## jinfo

### 在运行时开启GC日志 - ImportNew

http://www.importnew.com/15722.html

我们经常会遇到JVM运行时出错的情况。若能在启动时加入一些启动选项（startup option），便可以获取与bug相关的重要线索，从而有希望根治它们。但在实际操作时，我们总是忘记添加-XX:+HeapDumpOnOutOfMemoryError 或 -XX:+PrintGCDetails这样必要的flag。

每当面对如此窘境，我们只能关闭JVM，修改启动参数（startup parameter），然后默默祈祷，希望问题场景（problematic situation）能在重启之后得以重现。这种方法偶尔奏效，在场景重现后你或许还能收集到足够的证据，以便动手根治潜在的问题。

不难看出，前文所述的方法问题显著——你必须执行一次额外的重启才能加入那烦人的debug选项，而不能借助中断（outage）实现。事实上，JDK bundle提供了一种可行的变通方案，如果将之收入麾下，偶尔还能从中获益。

jinfo

在JDK bundle中隐藏着一个精悍的小工具——[jinfo](http://www.oracle.com/technetwork/java/javase/tooldescr-136044.html#gbdio)。作为一个命令行工具，jinfo用于收集正在运行的Java进程的配置信息。jinfo吸引眼球的地方在于，它能通过-flag选项动态修改指定的Java进程中的某些JVM flag的值。虽然这样的flag数量有限，但它们偶尔能够帮助到你。通过以下的命令你便能看到JVM中哪些flag可以被jinfo动态修改：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | my-precious me$ java -XX:+PrintFlagsFinal -version|grep manageable       intx CMSAbortablePrecleanWaitMillis            = 100                                 {manageable}       intx CMSWaitDuration                           = 2000                                {manageable}       bool HeapDumpAfterFullGC                       = false                               {manageable}       bool HeapDumpBeforeFullGC                      = false                               {manageable}       bool HeapDumpOnOutOfMemoryError                = false                               {manageable}       ... cut for brevity ...       bool PrintGC                                   = false                               {manageable}       bool PrintGCDateStamps                         = false                               {manageable}       bool PrintGCDetails                            = false                               {manageable}       bool PrintGCTimeStamps                         = false                               {manageable} |

通过选项-XX:+PrintFlagsFinal可以列出所有的JVM flag，而其中的标注为manageable 的flag则是值得我们关注的部分。这些flag可通过JDK management interface（-XX:+PrintFlagsFinal）动态修改。虽然在JConsole中也能查到与其十分相似的MBean。但在我看来，以命令行的方式查看它们更加的便捷。

如何使用jinfo

让我们通过实战来学习如何使用jinfo。在下面的例子中，我们将在一个正在运行的JVM中动态开启GC日志功能：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | my-precious me$ jps  12278 HighAllocationRate  12279 Jps  12269 JConsole  my-precious me$ jinfo -flag +PrintGCDetails 12278  my-precious me$ jinfo -flag +PrintGC 12278  my-precious me$ |

在jinfo中需要打开-XX:+PrintGC和 -XX:+PrintGCDetails两个选项才能开启GC日志，这与用命令行参数的方式实现有着细微的差别——如果你通过启动脚本（startup script）来设置参数，仅需-XX:+PrintGCDetails即可，因为-XX:+PrintGC会被自动打开。

不过，从standard output的结果来看，PID为12278的进程的GC日志的确能够滚动地显示出来：

|  |  |
| --- | --- |
| 1  2  3  4  5 | ...  [GC (Allocation Failure) [PSYoungGen: 876416K->102624K(909312K)] 1094420K->320820K(1161216K), 0.2173131 secs] [Times: user=0.74 sys=0.00, real=0.22 secs]  ...  [GC (Allocation Failure) [PSYoungGen: 890304K->102240K(917504K)] 1108924K->320956K(1169408K), 0.2446639 secs] [Times: user=0.82 sys=0.01, real=0.25 secs]  ... |

同理，若想关闭GC日志功能，只需要执行jinfo -flag -PrintGCDetails 12278和 jinfo -flag -PrintGC 12278命令即可。

在我看来，学会使用jinfo，只能让你在故障排除的工作中少花一到两天的时间。若打一开始就使用[Plumbr](https://plumbr.eu/)这类性能监控工具，你可能从未有过这样的烦恼。

原文链接： [JavaCodeGeeks](http://www.javacodegeeks.com/2015/03/turning-on-gc-logging-at-runtime.html) 翻译： [ImportNew.com](http://www.importnew.com/)- [elviskang](http://www.importnew.com/author/elviskang)  
译文链接： <http://www.importnew.com/15722.html>  
[ **转载请保留原文出处、译者和译文链接。**]