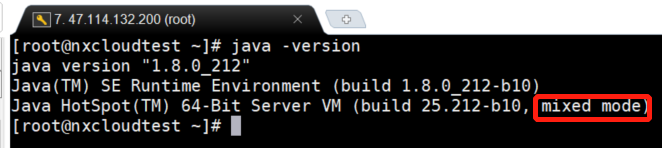
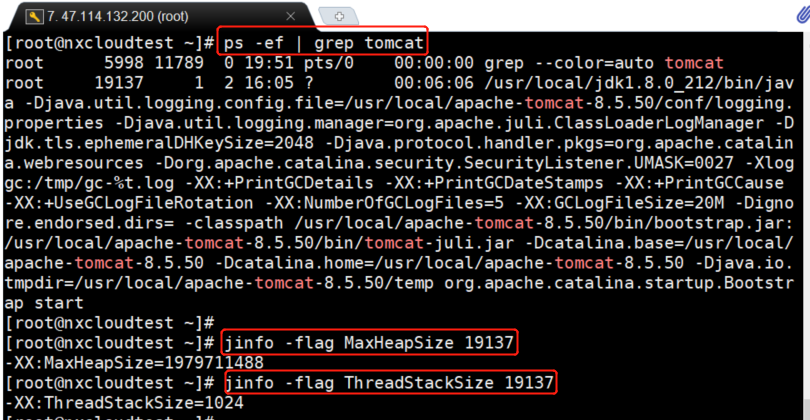
#### 基于JDK命令行工具的监控

1. JVM的参数类型
   1. 标准参数
      1. -help
      2. -server -client
      3. -version -showversion
      4. -cp -classpath
   2. X参数（非标准化参数）
      1. -Xint：解释执行
      2. -Xcomp：第一次使用就编译成本地代码
      3. -Xmixed：混合模式，JVM自己来决定是否编译成本地代码
      4. 示例：



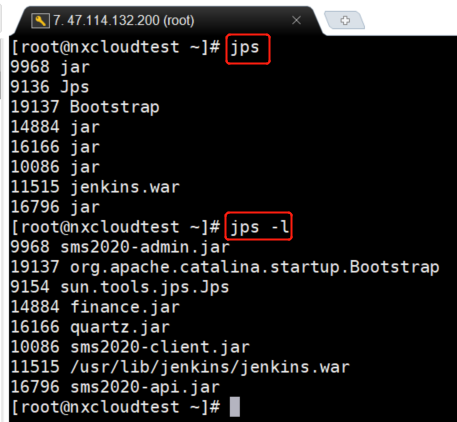
* 1. XX参数
     1. Boolean类型
        1. 格式：-XX:[+-]<name>表示启用或者禁用name属性
        2. 示例：
           1. -XX:+UseConcMarkSweepGC 启用CMS垃圾收集器
           2. -XX:+UseG1GC 启用G1垃圾收集器
     2. 非Boolean类型
        1. 格式：-XX:<name>=<value> 表示name属性的值是value
        2. 示例：
           1. -XX:MaxGCPauseMillis=500 GC的最大停顿时间是500毫秒
           2. XX:GCTimeRatio=19
  2. 参数示例说明
     1. -Xmx -Xms 设置JVM的最大/最小内存
        1. 说明：不是X参数，而是XX参数
        2. 原型：
           1. -Xms 等价于 -XX:InitialHapSize
           2. -Xmx 等价于 -XX:MaxHeapSize
        3. 示例查询



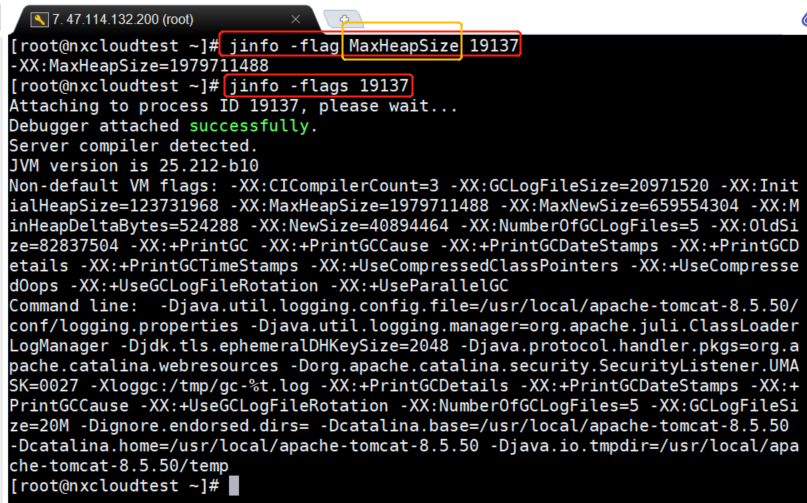
1. 运行时JVM参数查看
   1. [Oracl官网查询JVM命令](https://docs.oracle.com/javase/8/docs/technotes/tools/unix/index.html)
   2. -XX:+PrintFlagsInitial 查看初始值
   3. -XX:+PrintFlagsFinal 查看最终的值
      1. 输入示例：java -XX:+PrintFlagsFinal
      2. 结果说明：



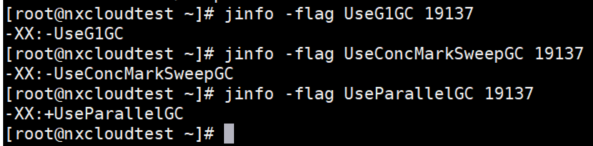
* 1. -XX:+UnlockExperimenttalVMOptions 解锁实验参数
  2. -XX:+UnlockDiagnosticVMOptions 解锁诊断参数
  3. -XX:+PrintCommandLineFlags 打印命令行参数
  4. 补充命令说明
     1. jps 查询在运行时程序的进程



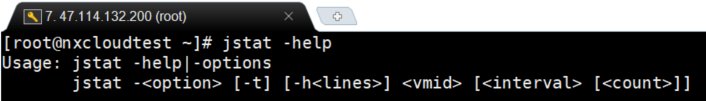
* + 1. jinfo 查询在运行中的JVM参数



说明：可查询指定进程明确/全部的参数信息

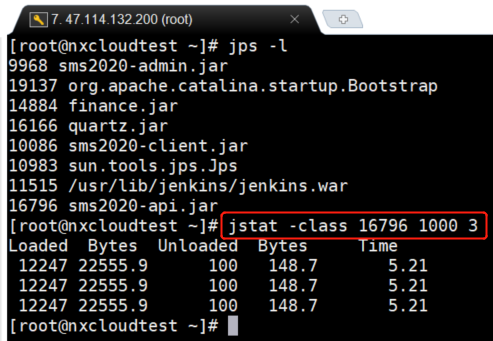


1. jstat查看虚拟机统计信息
   1. 命令格式



options：-class，-compiler，-gc，-printcompilation

* 1. 类加载信息(-class)



填写参数说明：

16796：进程编号

1000：1000毫秒统计一次

3：总共统计三次

[返回信息说明](https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstat.html" \l "BEHHGFAE) - 搜索“-class option”：

Loaded: 已加载的类数.

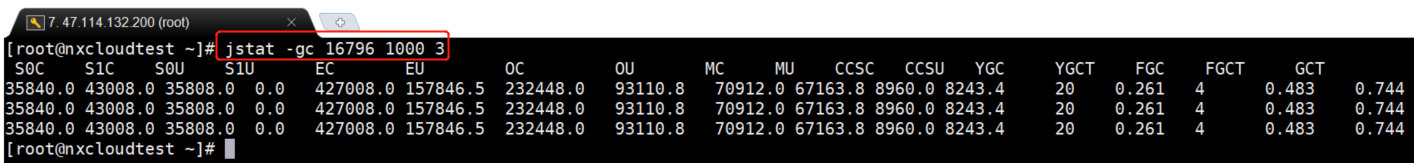
Bytes: 已加载的字节数.

Unloaded: 卸载的类数.

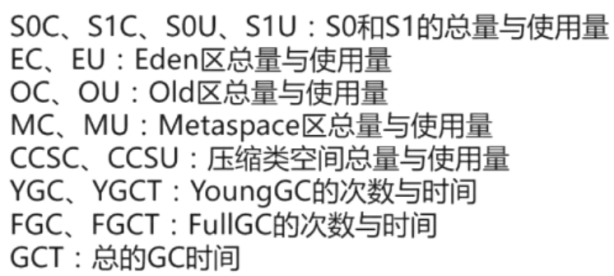
Bytes: 卸载的字节数.

Time: 执行类加载和卸载操作所花费的时间.

* 1. 垃圾收集信息(-gc、-gcutil、-gccause、-gcnew、-gcold.....)



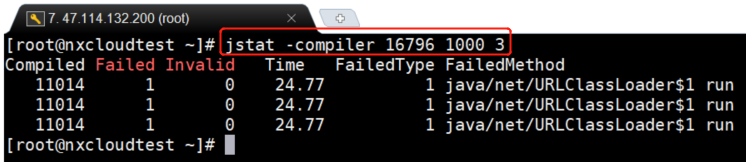
[返回信息说明](https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstat.html" \l "BEHHGFAE) - 搜索“-gc option”



标注：至于S0、S1有什么含义，后面讲

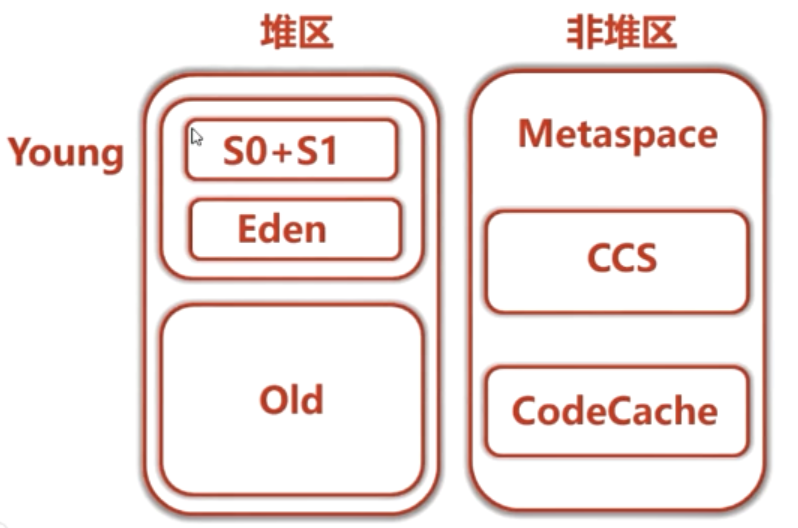
简单说明：S0 + S1 + E =young区

* 1. JIT编译信息(-compiler、-printcompilation)



[返回信息说明](https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstat.html" \l "BEHHGFAE) - 搜索“-compiler option”

* 1. JVM的内存结构

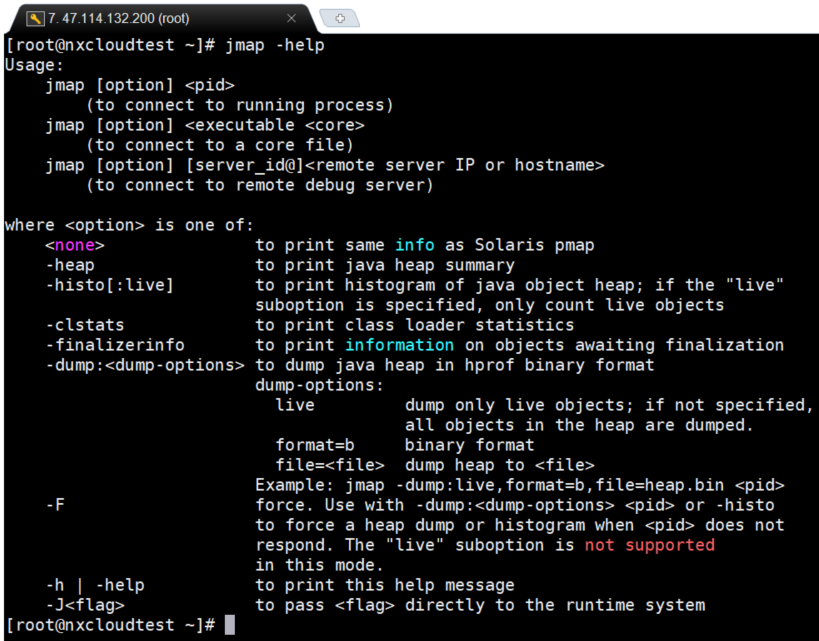


说明：S0 与 S1 大小一致，同个时间点上只有一个是启用的；CodeCache保存JIT的代码信息

1. jmap + MAT内存溢出(案例实战)
   1. 代码编写堆内存溢出和非堆内存溢出

[MemoryDemo 内存溢出案例](https://github.com/gitjiangxb/studentCode/blob/java-juc/src/main/java/jvm/MemoryDemo.java)

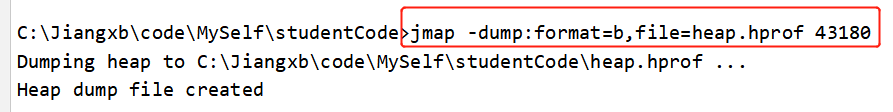
* 1. 如何导出内存映像文件
     1. 内存溢出自动导出(只需要配置参数)
        1. -XX:+HeapDumpOnOutOfMemoryError
        2. -XX:HeapDumpPath=./
     2. 使用jmap命令手动导出



option:-heap,-clstats,-dump:<dump-options>,-F

示例命令：

jmap -dump:format=b,file=heap.hprof 43180



参数说明：

format：格式

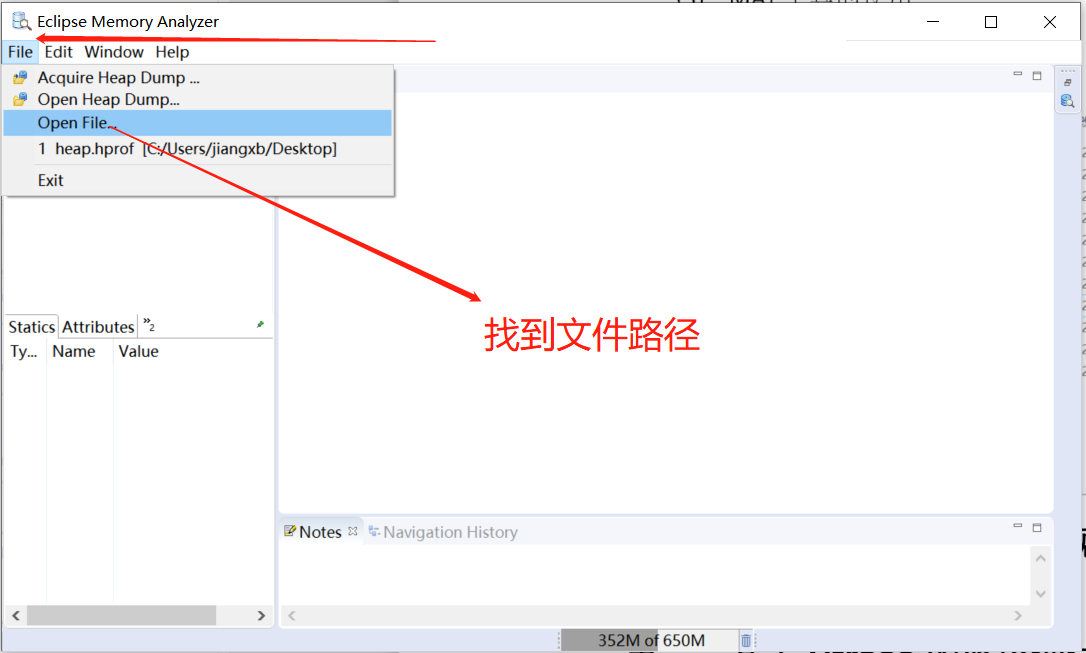
file：文件名称

43180：当前项目进程编号

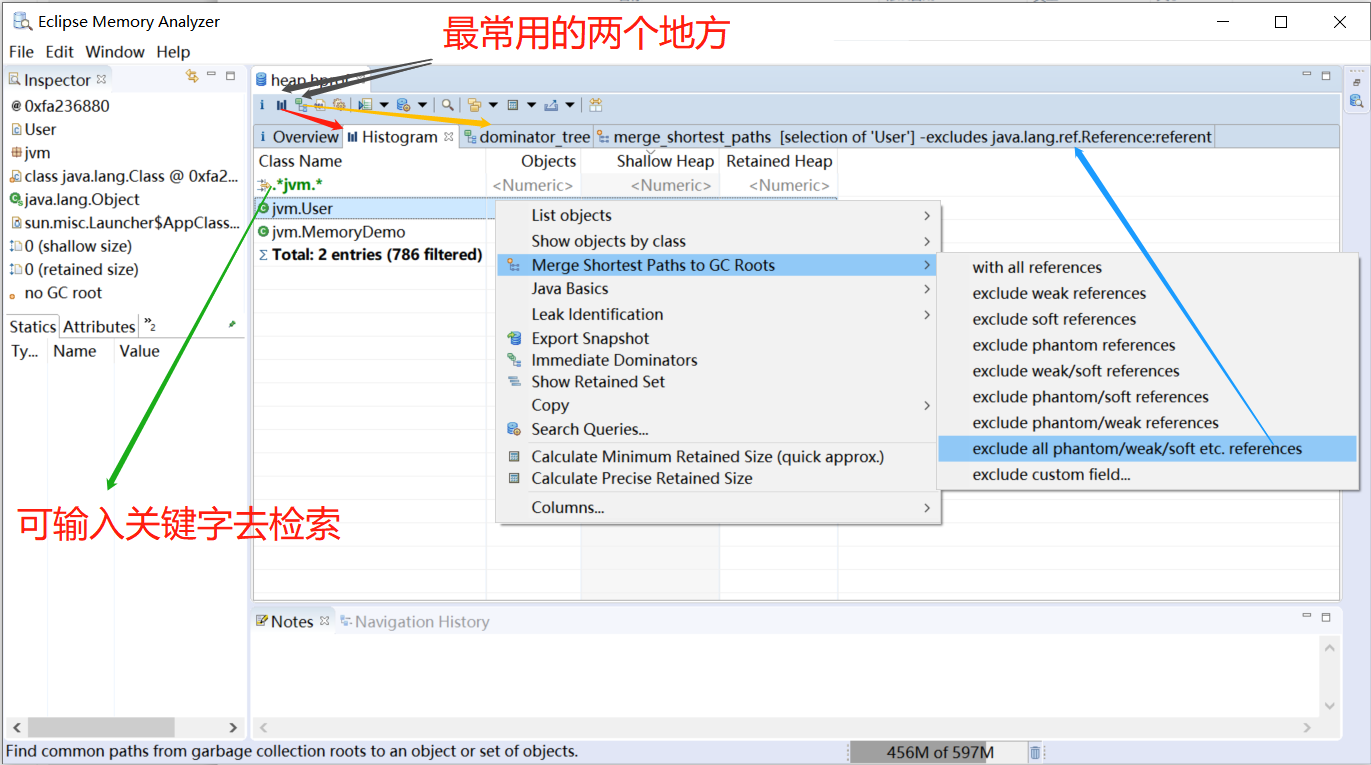
* 1. MAT工具的使用
     1. [下载地址](https://www.eclipse.org/downloads/download.php?file=/mat/1.10.0/rcp/MemoryAnalyzer-1.10.0.20200225-win32.win32.x86_64.zip)
     2. 解压完双击运行



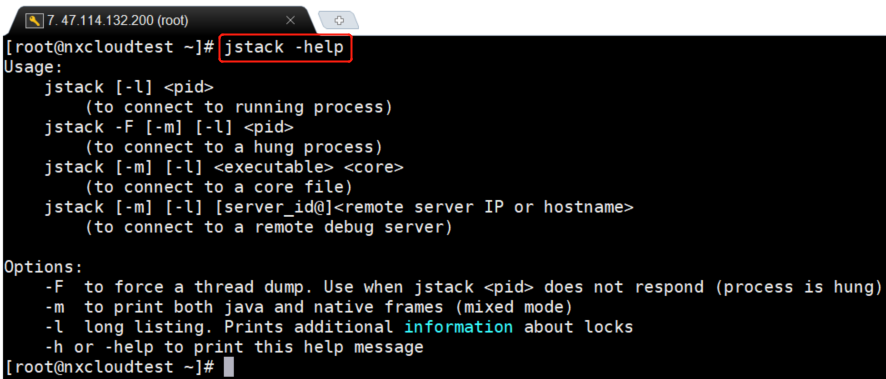
* + 1. 导入内存映像文件



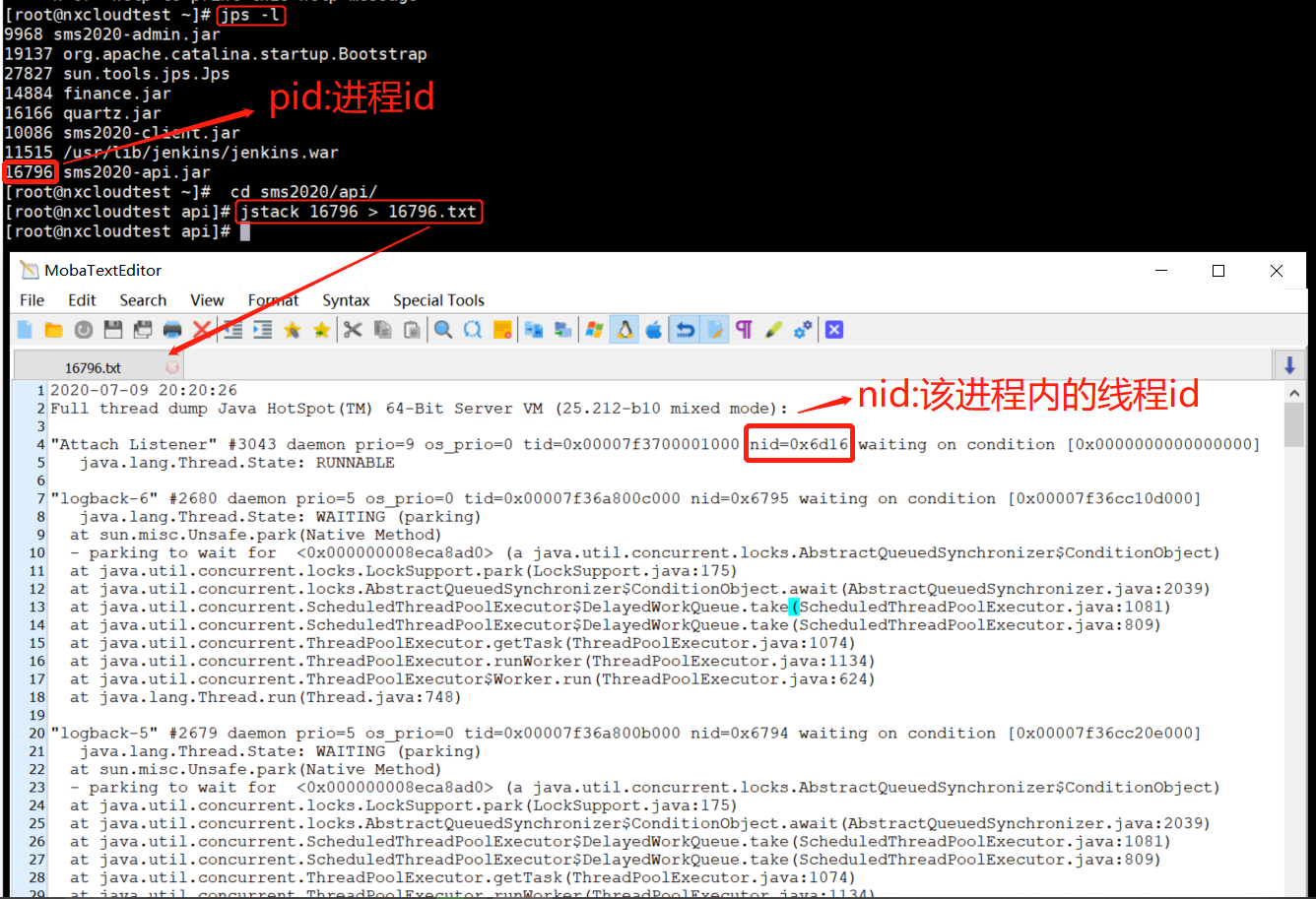
* + 1. 常用分析按钮



1. jstack 死循环与死锁(案例实战)
   1. 命令格式

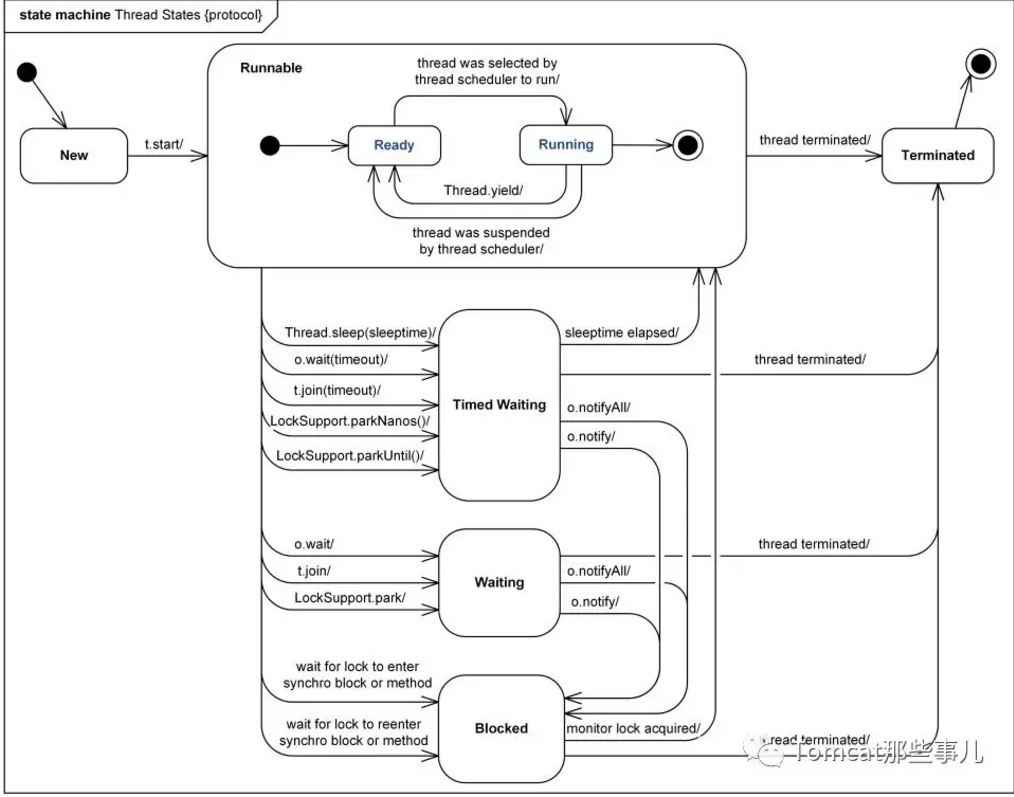


* 1. 案例分析



说明：注意跟java面试题(JUC+JVM+GC+Github).docx 的第二大点的第11点的区别

* 1. JAVA线程状态 [官网解释](https://docs.oracle.com/javase/8/docs/technotes/guides/troubleshoot/tooldescr034.html)
     1. NEW
     2. RUNNABLE
     3. BLOCKED
     4. WAITING
     5. TIMED\_WAITING
     6. TERMINATED
     7. [示例图地址](https://mp.weixin.qq.com/s/GsxeFM7QWuR--Kbpb7At2w)



#### 基于JVisualVM的可视化监控

#### 基于Btrace的监控调试

#### Tomcat性能监控与调优

#### Nginx性能监控与调优

#### JVM层GC调优

#### JVM字节码与Java代码层调优