图文 033、案例实战：百万级用户的在线教育平台，如何基于G性能（下）？

详情 评论

案例实战：百万级用户的在线教育平台

如何基于G1垃圾回收器优化性能（下）？

狸猫技术窝专栏上新，基于**真实订单系统**的消息中间件（mq）实战，重磅推荐：

未来3个月，我的好朋友原子弹大侠将带你一起，全程实战，360度死磕MQ

（点击下方蓝字进行试听）

从 0 开始带你成为消息中间件实战高手

重要说明：

如何提问：每篇文章都有评论区，大家可以尽情在评论区留言提问，我都会逐一答疑

（ps：评论区还精选了一些小伙伴对**专栏每日思考题的作答**，有的答案真的非常好！大家可以通过看别人的思路，启发一下自己，从而加深理解**）**

如何加群：购买了狸猫技术窝专栏的小伙伴都可以加入**狸猫技术交流群。**

（群里有不少**一二线互联网大厂的助教**，大家可以一起讨论交流各种技术）

具体**加群方式**请参见文末。

（**注**：以前通过其他专栏加过群的同学就不要重复加了）

1、前文回顾

上篇文章我们针对一个案例背景进行了系统的分析，其实这套系统分析方法论大家如果一路跟着学习到如今，都已经非常的熟练了，可以说这是一个优秀工程师必须具备的JVM压力分析的能力。

作为开发业务系统的工程师，不一定说要深入理解JVM的各种底层原理和源码，因为也没那么多精力去研究那些

但是务必要能够合理的分析自己系统的内存压力，然后合理的优化JVM的参数，尽可能降低JVM GC的频率，同时降低JVM GC导致的系统停顿的时间。

本文我们接着上文的案例继续来分析，在这个案例背景之下来看看，G1垃圾回收器在使用的时候有哪些地方是值得优化的。

2、G1垃圾回收器的默认内存布局

接着我们来看看G1垃圾回收器的默认内存布局，之前说过我们采用的是4核8G的机器来部署系统，然后每台机器每秒会有600个请求会占用3MB左右的内存空间。

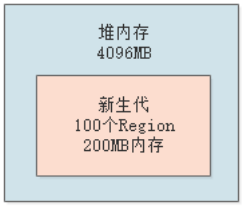
那么假设我们对机器上的JVM，分配4G给堆内存，其中新生代默认初始占比为5%，最大占比为60%，每个Java线程的栈内存为1MB，元数据区域（永久代）的内存为256M，此时JVM参数如下：

“-Xms4096M -Xmx4096M -Xss1M -XX:PermSize=256M -XX:MaxPermSize=256M -XX:+UseG1GC“

“-XX:G1NewSizePercent”参数是用来设置新生代初始占比的，不用设置，维持默认值为5%即可。

“-XX:G1MaxNewSizePercent”参数是用来设置新生代最大占比的，也不用设置，维持默认值为60%即可。

此时堆内存共4G，那么此时会除以2048，计算出每个Region的大小，此时每个Region的大小就是2MB，刚开始新生代就占5%的Region，可以认为新生代就是只有100个Region，有200MB的内存空间，如下图所示。



3、GC停顿时间如何设置?

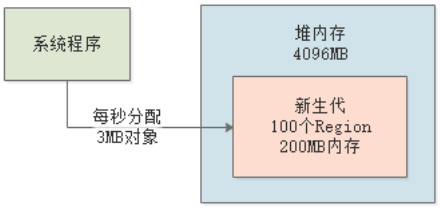
在G1垃圾回收器中有一个至关重要的参数会影响到GC的表现，就是“-XX:MaxGCPauseMills”，他的默认值是200毫秒

也就是说咱们希望每次触发一次GC的时候导致的系统停顿时间（也就是“Stop the World”）不要超过200毫秒，避免系统因为GC长时间卡死。

这个参数我们可以先保持一个默认值，继续往下分析看看，不着急忙下结论。

4、到底多长时间会触发新生代GC?

有一个问题，就是系统运行起来之后，会不停的在新生代的Eden区域内分配对象，按照之前的推算是每秒分配3MB的对象，如下图。



那么之前是说“Eden区域的空间不够了，就触发新生代gc”，但是到底什么时候Eden区域会内存不够呢?

之前说过“-XX:G1MaxNewSizePercent”参数限定了新生代最多就是占用堆内存60%的空间

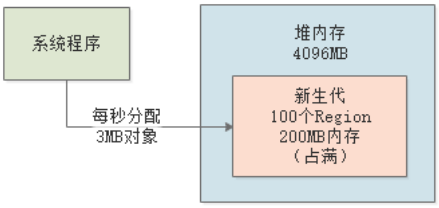
那么难道必须得随着系统运行一直给新生代分配更多的Region，直到新生代占据了60%的Region之后，无法再分配更多的Region了，再触发新生代gc?

G1肯定不是这么搞的，后续我们会通过几十个案例带着大家来实操体验各种JVM运行场景和通过工具来查看内存占用情况，GC频率和效果，但是现在我们就初步给大家说说G1的运行原理。

我们首先假设一个前提，这个纯碎就是我们人为设定的，就是假设在这个系统里，G1回收掉300个Region（600MB内存），大致需要200ms。

那么很有可能系统运行时，G1呈现出如下的运行效果。

首先，随着系统运行，每秒创建3MB的对象，大概1分钟左右就会塞满100个Region（200MB内存），如下图所示。

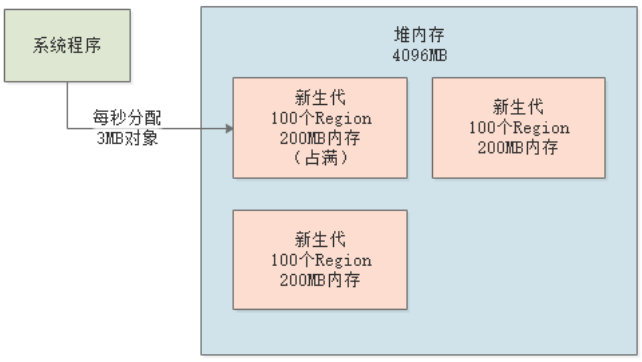


此时很可能G1会觉得，要是我现在就触发一次新生代gc，那么回收区区200MB只需要大概几十ms，最多就让系统停顿几十ms而已，跟我的主人设定的“-XX:MaxGCPauseMills”参数限制的200ms停顿时间相差甚远。

要是我现在就触发新生代gc，那岂不是会导致回收完过后接着1分钟再次让新生代这100个Region塞满，接着又触发新生代gc?

那这样算下来，岂不是每分钟都要执行一次新生代gc？是不是太频繁了？好像没这个必要吧！

所以还不如给新生代先增加一些Region，然后让系统继续运行着在新生代Region中分配对象好了，这样就不用过于频繁的触发新生代gc了，此时如下图。



然后系统继续运行，一直到可能300个Region都占满了，此时通过计算发现回收这300个Region大概需要200ms，那么可能这个时候就会触发一次新生代gc了。

所以大家通过这一小节的分析就明白了，其实G1里是很动态灵活的，他会根据你设定的gc停顿时间给你的新生代不停分配更多Region

然后到一定程度，感觉差不多了，就会触发新生代gc，保证新生代gc的时候导致的系统停顿时间在你预设范围内。

但是大家觉得上述的数字一定精准吗？

No!

完全只是示范一下做一个示例，其实这个G1到底会分配多少个Region给新生代，多久触发一次新生代gc，每次耗费多长时间，这些都是不确定的，必须通过一些工具去查看系统实际情况才知道，这个提前是无法预知的。

但是大家需要知道的，就是G1它本身是这样的一个运行原理，他会根据你预设的gc停顿时间，给新生代分配一些Region，然后到一定程度就触发gc，并且把gc时间控制在预设范围内，尽量避免一次性回收过多的Region导致gc停顿时间超出预期。

5、新生代gc如何优化?

此时大家就可以思考一下了，那么新生代gc如何优化?

其实，垃圾回收器是一代比一代先进，内部实现机制越来越复杂，但是对我们来说优化的时候越来越简单了。

比如对于G1而言，我们首先应该给整个JVM的堆区域足够的内存，比如我们在这里就给了JVM超过5G的内存，其中堆内存有4G的内存。

接着就应该合理设置“-XX:MaxGCPauseMills”参数

如果这个参数设置的小了，那么说明每次gc停顿时间可能特别短，此时G1一旦发现你对几十个Region占满了就立即触发新生代gc，然后gc频率特别频繁，虽然每次gc时间很短。

比如说30秒触发一次新生代gc，每次就停顿30毫秒。

如果这个参数设置大了呢?

那么可能G1会允许你不停的在新生代理分配新的对象，然后积累了很多对象了，再一次性回收几百个Region

此时可能一次GC停顿时间就会达到几百毫秒，但是GC的频率很低。比如说30分钟才触发一次新生代GC，但是每次停顿500毫秒。

所以这个参数到底如何设置，需要结合后续给大家讲解的系统压测工具、gc日志、内存分析工具结合起来进行考虑，尽量让系统的gc频率别太高，同时每次gc停顿时间也别太长，达到一个理想的合理值。

6、mixed gc如何优化?

说完了这个新生代gc之后，那接下来就是mixed gc的优化了

对于这个mixed gc的触发，大家都知道是老年代在堆内存里占比超过45%就会触发。

大家之前都很清楚了年轻代的对象进入老年代的几个条件了，要不然是新生代gc过后存活对象太多没法放入Survivor区域，要不然是对象年龄太大，要不然是动态年龄判定规则。

其中尤其关键的，就是新生代gc过后存活对象过多无法放入Survivor区域，以及动态年龄判定规则

这两个条件尤其可能让很多对象快速进入老年代，一旦老年代频繁达到占用堆内存45%的阈值，那么就会频繁触发mixed gc。

所以mixed gc本身很复杂，很多参数可以优化，但是优化mixed gc的核心不是优化他的参数，而是跟我们之前分析的思路一样，尽量避免对象过快进入老年代，尽量避免频繁触发mixed gc，就可以做到根本上优化mixed gc了。

那么G1里面跟之前的ParNew+CMS的组合是不同的，我们到底应该如何来优化参数呢?

其实核心的点，还是“-XX:MaxGCPauseMills”这个参数。

大家可以想一下，假设你“-XX:MaxGCPauseMills”参数设置的值很大，导致系统运行很久，新生代可能都占用了堆内存的60%了，此时才触发新生代gc。

那么存活下来的对象可能就会很多，此时就会导致Survivor区域放不下那么多的对象，就会进入老年代中。

或者是你新生代gc过后，存活下来的对象过多，导致进入Survivor区域后触发了动态年龄判定规则，达到了Survivor区域的50%，也会快速导致一些对象进入老年代中。

所以这里核心还是在于调节“-XX:MaxGCPauseMills”这个参数的值，在保证他的新生代gc别太频繁的同时，还得考虑每次gc过后的存活对象有多少，避免存活对象太多快速进入老年代，频繁触发mixed gc。

至于到底如何优化这个参数，**一切都要结合后续大量工具的讲解和实操演练了**，到这里为止，至少大家对原理性的东西都很了解了。

7、今日思考题

到底为止，大家已经基本学明白了G1的运行原理以及基本的优化思路，那么**我想问大家两个问题：**

G1这种垃圾回收器到底在什么场景下适用呢？
有了G1以后，是不是还有一些场景采用“ParNew+CMS”垃圾回收器也可以呢？

End

如何加群？

添加微信号：Lvgu0715_ （微信名：绿小九），狸猫技术窝管理员
发送 Jvm专栏的购买截图
由于是人工操作，发送截图后请耐心等待被拉群

最后再次提醒：通过其他专栏加过群的同学，就不要重复加了

狸猫技术窝其他精品专栏推荐：

21天互联网java进阶面试训练营（分布式篇）