**JVM性能调优监控工具总结**

<http://blog.csdn.net/wisgood/article/details/25343845>

<http://blog.csdn.net/lychbeyond/article/details/50999025>

**一、概述**

JDK本身提供了很多方便的**JVM性能调优监控工具**，除了集成式的VisualVM和jConsole外，还有jps、jstack、jmap、jhat、jstat、hprof等小巧的工具，每一种工具都有其自身的特点，用户可以根据需要检测的应用或者程序片段的状况，适当的选择相应的工具进行检测。

**问题：**

* OutOfMemoryError，内存不足
* 内存泄露
* 线程死锁
* 锁争用（Lock Contention）
* Java进程消耗CPU过高
* …..

**二、工具介绍**

**1、jps：输出JVM中运行的进程状态信息**

jps主要用来输出JVM中运行的进程状态信息。下面要介绍的小工具更多的都是先要使用jps查看出当前有哪些Java进程，获取该Java进程的id后再对该进程进行处理

语法格式：jps [options] [hostid]

如果不指定hostid就默认为当前主机或服务器。命令行参数选项说明如下：

* -q 不输出类名、Jar名和传入main方法的参数
* -m 输出传入main方法的参数
* -l 输出main类或Jar的全限名
* -v 输出传入JVM的参数

**2、jstack：查看某个Java进程内的线程堆栈信息**

jstack主要用来**查看某个Java进程内的线程堆栈信息**。jstack可以定位到线程堆栈，根据堆栈信息我们可以定位到具体代码，所以它在JVM性能调优中使用得非常多。

语法格式如下：

* jstack [option] pid
* jstack [option] executable core
* jstack [option] [server-id@]remote-hostname-or-ip

命令行参数选项说明如下：

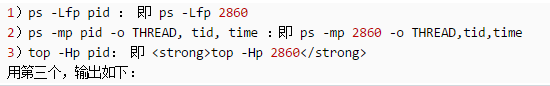
* -l long listings，会**打印出额外的锁信息**，在发生死锁时可以用 **stack -l pid** 来观察锁持有情况
* -m mixed mode，不仅会输出Java堆栈信息，还会输出C/C++堆栈信息（比如Native方法）

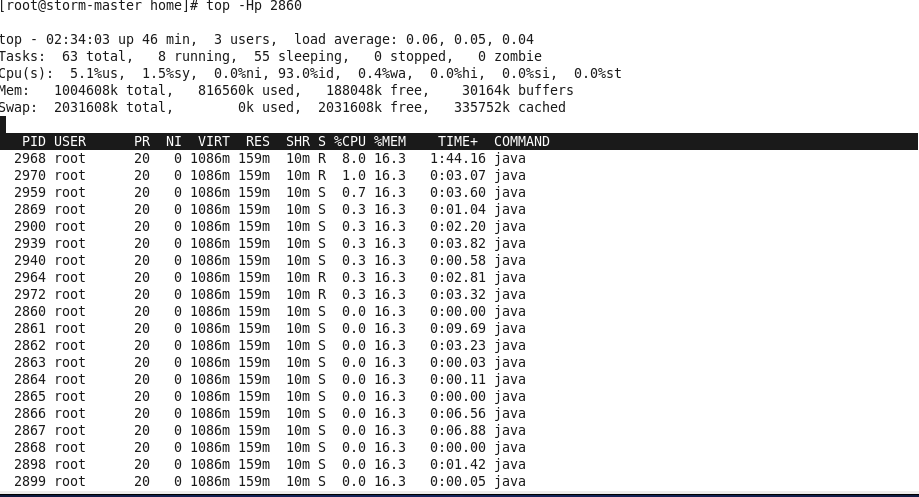
**实例：**找出某个Java进程中最耗费CPU的Java线程并定位堆栈信息，用到的命令有ps、top、printf、jstack、grep

* 第一步： 先找出Java进程ID，服务器上的Java应用名称为wordcount.jar：得到进程ID为2860,



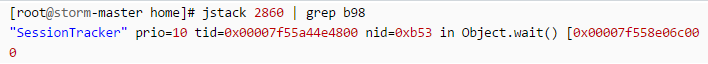
* 第二步：找出该进程内**最耗费CPU的线程**，可以使用如下3个命令，这里我们使用第3个命令得出如下结果：





TIME列就是各个Java线程耗费的CPU时间，显然CPU时间最长的是ID为2968的线程，用printf "%x\n" 2968，得到2968的十六进制值为b98。

* 第三步：终于轮到jstack上场了，它用来输出进程2860的堆栈信息，然后根据线程ID的十六进制值grep，如下：



可以看到CPU消耗在SessionTracker这个类的Object.wait()，于是就能很**容易的定位到相关的代码了**。

**3、jmap和jhat**

jmap（Memory Map）和 jhat（Java Heap Analysis Tool）：jmap导出堆内存，然后使用jhat来进行分析。jmap用来**查看堆内存使用状况**，一般结合jhat使用。

jmap语法格式如下：

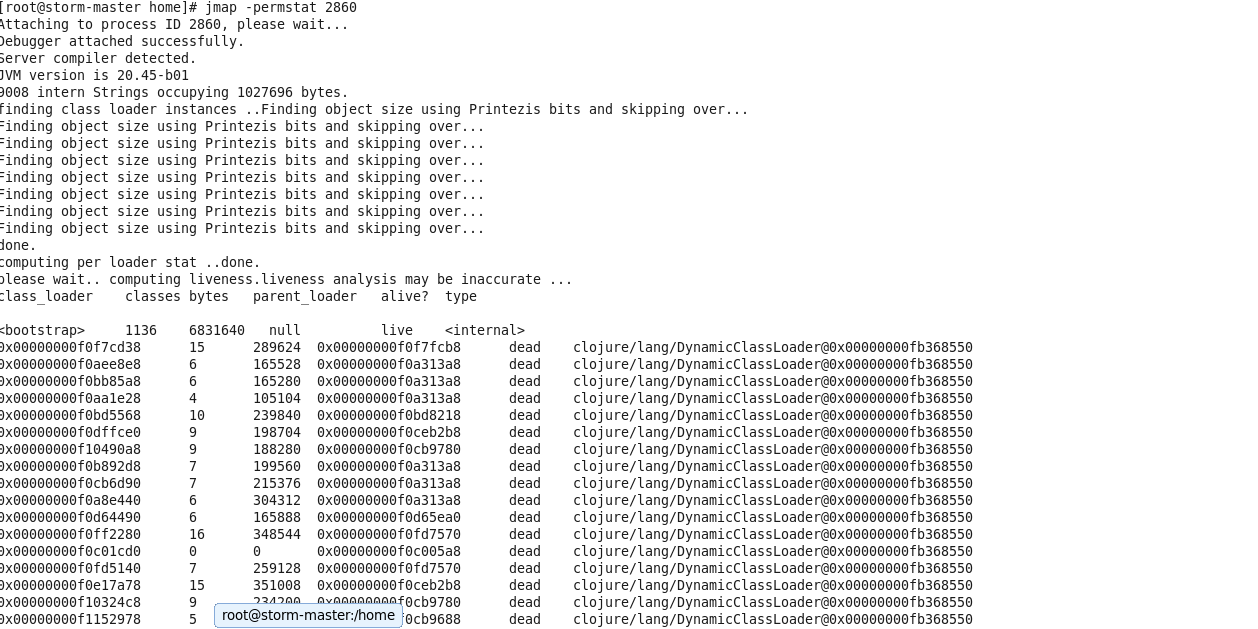
* jmap [option] pid
* jmap [option] executable core
* jmap [option] [server-id@]remote-hostname-or-ip

如果运行在64位JVM上，由于linux操作系统的不同，可能需要指定-J-d64命令选项参数。

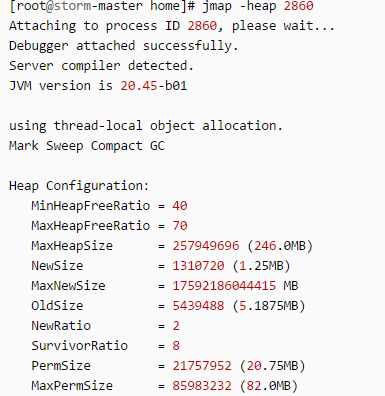
**作用示例：**

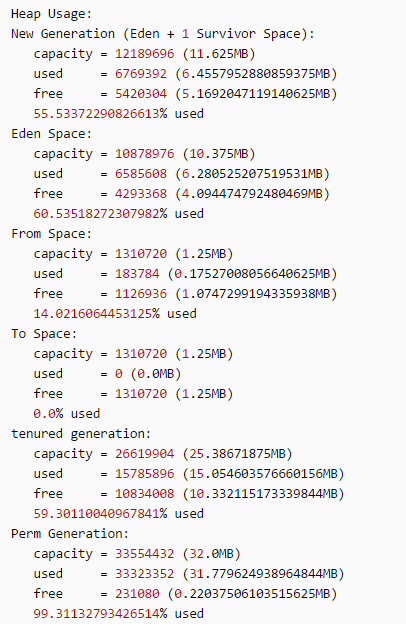
* 打印进程的类加载器和类加载器加载的持久代对象信息：**jmap -permstat pid**

输出：类加载器名称、对象是否存活（不可靠）、对象地址、父类加载器、已加载的类大小等信息，如图：



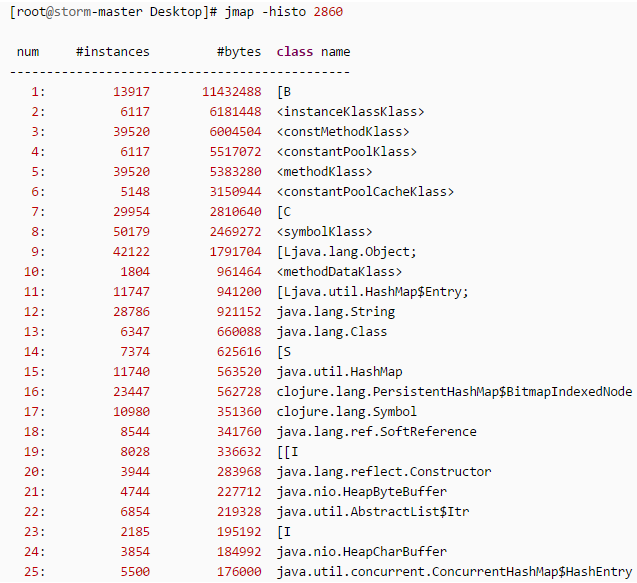
* 查看进程堆内存使用情况：包括使用的GC算法、堆配置参数和各代中堆内存使用：**jmap -heap pid**



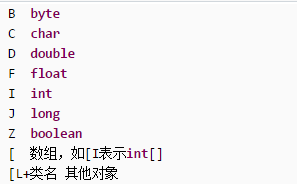


* 查看**堆内存中的对象数目、大小统计直方图**，如果带上live则只统计活对象：

**jmap -histo[:live] pid**



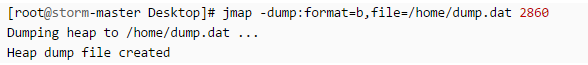
class name是对象类型，说明如下:



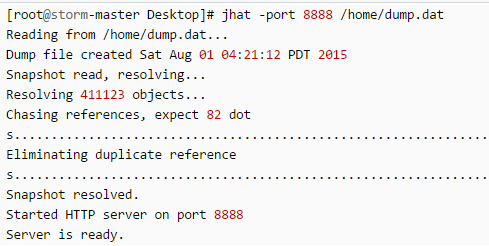
* 还有一个很常用的情况是：用jmap把进程内存使用情况dump到文件中，再用jhat分析查看。需要注意的是 dump出来的文件还可以用MAT、VisualVM等工具查看。jmap进行dump命令格式如下：

jmap -dump:format=b,file=dumpFileName pid

对上面进程ID为2860进行Dump：



然后使用jhat来对上面dump出来的内容进行分析



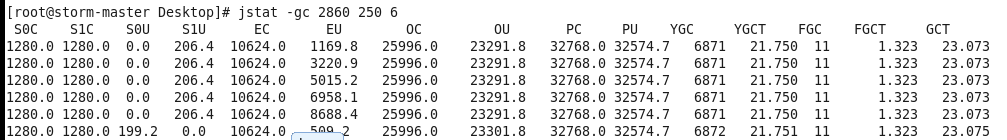
**4、jstat（JVM统计监测工具）: 看看各个区内存和GC的情况**

语法格式如下：

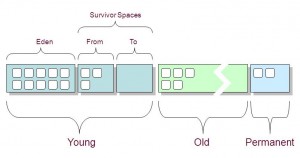
jstat [ generalOption | outputOptions vmid [interval[s|ms] [count]] ]

vmid是Java虚拟机ID，在Linux/Unix系统上一般就是进程ID。interval是采样时间间隔。count是采样数目。比如下面输出的是GC信息，采样时间间隔为250ms，采样数为6：

[root@storm-master Desktop]# jstat -gc 2860 250 6



**JVM堆内存布局：**



可以看出：

**堆内存 = 年轻代 + 年老代 + 永久代**

**年轻代 = Eden区 + 两个Survivor区（From和To）**

现在来解释各列含义：

