[译]Java8官方GC调优指南 --(五)可用的收集器 - 掘金

juejin.cn/post/6844904053533573128

2020年1月28日

> 本套文章是Java8官方GC调优指南的全文翻译，点击查看原文,原文章名称《Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide》

5 Available Collectors 可用的收集器

到目前为止，讨论的都是serial collector。Java HotSpot VM包括三种不同的collector，每一种都有不同的性能特征。

- serial collector使用单线程处理所有的gc工作，因为没有多线程的开销，所以相对来说也是比较有效的。它最审核单线程的机器，因为本来也没办法利用多核，不过也适用于多处理器的环境，当然，应用要小(100MB内存左右)。serial collector在某些特定的系统设置下是默认的，或者可以显示配置 `-XX:+UseSerialGC`
- parallel collector(AKA throughput collector，也叫吞吐量收集器)提供并行的minor gc，可以显著减少gc的消耗。这个收集器适用于多处理器部署的大中型应用。使用参数 `-XX+UseParallelGC` 启用
  parallel collector使用并行压缩特性来并行化major gc。没有并行压缩的话，major gc将会使用单线程执行，这样就限制了伸缩性(多核的时候性能没有提升)。并行压缩特性默认是开启的，如果需要关掉可以使用参数 `-XX:-UseParallelOldGC` 。
- The Mostly Concurrent Colletor基于并行方式来缩短gc停顿时间。基于响应时间优先的中大型应用设计，这种gc策略认为响应时间优先于应用的总吞吐量。不过吞吐量一定是有所降低的，因为降低停顿时间一定会降低总吞吐量。JVM提供了两种并发收集器， `-XX:+UseConcMarkSweepGC` 和 `-XX：UseG1GC` 。一个是我们熟知的CMS，另一个就是G1。

Selecting a Collector 选择一个收集器

除非你的应用对停顿时间有严格的要求，否则让JVM去选择收集器。如果有必要的话，调整heap size来提升性能。如果性能还是没达到预期，参考下面几点来选择一个收集器：

- 如果程序很小，内存占用也就100MB，直接用 `-XX:+UseSerialGC` 。
- 如果程序运行在单处理器平台而且没有什么停顿时间的需求，就让JVM选收集器或者选择 `-XX:+UseSerialGC` 。
- 如果程序性能是第一优先级，而且没有停顿时间的需求，或者停顿时间大于1秒以上也是可以接受的，那就让JVM选择收集器，或者用 `-XX:+UseParallelGC`
- 如果程序的响应时间比整体吞吐量更重要，而且程序停顿时间要尽可能短，最好少于1秒，就选择 `-XX:+UseConcMarkSweepGC` 和 `-XX：UseG1GC`

> 总结一下，如果你程序不大，而且是那种批处理，或者消费者之类的，可以选择ParallelGC，吞吐量会很高。 如果是那种web服务器，直接和用户交互的，响应时间是最优先的，而且停顿时间肯定也不能太长，这种时候就选CMS和G1.

这几个提示只是让你先选一个收集器，具体性能取决于heap size还有应用的存活对象数量，还有你CPU的主频、核数等待。停顿时间尤其依赖这几个因素，所以之前说的1秒只是近似值而已，根据经验来看，parallel collector的GC停顿时间在对象数量多的情况下会经常大于1秒；CMS在某些场景也无法保证每次停顿都小于1秒。

> 反正就是我没办法给你保证性能

如果推荐的收集器没有达到预期的性能，先调整heap和generation大小。如果性能还是不够理想，那就换个收集器吧。