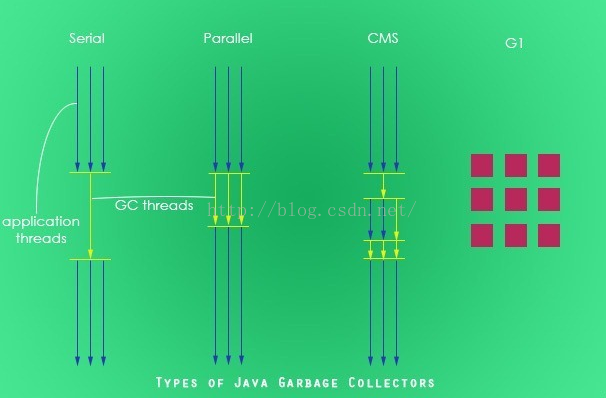
G1,CMS及Par GC性能对比

目前以后的几种GC，从收集器着重点上可以进行如下分类：

* 串行垃圾收集器，最基本的垃圾收集器，进行垃圾回收时会STW
* 吞吐量垃圾收集器，Parallel Scavenge垃圾收集器，达到一个可控制的吞吐量。
* 并发低停顿收集器（低延迟垃圾收集器），CMS，只有ParNew能与CMS收集器配合，默认使用ParNew。CMS垃圾收集器以获取最短回收停顿时间为目标。
* 垃圾首先收集器（G1），独立管理这个GC堆，包括新生代和老生代，不需要与其他收

在服务端常用的是后三种，下面对这三种GC进行测试，



启用的配置如下：

*-XX:+UseParallelOldGC*

*-XX:+UseConcMarkSweepGC*

*-XX:+UseG1GC*

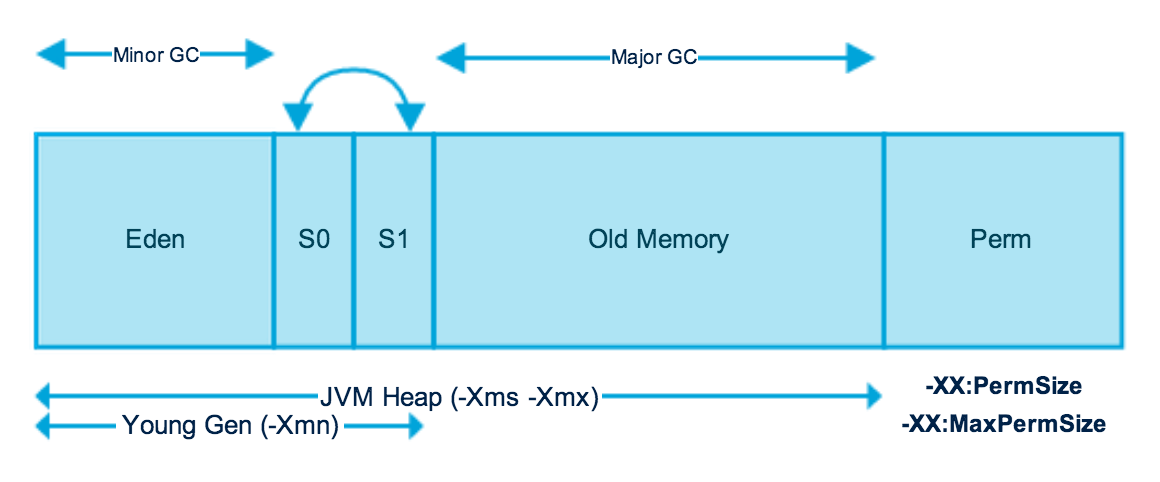
测试过程中，通过配置：

*-XX:+PrintGCTimeStamps -Xloggc:/tmp/gc.log -XX:+PrintGCDetails*

来收集GC参数，然后通过GCLogViewer来分析里面的数据。

# 参数及评估性能

JVM内存相关配置如下图所示：



JVM整个内存大小=年轻代大小+年老代大小+持久代大小。

* -Xms，初始堆内存，等价于-XX:InitialHeapSize，示例：-Xms2G
* -Xmx，最大堆内存，等价于-XX:MaxHeapSize，最好将-Xms和-Xmx设置一样大，减少申请内存的消耗
* -Xmn，设置年轻代的大小
* -XX:NewRatio，设置老生代与新生代的比例
* -XX:SurvivorRatio，Eden区与Survivor区的大小比例，默认是8
* -XX:PermSize，永久代初始大小
* -XX:MaxPermSize，永久代最大值，永久代的大小并会被包括到使用参数-Xmx设置的堆内存大小中，两者是独立的。
* -Xss，设置每个线程的堆栈大小，相同物理内存下，减少这个值能生成更多的线程。

1. Parallel

* -XX:MaxGCPauseMillis，设置最大垃圾收集停顿时间，其优先级高
* -XX:GCTimeRatio，设置吞吐量大小
* **-XX:ParallelGCThread，**GC线程数

1. CMS ，年轻代进行并发GC，会有STW

* -XX:CMSInitiatingOccupancyFraction，触发老年代回收的触发CMS的百分比
* XX:+CMSPermGenSweepingEnabled，持久代使用CMS收集器
* -XX:+CMSClassUnloadingEnabled，持久代能真正释放不再使用的类
* -XX:ParallelGCThreads=n
* -XX:+UseCMSCompactAtFullCollection：使用并发收集器时，开启对年老代的压缩

3）G1

* -XX:MaxGCPauseMillis=50，暂停时间目标
* -XX:GCPauseIntervalMillis=200，暂停间隔目标
* -XX:+G1YoungGenSize=512m，年轻代大小

# 测试

## 2.1 测试环境

http://10.254.10.116:8080

root

CMH@bdi123

admin 6zlHSc\_6x@

主机配置：

磁盘：120T

内存： 256G/128G

CPU:

## 2.2 测试程序

NN：

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples-2.8.2-bc1.4.0.jar randomtextwriter –Dmapreduce.randomtextwriter.mapsperhost=50 -D mapreduce.randomtextwriter.bytespermap=32212254720 /randomwrite/input

hadoop org.apache.hadoop.fs.slive.SliveTest -baseDir /benchmarks/slive -ops 100000 -files 10000 -maps 100 -reduces 10 -create 50 -append 0 -delete 50 -read 0 -rename 0 -truncate 0 -mkdir 0 -ls 0

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-\*-tests.jar nnbench -operation create\_write -maps 100 -reduces 15 -bytesToWrite 1 -numberOfFiles 10000 -blockSize 1048576 -replicationFactorPerFile 3 -readFileAfterOpen true -baseDir /benchmarks/NNBench2

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-\*-tests.jar nnbench -operation delete -maps 100 -reduces 15 -bytesToWrite 1 -numberOfFiles 10000 -blockSize 1048576 -replicationFactorPerFile 3 -readFileAfterOpen true -baseDir /benchmarks/NNBench5

## 2.3 测试流程

1）通用配置：

-Xloggc:/var/log/hadoop/hdfs/gc.log-201803080929

-verbose:gc

-XX:+PrintGCDetails

-XX:+PrintGCTimeStamps

-XX:+PrintGCDateStamps

-XX:+PrintGCApplicationStoppedTime

-XX:ErrorFile=/var/log/hadoop/hdfs/hs\_err\_pid%p.log

2）进行三组内存大小测试

|  |  |  |  |
| --- | --- | --- | --- |
|  | 4G | 8G | 16G |
| 堆内存 | -Xms4096m  -Xmx4096m | -Xms8192m  -Xmx8192m | -Xms16384m  -Xmx16384m |
| -XX:SurvivorRatio | 8 | 8 | 8 |
| 新生代 | -XX:NewSize=512m  -XX:MaxNewSize=512m | -XX:NewSize=2048m  -XX:MaxNewSize=2048m | -XX:NewSize=4096m  -XX:MaxNewSize=4096m |
| 老生代 |  |  |  |
| -XX:NewRatio | 2 | 2 | 2 |

3）进行3种GC算法测试

|  |  |  |  |
| --- | --- | --- | --- |
|  | Parallel | CMS | G1 |
| 启用配置 | -XX:+UseParallelOldGC | -XX:+UseConcMarkSweepGC  –XX:+UuseParNewGC | -XX:+UseG1GC |
| GC线程 | -XX:ParallelGCThreads=8 | -XX:ParallelGCThreads=8 | -XX:ConcGCThreads=8 |
|  |  | -XX:CMSInitiatingOccupancyFraction=70  -XX:+UseCMSInitiatingOccupancyOnly |  |
|  | -XX:MaxGCPauseMillis=100  **-XX:GCTimeRatio=19** |  | -XX:MaxGCPauseMillis=n  **-XX:GCTimeRatio=n** |

1. 生成初始数据20T，大概占用内存600M
2. NNBench Write生成数据，占用内存：
3. NNBench Write和Delete同时执行，查看GC日志

# 结果分析

### 3.3.1 吞吐量（Throughput）

使用-XX:GCTimeRatio参数指定可接受的GC时间占比（目标吞吐量），吞吐量计算公式为：

*吞吐量=运行用户代码时间/（运行用户代码时间+垃圾收集时间）*

高吞吐量即减少垃圾收集时间，让用户代码获得更长的运行时间。

### 3.3.2 软实时目标(Soft Real-Time Goal) ，反应停顿时间

JVM堆容量下的测试结果，V%，avgV%和wV%分别代表的含义如下：

1. V%，表示测试过程中，软实时目标失败的概率，目标失败表示GC时间超过允许的最大GC时间，配置参数：-XX:MaxGCPauseMillis。
2. avgV%，表示在所有实际GC时间超标的时间片段里，实际GC时间超过最大的GC时间的平均百分比，实际GC时间减去允许最大GC时间，再除以总时间片段。
3. wV%，表示在测试结果最差的时间片段里，实际GC时间占用执行时间的百分比

HDFS元数据都会存在于Namenode的内存中，因此NameNode的内存大小直接决定了集群支持最大容量，如何估算namenode需要内存大小？

元数据占用内存空间说明   
对象类别 估算大小（bytes） 计算方法 估计总大小（bytes）   
文件 224 224+2\*文件名长度 250   
目录 264 264+2\*文件名长度 290   
块 152 152+72\*副本数 368

NameNode元数据内存估算举例   
一个典型集群拥有如下信息   
文件数量：10 600 000   
目录数量：310 000   
块数量：13 300 000

计算元数据需要内存： 250x 10600000+290x310000 +368x13300000

= 7 634 300 000（bytes）≈ 7.63G

13300000(block数目) =》 4G （大概）=> 200T数据

10600000（文件数目目） =》2.5G(大概) =》 400万个文件大概1G内存