JVM GC测试和NameNode GC优化方案

垃圾回收机制是Java语言的一个优势，在JVM中垃圾回收由GC(守护进程)来实现，其根据内存使用情况自动运行。由于GC需要消耗一定资源和时间，采用分代的方式进行对象收集，按照新生代、旧生代对对象进行收集，尽可能的缩短GC对应用造成的暂停。常用的垃圾收集器包括：Parallel、CMS和G1收集器，下面主要针对这几种收集器进行测试。

# 测试结论

根据分析和测试结果，目前JVM常用的垃圾收集器包括：Parallel、CMS和G1收集器

* Parallel，JVM缺省收集器，使用多线程扫描并压缩堆，但是GC时会暂停其他所有工作线程，CPU利用率高，适用于高吞吐量的应用，但是停顿时间会比较长。
* CMS，基于标记-清除算法实现的收集器，使用多线程算法扫描堆标记并对发现的未使用的对象进行清除。CMS收集器在收集结束时会产生大量空间碎片，而且其无法处理浮动垃圾，可能出现Concurrent Mode Failure失败而导致Full GC，临时启用Serial Old收集器重新进行老年代的垃圾收集，停顿时间会过长。
* G1(Garbage-First)收集器, 旨在替代CMS的收集器方案。面向服务端应用的收集器，针对多处理大容量内存服务器短的垃圾收集器，其目标在实现高吞吐量的同时，尽可能满足GC暂停时间的要求。与CMS相比不会产生空间碎片，这堆长时间允许的应用系统非常重要，另一方面可以精确的控制GC停顿时间。

根据下面的测试结果和社区堆G1的测试来看，G1收集器在GC暂停时间指标：最大暂停时间、平均暂停时间和最小暂停时间等的测试结果指标都要优于其他垃圾收集器。在吞吐量指标上一定程度上实现了高吞吐量，不会有明显下降。

集中化经分的NameNode使用的内存已经达到上百G（未来可能更大），而且NameNode属于服务端应用，需要尽可能短的暂停时间。使用G1垃圾收集在一定程度上会提升性能，而且能够避免CMS收集器造成的Concurrent Mode Failure问题而导致业务暂停的情况，建议使用G1垃圾收集器。测试结果和参数调整方案在下面进行介绍。

# 测试程序及测试结果汇总

## 2.1 测试程序

* 针对GC测试的测试程序，TestApp，代码见附录
* Hadoop MR作业，模拟生产环境，测试NameNode的GC
* 社区对G1和CMS的测试

## 2.2 测试结果

### 2.2.1 TestApp程序测试结果

表格1：-Xmx2g，执行时间5分钟，每轮写入消息条数：5.000,000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC算法 | Serial | Parallel | CMS | G1 |
| 占用总量 | 1945 | 1928 | 1944 | 2048 |
| 平均暂停 | 0.28353 | 0.10062 | 0.20142 | 0.0886 |
| 最小暂停 | 0.00437 | 0.00166 | 0.00489 | 0.00154 |
| 最大暂停 | 1.28869 | 1.41787 | 3.34579 | 1.01982 |
| 总暂停时间 | 119.65 | 101.82 | 110.78 | 138.13 |
| 吞吐量 | 80% | 84.00% | 81.53% | 76.98% |

表格2：-Xmx4g，执行时间10分钟，每轮写入消息条数：10,000,000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC算法 | Serial | Parallel | CMS | G1 |
| 占用总量 | 2286 | 3641 | 3959 | 4096 |
| 平均暂停 | 0.81596 | 0.24596 | 0.4146 | 0.11451 |
| 最小暂停 | 0.0331 | 0.0049 | 0.00191 | 0.00166 |
| 最大暂停 | 3.06851 | 2.60713 | 3.41563 | 0.68087 |
| 总暂停时间 | 709.07 | 259 | 254.9 | 249.51 |
| 吞吐量 | 25% | 56.85% | 51.94% | 58.44% |

表格3：-Xmx8g，执行时间20分钟，每轮写入条数：10,000,000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC算法 | Serial | Parallel | CMS | G1 |
| 占用总量 | 2288 | 4218 | 8000 | 8192 |
| 平均暂停 | 0.83424 | 0.20114 | 0.40625 | 0.17414 |
| 最小暂停 | 0.02332 | 0.01496 | 0.00194 | 0.00254 |
| 最大暂停 | 2.69648 | 3.91097 | 8.22612 | 0.97166 |
| 总暂停时间 | 1449 | 166.75 | 279.9 | 497.34 |
| 吞吐量 | 24% | 86.11% | 76.67% | 58.58% |

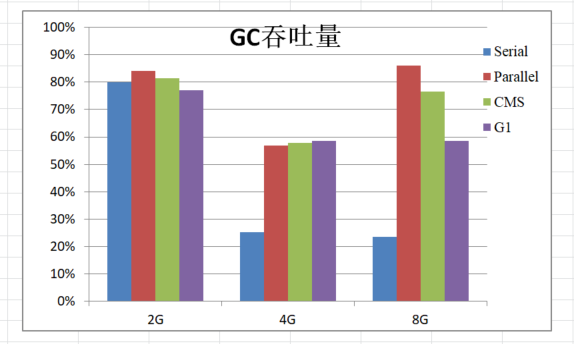
表格中红色代表测试结果最差，绿色代表测试结果最好。

**TestApp测试程序的GC Pause统计结果汇总：**

|  |  |
| --- | --- |
|  |  |
|  | |

G1垃圾回收算法在平均暂停时间、最小暂停时间及最大暂停时间各个指标上都明显优于其他垃圾回收算法。

吞吐量测试统计图如下：



Parallel 和CMS 的吞吐量在几轮测试中都是最高的，G1在内存越大的情况下吞吐量会有所下降。

下面补充了G1和CMS在64GB和128GB的测试指标：

表格4：-Xmx64g，执行时间30分钟，每轮写入条数：400,000,000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC算法 | 平均暂停 | 最小暂停 | 最大暂停 | 吞吐量 |
| G1 | 1.26139 | 0.00064 | 5.46501 | 78.77 |
| CMS | 6.51343 | 0.00187 | 159.3576 | 9.46 |

表格5：-Xmx128g，执行时间30分钟，每轮写入条数：400,000,000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC算法 | 平均暂停 | 最小暂停 | 最大暂停 | 吞吐量 |
| G1 | 2.30625 | 0.00123 | 73.84419 | 76.35 |
| CMS | 3.2527 | 0.00257 | 200.7105 | 15.02 |

在内存较大的情况下，G1的GC优势更加明显

### 2.2.2 NameNode仿真测试

测试程序在NameNode堆内存使用接近16GB的情况下，使用NNBench程序进行测试，测试结果如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GC算法 | 平均暂停 | 最小暂停 | 最大暂停 | 吞吐量 |
| CMS | 0.20768 | 0.00541 | 2.63031 | 98.13 |
| G1 | 0.14601 | 0.00243 | 2.46684 | 99.1 |

G1在GC Pause的测试指标上要明显优于CMS GC算法，在吞吐量上指标上基本差不多。

### 2.2.3 社区测试结果

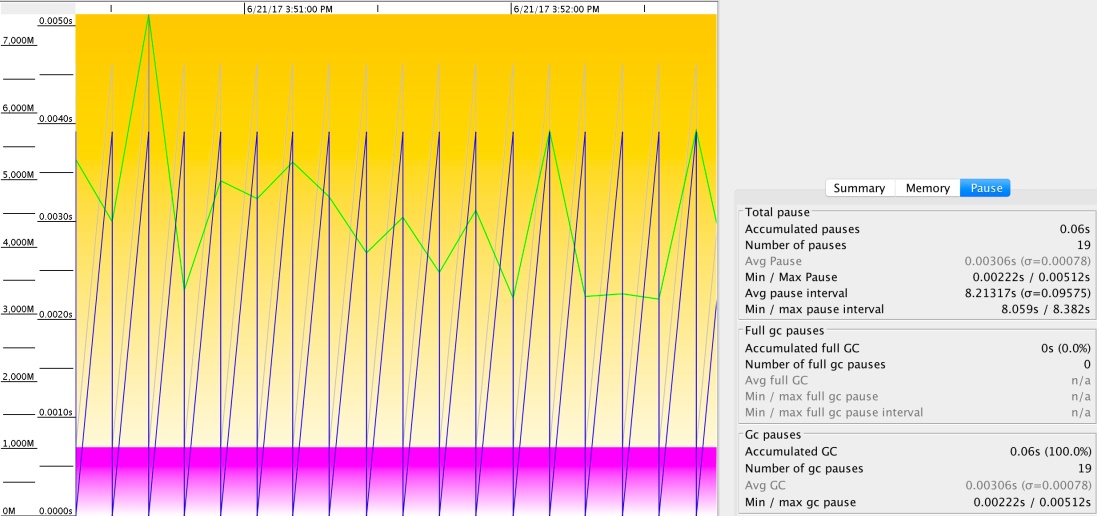
1）G1与CMS收集器对比

参考链接：http://blog.chriscs.com/2017/06/20/g1-vs-cms/

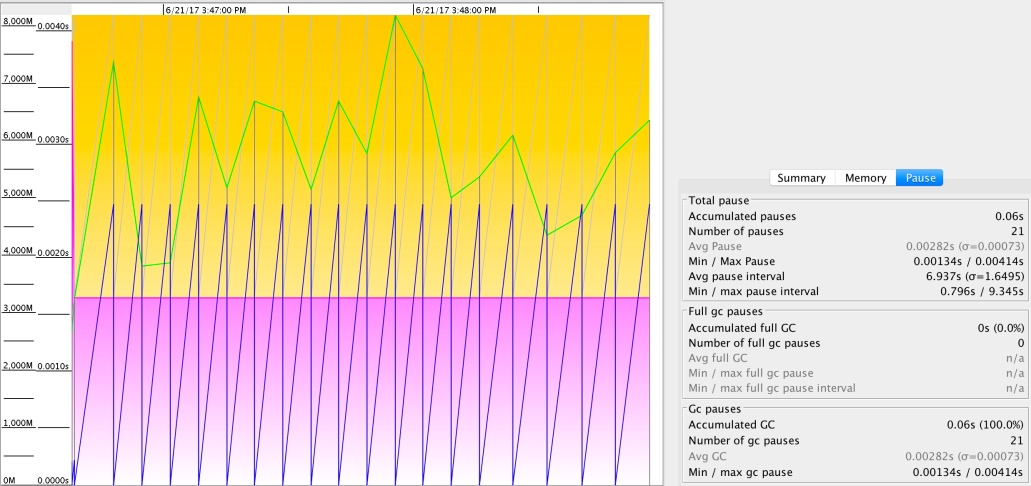
测试结果如下表：

1. 新生代测试

* CMS执行情况



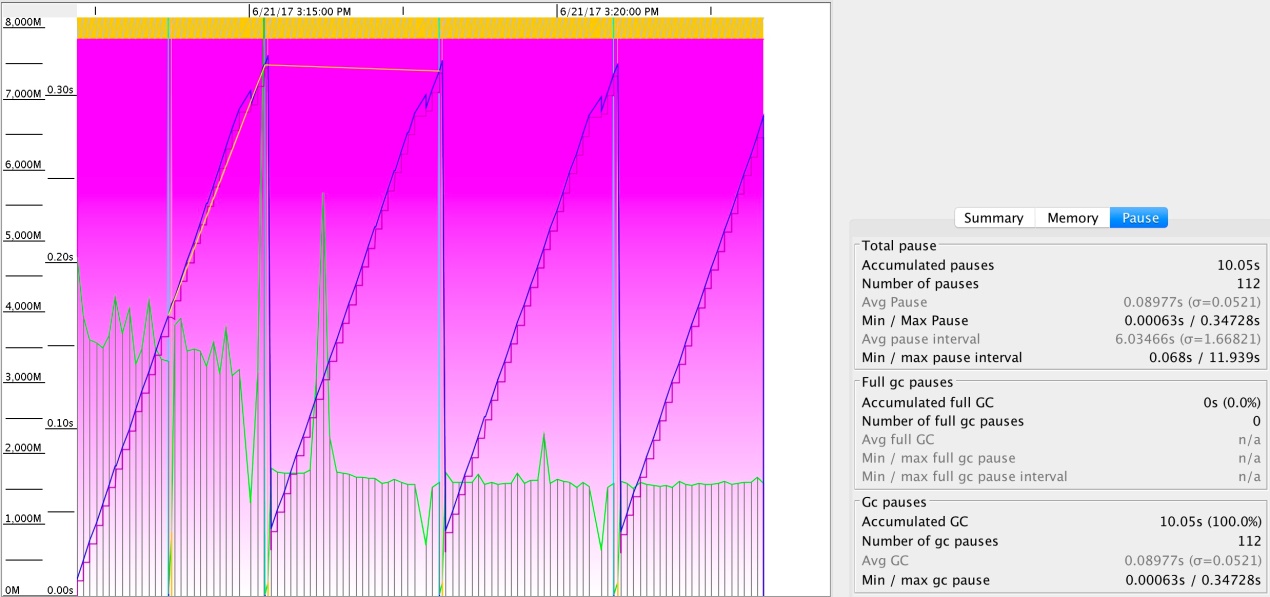
* G1执行情况



在新生代执行测试中，CMS和G1的吞吐率测试结果相近，最大暂停时间G1更好，G1对新生代的回收性能是足够的。

1. 老生代测试

* CMS执行情况



* GC执行情况



总结：G1和CMS测试性能对比如下表

|  |  |  |  |
| --- | --- | --- | --- |
| GC算法 | 总暂停 | 最大暂停 | 吞吐量 |
| CMS | 10.05 | 0.34 | 98.5 |
| G1 | 7.93 | 0.32s | 98.81 |

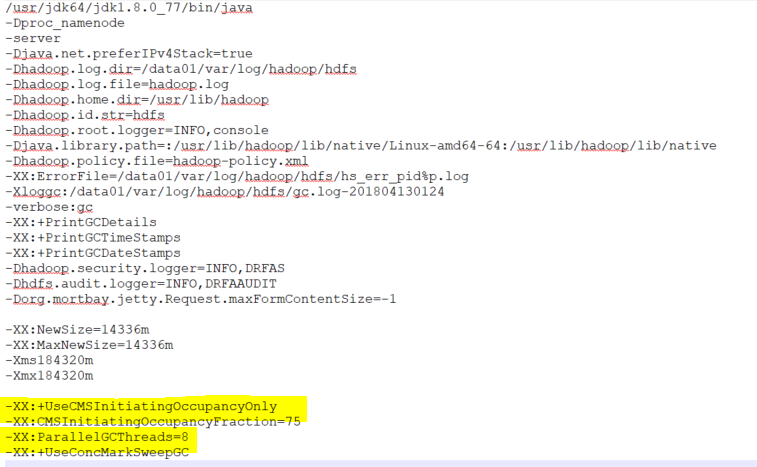
G1在总暂停时间、最大暂停时间和吞吐量指标上都要优于CMS

## 2.3 测试结果总结

以上测试主要针对CMS和G1两种收集器进行的测试对比，根据测试结果可以得出G1收集器在保证吞吐量的同时，在GC暂停时间指标（最大、最小和平均暂停）上要优于CMS收集器。在高内存、服务端应用来说G1收集器的优势会更加明显。

# 4. JVM GC参数调整方案

目前集中化经分的NameNode的GC使用CMS收集器，配置如下：



上面标注的GC参数时CMS相关参数，修改为G1相关参数，如下所示：

*-XX:+UseG1GC*

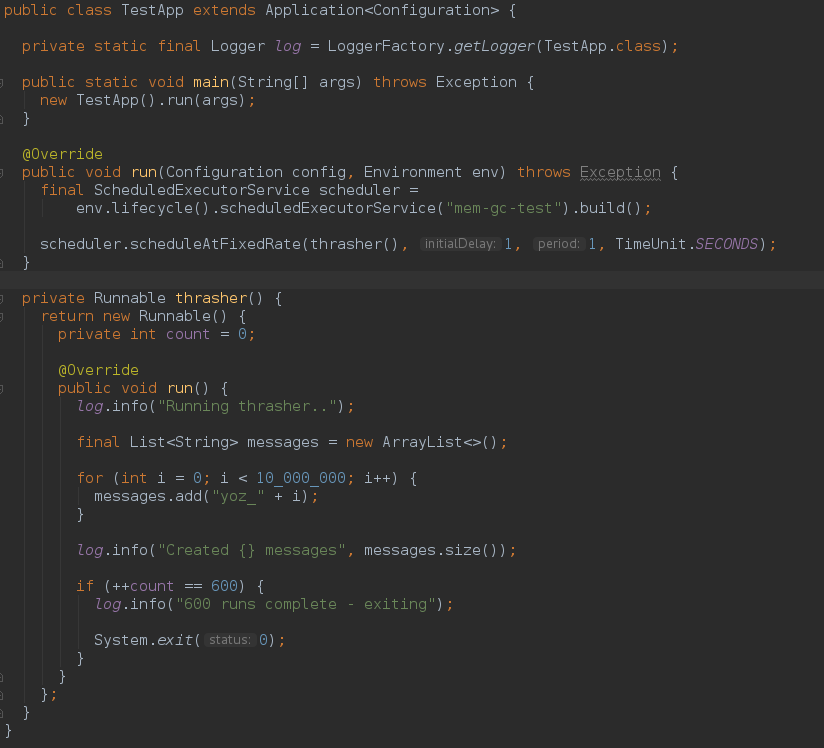
*-XX:ParallelGCThreads=20*

*-XX:ConcGCThreads=20*

*-XX:MaxGCPauseMillis=5000*

附录1：

1. TestApp测试程序



参考文献：

https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.4/bk\_hdfs-administration/content/ch\_g1gc\_garbage\_collector\_tech\_preview.html