软件信息

|  |  |
| --- | --- |
| 软件名称 | 版本号 |
| Tomcat | 8.5.5 |

软件安装

**1.安装目录**

Tips: ${HOME}表示用户家目录 ${PRO\_NAME}表示工程名，${i}表示第i个节点，同一工程Tomcat一般部署多个节点。 一般以${HOME}/ ${PRO\_NAME}/ tomcat${i}作为部署路径。

**2.安装步骤**

(1)安装依赖软件  
Tomcat依赖于jdk，jdk采用压缩包进行部署，过程如下：

tar –xzf jdk1.7.0\_76.tar.gz –C /home/zyzx/ jdk1.7.0\_76

创建link，方便后续对JDK进行升级 ln –s /home/zyzx/ jdk1.7.0\_76 jdk

修改~/.bash\_profile 添加

export JAVA\_HOME=~/jdk

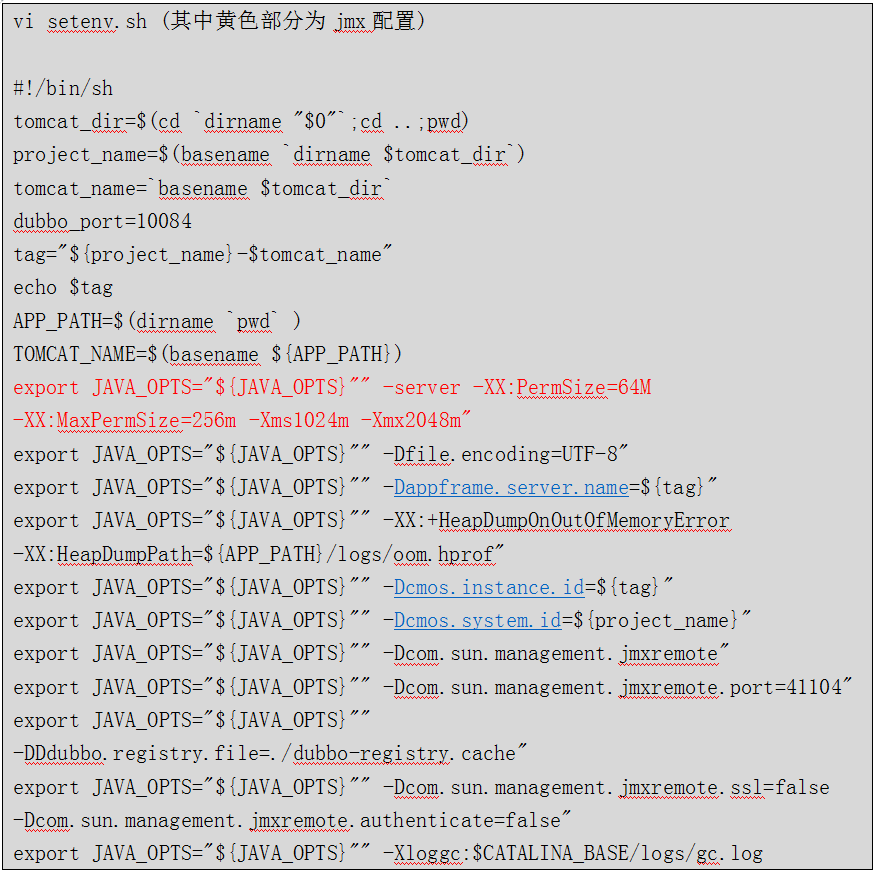
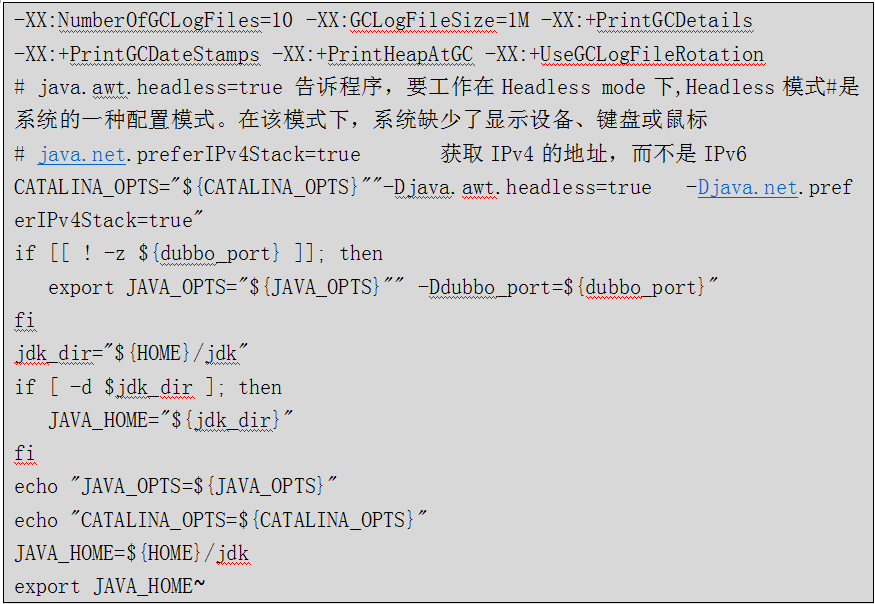
export PATH=${JAVA\_HOME}/bin:$PATH

然后执行source ~/.bash\_profile使环境变量生效。 后续对jdk升级时只需要修改链接，其他配置不需要修改即可。

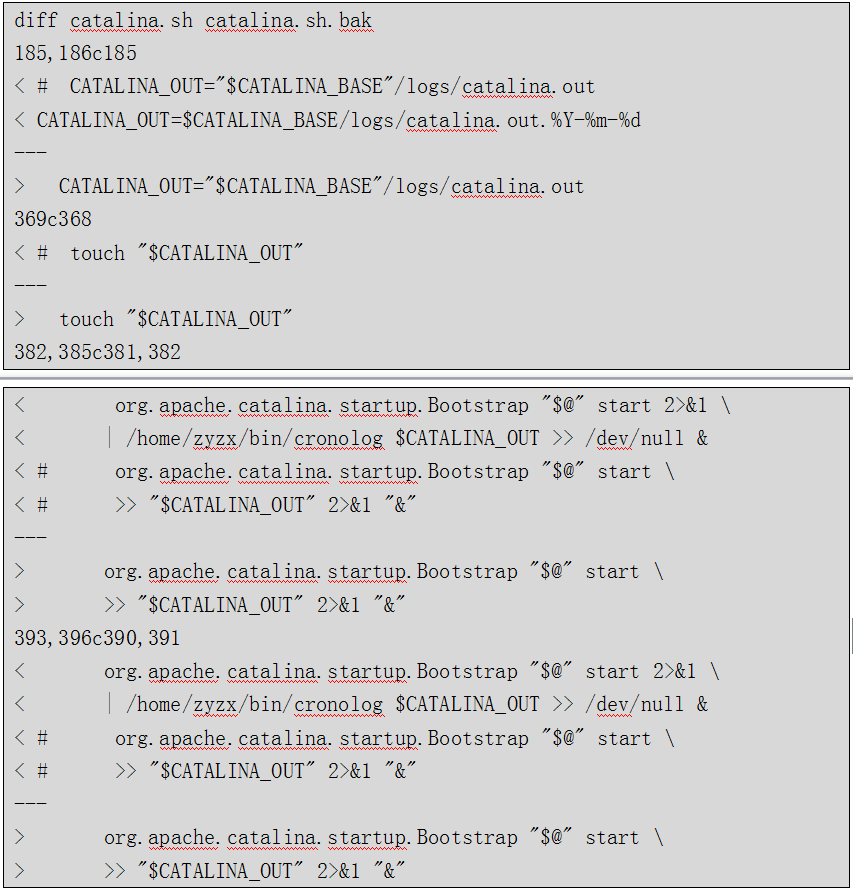
(2)安装tomcat  
Tomcat为java运行程序，在java部署后可直接启动，采用压缩包方式安装。根据工程名在家目录下创建目录${PRO\_NAME}，解压tomcat并将其重命名后放到${HOME}/ ${PRO\_NAME}下： wget https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.5/bin/apache-tomcat-8.5.5.tar.gz mkdir -p ${HOME}/ ${PRO\_NAME} tar –xzf apache-tomcat-8.5.5.tar.gz mv apache-tomcat-8.5.5 ~/${PRO\_NAME}/tomcat${i} 同一工程为保证服务一致，多个tomcat共享同一份代码，一般将代码放到家目录下app-share/${PRO\_NAME}目录下，在tomcat/webapps/${PRO\_NAME}下创建软连接到app-share目录下的代码存放目录。

软件配置

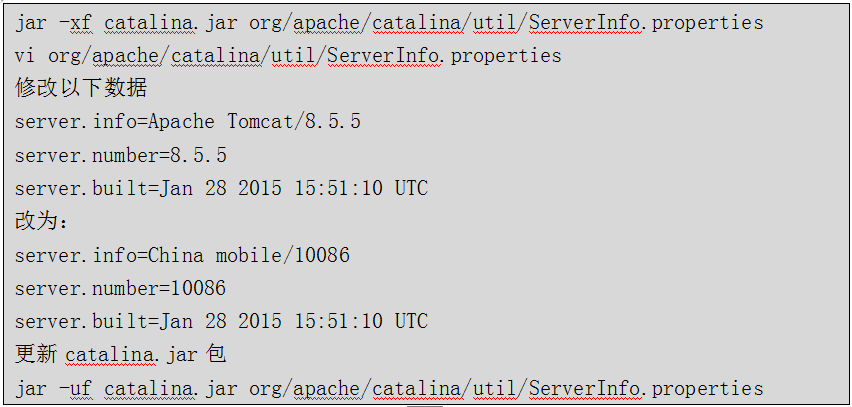
**1.setenv.sh文件配置**

在tomcat下bin目录添加setenv.sh文件，对tomcat的内存分配以及自定义变量都配置在setenv.sh文件里，避免多个tomcat相互不同时有影响，避免把自定义变量配置在用户环境变量里，影响其他进程。 其中黄色部分为jmx配置。  
cd ${HOME}/tomcat\_${PRO\_NAME}/ tomcat${i}/bin  
  


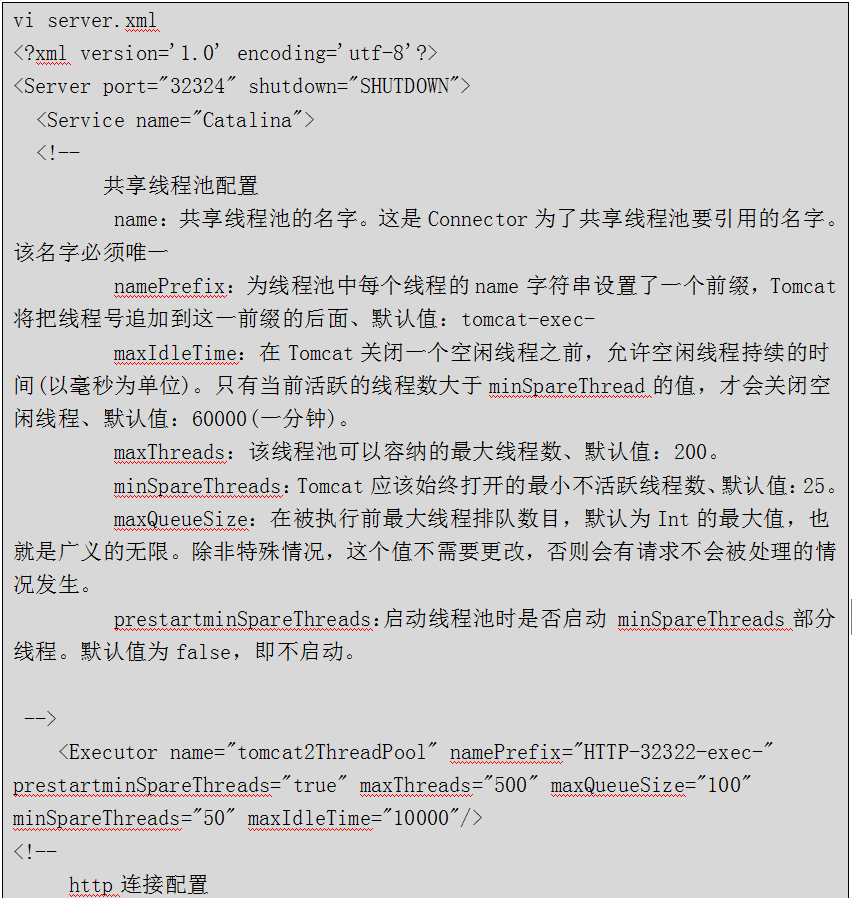
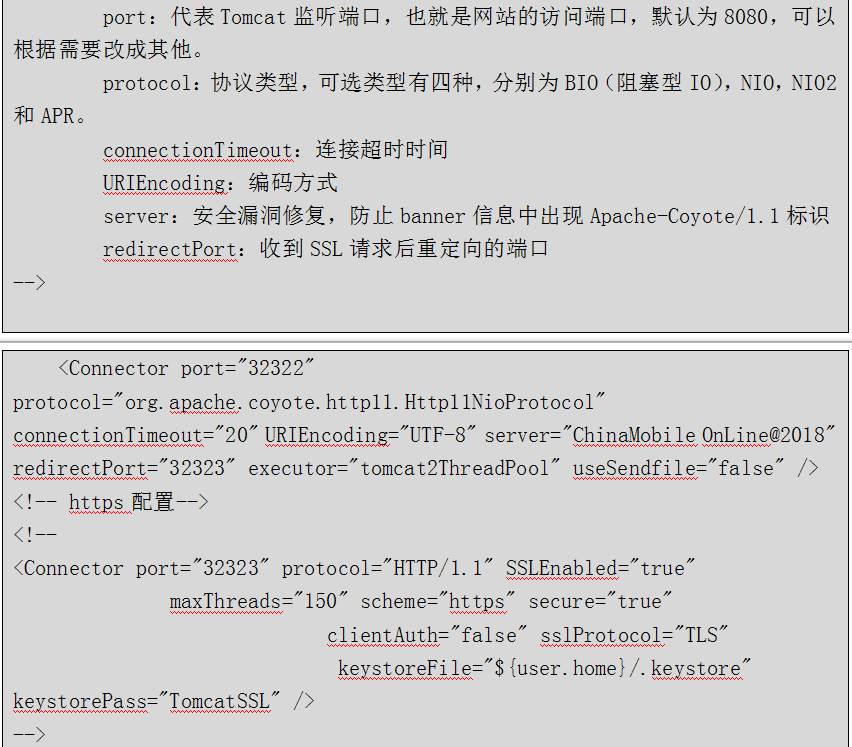
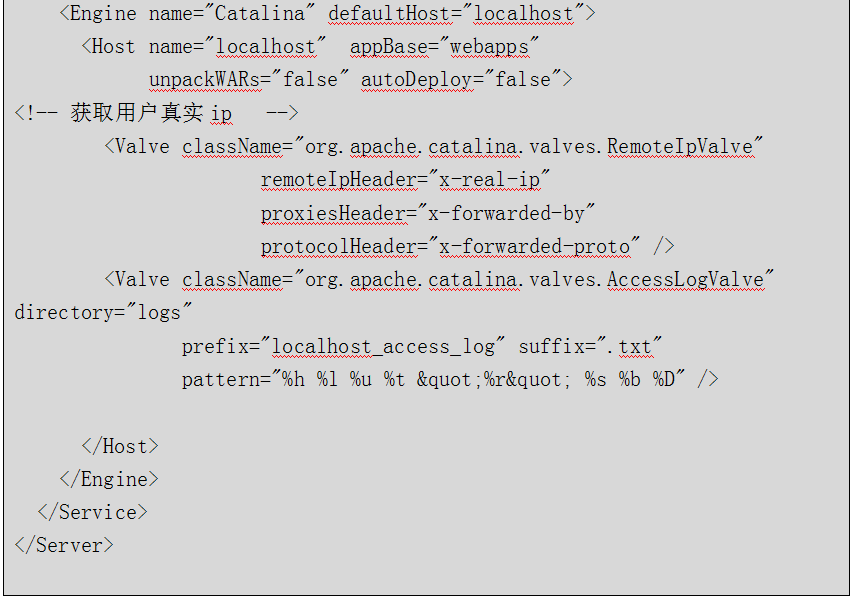
**2.cronolog日志分割**

修改tomcat下bin目录catalina.sh文件，添加cronolog日志分割配置。主要有以下几点需要修改： 日志按天分割catalina.out 命名改为catalina.out.%Y-%m-%d；注释掉touch "$CATALINA\_OUT"；记录日志增加cronolog配置；修改后的配置对比如下：  
  
其中/home/zyzx/bin/cronolog为cronolog存放路径，可以采用相对路径，放到tomcat下bin目录，配置中修改为./cronolog，这样修改后，启动tomcat要进入到bin目录下执行startup.sh，否则会报找不到cronolog文件。

**3.隐藏tomcat版本号**

修改tomcat下lib目录中catallina.jar文件  


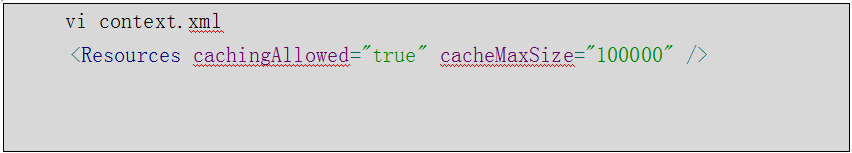
**4.server.xml配置**

修改shutdown http https(如果需要)端口号，以免出现端口冲突。Server.xml部分配置项，主要为各个端口以及http配置：  
  
  


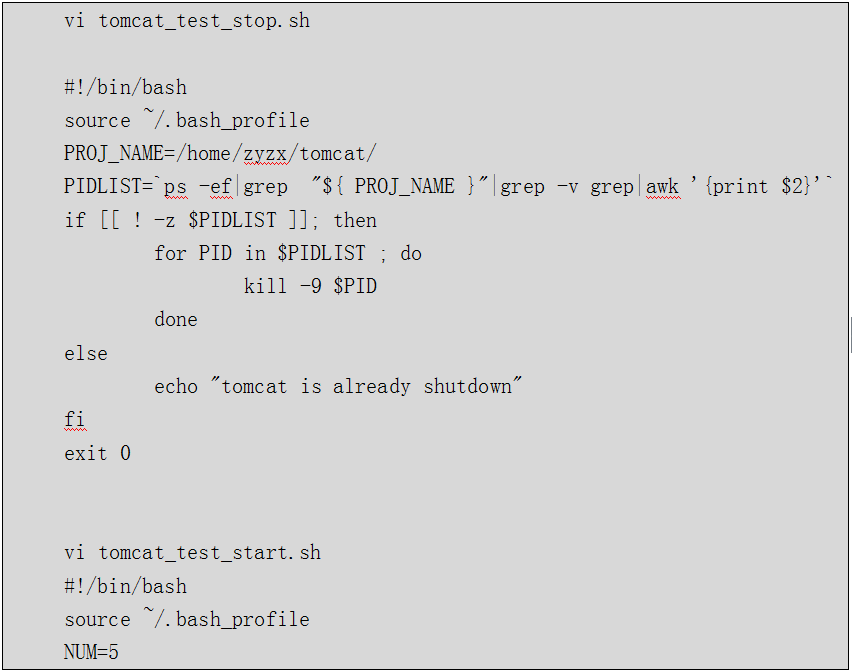
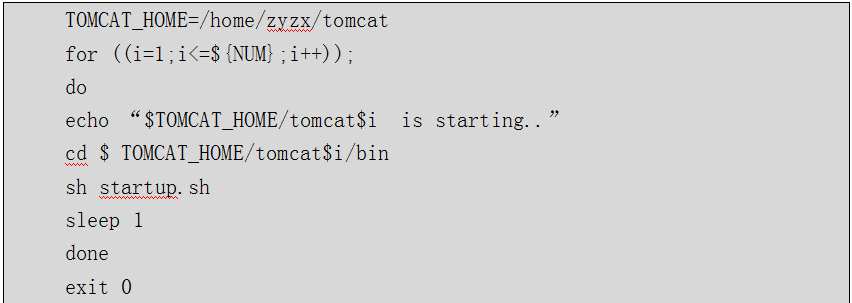
**5.修改conf/logging.properties，删除不必要的日志**

将manager host-manager日志信息删除，只保留catalina localhost

**6.修改context.xml，增大缓存空间**

Tomcat8.5.5运行时会输出Unable to add the resource at [/WEB-INF/lib/springwebmvc-3.0.2.RELEASE.jar] to the cache because there was insufficient free space available after evicting expired cache entries - consider increasing the maximum size of the cache的Warning信息，可修改context.xml增加缓存空间。  


**7.启停脚本编写**

Tomcat加载应用后，有可能出现执行shutdown.sh关不掉的情况，因此需要使用kill -9来杀掉应用。 启停脚本统一放到${HOME}/shell目录下。  
  


#### 日常维护操作

#### 1.操作手册总则

【强制】 除紧急情况，Tomcat配置修改务必通过业务运维组变更操作评审，且在上线期间进行  
【强制】 所有生产配置变更必须备份

#### 2.端口规划手册

**端口定义：**  
Tomcat现网中暴露的端口有http、https、shutdown、ajp（一般不用）、jmx和dubbo共5个端口，原则上Tomcat端口选用10000以上：  
第一位对应三层架构的位置，web、control、core分别为1，2，3，如ngcs第一位为1，ngcscontrol第一位为2，ngcscore第一位为3；  
第二位用于区分同一服务器中的不同应用，如ngcscontrol为0，ngbacontrol为1，ngcctcontrol为2，……；  
第三位保留；  
第四位分别对应http、https、shutdown、jmx和dubbo端口，分别为1，2，4，5，6；  
第五位用于区分同一tomcat节点，如tomcat1为1，tomcat2为2……。

#### 3.启动停止手册

**(1)启停操作**  
启停脚本置于~/shell下，并以${project-name}\_start.sh、${project-name}\_stop.sh或start\_group.sh、start\_group.sh为命名规范

**(2)批量启动**  
执行sh ${project-name}\_start.sh或start\_group.sh ${project-name}启动；单个启动时，先cd到该tomcat节点的bin目录，执行sh startup.sh启动。

**(3)批量停止**  
执行sh ${project-name}\_stop.sh或stop\_group.sh ${project-name}启动；单个停止时，先ps找出该tomcat的pid，然后执行kill -9 pid的形式停止。

#### 4.日志操作规范

**(1)tail -f使用场景**  
tomcat启动后，实时查看日志输出，观察tomcat是否运行正常；

**(2)grep使用场景**  
用于过滤指定关键字进行日志查看、统计等，建议输出重定向于临时文件进行操作；

**(3)less使用场景**  
用于查看整个日志文件，相较于vim有占用内存小、打开速度快等优点

#### 5.维护类

**(1)内存溢出**  
问题定界：按照《中移在线Tomcat安装部署规范》中setenv.sh配置，进行GC日志打印输出，用于内存问题预警、定界和GC情况查看；执行jmap -heap pid查看堆内存使用情况；执行jstat -gc pid 1000 10查看堆内存使用情况和YoungGC、FullGC次数和时间，确定是否为内存问题；  
OutOfMemoryError: PermGen space：永久代内存溢出，主要是由于加载的Class太多导致的，如果你的应用中没有使用到动态类加载(如asm、cglib或者spring的动态代理等技术)，只需提高-XX:MaxPermSize=128M参数的值即可；  
OutOfMemoryError: unable to create new native thread: 创建的线程太多了，需要减少应用程序创建的线程数量。线程的数量可以通过jstack pid或者top –Hp pid命令计算出来。比如：jstack -l pid |grep "nid=" | wc –l  
OutOfMemoryError: Java heap space: 堆内存溢出，这种是最常见的，遇到这种类型我们通常分为几个步骤进行排查:  
①使用jmap –histo pid命令导出内存使用直方图  
②使用jmap –dump:format=b,file=heap.hprof pid命令导出内存dump文件，然后再使用Eclipse Memory Analyze工具分析导出来的内存dump文件，这个软件的下载地址：https://www.eclipse.org/mat，通过该工具的强大引用链分析能力最终可以定位出具体是哪个类里的Map泄漏了。 [点击下载"java性能问题"文档](http://117.159.206.246:28080/operation-resource/fujian/java.rar)

**(2)高CPU**  
问题定界: 该类问题对外体现就是回响应慢，外部模块有大量的超时。如果单板CPU核数较少，内存泄漏也会导致高CPU使用率。使用top命令可以看出各个应用进程CPU的使用率。  
①使用top –Hp pid命令查看各个线程的CPU使用率；  
   ②使用jstack –l pid命令导出java进程的线程堆栈。  
  根据步骤①中的PID列(十进制)可以找到对应的线程，比如115644对应的线程是0x1c3bc，在②中查找对应线程内容进行排查，重复这个步骤多次(如5次)，对比jstack命令的输出结果，确定步骤2中统计到的高CPU占用线程执行的高频函数。

**(3)线程阻塞**  
问题定界: 典型的现象性能测试TPS压不上去，但是CPU使用率不高。  
常见的导致线程阻塞的操作有数据库操作、文件操作、object.wait、Thread.sleep、socket读写操作等，首先可以先尝试使用jstack –l pid命令打印出线程堆栈比如下面显示了数据库操作的线程堆栈：线程的状态是RUNNABLE的，但是其实我们知道这个线程实际上是阻塞在网络IO上的。也就是说RUNNABLE状态的线程不一定真是运行的(正在使用CPU)，但是WAITING状态的线程一定是阻塞的，我们多执行几次jstack命令，如果发现线程“plugin-continer-16-1“堆栈阻塞在socketRead0方法上的概率很高，那很有可能这个数据库操作就是瓶颈所在。  


#### 高危风险操作

#### 1.起停中的高危操作

**严禁ansible批量启动**  
Core层服务启动脚本中多节点启动时，如脚本中无sleep操作，严禁用ansible批量启动，防止将数据库顶死；

#### 2.日志操作中的高危操作

**(1)使用less代替vim**  
严禁用Vim操作超过2G的文件，防止占用过多内存导致服务器挂死；

**(2)配置文件备份**  
配置文件批量修改务必备份，find命令操作前先确认目标文件是否准确；

#### 其他注意事项

#### 1.观察启动日志

启动后注意观察日志是否正常输出，如未成功杀掉进程可能会有端口占用等问题；

#### 2.定时清理日志文件

日志清理脚本中注意清理dump大文件。