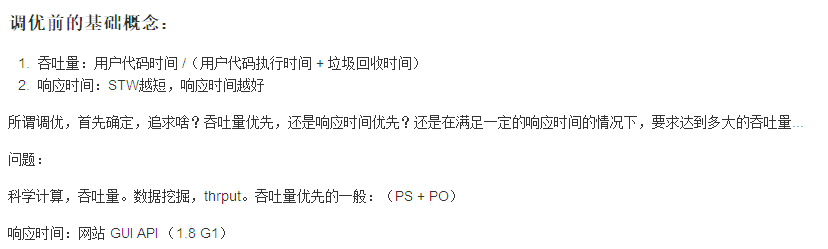
1. 调优的追求



1. 解决问题常用工具

命令行工具：jstack、jmap、jhat、jinfo、jstat

可视化工具：jvisualvm、jconsole

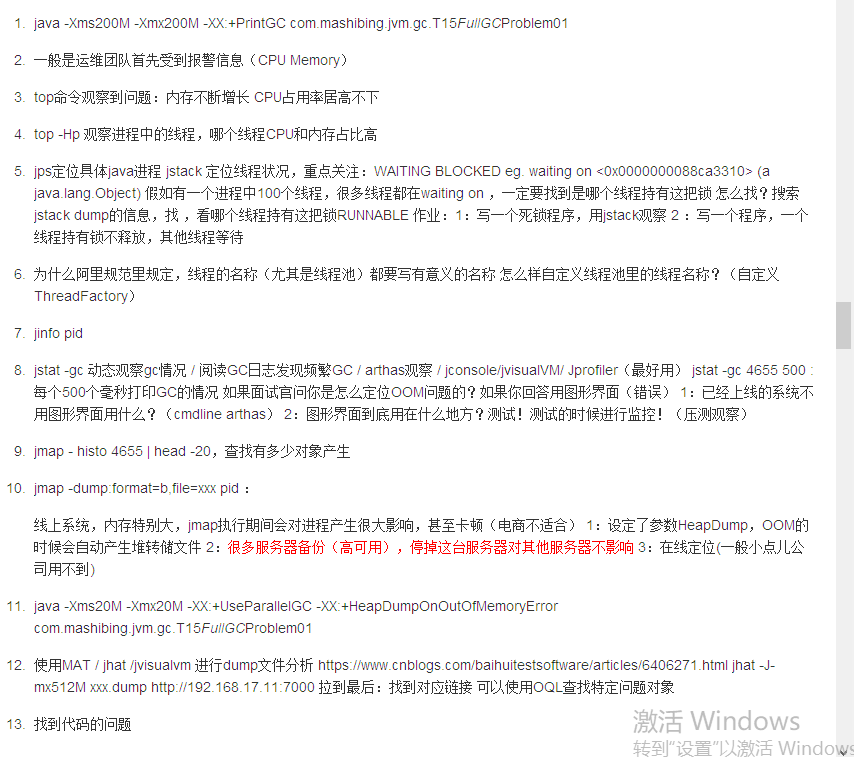
命令行在线排查：Arthas（可以进行热部署、反编译）

查看GC日志（频繁GC、FGC）

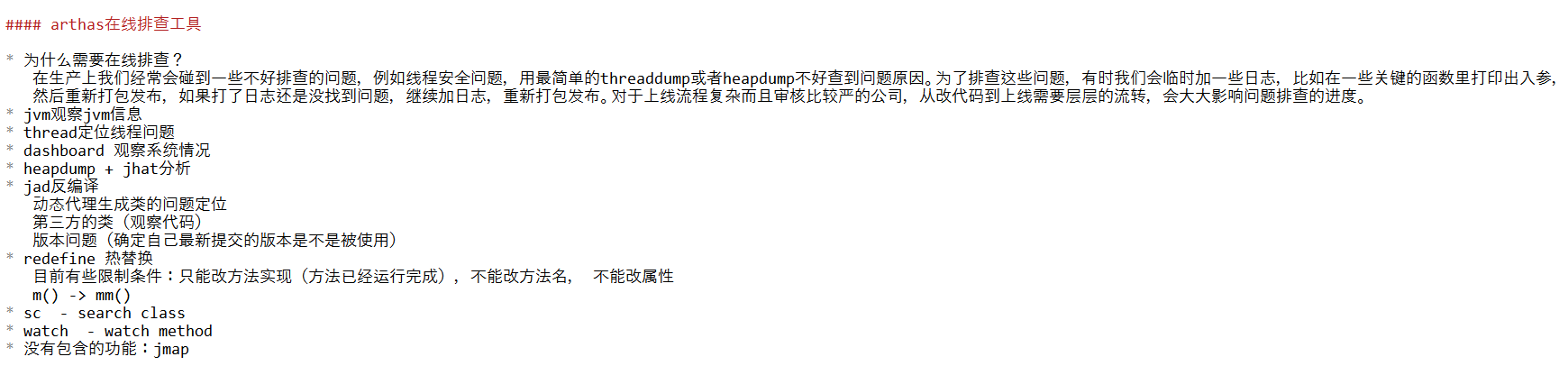
CPU问题：jstack，线程堆栈

内存问题：jmap查看堆中对象，用jhat、jvisualvm分析

频繁GC：设置参数查看GC日志（内存过小、代码不合理、显式调用System.gc）







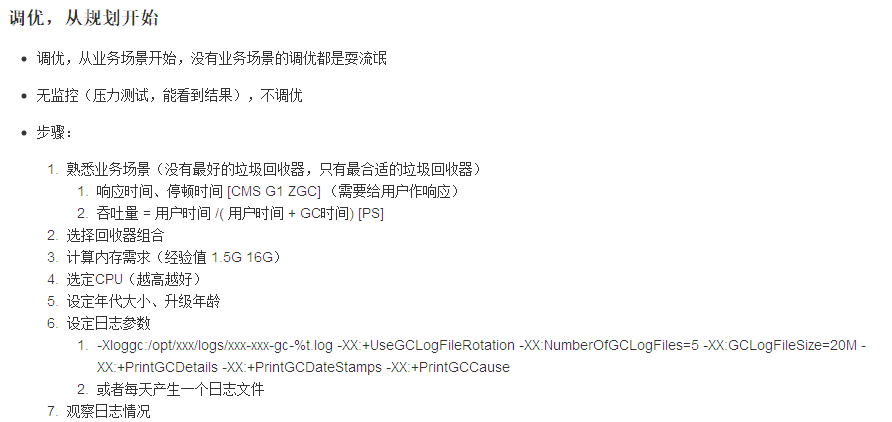
1. 调优场景



1. 调优->预调优
2. 步骤

选择垃圾回收器

计算内存



1. 具体场景



1. 优化环境（慢、卡顿）
2. CPU占用高：

top查看进程ID及占用CPU

top -Hp查看线程ID及占用CPU

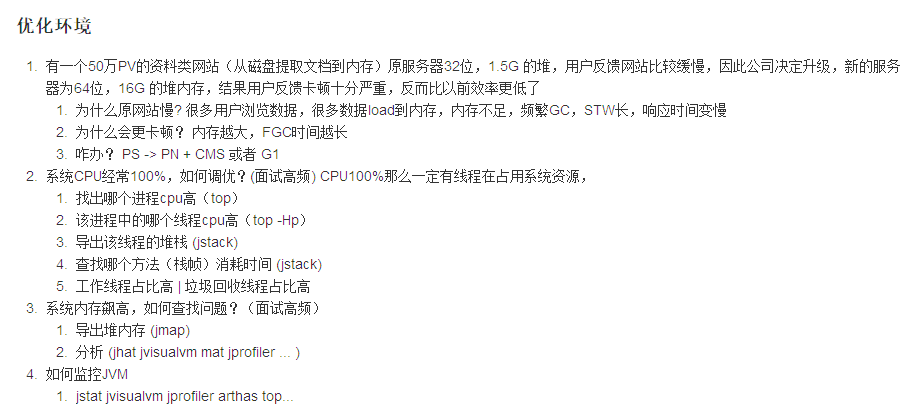
jstack查看进程的所有线程栈帧（再定位到占用CPU高的线程）

1. 内存占用高

先进行隔离，再heap dump分析

Or

jmap -histo



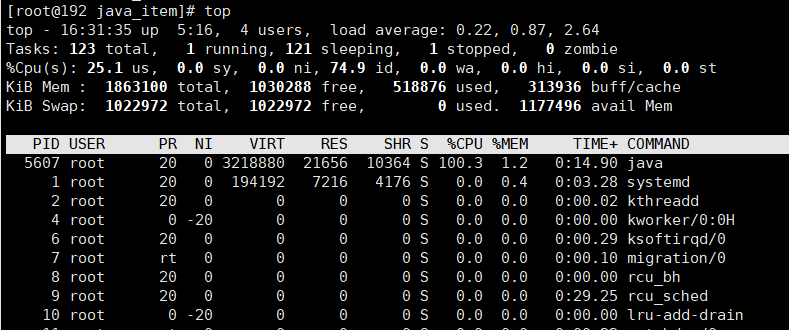
1. 解决JVM运行期间的问题------->死锁，jstack

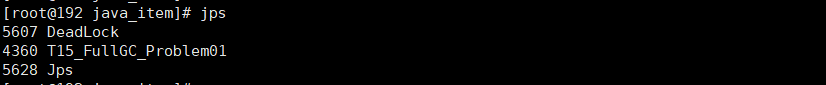
（1）代码



（2）定位

* jps或者top查看进程ID

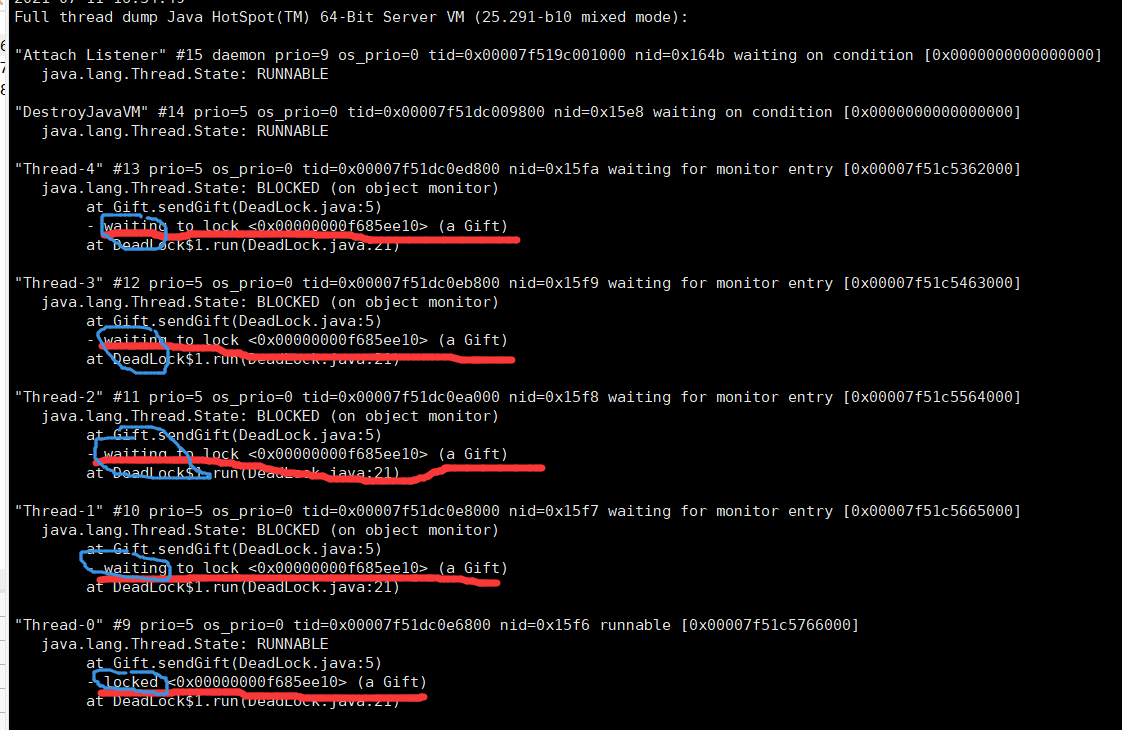




* jstack查看进程所有栈帧

可以看到线程0在lock，其他4个线程在wating to lock

就可以定位到线程0的具体死锁的栈帧



1. 解决JVM运行期间的问题------->OOM，jmap

jmap -histo，影响比heap dump小

jmap -dump执行期间确实会对进程产生很大的影响甚至卡顿，所以做了很多服务器备份，把那台机器先隔离开，然后再去做jmap分析。

1. 生成heap dump文件
2. jhat、jvisualvm分析dump文件



1. 案例总结

