

Tomcat打印线程信息和堆Dump文件



使用背景：

一般应用于服务器繁忙或是无响应时，我们需要看一下线程的状态和详细信息。



注意事项：

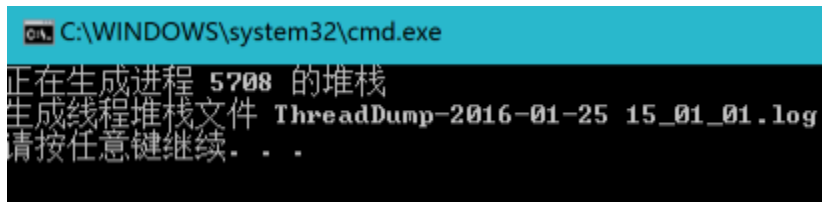
先不要重启服务器，在服务器无响应的时候执行下面获取步骤。

执行ThreadDump.cmd\ThreadDump.sh生成日志

如果是使用Smartbi的安装包部署的，打开<smartbi>\Tomcat\bin目录并双击ThreadDump.cmd就可以生成线程堆栈日志文件。

如果单独使用war部署或Tomcat\bin目录中不存在ThreadDump.cmd，可以点击下载[ThreadDump.cmd](#)（Linux系统使用文件：[ThreadDump.sh](#)）和[ThreadDump.jar](#)，并**放到Tomcat\bin目录**中，先在cmd添加“cd tomcat\bin目录”，然后右键使用管理员身份执行该cmd文件生成线程堆栈（**注意：请将ThreadDump.cmd或ThreadDump.sh文件中的jdk路径修改为服务器上的jdk路径**）

使用Linux系统时将ThreadDump.sh和ThreadDump.jar通过FTP、SFTP等上传到Tomcat/bin目录中，通过SSH等连接到服务器进入Tomcat/bin目录并执行chmod 755 ThreadDump.sh令它可执行，再执行./ThreadDump.sh生成日志。



如果通过上述命令无法正常获取线程堆栈日志，请再以下步骤操作。

执行命令行获取线程堆栈信息

Windows

一、线程信息

请在服务器无响应时，执行打印进程信息，打印线程信息有两个方法：

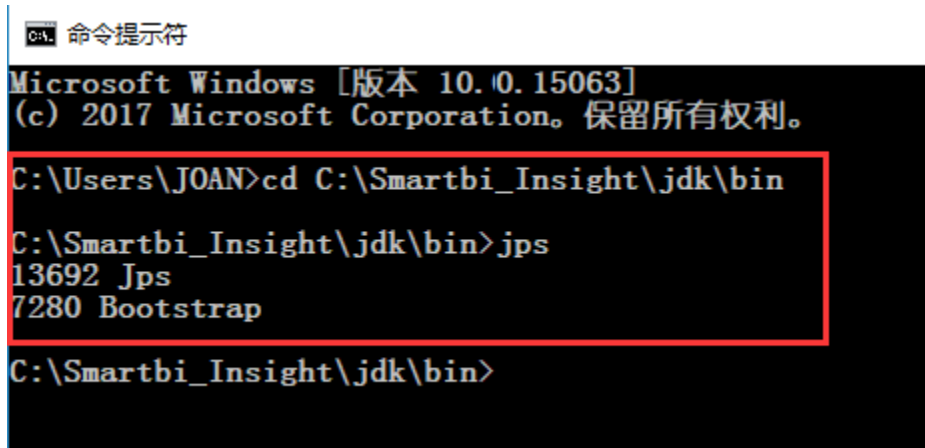
方法一

要求JDK版本为1.6及其以上版本。

- 1、在运行中打开cmd命令行窗口。
- 2、在cmd窗口进入JDK的bin目录下，执行jps获取进程信息，此处要保证执行的JDK是服务器使用的JDK。

如命令行：cd C:\Smartbi_Insight\jdk\bin

jps



3、获取对应线程号，然后执行jstack +进程号 > 进程号.log 获取线程信息：Tomcat显示的名称应该是Bootstrap;

C:\> 命令提示符

```
Microsoft Windows [版本 10.0.15063]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\JOAN>cd C:\Smartbi_Insight\jdk\bin

C:\Smartbi_Insight\jdk\bin>jps
13692 Jps
7280 Bootstrap

C:\Smartbi_Insight\jdk\bin>jstack 7280 >7280.log

C:\Smartbi_Insight\jdk\bin>
```

4、可以在 C:\Smartbi_Insight\jdk\bin 目录下看到生成的线程文件。

> 此电脑 > Windows8_OS (C:) > Smartbi_Insight > jdk > bin

名称	修改日期	类型	大小
7280.log	2017/7/20 11:33	文本文档	38 KB
appletviewer.exe	2014/2/20 18:59	应用程序	16 KB
apt.exe	2014/2/20 18:59	应用程序	16 KB

方法二

- 1、通过CMD命令行窗口进入JDK的bin目录下
- 2、使用netstat -ano|find “访问的端口号”|find “LISTENING” 获取进程号（注意双引号是必须的）
如访问smartbi的端口号是18080，则输入命令：netstat -ano|find “18080”|find “LISTENING”

C:\> 命令提示符

```
Microsoft Windows [版本 10.0.15063]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\JOAN>cd C:\Smartbi_Insight\jdk\bin

C:\Smartbi_Insight\jdk\bin>netstat -ano|find "18080"|find "LISTENING"
TCP 0.0.0.0:18080 0.0.0.0:0 LISTENING 7280

C:\Smartbi_Insight\jdk\bin>
```

- 3、再使用jstack 进程号 >进程号.log 获取线程信息，如果提示无法连接到JVM，可以增加-F参数强制生成jstack -F 进程号 >进程号.log
- 上边截图获取到进程号是7280，可输入命令行：jstack 7280 >7280.log

C:\ 命令提示符

```
Microsoft Windows [版本 10.0.15063]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\JOAN>cd C:\Smartbi_Insight\jdk\bin

C:\Smartbi_Insight\jdk\bin>netstat -ano|find "18080"|find "LISTENING"
TCP        0.0.0.0:18080          0.0.0.0:0             LISTENING     7280

C:\Smartbi_Insight\jdk\bin>jstack 7280 >7280.log

C:\Smartbi_Insight\jdk\bin>
```

此电脑 > Windows8_OS (C:) > Smartbi_Insight > jdk > bin

名称	修改日期	类型	大小
7280.log	2017/7/20 11:38	文本文档	38 KB
appletviewer.exe	2014/2/20 18:59	应用程序	16 KB
apt.exe	2014/2/20 18:59	应用程序	16 KB

或者：输入命令行：jstack -F 7280 >test.log

```
C:\Smartbi_Insight\jdk\bin>jstack -F 7280 >test.log
Attaching to process ID 7280, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 24.51-b03

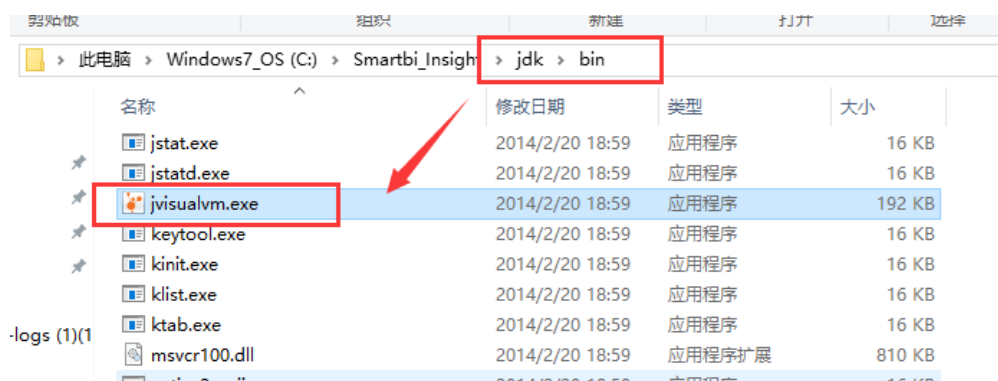
C:\Smartbi_Insight\jdk\bin>
```

此电脑 > Windows8_OS (C:) > Smartbi_Insight > jdk > bin

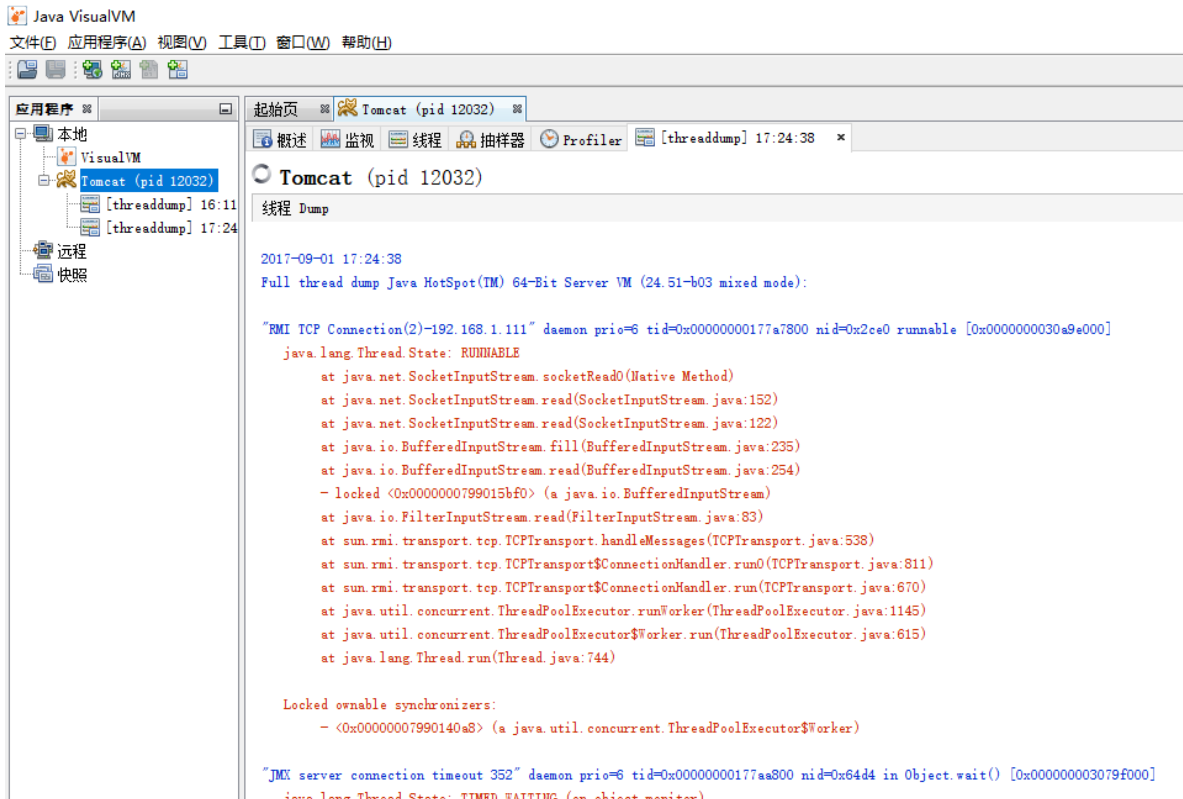
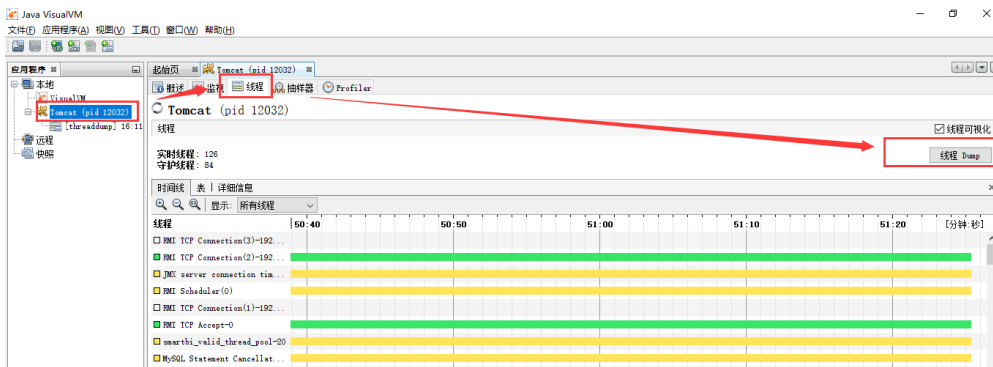
名称	修改日期	类型	大小
test.log	2017/7/20 11:40	文本文档	24 KB
7280.log	2017/7/20 11:38	文本文档	38 KB
appletviewer.exe	2014/2/20 18:59	应用程序	16 KB
apt.exe	2014/2/20 18:59	应用程序	16 KB

方法三

1、进入jdk/bin，执行jvisualvm.exe



2、找到对应的线程号，然后进入线程 --> 线程 dump



3、将线程信息复制到文本发回即可

二、堆dump文件

1、同“一、线程信息”打印线程信息步骤一样，需要先进入对应的JDK的bin目录下，通过jps或netstat获取Java进程号：

2、在JDK的bin目录运行【jmap -histo:live 进程号 >进程号.map】

假设获取到的进程号是7280，则输入命令行为：jmap -histo:live 7280 >7280.map

```
C:\Smartbi_Insight\jdk\bin>jmap -histo:live 7280 >7280.map
```

3、反馈文件以跟踪问题

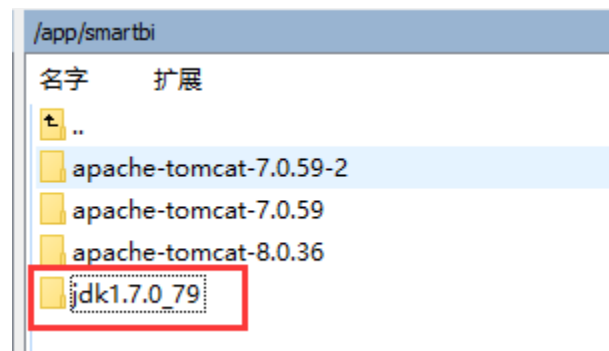
[illegible]

Linux

一、线程信息

执行命令行生成日志:

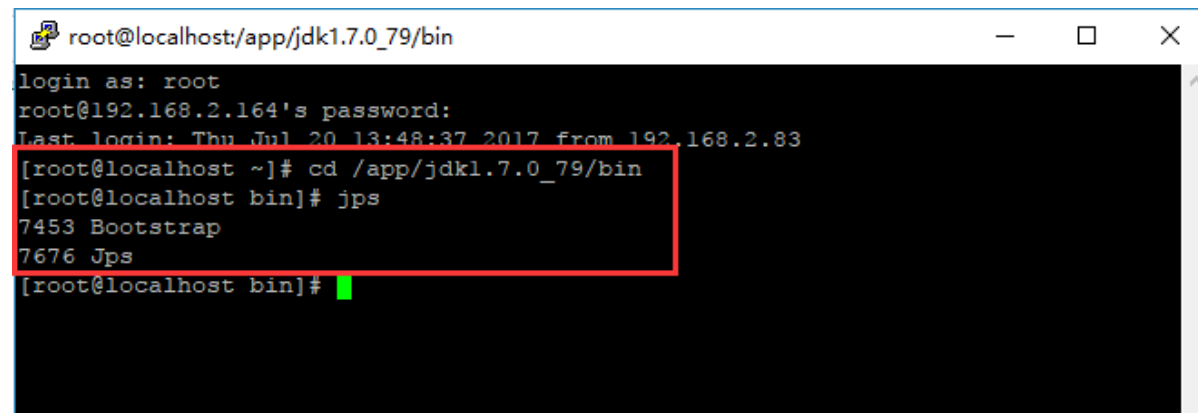
1) 另外启动一个xshell连接窗口, 进入到JDK的路径下 (不知道如何选择适合的JDK, 可看最下边备注信息), 如:



2) 进入到 /app/jdk1.7.0_79/bin 目录下, 并输入 jps 命令, Tomcat显示的名称Bootstrap:

如命令行: cd /app/jdk1.7.0_79/bin

jps



3) 在窗口继续输入 jstack 7453 (就是输入第二步命令获取到的tomcat的线程号), 开始打印线程 (这一步也可以直接跳过进行第四步操作)

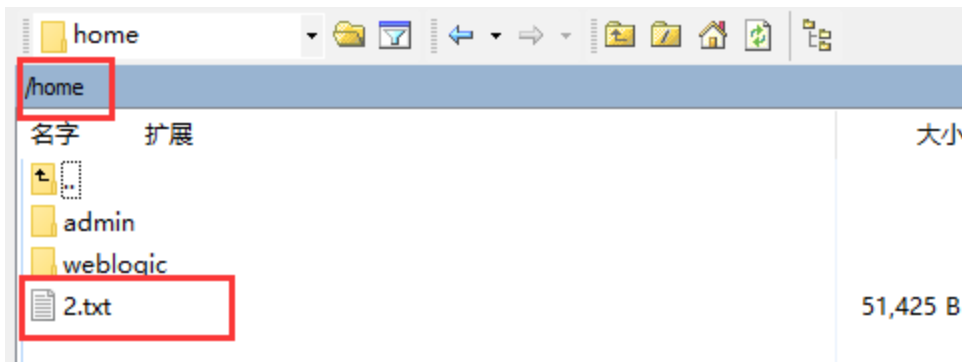
```
root@localhost:/app/jdk1.7.0_79/bin
login as: root
root@192.168.2.164's password:
Last login: Thu Jul 20 13:50:39 2017 from 192.168.2.83
[root@localhost ~]# cd /app/jdk1.7.0_79/bin
[root@localhost bin]# jps
7906 Jps
7453 Bootstrap
[root@localhost bin]# jstack 7453
2017-07-20 13:55:44
Full thread dump Java HotSpot(TM) 64-Bit Server VM (24.79-b02 mixed mode):

"RMI TCP Connection(3)-127.0.0.1" daemon prio=10 tid=0x00007f0c78003000 nid=0x1e
ec runnable [0x00007f0ca20e5000]
   java.lang.Thread.State: RUNNABLE
       at java.net.SocketInputStream.socketRead0(Native Method)
       at java.net.SocketInputStream.read(SocketInputStream.java:152)
       at java.net.SocketInputStream.read(SocketInputStream.java:122)
       at java.io.BufferedInputStream.fill(BufferedInputStream.java:235)
       at java.io.BufferedInputStream.read(BufferedInputStream.java:254)
       - locked <0x0000000076aa99148> (a java.io.BufferedInputStream)
       at java.io.FilterInputStream.read(FilterInputStream.java:83)
       at sun.rmi.transport.tcp.TCPTransport.handleMessages(TCPTransport.java:5
```

4) 将线程信息输出，如输出到home 路径下，输入命令为 `jstack 7453 >> /home/2.txt`

```
[root@localhost bin]# jstack 7453 >> /home/2.txt
[root@localhost bin]#
```

这时候就会在服务器上home目录下生成对应的文件了，如下图，将这个文件发回分析。



二、堆Dump文件

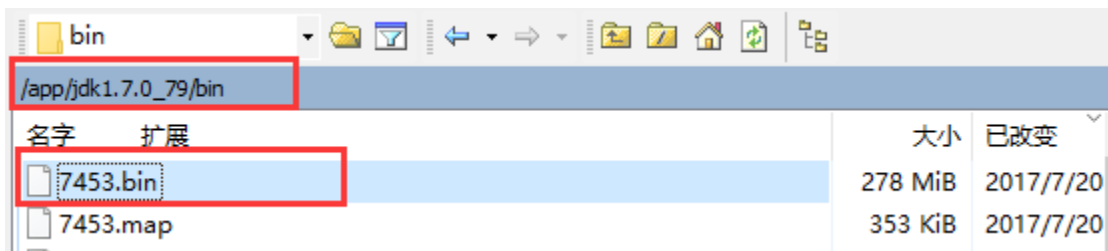
1、同“一、线程信息”打印线程信息步骤一样，需要先进入对应的JDK的bin目录下，通过jps获取Java进程号，Tomcat显示的名称应该是Bootstrap:

2、在JDK的bin目录运行，可生成完整堆文件 .bin文件 【`jmap -dump:live,format=b,file=进程号.bin 进程号`】

如输入命令行: `jmap -dump:live,format=b,file=7453.bin 7453`

```
[root@localhost bin]# jmap -dump:live,format=b,file=7453.bin 7453
Dumping heap to /app/jdk1.7.0_79/bin/7453.bin ...
Heap dump file created
[root@localhost bin]#
```

3、该文件非常大，与Java进程的运行内存一样的大，必须压缩以后再反馈



注：反馈以上的信息有助于解决系统无响应的问题。

另，堆Dump文件还有一个map文件，为简要堆文件，但一般情况用处不大（只能看出大概的内存占用总数，无法看具体报表占用的详细信息）。

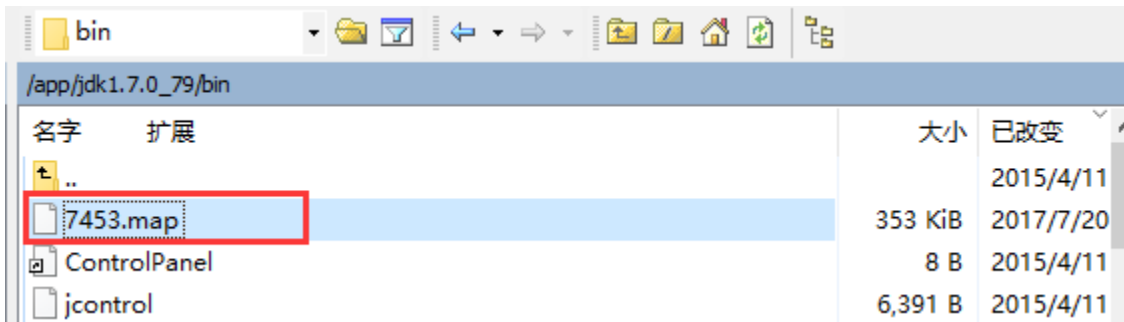
4、【jmap -histo:live 进程号 >进程号.map】

假设获取到的进程号是7453，则输入命令行为：jmap -histo:live 7453 >7453.map

这样会在JDK的bin目录下直接生成相应文件：

```
[root@localhost bin]# jmap -histo:live 7453 >7453.map
[root@localhost bin]#
```

5、反馈文件以跟踪问题



三、GC参数

1、在JDK的bin目录中运行【jstat -gcutil 进程号 5000 1000】

```
C:\jdk1.8.0_40\bin>jstat -gcutil 720 5000 1000
```

2、若最后一段GCT段（GC所花费的总时间）在持续快速增加则表示服务器的内存已经不足，若只是少量增加是正常现象：

6.662
6.662
6.662
38.682
38.682
42.734
46.856
51.466
51.466
51.466
51.466
51.466
51.466
51.466
51.466
51.466
51.466
51.466
103.320
108.377
113.070
113.070
113.070
113.070
113.070
113.070

```

/app/smartbi/apache-tomcat-7.0.59/bin/catalina.sh root@192.168.2.164
PRG=`dirname "$PRG"`${/"$link"
fi
done

# Get standard environment variables
PRGDIR=`dirname "$PRG"`
JAVA_HOME="/app/smartbi/1dk1.7.0_79"
JAVA_OPTS="$JAVA_OPTS -Xms512m -Xmx2048m -XX:PermSize=256M -XX:MaxNewSize=256M -XX:MaxPermSize=512m -Dmail.mime.splitlongparameters=false"

# Only set CATALINA_HOME if not already set
[ -z "$CATALINA_HOME" ] && CATALINA_HOME=`cd "$PRGDIR/.." >/dev/null; pwd`

# Copy CATALINA_BASE from CATALINA_HOME if not already set
[ -z "$CATALINA_BASE" ] && CATALINA_BASE="$CATALINA_HOME"

```