switching to thread 2 (Thread 47057226893584 (LWP 11073))]#0  CSendFile::SendFile (this=0x2acc5d4aff40, pathname=@0x2acc5d4afee0) http://www.cppblog.com/Images/OutliningIndicators/None.gif    at ../src/csendfile.cpp:101 http://www.cppblog.com/Images/OutliningIndicators/None.gif101             while(1) http://www.cppblog.com/Images/OutliningIndicators/None.gif

(gdb)

bt一下：

http://www.cppblog.com/Images/OutliningIndicators/None.gif(gdb) bt http://www.cppblog.com/Images/OutliningIndicators/None.gif

#0  CSendFile::SendFile (this=0x2acc5d4aff40, pathname=@0x2acc5d4afee0) at ../src/csendfile.cpp:101 http://www.cppblog.com/Images/OutliningIndicators/None.gif

#1  0x000000000040592e in CIcdn::TaskThread (pParam=0x7fff617eafe0) at ../src/cicdn.cpp:128 http://www.cppblog.com/Images/OutliningIndicators/None.gif

#2  0x00002acc4a90b73a in start\_thread () from /lib/libpthread.so.0 http://www.cppblog.com/Images/OutliningIndicators/None.gif

#3  0x00002acc4a40d6dd in clone () from /lib/libc.so.6 http://www.cppblog.com/Images/OutliningIndicators/None.gif

#4  0x0000000000000000 in ?? ()

来看一下101行的代码：

http://www.cppblog.com/Images/OutliningIndicators/None.gif(gdb) l http://www.cppblog.com/Images/OutliningIndicators/None.gif

96      } http://www.cppblog.com/Images/OutliningIndicators/None.gif

97 http://www.cppblog.com/Images/OutliningIndicators/None.gif

98      int CSendFile::SendFile(const string& pathname) http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gif

99      {

100             int n;

101             while(1)

102             {

103                     n++;

104             }

105             //read file and send

现在我们定位到了出问题的代码位置，这里的循环只用来演示的。   
最后别忘了detach（）

调试完指定进程后，可以运行detach命令来让GDB释放该进程，该进程得以继续运行。当回车时，detach不会重复。当执行完detach后，进程和GDB不再相关，GDB可以attach其他进程。

2. 使用 pstack

1.安装pstack

yum install pstack

使用方式：

>pstack pid 根据pid找出有问题的线程

>strace -p pid //这个线程所有系统调用