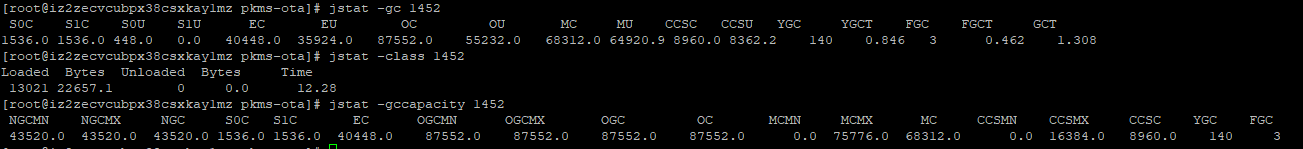
# Jstat -gc pid 指令测试。



# jstat命令详解

2013-02-28 19:49:23 [zhaozheng7758](https://me.csdn.net/zhaozheng7758" \t "https://blog.csdn.net/zhaozheng7758/article/details/_blank) 阅读数 72317更多

分类专栏： [java编程](https://blog.csdn.net/zhaozheng7758/article/category/447970" \t "https://blog.csdn.net/zhaozheng7758/article/details/_blank)

版权声明：本文为博主原创文章，遵循[CC 4.0 BY-SA](http://creativecommons.org/licenses/by-sa/4.0/" \t "https://blog.csdn.net/zhaozheng7758/article/details/_blank)版权协议，转载请附上原文出处链接和本声明。

本文链接：<https://blog.csdn.net/zhaozheng7758/article/details/8623549>

Jstat是JDK自带的一个轻量级小工具。全称“Java Virtual Machine statistics monitoring tool”，它位于java的bin目录下，主要利用JVM内建的指令对Java应用程序的资源和性能进行实时的命令行的监控，包括了对Heap size和垃圾回收状况的监控。可见，Jstat是轻量级的、专门针对JVM的工具，非常适用。

jstat工具特别强大，有众多的可选项，详细查看堆内各个部分的使用量，以及加载类的数量。使用时，需加上查看进程的进程id，和所选参数。参考格式如下：

jstat -options

可以列出当前JVM版本支持的选项，常见的有

* l  class (类加载器)
* l  compiler (JIT)
* l  gc (GC堆状态)
* l  gccapacity (各区大小)
* l  gccause (最近一次GC统计和原因)
* l  gcnew (新区统计)
* l  gcnewcapacity (新区大小)
* l  gcold (老区统计)
* l  gcoldcapacity (老区大小)
* l  gcpermcapacity (永久区大小)
* l  gcutil (GC统计汇总)
* l  printcompilation (HotSpot编译统计)

1、jstat –class<pid> : 显示加载class的数量，及所占空间等信息。

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| Loaded | 装载的类的数量 |
| Bytes | 装载类所占用的字节数 |
| Unloaded | 卸载类的数量 |
| Bytes | 卸载类的字节数 |
| Time | 装载和卸载类所花费的时间 |

2、jstat -compiler <pid>显示VM实时编译的数量等信息。

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| Compiled | 编译任务执行数量 |
| Failed | 编译任务执行失败数量 |
| Invalid | 编译任务执行失效数量 |
| Time | 编译任务消耗时间 |
| FailedType | 最后一个编译失败任务的类型 |
| FailedMethod | 最后一个编译失败任务所在的类及方法 |

3、jstat -gc <pid>: 可以显示gc的信息，查看gc的次数，及时间。

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| S0C | 年轻代中第一个survivor（幸存区）的容量 (字节) |
| S1C | 年轻代中第二个survivor（幸存区）的容量 (字节) |
| S0U | 年轻代中第一个survivor（幸存区）目前已使用空间 (字节) |
| S1U | 年轻代中第二个survivor（幸存区）目前已使用空间 (字节) |
| EC | 年轻代中Eden（伊甸园）的容量 (字节) |
| EU | 年轻代中Eden（伊甸园）目前已使用空间 (字节) |
| OC | Old代的容量 (字节) |
| OU | Old代目前已使用空间 (字节) |
| PC | Perm(持久代)的容量 (字节) |
| PU | Perm(持久代)目前已使用空间 (字节) |
| YGC | 从应用程序启动到采样时年轻代中gc次数 |
| YGCT | 从应用程序启动到采样时年轻代中gc所用时间(s) |
| FGC | 从应用程序启动到采样时old代(全gc)gc次数 |
| FGCT | 从应用程序启动到采样时old代(全gc)gc所用时间(s) |
| GCT | 从应用程序启动到采样时gc用的总时间(s) |

4、****jstat -gccapacity <pid>:****可以显示，VM内存中三代（young,old,perm）对象的使用和占用大小

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| NGCMN | 年轻代(young)中初始化(最小)的大小(字节) |
| NGCMX | 年轻代(young)的最大容量 (字节) |
| NGC | 年轻代(young)中当前的容量 (字节) |
| S0C | 年轻代中第一个survivor（幸存区）的容量 (字节) |
| S1C | 年轻代中第二个survivor（幸存区）的容量 (字节) |
| EC | 年轻代中Eden（伊甸园）的容量 (字节) |
| OGCMN | old代中初始化(最小)的大小 (字节) |
| OGCMX | old代的最大容量(字节) |
| OGC | old代当前新生成的容量 (字节) |
| OC | Old代的容量 (字节) |
| PGCMN | perm代中初始化(最小)的大小 (字节) |
| PGCMX | perm代的最大容量 (字节) |
| PGC | perm代当前新生成的容量 (字节) |
| PC | Perm(持久代)的容量 (字节) |
| YGC | 从应用程序启动到采样时年轻代中gc次数 |
| FGC | 从应用程序启动到采样时old代(全gc)gc次数 |

5、jstat -gcutil <pid>:统计gc信息

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| S0 | 年轻代中第一个survivor（幸存区）已使用的占当前容量百分比 |
| S1 | 年轻代中第二个survivor（幸存区）已使用的占当前容量百分比 |
| E | 年轻代中Eden（伊甸园）已使用的占当前容量百分比 |
| O | old代已使用的占当前容量百分比 |
| P | perm代已使用的占当前容量百分比 |
| YGC | 从应用程序启动到采样时年轻代中gc次数 |
| YGCT | 从应用程序启动到采样时年轻代中gc所用时间(s) |
| FGC | 从应用程序启动到采样时old代(全gc)gc次数 |
| FGCT | 从应用程序启动到采样时old代(全gc)gc所用时间(s) |
| GCT | 从应用程序启动到采样时gc用的总时间(s) |

6、jstat -gcnew <pid>:年轻代对象的信息。

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| S0C | 年轻代中第一个survivor（幸存区）的容量 (字节) |
| S1C | 年轻代中第二个survivor（幸存区）的容量 (字节) |
| S0U | 年轻代中第一个survivor（幸存区）目前已使用空间 (字节) |
| S1U | 年轻代中第二个survivor（幸存区）目前已使用空间 (字节) |
| TT | 持有次数限制 |
| MTT | 最大持有次数限制 |
| EC | 年轻代中Eden（伊甸园）的容量 (字节) |
| EU | 年轻代中Eden（伊甸园）目前已使用空间 (字节) |
| YGC | 从应用程序启动到采样时年轻代中gc次数 |
| YGCT | 从应用程序启动到采样时年轻代中gc所用时间(s) |

7、jstat -gcnewcapacity<pid>: 年轻代对象的信息及其占用量。

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| NGCMN | 年轻代(young)中初始化(最小)的大小(字节) |
| NGCMX | 年轻代(young)的最大容量 (字节) |
| NGC | 年轻代(young)中当前的容量 (字节) |
| S0CMX | 年轻代中第一个survivor（幸存区）的最大容量 (字节) |
| S0C | 年轻代中第一个survivor（幸存区）的容量 (字节) |
| S1CMX | 年轻代中第二个survivor（幸存区）的最大容量 (字节) |
| S1C | 年轻代中第二个survivor（幸存区）的容量 (字节) |
| ECMX | 年轻代中Eden（伊甸园）的最大容量 (字节) |
| EC | 年轻代中Eden（伊甸园）的容量 (字节) |
| YGC | 从应用程序启动到采样时年轻代中gc次数 |
| FGC | 从应用程序启动到采样时old代(全gc)gc次数 |

8、jstat -gcold <pid>：old代对象的信息。

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| PC | Perm(持久代)的容量 (字节) |
| PU | Perm(持久代)目前已使用空间 (字节) |
| OC | Old代的容量 (字节) |
| OU | Old代目前已使用空间 (字节) |
| YGC | 从应用程序启动到采样时年轻代中gc次数 |
| FGC | 从应用程序启动到采样时old代(全gc)gc次数 |
| FGCT | 从应用程序启动到采样时old代(全gc)gc所用时间(s) |
| GCT | 从应用程序启动到采样时gc用的总时间(s) |

9、stat -gcoldcapacity <pid>: old代对象的信息及其占用量。

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| OGCMN | old代中初始化(最小)的大小 (字节) |
| OGCMX | old代的最大容量(字节) |
| OGC | old代当前新生成的容量 (字节) |
| OC | Old代的容量 (字节) |
| YGC | 从应用程序启动到采样时年轻代中gc次数 |
| FGC | 从应用程序启动到采样时old代(全gc)gc次数 |
| FGCT | 从应用程序启动到采样时old代(全gc)gc所用时间(s) |
| GCT | 从应用程序启动到采样时gc用的总时间(s) |

10、jstat -gcpermcapacity<pid>: perm对象的信息及其占用量。

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| PGCMN | perm代中初始化(最小)的大小 (字节) |
| PGCMX | perm代的最大容量 (字节) |
| PGC | perm代当前新生成的容量 (字节) |
| PC | Perm(持久代)的容量 (字节) |
| YGC | 从应用程序启动到采样时年轻代中gc次数 |
| FGC | 从应用程序启动到采样时old代(全gc)gc次数 |
| FGCT | 从应用程序启动到采样时old代(全gc)gc所用时间(s) |
| GCT | 从应用程序启动到采样时gc用的总时间(s) |

11、jstat -printcompilation <pid>：当前VM执行的信息。

|  |  |
| --- | --- |
| 显示列名 | 具体描述 |
| Compiled | 编译任务的数目 |
| Size | 方法生成的字节码的大小 |
| Type | 编译类型 |
| Method | 类名和方法名用来标识编译的方法。类名使用/做为一个命名空间分隔符。方法名是给定类中的方法。上述格式是由-XX:+PrintComplation选项进行设置的 |

参考文献：

[http://docs.oracle.com/javase/1.5.0/docs/tooldocs/share/jstat.html#class\_option](http://docs.oracle.com/javase/1.5.0/docs/tooldocs/share/jstat.html" \l "class_option)

<http://nassir.iteye.com/blog/1517484>