unixboy

unixboy.iteye.com/blog/174173

1. **堆大小設置**
   JVM 中最大堆大小有三方面限制：相關操作系統的數據模型（32-bt還是64-bit）限制；系統的可用虛擬內存限制；系統的可用物理內存限制。32位系統下，一般限制在1.5G~2G；64為操作系統對內存無限制。我在Windows Server 2003 系統，3.5G物理內存，JDK5.0下測試，最大可設置為1478m。
   典型設置：

   - java **-Xmx3550m -Xms3550m -Xmn2g -Xss128k**
     -Xmx3550m：設置JVM最大可用內存為3550M。
     -Xms3550m：設置JVM促使內存為3550m。此值可以設置與-Xmx相同，以避免每次垃圾回收完成後JVM重新分配內存。
     -Xmn2g：設置年輕代大小為2G。**整個JVM內存大小=年輕代大小 + 年老代大小 + 持久代大小**。持久代一般固定大小為64m，所以增大年輕代後，將會減小年老代大小。此值對系統性能影響較大，Sun官方推薦配置為整個堆的3/8。
     -Xss128k：設置每個線程的堆棧大小。JDK5.0以後每個線程堆棧大小為1M，以前每個線程堆棧大小為256K。更具應用的線程所需內存大小進行調整。在相同物理內存下，減小這個值能生成更多的線程。但是操作系統對一個進程內的線程數還是有限制的，不能無限生成，經驗值在3000~5000左右。
   - java -Xmx3550m -Xms3550m -Xss128k **-XX:NewRatio=4 -XX:SurvivorRatio=4 -XX:MaxPermSize=16m -XX:MaxTenuringThreshold=0**
     -XX:NewRatio=4:設置年輕代（包括Eden和兩個Survivor區）與年老代的比值（除去持久代）。設置為4，則年輕代與年老代所佔比值為1：4，年輕代佔整個堆棧的1/5
     -XX:SurvivorRatio=4：設置年輕代中Eden區與Survivor區的大小比值。設置為4，則兩個Survivor區與一個Eden區的比值為2:4，一個Survivor區佔整個年輕代的1/6
     -XX:MaxPermSize=16m:設置持久代大小為16m。
     -XX:MaxTenuringThreshold=0：設置垃圾最大年齡。**如果設置為0的話，則年輕代對象不經過Survivor區，直接進入年老代**。對於年老代比較多的應用，可以提高效率。**如果將此值設置為一個較大值，則年輕代對象會在Survivor區進行多次複製，這樣可以增加對象再年輕代的存活時間**，增加在年輕代即被回收的概論。

2. **回收器選擇**
   JVM給了三種選擇：**串行收集器、並行收集器、並發收集器**，但是串行收集器只適用於小數據量的情況，所以這裡的選擇主要針對並行收集器和並發收集器。默認情況下，JDK5.0以前都是使用串行收集器，如果想使用其他收集器需要在啟動時加入相應參數。JDK5.0以後，JVM會根據當前系統配置進行判斷。

   1. **吞吐量優先**的並行收集器
      如上文所述，並行收集器主要以到達一定的吞吐量為目標，適用於科學技術和後台處理等。
      典型配置：

      - java -Xmx3800m -Xms3800m -Xmn2g -Xss128k **-XX:+UseParallelGC -XX:ParallelGCThreads=20**
        -XX:+UseParallelGC：選擇垃圾收集器為並行收集器。**此配置僅對年輕代有效。即上述配置下，年輕代使用並發收集，而年老代仍舊使用串行收集。**
        -XX:ParallelGCThreads=20：配置並行收集器的線程數，即：同時多少個線程一起進行垃圾回收。此值最好配置與處理器數目相等。
      - java -Xmx3550m -Xms3550m -Xmn2g -Xss128k -XX:+UseParallelGC -XX:ParallelGCThreads=20 **-XX:+UseParallelOldGC**
        -XX:+UseParallelOldGC：配置年老代垃圾收集方式為並行收集。JDK6.0支持對年老代並行收集。
      - java -Xmx3550m -Xms3550m -Xmn2g -Xss128k -XX:+UseParallelGC **-XX:MaxGCPauseMillis=100**
        -XX:MaxGCPauseMillis=100:設置每次年輕代垃圾回收的最長時間，如果無法滿足此時間，JVM會自動調整年輕代大小，以滿足此值。
      - java -Xmx3550m -Xms3550m -Xmn2g -Xss128k -XX:+UseParallelGC -XX:MaxGCPauseMillis=100 **-XX:+UseAdaptiveSizePolicy**
        -XX:+UseAdaptiveSizePolicy：設置此選項後，並行收集器會自動選擇年輕代區大小和相應的Survivor區比例，以達到目標系統規定的最低相應時間或者收集頻率等，此值建議使用並行收集器時，一直打開。

   2. **響應時間優先**的並發收集器
      如上文所述，並發收集器主要是保證系統的響應時間，減少垃圾收集時的停頓時間。適用於應用服務器、電信領域等。
      典型配置：

      - java -Xmx3550m -Xms3550m -Xmn2g -Xss128k -XX:ParallelGCThreads=20 **-XX:+UseConcMarkSweepGC -XX:+UseParNewGC**
        -XX:+UseConcMarkSweepGC：設置年老代為並發收集。測試中配置這個以後，-XX:NewRatio=4的配置失效了，原因不明。所以，此時年輕代大小最好用-Xmn設置。
        -XX:+UseParNewGC:設置年輕代為並行收集。可與CMS收集同時使用。JDK5.0以上，JVM會根據系統配置自行設置，所以無需再設置此值。
      - java -Xmx3550m -Xms3550m -Xmn2g -Xss128k -XX:+UseConcMarkSweepGC **-XX:CMSFullGCsBeforeCompaction=5 -XX:+UseCMSCompactAtFullCollection**
        -XX:CMSFullGCsBeforeCompaction：由於並發收集器不對內存空間進行壓縮、整理，所以運行一段時間以後會產生「碎片」，使得運行效率降低。此值設置運行多少次GC以後對內存空間進行壓縮、整理。
        -XX:+UseCMSCompactAtFullCollection：打開對年老代的壓縮。可能會影響性能，但是可以消除碎片

3. **輔助信息**

JVM提供了大量命令行參數，打印信息，供調試使用。主要有以下一些：

- **-XX:+PrintGC**
  輸出形式：**[GC 118250K->113543K(130112K), 0.0094143 secs]**

  [Full GC 121376K->10414K(130112K), 0.0650971 secs]

- **-XX:+PrintGCDetails**
  輸出形式：**[GC [DefNew: 8614K->781K(9088K), 0.0123035 secs] 118250K->113543K(130112K), 0.0124633 secs]**

  [GC [DefNew: 8614K->8614K(9088K), 0.0000665 secs][Tenured: 112761K->10414K(121024K), 0.0433488 secs] 121376K->10414K(130112K), 0.0436268 secs]

- **-XX:+PrintGCTimeStamps** -XX:+PrintGC：PrintGCTimeStamps可與上面兩個混合使用
  輸出形式：**11.851: [GC 98328K->93620K(130112K), 0.0082960 secs]**
- **-XX:+PrintGCApplicationConcurrentTime:**打印每次垃圾回收前，程序未中斷的執行時間。可與上面混合使用
  輸出形式：**Application time: 0.5291524 seconds**
- **-XX:+PrintGCApplicationStoppedTime**：打印垃圾回收期間程序暫停的時間。可與上面混合使用
  輸出形式：**Total time for which application threads were stopped: 0.0468229 seconds**
- **-XX:PrintHeapAtGC**:打印GC前後的詳細堆棧信息
  輸出形式：
  34.702: [GC {Heap before gc invocations=7:
  def new generation total 55296K, used 52568K [0x1ebd0000, 0x227d0000, 0x227d0000)
  eden space 49152K, 99% used [0x1ebd0000, 0x21bce430, 0x21bd0000)
  from space 6144K, 55% used [0x221d0000, 0x22527e10, 0x227d0000)
  to space 6144K, 0% used [0x21bd0000, 0x21bd0000, 0x221d0000)
  tenured generation total 69632K, used 2696K [0x227d0000, 0x26bd0000, 0x26bd0000)
  the space 69632K, 3% used [0x227d0000, 0x22a720f8, 0x22a72200, 0x26bd0000)
  compacting perm gen total 8192K, used 2898K [0x26bd0000, 0x273d0000, 0x2abd0000)
  the space 8192K, 35% used [0x26bd0000, 0x26ea4ba8, 0x26ea4c00, 0x273d0000)
  ro space 8192K, 66% used [0x2abd0000, 0x2b12bcc0, 0x2b12be00, 0x2b3d0000)
  rw space 12288K, 46% used [0x2b3d0000, 0x2b972060, 0x2b972200, 0x2bfd0000)
  34.735: [DefNew: 52568K->3433K(55296K), 0.0072126 secs] 55264K->6615K(124928K)**Heap after gc invocations=8:**
  def new generation total 55296K, used 3433K [0x1ebd0000, 0x227d0000, 0x227d0000)
  eden space 49152K, 0% used [0x1ebd0000, 0x1ebd0000, 0x21bd0000)
  from space 6144K, 55% used [0x21bd0000, 0x21f2a5e8, 0x221d0000)
  to space 6144K, 0% used [0x221d0000, 0x221d0000, 0x227d0000)
  tenured generation total 69632K, used 3182K [0x227d0000, 0x26bd0000, 0x26bd0000)
  the space 69632K, 4% used [0x227d0000, 0x22aeb958, 0x22aeba00, 0x26bd0000)
  compacting perm gen total 8192K, used 2898K [0x26bd0000, 0x273d0000, 0x2abd0000)
  the space 8192K, 35% used [0x26bd0000, 0x26ea4ba8, 0x26ea4c00, 0x273d0000)
  ro space 8192K, 66% used [0x2abd0000, 0x2b12bcc0, 0x2b12be00, 0x2b3d0000)
  rw space 12288K, 46% used [0x2b3d0000, 0x2b972060, 0x2b972200, 0x2bfd0000)
  }
  , 0.0757599 secs]
- **-Xloggc:filename**:與上面幾個配合使用，把相關日誌信息記錄到文件以便分析。

4. **常見配置彙總**

    1. 堆設置

        - **-Xms**:初始堆大小
        - **-Xmx**:最大堆大小
        - **-XX:NewSize=n**:設置年輕代大小
        - **-XX:NewRatio=n:**設置年輕代和年老代的比值。如:為3，表示年輕代與年老代比值為1：3，年輕代佔整個年輕代年老代和的1/4
        - **-XX:SurvivorRatio=n**:年輕代中Eden區與兩個Survivor區的比值。注意Survivor區有兩個。如：3，表示Eden：Survivor=3：2，一個Survivor區佔整個年輕代的1/5
        - **-XX:MaxPermSize=n**:設置持久代大小

    2. 收集器設置

        - **-XX:+UseSerialGC**:設置串行收集器
        - **-XX:+UseParallelGC**:設置並行收集器
        - **-XX:+UseParalledlOldGC**:設置並行年老代收集器
        - **-XX:+UseConcMarkSweepGC**:設置並發收集器

    3. 垃圾回收統計信息

        - **-XX:+PrintGC**
        - **-XX:+PrintGCDetails**
        - **-XX:+PrintGCTimeStamps**
        - **-Xloggc:filename**

    4. 並行收集器設置

        - **-XX:ParallelGCThreads=n**:設置並行收集器收集時使用的CPU數。並行收集線程數。
        - **-XX:MaxGCPauseMillis=n**:設置並行收集最大暫停時間
        - **-XX:GCTimeRatio=n**:設置垃圾回收時間佔程序運行時間的百分比。公式為1/(1+n)

    5. 並發收集器設置

        - **-XX:+CMSIncrementalMode**:設置為增量模式。適用於單CPU情況。
        - **-XX:ParallelGCThreads=n**:設置並發收集器年輕代收集方式為並行收集時，使用的CPU數。並行收集線程數。

四、調優總結

1. **年輕代大小選擇**

    - **響應時間優先的應用：儘可能設大，直到接近系統的最低響應時間限制**（根據實際情況選擇）。在此種情況下，年輕代收集發生的頻率也是最小的。同時，減少到達年老代的對象。
    - **吞吐量優先的應用**：儘可能的設置大，可能到達Gbit的程度。因為對響應時間沒有要求，垃圾收集可以並行進行，一般適合8CPU以上的應用。

2. **年老代大小選擇**

   - **響應時間優先的應用**：年老代使用並發收集器，所以其大小需要小心設置，一般要考慮**並發會話率**和**會話持續時間**等一些參數。如果堆設置小了，可以會造成內存碎片、高回收頻率以及應用暫停而使用傳統的標記清除方式；如果堆大了，則需要較長的收集時間。最優化的方案，一般需要參考以下數據獲得：

     - 並發垃圾收集信息
     - 持久代並發收集次數
     - 傳統GC信息
     - 花在年輕代和年老代回收上的時間比例

     減少年輕代和年老代花費的時間，一般會提高應用的效率

   - **吞吐量優先的應用**：一般吞吐量優先的應用都有一個很大的年輕代和一個較小的年老代。原因是，這樣可以儘可能回收掉大部分短期對象，減少中期的對象，而年老代盡存放長期存活對象。

3. **較小堆引起的碎片問題**
   因為年老代的並發收集器使用標記、清除算法，所以不會對堆進行壓縮。當收集器回收時，他會把相鄰的空間進行合併，這樣可以分配給較大的對象。但是，當堆空間較小時，運行一段時間以後，就會出現「碎片」，如果並發收集器找不到足夠的空間，那麼並發收集器將會停止，然後使用傳統的標記、清除方式進行回收。如果出現「碎片」，可能需要進行如下配置：

   - **-XX:+UseCMSCompactAtFullCollection**：使用並發收集器時，開啟對年老代的壓縮。
   - **-XX:CMSFullGCsBeforeCompaction=0**：上面配置開啟的情況下，這裡設置多少次Full GC後，對年老代進行壓縮