Java GC日誌查看 - mmaa's blog

Java GC類型

Java中的GC有哪幾種類型？

參數	描述
UseSerialGC	虛擬機運行在Client模式的默認值，打開此開關參數後， 使用Serial+Serial Old收集器組合進行垃圾收集。
UseParNewGC	打開此開關參數後，使用ParNew+Serial Old收集器組合進行垃圾收集。
UseConcMarkSweepGC	打開此開關參數後，使用ParNew+CMS+Serial Old收集器組合進行垃圾收集。Serial Old作為CMS收集器出現Concurrent Mode Failure的備用垃圾收集器。
UseParallelGC	虛擬機運行在Server模式的默認值，打開此開關參數後，使用Parallel Scavenge+Serial Old收集器組合進行垃圾收集。
UseParallelOldGC	打開此開關參數後，使用Parallel Scavenge+Parallel Old收集器組合進行垃圾收集。

在Java程序啟動完成後，通過jps觀察進程來查詢到當前運行的java進程，使用

Java代碼

1. jinfo –flag UseSerialGC 進程

的方式可以定位其使用的gc策略，因為這些參數都是boolean型的常量，如果使用該種gc策略會出現 + 號，否則 – 號。

使用-XX:+上述GC策略可以開啟對應的GC策略。

GC日誌查看

可以通過在java命令種加入參數來指定對應的gc類型，打印gc日誌信息並輸出至文件等策略。

GC的日誌是以替換的方式(>)寫入的，而不是追加(>>)，如果下次寫入到同一個文件中的話，以前的GC內容會被清空。

對應的參數列表

Java代碼

1. -XX:+PrintGC 輸出GC日誌
2. -XX:+PrintGCDetails 輸出GC的詳細日誌
3. -XX:+PrintGCTimeStamps 輸出GC的時間戳（以基準時間的形式）
4. -XX:+PrintGCDateStamps 輸出GC的時間戳（以日期的形式，如 2013-05-04T21:53:59.234+0800）
5. -XX:+PrintHeapAtGC 在進行GC的前後打印出堆的信息
6. -Xloggc:../logs/gc.log 日誌文件的輸出路徑

這裡使用如下的參數來進行日誌的打印：

Java代碼

1. -XX:+PrintGCDateStamps -XX:+PrintGCDetails -Xloggc:./gclogs

對於新生代回收的一行日誌，其基本內容如下：

Java代碼

1. 2014-07-18T16:02:17.606+0800: 611.633: [GC 611.633: [DefNew: 843458K->2K(948864K), 0.0059180 secs] 2186589K->1343132K(3057292K), 0.0059490 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]

其含義大概如下：

Java代碼

1. 2014-07-18T16:02:17.606+0800（當前時間戳）: 611.633（時間戳）: [GC（表示Young GC） 611.633: [DefNew（單線程Serial年輕代GC）: 843458K（年輕代垃圾回收前的大小）->2K（年輕代回收後的大小）(948864K（年輕代總大小）), 0.0059180 secs（本次回收的時間）] 2186589K（整個堆回收前的大小）->1343132K（整個堆回收後的大小）(3057292K（堆總大小）), 0.0059490 secs（回收時間）] [Times: user=0.00（用戶耗時） sys=0.00（系統耗時）, real=0.00 secs（實際耗時）]

老年代回收的日誌如下：

Java代碼

1. 2014-07-18T16:19:16.794+0800: 1630.821: [GC 1630.821: [DefNew: 1005567K->111679K(1005568K), 0.9152360 secs]1631.736: [Tenured:
2. 2573912K->1340650K(2574068K), 1.8511050 secs] 3122548K->1340650K(3579636K), [Perm : 17882K->17882K(21248K)], 2.7854350 secs] [Times: user=2.57 sys=0.22, real=2.79 secs]

gc日誌中的最後貌似是系統運行完成前的快照：

Java代碼

```
1. Heap
2.  def new generation   total 1005568K, used 111158K [0x00000006fae00000, 0x000000073f110000, 0x0000000750350000)
3.   eden space 893888K,  12% used [0x00000006fae00000, 0x0000000701710e90, 0x00000007316f0000)
4.   from space 111680K,   3% used [0x0000000738400000, 0x000000073877c9b0, 0x000000073f110000)
5.   to   space 111680K,   0% used [0x00000007316f0000, 0x00000007316f0000, 0x0000000738400000)
6.  tenured generation   total 2234420K, used 1347671K [0x0000000750350000, 0x00000007d895d000, 0x00000007fae00000)
7.    the space 2234420K,  60% used [0x0000000750350000, 0x00000007a2765cb8, 0x00000007a2765e00, 0x00000007d895d000)
8.  compacting perm gen  total 21248K, used 17994K [0x00000007fae00000, 0x00000007fc2c0000, 0x0000000800000000)
9.    the space 21248K,  84% used [0x00000007fae00000, 0x00000007fbf92a50, 0x00000007fbf92c00, 0x00000007fc2c0000)
10. No shared spaces configured.
```

GC日誌的離線分析

可以使用一些離線的工具來對GC日誌進行分析，比
如sun的gchisto(https://java.net/projects/gchisto)，gcviewer（https://github.com/chewiebug/GCViewer ），這些都是開源的工具，用戶可以直接通過版本控制工具下載其源碼，進行離線分析。

下面就已gcviewer為例，簡要分析一下gc日誌的離線分析，gcviewer源代碼工程是maven結構的，可以直接用maven進行package，這裡編譯的是1.34版本，本版本的快照已經上傳至附件中。

需要說明的是，gcviewer支持多種參數生成的gc日誌，直接通過java –jar的方式運行，加載生成的gc日誌即可：