JVM中最大堆大小与三个方面相关：

1.操作系统的数据模型(32位还是64位)

2.系统的可用虚拟内存限制

3.系统的可用物理内存限制

--Xmx3550m 设置了jvm的最大可用内存为3550m

-Xms3550m 设置了 最大促使内存 可以设置和最大可用内存相等 以避免jvm每次垃圾回收完之后虚拟机重新分配内存

-Xmn2g 设置年轻带的大小为2g 整个jvm的内存大小=年轻代+老年代+持久代 持久代 一般规定为64m 所以修改年轻带的大小 会影响到老年代的大小

一般官方推荐 此值的配置为3/8

-Xss128k 设置了每条线程所占用的内存 这个值设置小的话 可以产生更多的线程数 但是一般操作系统堆每个进程都有最大线程数量 一般经验值在3000-5000

在jdk5.0之后 每条线程分配的内存在1m 之前是256k

-XX:NewRatio=4 设置年轻代(包括两个区)和老年代的比值 设置为4 则表示年轻代比上老年代的比值为4:1

-XX:SurvivorRatio=4 设置了年轻代中 新生代和存活代两个区的比值 设置为4 则表示 存活代中两个区与新生代的比值为2:4 一个新生代占整个年轻带的1/6

-XX:MaxPermSize=16m 设置持久代 大小为16m

-XX:MaxTenuringThreshold=0:设置为0则表示年轻代中的对象**不经过Survivor区 直接进入老年代 对于老年代比较多的应用 可以提升应用的效率**

如果将此值设置为一个较大值 那么对象会在Survivor中进行多次复制 这样可以增加对象在年轻带中的存活时间

-XX:+HeapDumpOnOutOfMemoryError -Dfile.encoding=UTF-8

当程序发生oom是自动生成DUMP文件。

-XX:HeapDumpPath=${目录}参数表示生成DUMP文件的路径，也可以指定文件名称，例如：-XX:HeapDumpPath=${目录}/java_heapdump.hprof。如果不指定文件名，默认为：java_<pid>_<date>_<time>_heapDump.hprof。

回收器选择 一般有两个标准 一是 响应速度 二时吞吐量

调优总结

1.年轻带大小选择

响应时间优先的应用 尽可能设置大 达到系统的最低响应时间 这种情况下年轻带的垃圾收集频率会最小 同时减少到达老年代的对象

吞吐量优先的应用 尽可能的设置大 适合8cpu以上的应用

2.年老带大小选择

响应时间优先应用： 年老带的垃圾收集使用的是并发收集器 所以要注意 一般要考虑并发会话率和会话持续时间等一些原因 如果堆设置小了就会产生内存碎片化 ，高回收率 以及应用暂停使用标记式的垃圾回收方式 如果堆设置的过大 则需要较长的垃圾收集时间

一般需要考虑一下集中情况

并发垃圾收集信息

持久带并发垃圾收集次数

传统GC收集信息

花费在年轻带和老年代的时间比例

吞吐量优先应用：一般吞吐量优先得应用 都拥有一个很大的年轻带和一个很小的老年代 原因是这样可以尽可能的回收掉大量的短期对象 减少中期的对象 持久使用的对象放在老年代中

3.较小堆所引起的内存碎片化问题

老年代使用的时并发收集器 所以不会对堆进行压缩 当收集器回收时 他会把相邻的空间进行合并 从而预留给较大的对象 当堆空间较小时 就会出现许多碎片问题 如果并发收集器找不到足够的空间 那么并发收集器就会停止 使用传统的标记 清除收集器 如果出现碎片 可能会需要如下配置

-XX:+UseCMSCompactAtFullCollection 使用并发收集器时 开启堆老年代的压缩

-XX:CMSFullGCsBeforeCompaction=0 在上面的状态设置的基础上 此状态可以设置 在进行过多少次Full GC 之后对老年底啊进行压缩

org.gradle.parallel=true

脱机 build 这是只有在本地存在下载好的文件后 才会生效 可以加快build速度

05-03 12:43:46.610 15310-15329/com.example.huangyi.pagerscroller I/art: Background sticky concurrent mark sweep GC freed 14(448B) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 127.167ms total 214.275ms
05-03 12:43:46.610 15310-15310/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 121.628ms for cause Alloc
05-03 12:43:48.440 15310-15329/com.example.huangyi.pagerscroller W/art: Suspending all threads took: 1.608s
05-03 12:43:48.570 15310-15324/com.example.huangyi.pagerscroller I/art: Clamp target GC heap from 259MB to 256MB
05-03 12:43:48.570 15310-15324/com.example.huangyi.pagerscroller I/art: Alloc

partial concurrent mark sweep GC freed 26(832B) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 1.375ms total 1.617s
05-03 12:43:48.570 15310-15326/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 1.737s for cause Alloc
05-03 12:43:48.780 15310-15310/com.example.huangyi.pagerscroller I/art: Alloc sticky concurrent mark sweep GC freed 5(160B) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 1.396ms total 96.267ms
05-03 12:43:48.780 15310-15324/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 96.498ms for cause Alloc
05-03 12:43:49.000 15310-15324/com.example.huangyi.pagerscroller I/art: Alloc sticky concurrent mark sweep GC freed 1(32B) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 1.691ms total 213.608ms
05-03 12:43:49.000 15310-15326/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 308.980ms for cause Alloc
05-03 12:43:49.130 15310-15326/com.example.huangyi.pagerscroller I/art: Alloc sticky concurrent mark sweep GC freed 0(0B) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 1.374ms total 129.774ms
05-03 12:43:49.130 15310-15329/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 437.013ms for cause Background
05-03 12:43:49.130 15310-15310/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 224.626ms for cause Alloc
05-03 12:43:49.130 15310-15324/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 13.270ms for cause Alloc
05-03 12:43:50.890 15310-15324/com.example.huangyi.pagerscroller I/art: Clamp target GC heap from 259MB to 256MB
05-03 12:43:50.890 15310-15324/com.example.huangyi.pagerscroller I/art: Alloc concurrent mark sweep GC freed 5(12KB) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 1.566ms total 1.760s
05-03 12:43:50.890 15310-15329/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 1.640s for cause Background
05-03 12:43:50.890 15310-15326/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 1.640s for cause Alloc
05-03 12:43:50.890 15310-15310/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 1.641s for cause Alloc
05-03 12:43:52.630 15310-15324/com.example.huangyi.pagerscroller W/art: Suspending all threads took: 1.631s
05-03 12:43:52.770 15310-15324/com.example.huangyi.pagerscroller W/art: Suspending all threads took: 7.673ms
05-03 12:43:52.900 15310-15310/com.example.huangyi.pagerscroller I/art: Clamp target GC heap from 259MB to 256MB
05-03 12:43:52.900 15310-15310/com.example.huangyi.pagerscroller I/art: Alloc partial concurrent mark sweep GC freed 0(0B) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 1.184ms total 1.886s
05-03 12:43:52.900 15310-15328/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 1.746s for cause HeapTrim
05-03 12:43:53.010 15310-15329/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 251.365ms for cause Background
05-03 12:43:53.150 15310-15310/com.example.huangyi.pagerscroller W/art: Suspending all threads took: 23.303ms

05-03 12:43:53.400 15310-15329/com.example.huangyi.pagerscroller I/art: Background sticky concurrent mark sweep GC freed 8(256B) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 2.471ms total 149.864ms
05-03 12:43:55.310 15310-15329/com.example.huangyi.pagerscroller I/art: Clamp target GC heap from 259MB to 256MB
05-03 12:43:55.310 15310-15329/com.example.huangyi.pagerscroller I/art: Background partial concurrent mark sweep GC freed 120(3KB) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 1.241ms total 1.667s
05-03 12:43:55.310 15310-15310/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 1.667s for cause Alloc
05-03 12:43:55.310 15310-15324/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 1.667s for cause Alloc
05-03 12:43:55.520 15310-15310/com.example.huangyi.pagerscroller I/art: Alloc sticky concurrent mark sweep GC freed 28(896B) AllocSpace objects, 0(0B) LOS objects, 0% free, 255MB/256MB, paused 1.213ms total 96.644ms
05-03 12:43:55.520 15310-15329/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 97.253ms for cause Background
05-03 12:43:55.520 15310-15324/com.example.huangyi.pagerscroller I/art: WaitForGcToComplete blocked for 84.298ms for cause Alloc

hprof-conv com.ourslook.changyu:ipc.hprof a.hprof
处理hprof文件

finalizerReference 强引用的集合 属于强引用的子类

强引用可以直接访问目标对象
强引用所指向的对象在任何时候都不会被系统回收
强引用可能导致内存泄漏

1.软引用，引用类型表现为当内存接近满负荷，或对象由softreference.get()方法的调用没有发生一段时间后，垃圾回收器将会清理该对象，在运行对象的finalize方法前，会将软引用对象加入referenceQueue中去
2.弱引用，引用类型表现为当系统垃圾回收器开始回收时，则立即会回收该对象的引用与软引用一样，弱引用也会在运行对象的finalize方法之前将弱引用对象加入referenceQueue.
3.强引用，这是最常用的引用类型，jvm系统采用Finalizer来管理每个强引用对象，并将其被标记要清理时加入ReferenceQueue,并逐一调用该对象的finalize（）方法
4.虚引用，这是一个最虚幻的引用类型，无论是从哪里进去都无法再次返回被虚引用所引用的对象，虚引用在垃圾回收器开始回收对象时，将直接调用finalize()方法，但不会立即将其加入回收队列中去