拟上亿请求的支付系统 jvm 调优

1. 假设案例背景

   假设我们的项目请求量能达到亿级，对我们项目的压力有很多，比如高并发访问、高性能处理请求、大量的信息存储等。此处我们就单一浅谈我们的核心支付系统。

   如果我们有 1 亿的请求量，那么估算每个用户请求 20 次左右，那么就是 500w 人浏览，但是最终下单可能 10%,也就是 50w 订单。如果 50w 订单，在中午和下午一共 4 个小时的高峰期下单。最终大概估算出来每秒 30 或者 40 个订单，一般压力不大。

2. 双 11 促销

   如果遇到双 11，可能在双 11 的前十分钟就有 50w 订单了。那么可以推算每秒就有近 1000 个订单

3. 模拟机器数量

   在双 11 这种节日肯定需要扩展机器，我们大且用作 3 台计算 4 核 8G，那么每机器每秒大概需要抗住 300 个支付请求,机器的配置应该是能够抗住这么多请求，但是我们还是要对根据 JVM 有限的内存资源进行合理的分配和优化，让 JVM 的 GC 尽可能的减少，而且避免 full GC,这样才能减少 JVM 的 GC 对这种高峰期的影响

4. 内存使用估算

   假设：

   1. 每个订单中实例变量 20 个，char2 个字节，int4 个字节，往大了估算一个订单 500 字节，不到 1kb
   2. 300 个订单：那么大概 300kb
   3. 加上订单对象的库存、促销、优惠券、分享等等业务对象，我们可以把内存开销扩大个 20 倍
   4. 加上订单的其他相关操作，例如查询等操作，再扩大 10 倍

      最终我们可以估算出：300*20*10=60MB 内存开销

5. 内存分配

   采用 4 核 8G，可以分配 4G 给 JVM，其他要留给操作系统使用

   堆内存 3G，新生代 1.5G，老年代 1.5G

   每个线程的虚拟机栈 1M,那么可能有几百个线程，那就是几百 M,设置个 256M

   永久代 256M

   参数为：

   ```
   -Xms3072M
   -Xmx3072M
   -Xmn1536M
   -xx:PerSize=256M
   -xx:MaxPerSize=256M
   -xx:Survivor=8
   ```

6. 新生代调优

   以上参数，新生代 Eden1.2G,两个 Survivor150M，如果每秒 60MB 内存开销，那么 20 多秒就会 minor GC 一次。假设存活对象 100MB(正在引用的+minor GC 剩下的)

1. 考虑 Survivor 内存够不够
   按照这个逻辑，大致剩余 100MB，也可能大于 150MB 进入老年代，也有可能因为 100MB 大于了 Survivor 的一半(动态年龄判断)进入老年代，这样的话肯定是不够的，在这里便可以调整新生代大小，可以调高新生代，调低老年代：-Xmn2048
2. 考虑 minor GC 次数进入老年代
   JVM 默认为 15，如果是 15 次，那么这些对象可能在新生代里呆了 20*15=300s,也就是好几分钟了，这种对象进入老年代也是应该的
   但是，-xx:MaxTenuringThreshold 这个值应该要根据我们系统的业务来调整，像我们这种业务系统，类似于 15 次还没回收的多数是@Service、@Controller 这种业务组件
   我们这里可以调低-xx: MaxTenuringThreshold=5,让他提前进入老年代，看了很多博客，也有说可以调高这些参数的，确实是可以延迟进入老年代，但是越到后面可能对象会越来越大，可能没有到次数就会到老年代
3. 多大的对象进入老年代
   大对象进入老年代，说明大对象是长期存活的，我们可以根据自己的系统设置，这种支付系统姑且设置为 1MB，可能也就是超大集合
   -xx:PretenureSizeThreshold=1M
4. 指定垃圾回收器
   -xx:UserParNewGC
   -xx:UserConcMarkSweepGC

核心在于尽可能的让每次 minor GC 后的对象都留在 Survivor 里，不要进入老年代，延缓 full GC 的频率

7. 老年代调优

1.进入老年代的可能性：

1.-xx: MaxTenuringThreshold=5 这种业务组件对象，长期被 GC Root 使用

2.-xx: PretenureSizeThreshold=1 大对象

3.minor GC 存活对象超过 200MB(目前 Survivior 容量 200MB)，或者是同一存活年龄超过 100MB 的。我们新生代的优化就是为了避免这些情况

2.双 11 期间多久触发 full GC 一次

1.每次 minor GC 之前都检查一下

老年代可用空间 < 历次 minor GC 进入老年代的平均大小

这种背景条件下是有可能的，但是几率是很小的

2.可能某次 minor GC 进入老年代的对象有几百 MB，老年代空间不足

3.设置了 -xx:CMSIntiatingOccupancyFaction 参数，默认为 92%，预测这种背景下可能需要 30min-60min 才会有近 1G 的对象在老年代