01、高cpu占用

1、代码

```
public class FindJavaThreadInTaskManager {
    public static void main(String[] args) {
          Thread thread = new Thread(new Worker());
          thread.start();
    }
 static class Worker implements Runnable {
        @Override
        public void run() {
            while (true) {
                System.out.println("Thread Name:" + Thread.currentThread().getName());
             }
        }
    }
}
```

2、运行

java -jar jstacktopcpu.jar

3、top

选择大写P，进程按照cpu排序，找到最耗内存的java进程。

4、获取线程号

ps p 28348 -L -o pcpu,pid,**tid**,time,tname,cmd

5、将获取的线程号（十进制数）转换成十六进制,此处为0x**b46**

 printf "%x\n" 28376

0x6ed8

6、查看进程PID为28348 中

nid为0x6ed8的线程信息。

命令：

jstack -l 28348 >28348.jstack

完整文档:

java程序CPU利用率高怎么办

请jstack神器来帮忙

本文介绍Linux环境下使用jstack定位问题的秘笈

工具/原料

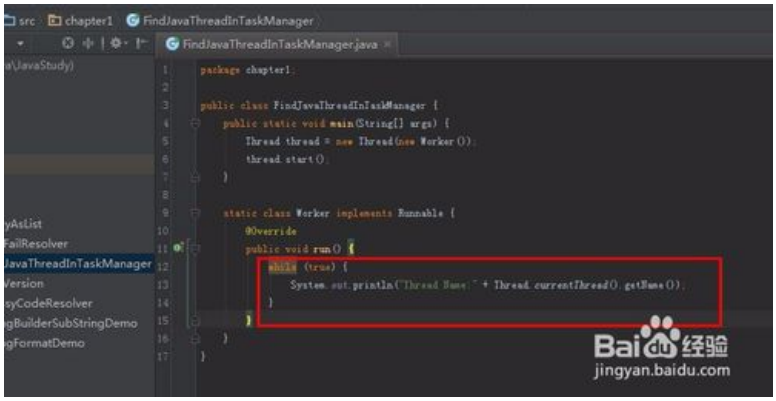
- Linux
- java
- thread
- jstack
- top
- ps

- printf
- Runnable

方法/步骤

1. 一个CPU密集型线程的demo：

```
package chapter1;
public class FindJavaThreadInTaskManager {
    public static void main(String[] args) {
            Thread thread = new Thread(new Worker());
            thread.start();
    }
 static class Worker implements Runnable {
        @Override
        public void run() {
            while (true) {
                System.out.println("Thread Name:" + Thread.currentThread().getName());
             }
         }
     }
}
```



2. 将上述代码打成Jar。

在Linux上执行上述代码

命令：

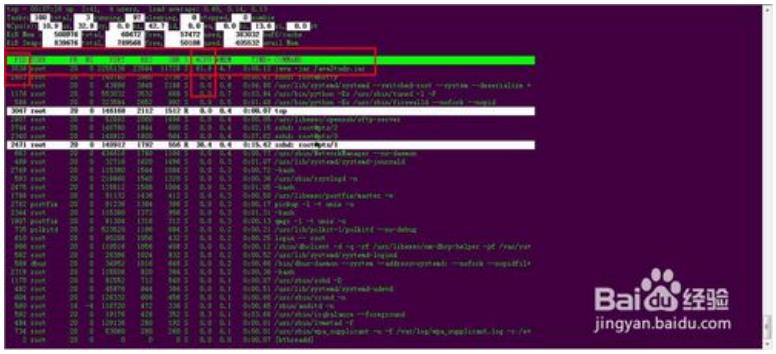
java -jar jstack.jar

ps p 27892 -L -o pcpu,pid,**tid**,time,tname,cmd

3. 找到CPU利用率持续比较高的进程，获取进程号，此处PID为 27892

命令：

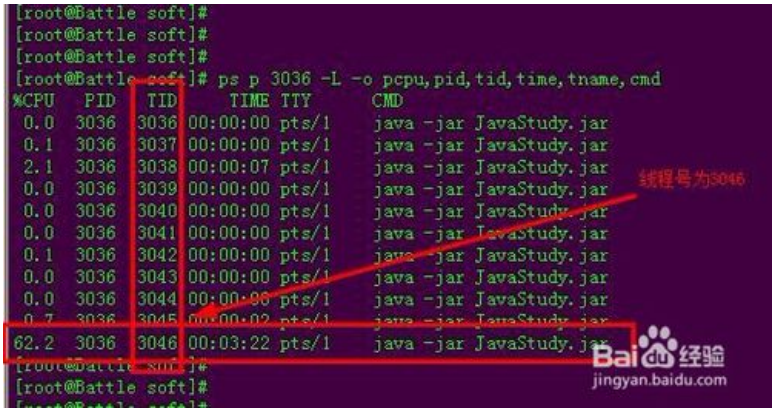
top



4. 找到上述进程中，CPU利用率比较高的线程号TID（十进制数）,此处为3046

命令：

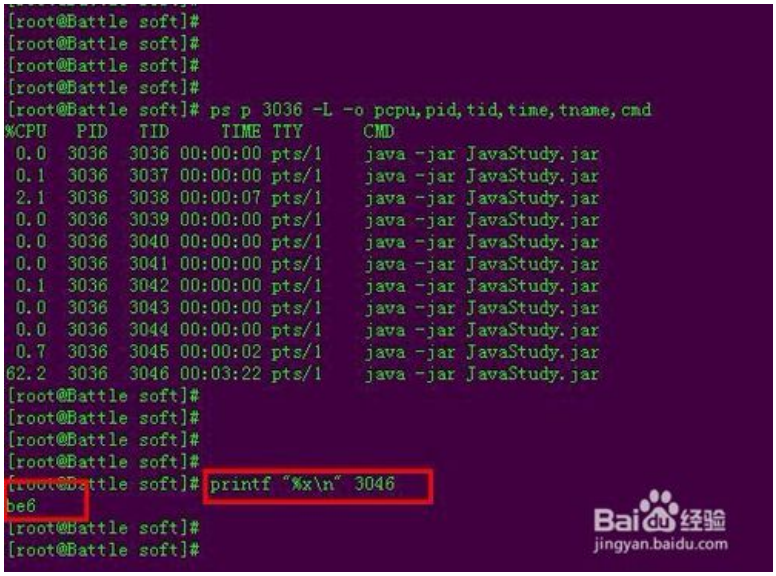
ps p 27892 -L -o pcpu,pid,**tid**,time,tname,cmd



5. 将获取的线程号（十进制数）转换成十六进制,此处为0x**b46**

命令：

printf "%x\n" 27892

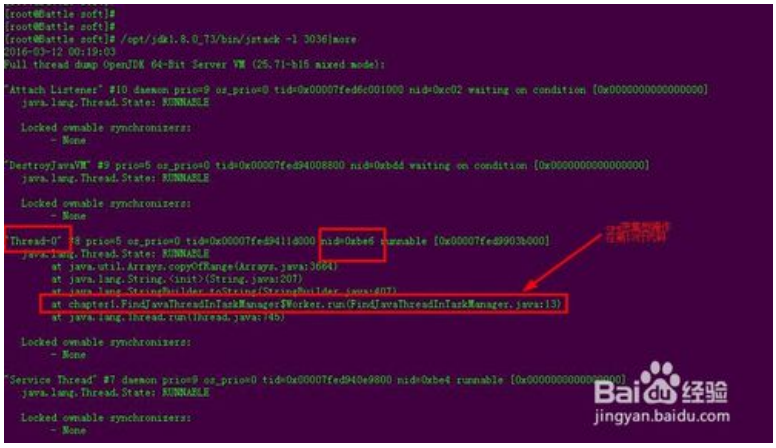


6. 查看进程PID为3036中

nid为0x**6cf4**的线程信息。

命令：

jstack -l 27892



7. 总结：

可以看到jstack命令的输出结果是相当准确的：
显示耗CPU比较高的代码与实际情况相同，都是第13行。
放心的用吧。

Enjoy youself!