**垃圾回收调优**

JVM的垃圾回收在某些情况下可能会造成瓶颈，比如，你的RDD存储经常需要“换入换出”（新RDD抢占了老RDD内存，不过如果你的程序没有这种情况的话那JVM垃圾回收一般不是问题，比如，你的RDD只是载入一次，后续只是在这一个RDD上做操作）。当Java需要把老对象逐出内存的时候，JVM需要跟踪所有的Java对象，并找出那些对象已经没有用了。概括起来就是，垃圾回收的开销和对象个数成正比，所以减少对象的个数（比如用 Int数组取代 LinkedList），就能大大减少垃圾回收的开销。当然，一个更好的方法就如前面所说的，以序列化形式存储数据，这时每个RDD分区都只包含有一个对象了（一个巨大的字节数组）。在尝试其他技术方案前，首先可以试试用序列化RDD的方式（[serialized caching](http://spark.apache.org/docs/latest/tuning.html" \l "serialized-rdd-storage)）评估一下GC是不是一个瓶颈。

如果你的作业中各个任务需要的工作内存和节点上存储的RDD缓存占用的内存产生冲突，那么GC很可能会出现问题。下面我们将讨论一下如何控制好RDD缓存使用的内存空间，以减少这种冲突。

**衡量GC的影响**

GC调优的第一步是统计一下，垃圾回收启动的频率以及GC所使用的总时间。给JVM设置一下这几个参数（参考Spark配置指南 –  [configuration guide](http://spark.apache.org/docs/latest/configuration.html#Dynamically-Loading-Spark-Properties)，查看Spark作业中的Java选项参数）：-verbose:gc -XX:+PrintGCDetails，就可以在后续Spark作业的worker日志中看到每次GC花费的时间。注意，这些日志是在集群worker节点上（在各节点的工作目录下stdout文件中），而不是你的驱动器所在节点。

**高级GC调优**

为了进一步调优GC，我们就需要对JVM内存管理有一个基本的了解：

* Java堆内存可分配的空间有两个区域：新生代（Young generation）和老生代（Old generation）。新生代用以保存生存周期短的对象，而老生代则是保存生存周期长的对象。
* 新生代区域被进一步划分为三个子区域：Eden，Survivor1，Survivor2。
* 简要描述一下垃圾回收的过程：如果Eden区满了，则启动一轮minor GC回收Eden中的对象，生存下来（没有被回收掉）的Eden中的对象和Survivor1区中的对象一并复制到Survivor2中。两个Survivor区域是互相切换使用的（就是说，下次从Eden和Survivor2中复制到Survivor1中）。如果某个对象的年龄（每次GC所有生存下来的对象长一岁）超过某个阈值，或者Survivor2（下次是Survivor1）区域满了，则将对象移到老生代（Old区）。最终如果老生代也满了，就会启动full GC。

Spark GC调优的目标就是确保老生代（Old generation ）只保存长生命周期RDD，而同时新生代（Young generation ）的空间又能足够保存短生命周期的对象。这样就能在任务执行期间，避免启动full GC。以下是GC调优的主要步骤：

* 从GC的统计日志中观察GC是否启动太多。如果某个任务结束前，多次启动了full GC，则意味着用以执行该任务的内存不够。
* 如果GC统计信息中显示，老生代内存空间已经接近存满，可以通过降低 spark.memory.storageFraction 来减少RDD缓存占用的内存；减少缓存对象总比任务执行缓慢要强！
* 如果major GC比较少，但minor GC很多的话，可以多分配一些Eden内存。你可以把Eden的大小设为高于各个任务执行所需的工作内存。如果要把Eden大小设为E，则可以这样设置新生代区域大小：-Xmn=4/3\*E。（放大4/3倍，主要是为了给Survivor区域保留空间）
* 举例来说，如果你的任务会从HDFS上读取数据，那么单个任务的内存需求可以用其所读取的HDFS数据块的大小来评估。需要特别注意的是，解压后的HDFS块是解压前的2~3倍大。所以如果我们希望保留3~4个任务并行的工作内存，并且HDFS块大小为64MB，那么可以评估Eden的大小应该设为 4\*3\*64MB。
* 最后，再观察一下垃圾回收的启动频率和总耗时有没有什么变化。

我们的很多经验表明，GC调优的效果和你的程序代码以及可用的总内存相关。网上还有不少调优的选项说明（[many more tuning options](http://www.oracle.com/technetwork/java/javase/gc-tuning-6-140523.html)），但总体来说，就是控制好full GC的启动频率，就能有效减少垃圾回收开销。