

简单，可复制

点点滴滴，尽在文中

:: 首页 :: 博文 :: 闪存 :: 新随笔 :: 联系 :: 订阅  :: 管理 ::  431 随笔 :: 0 文章 :: 538 评论 :: 0 引用

## 公告

## 史上最好的免费svn空间

昵称: [ggjucheng](#)

园龄: 6年8个月

粉丝: 1627

关注: 6

[+加关注](#)

## 博客地图

[c/c++笔记](#)

本人学习c/c++的一些笔记

[db笔记](#)

[mysql nosql](#)

[hadoop笔记](#)

本人工作中hadoop的心得

[internet笔记](#)

[互联网学习笔记](#)

[java笔记](#)

[java平台笔记](#)

[Linux/Unix笔记](#)

本人学习linux/unix的笔记

[TCP/IP笔记](#)

本人学习TCP/IP的心得和笔记

[web开发](#)

html css js php etc.....

[技术花絮](#)

## JVM GC算法 CMS 详解(转)

### 前言

CMS，全称Concurrent Low Pause Collector，是jdk1.4后期版本开始引入的新gc算法，在jdk5和jdk6中得到了进一步改进，它的主要适合场景是对响应时间的重要性需求 大于对吞吐量的要求，能够承受垃圾回收线程和应用线程共享处理器资源，并且应用中存在比较多的长生命周期的对象的应用。CMS是用于对tenured generation的回收，也就是年老代的回收，目标是尽量减少应用的暂停时间，减少full gc发生的几率，利用和应用程序线程并发的垃圾回收线程来标记清除年老代。在我们的应用中，因为有缓存的存在，并且对于响应时间也有比较高的要求，因此希望尝试使用CMS来替代默认的server型JVM使用的并行收集器，以便获得更短的垃圾回收的暂停时间，提高程序的响应性。

### CMS收集周期

CMS并非没有暂停，而是用两次短暂停来替代串行标记整理算法的长暂停，它的收集周期是这样：

初始标记(CMS-initial-mark) -> 并发标记(CMS-concurrent-mark) -> 重新标记(CMS-remark) -> 并发清除(CMS-concurrent-sweep) -> 并发重设状态等待下次CMS的触发(CMS-concurrent-reset)。

其中的1，3两个步骤需要暂停所有的应用程序线程的。第一次暂停从root对象开始标记存活的对象，这个阶段称为初始标记；第二次暂停是在并发标记之后，暂停所有应用程序线程，重新标记并发标记阶段遗漏的对象（在并发标记阶段结束后对象状态的更新导致）。第一次暂停会比较短，第二次暂停通常会比较长，并且 remark这个阶段可以并行标记。

而并发标记、并发清除、并发重设阶段的所谓并发，是指一个或者多个垃圾回收线程和应用程序线程并发地运行，垃圾回收线程不会暂停应用程序的执行，如果你有多于一个处理器，那么并发收集线程将与应用线程在不同的处理器上运行，显然，这样的开销就是会降低应用的吞吐量。Remark阶段的并行，是指暂停了所有应用程序后，启动一定数目的垃圾回收进程进行并行标记，此时的应用线程是暂停的。

CMS的young generation的回收采用的仍然是并行复制收集器，这个跟Parallel gc算法是一致的。

非技术的技术

[其他笔记本](#)

比较零碎的技术文章归类

[学习指南](#)

IT技术学习路线,IT经典书籍学习和下载

## 友情链接

[IT短篇笑话](#)

百忙中, 可以看看it短篇笑话, 笑一笑, 放松下!

[相当好用的免费svn空间](#)

国内挺不错的svn免费空间, 很适合小团队使用

## 积分与排名

积分 - 1044869

排名 - 73

## 最新评论

1. [Re:linux awk命令详解](#)

讲得非常清楚, 谢谢楼主分享

--青儿哥哥

2. [Re:Linux vmstat命令实战详解](#)

可以看一下redhat的文档, Procs r: The number of processes waiting for run time. b: The number of process.....

--jcuan

3. [Re:JAVA正则表达式: Pattern类与Matcher类详解\(转\)](#)

Pattern

p=Pattern.compile("\\d+");  
Matcher

m3=m.matcher("2223bb");  
m.matches(); //匹配整个字符串  
m.start(); .....

--knn120

4. [Re:Linux网络流量实时监控](#)

[ifstat iftop命令详解](#)

666

--starRTC免费IM直播

5. [Re:Linux vmstat命令实战详解](#)

## 参数介绍

1. 启用CMS: -XX:+UseConcMarkSweepGC。

2. CMS默认启动的回收线程数目是 (ParallelGCThreads + 3)/4 , 如果你需要明确设定, 可以通过-XX:ParallelCMSThreads=20来设定, 其中ParallelGCThreads是年轻代的并行收集线程数

3. CMS是不会整理堆碎片的, 因此为了防止堆碎片引起full gc, 通过会开启CMS阶段进行合并碎片选项: -

XX:+UseCMSCompactAtFullCollection, 开启这个选项一定程度上会影响性能, 阿宝的blog里说也许可以通过配置适当的

CMSFullGCsBeforeCompaction来调整性能, 未实践。

4. 为了减少第二次暂停的时间, 开启并行remark: -XX:+CMSParallelRemarkEnabled。如果remark还是过长的话, 可以开启-

XX:+CMSScavengeBeforeRemark选项, 强制remark之前开始一次minor gc, 减少remark的暂停时间, 但是在remark之后也将立即开始又一次minor gc。

5. 为了避免Perm区满引起的full gc, 建议开启CMS回收Perm区选项:

+CMSPermGenSweepingEnabled -XX:+CMSClassUnloadingEnabled

6. 默认CMS是在tenured generation沾满68%的时候开始进行CMS收集, 如果你的年老代增长不是那么快, 并且希望降低CMS次数的话, 可以适当调高此值:

-XX:CMSInitiatingOccupancyFraction=80

这里修改成80%沾满的时候才开始CMS回收。

7. 年轻代的并行收集线程数默认是(cpu <= 8) ? cpu : 3 + ((cpu \* 5) / 8), 如果你希望降低这个线程数, 可以通过-

XX:ParallelGCThreads= N 来调整。

8. 进入重点, 在初步设置了一些参数后, 例如:

```
-server -Xms1536m -Xmx1536m -XX:NewSize=256m -XX:MaxNewSize=256m -XX:PermSize=64m  
-XX:MaxPermSize=64m -XX:-UseConcMarkSweepGC -XX:+UseCMSCompactAtFullCollection  
-XX:CMSInitiatingOccupancyFraction=80 -XX:+CMSParallelRemarkEnabled  
-XX:SoftRefLRUPolicyMSPerMB=0
```

需要在生产环境或者压测环境中测量这些参数下系统的表现, 这时候需要打开GC日志查看具体的信息, 因此加上参数:

```
-verbose:gc -XX:+PrintGCTimeStamps -XX:+PrintGCDetails -Xloggc:/home/test/logs/gc.log
```

在运行相当长一段时间内查看CMS的表现情况, CMS的日志输出类似这样:

例如在apache和nginx这种web服务器中，我们一般做性能测试时会进行几千并发甚至几万并发的测试，选择web服务器的进程可以由进程或者线程的峰值一直下调，压测，直到cs到一个比较小的值，这个进程.....

--201dom

#### 阅读排行榜

1. linux awk命令详解(1193409)
2. Linux tcpdump命令详解(857102)
3. Linux netstat命令详解(584850)
4. linux grep命令详解(425520)
5. linux sed命令详解(375109)

#### 评论排行榜

1. linux awk命令详解(40)
2. Linux tcpdump命令详解(27)
3. C++指针详解(24)
4. linux sed命令详解(23)
5. Linux netstat命令详解(21)

#### 推荐排行榜

1. linux awk命令详解(112)
2. Linux tcpdump命令详解(64)
3. Linux netstat命令详解(62)
4. Linux GCC常用命令(53)
5. Linux入门——适合初学者(52)



```
4391.322: [GC [1 CMS-initial-mark: 655374K(1310720K)] 662197K(1546688K), 0.0303050 secs] [Times: user=0.02 sys=0.02, real=0.03 secs]
4391.352: [CMS-concurrent-mark-start]
4391.779: [CMS-concurrent-mark: 0.427/0.427 secs] [Times: user=1.24 sys=0.31, real=0.42 secs]
4391.779: [CMS-concurrent-preclean-start]
4391.821: [CMS-concurrent-preclean: 0.040/0.042 secs] [Times: user=0.13 sys=0.03, real=0.05 secs]
4391.821: [CMS-concurrent-abortable-preclean-start]
4392.511: [CMS-concurrent-abortable-preclean: 0.349/0.690 secs] [Times: user=2.02 sys=0.51, real=0.69 secs]
4392.516: [GC[YG occupancy: 111001 K (235968 K)]4392.516: [Rescan (parallel) , 0.0309960 secs]4392.547: [weak refs processing, 0.0417710 secs] [1 CMS-remark: 655734K(1310720K)] 766736K(1546688K), 0.0932010 secs] [Times: user=0.17 sys=0.00, real=0.09 secs]
4392.609: [CMS-concurrent-sweep-start]
4394.310: [CMS-concurrent-sweep: 1.595/1.701 secs] [Times: user=4.78 sys=1.05, real=1.70 secs]
4394.310: [CMS-concurrent-reset-start]
4394.364: [CMS-concurrent-reset: 0.054/0.054 secs] [Times: user=0.14 sys=0.06, real=0.06 secs]
```



其中可以看到CMS-initial-mark阶段暂停了0.0303050秒，而CMS-remark阶段暂停了0.0932010秒，因此两次暂停的总共时间是0.123506秒，也就是123毫秒左右。两次短暂停的时间之和在200以下可以称为正常现象。

但是你很可能遇到两种fail引起full gc: Prommotion failed和Concurrent mode failed。

Prommation failed的日志输出大概是这样：

```
[ParNew (promotion failed): 320138K->320138K(353920K), 0.2365970 secs]42576.951: [CMS: 1139969K->1120688K(166784K), 9.2214860 secs] 1458785K->1120688K(2520704K), 9.4584090 secs]
```

这个问题的产生是由于救助空间不够，从而向年老代转移对象，年老代没有足够的空间来容纳这些对象，导致一次full gc的产生。解决这个问题的办法有两种完全相反的倾向：增大救助空间、增大年老代或者去掉救助空间。增大救助空间就是调整-XX:SurvivorRatio参数，这个参数是Eden区和Survivor区的大小比值，默认是32，也就是说Eden区是Survivor区的32倍大小，要注意Survivo是有两个区的，因此Survivor其实占整个young generation的1/34。调小这个参数将增大survivor区，让对象尽量在survivor区呆长一点，减少进入年老代的对象。去掉救助空间的想法是让大部分不能马上回收的数据尽快进入年老代，加快年老代的回收频率，减少年老代暴涨的可能性，这个是通过将-XX:SurvivorRatio 设置成比较大的值（比如65536）来做到。在我们的应用中，将young generation设置成256M，这个值相对来说比较大了，而救助空间设置成默认大小(1/34)，从压测情况来看，没有出现prommation failed的现象，年轻代比较大，从GC日志来

看, minor gc的时间也在5-20毫秒内, 还可以接受, 因此暂不调整。

Concurrent mode failed的产生是由于CMS回收年老代的速度太慢, 导致年老代在CMS完成前就被沾满, 引起full gc, 避免这个现象的产生就是调小-XX:CMSInitiatingOccupancyFraction参数的值, 让CMS更早更频繁的触发, 降低年老代被沾满的可能。我们的应用暂时负载比较低, 在生产环境上年老代的增长非常缓慢, 因此暂时设置此参数为80。在压测环境下, 这个参数的表现还可以, 没有出现过Concurrent mode failed。

分类: [Java](#)

好文要顶

关注我

收藏该文



ggjucheng

关注 - 6

粉丝 - 1627

+加关注

« 上一篇: [JVM1.6 GC详解](#)

» 下一篇: [rsync命令详解](#)

posted on 2014-09-17 17:17 [ggjucheng](#) 阅读(28246) 评论(4) [编辑](#) [收藏](#)

2

0

## 评论

### #1楼 2014-11-27 14:46 zhang\_test

很喜欢博主的文章, 刚刚用豆约翰博客备份专家备份了您的全部博文。

支持(1) 反对(0)

### #2楼 2015-07-22 10:22 漫步云端wjl

8.进入重点, 在初步设置了一些参数后, 例如:

-XX:-UseConcMarkSweepGC

这里应该是: -XX:+UseConcMarkSweepGC

减号害死人啊....

支持(0) 反对(0)

## #3楼 2016-03-30 13:44 猥琐致胜

很喜欢博主的文章，刚刚用豆约翰博客备份专家备份了您的全部博文。

支持(0) 反对(0)

## #4楼 2016-05-19 15:52 江小坤

很喜欢博主的文章，刚刚用豆约翰博客备份专家备份了您的全部博文。

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

腾讯云广告，背景为深蓝色，带有白色和浅蓝色的几何图形。左上角是腾讯云的标志和名称。中间偏上位置是标题“如何快速低成本完成网站搭建？”，下方是副标题“助力日PV1-100万网站部署，云产品3折起”。底部有一个蓝色的按钮，上面写着“立即购买”。**最新IT新闻：**

- [Yandex在俄推出无人驾驶出租车服务 欧洲首次公众测试](#)
- [富士康携手3家公司组建1亿美元规模风险投资基金](#)
- [Instagram支持申请蓝V验证 阻止假新闻](#)

- [蔚来更新招股书：已生产2200台蔚来ES8](#)
- [云米在美国提交IPO申请 计划最多募资1.5亿美元](#)
- » [更多新闻...](#)



**最新知识库文章：**

- [如何招到一个靠谱的程序员](#)
- [一个故事看懂“区块链”](#)
- [被踢出去的用户](#)
- [成为一个有目标的学习者](#)
- [历史转折中的“杭派工程师”](#)
- » [更多知识库文章...](#)

---

Powered by:

[博客园](#)

Copyright © ggjucheng