C:\Users\MINGZH~1.QUN\AppData\Local\Temp\Wiz\425fa01f-abba-4aa6-832f-aa3c744481ff_4_files\artType02[1].jpg 过Java thread dump分析找到耗费CPU最高的源代码

2010-09-22 18:00:23

标签：[thread dump](http://blog.51cto.com/tag-thread%20dump.html" \t "_blank) [休闲](http://blog.51cto.com/tag-%E4%BC%91%E9%97%B2.html) [职场](http://blog.51cto.com/tag-%E8%81%8C%E5%9C%BA.html)

最近产品在运行过程中出现了性 能问题，在很低的流量的情况下CPU就达到40%，流量稍高时CPU就达到98%。

产品是Java写的，运行于JBOSS平台。操作系统为redhat linux。当你通过top命令发现你的应用程序的进程占用CPU达98%时，我想你肯定想知道究竟是哪个地方耗费了如此的CPU处理时间。通过thread dump分析就可以找到，但这只是解决问题的第一步，即找到问题的所在。

首先：如何产生thread dump日志？

第一步：找到应用程序所在的进 程号，通过top命令可以找到，目的是找到cpu使用过高的进程pid。

第二步：执行kill -3 pid获取thread dump日志（pid就是第一步获取到的）。注意：cpu占用过高，只能用kill -3 命令，其他任何工具可能都连不上系统，导致无法使用。kill -3 的输出结果，一般在服务的log里。

第三步：获取线程信息

大多数服务器应用都是多线程， 因此必须查到具体是哪些线程占用的CPU高。通过top –H命令可以查看到应用程序的线程信息及占用CPU的情况。

如下所示：

 PID USER      PR NI VIRT RES SHR S %CPU %MEM    TIME+ COMMAND

 4280 nbg-syst 18   0 3608m 2.0g 21m R 93.6 25.9   5004:49 java

 4279 nbg-syst 18   0 3608m 2.0g 21m R 92.6 25.9   4876:40 java

 4281 nbg-syst 18   0 3608m 2.0g 21m R 92.6 25.9   3892:54 java

 4282 nbg-syst 18   0 3608m 2.0g 21m R 91.2 25.9   4954:40 java

 4244 nbg-syst 15   0 3608m 2.0g 21m S 3.3 25.9 168:34.04 java

PID所在的列即是对应的线程ID，这是十进制的。

第四步：找到耗费CPU高的线程及对应的源代码

取上面耗费CPU最高的第一行的PID 4280，将其转化为十六进制得到0x10b8。然后在thread dump日志中搜索0x10b8，将会搜到如下信息：

"Stack.ClientSelector-1" daemon prio=10 tid=0x000000004baeec00 nid=0x10b8 runnable [0x0000000053169000..0x0000000053169c90]

   java.lang.Thread.State: RUNNABLE

       at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)

       at sun.nio.ch.EPollArrayWrapper.poll(Unknown Source)

       at sun.nio.ch.EPollSelectorImpl.doSelect(Unknown Source)

       at sun.nio.ch.SelectorImpl.lockAndDoSelect(Unknown Source)

       - locked <0x00002aaac4105468> (a sun.nio.ch.Util$1)

       - locked <0x00002aaac4131670> (a java.util.Collections$UnmodifiableSet)

       - locked <0x00002aaac3f79c78> (a sun.nio.ch.EPollSelectorImpl)

       at sun.nio.ch.SelectorImpl.select(Unknown Source)

       at com.\*\*\*\*\*.warlock.protocolstack.impl.layer2.nio.ActiveSelectorImpl.callSelect(ActiveSelectorImpl.java:288)

       at com. \*\*\*\*\*.warlock.protocolstack.impl.layer2.nio.ActiveSelectorImpl.run(ActiveSelectorImpl.java:163)

上面日志中的nid即是线程号。这样可以清晰的看到耗费CPU的源代码的具体位置，可以精确到行号。

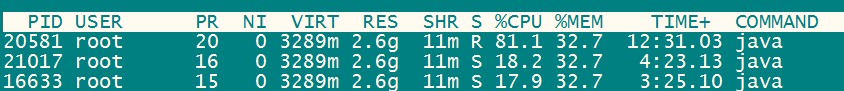
备注：上面有部分采用\*\*\*\*\*是为了屏蔽公司版权信息而设置的，不是\*哈。以上举例都是基于HP Blade硬件，redhat企业版操作系统和Sun的JDK。

示例：

1、先用top 看成cpu最高的进程号  
C:\Users\MINGZH~1.QUN\AppData\Local\Temp\Wiz\425fa01f-abba-4aa6-832f-aa3c744481ff_4_files\33c5c27af6279a0eacc1450b5669ac44.jpg  
2、再用kill -3 20568 生成jvm thread dump  
数据生成到/user/local/resin/log/jvm-default.log中

(

可以用jstate -gcutil 20568  1000看gc回收日期信息，1000是指打印1000行

)  
3、再用top H 查看用cpu最高的线程号  
  
4、然后把线程号转换成16进制（System.*out*.println(Integer.*toHexString*(20581)); ）  
   
  
5、在jvm-default.log中查找线程在干啥