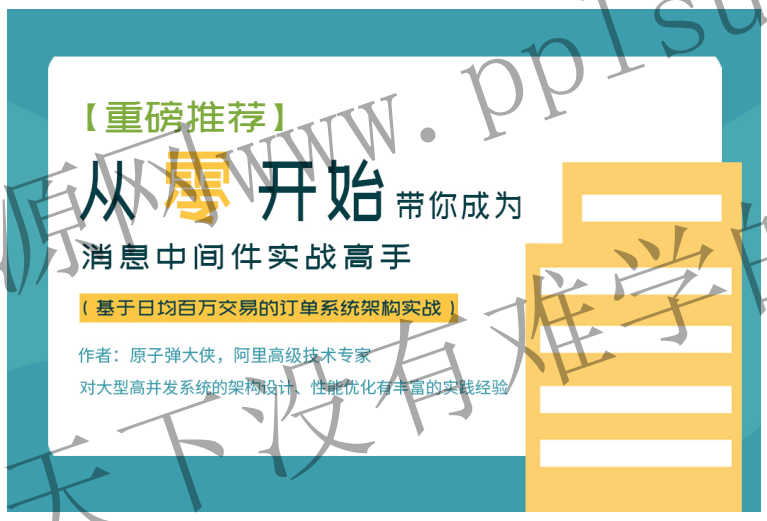
图文 061、案例实战：电商大促活动下，严重Full GC导致系统直接卡死的优化实战

详情 评论

案例实战：
电商大促活动下，严重Full GC导致系统直接卡死的优化实战！

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）

今天这个案例因为是延迟了一天发布在周末，而且周末很多同学一般都会出去玩儿，所以我们今天发一个比较简单轻松的优化小案例，让大家一下子就能看完的，不烧脑子的，对应了一个小的JVM优化的知识点。

案例是这样，有一次一个新系统上线，平时都还算正常，结果有一次大促活动的时候，这个系统就直接卡死不动了

大家注意，是直接卡死不动！也就是说，所有请求到这个系统就直接卡住无法处理，无论如何重启这个系统都没任何效果。

这个时候我们当然会想，是不是按照之前的思路，一点一点去分析JVM的GC问题，考虑是不是过于频繁的GC问题导致了系统被卡死?

那当然是会按照之前的思路去分析的，首先使用jstat去看一下系统运行情况，令人吃惊的事情是：JVM几乎每秒都执行一次Full GC，每次都耗时几百毫秒。

我们当时就惊呆了，为什么每秒都有一次Full GC?

结果更加令人吃惊的事情还在后面：我们通过jstat看了一下JVM各个内存区域的使用量，基本都没什么问题，年轻代对象增长并不快，老年代才占用了不到10%的空间，永久代也就使用了20%左右的空间

那。。。为什么会频繁触发Full GC呢?

这个时候我们立马想到了一个问题，是不是有人在系统里写了一行致命的代码：**System.gc()**

这个“System.gc()”可不能随便瞎写，他每次执行都会指挥JVM去尝试执行一次Full GC，连带年轻代、老年代、永久代都会去回收。

所以我们立马找到那个系统负责开发的同学，让他排查了一下自己的代码。

其实说是排查，无非就是在开发IDE中开启代码全局搜索找一下是否有“System.gc()”，结果没想到还真的找到了!

所以致命的问题就在这里，当时这个同学写这行代码想的特别的天真和可爱，出发点也是好的

他是这么考虑的，大家可以看看他当时的思路：他在代码里经常会一下子加载出来一大批数据，一旦请求处理完毕之后，他就觉得，一大批数据就废弃不用了，占据内存太多了，完全可以主动用“System.gc()”代码触发一次GC，把他们回收掉!

结果呢，平时系统运行时，访问量很低，基本还不会出大乱子！但是在大促活动的时候，访问量一高，立马由“System.gc()”代码频繁触发了Full GC，导致了这个系统直接被卡死了!

原来关键点就在于这里!

这个案例很短，也很简单，发在周末，让大家轻松看一看，不太烧脑，这周之前的几个案例，都略微有点烧脑，而且都在4000字以上，甚至需要大家看两遍才能理解其中的内容和精髓

这篇文章轻松一点，就带给大家一个小的知识点，相信大家看着也很容易。

所以，针对这个问题，一方面大家平时写代码的时候，不要自己使用“System.gc()”去随便触发GC，一方面可以在JVM参数中加入这个参数：-XX:+DisableExplicitGC。这个参数的意思就是禁止显式执行GC，不允许你来通过代码触发GC。

所以推荐大家将“**-XX:+DisableExplicitGC**”参数加入到自己的系统的JVM参数中，或者是加入到公司的JVM参数模板中去。避免有的开发工程师好心办坏事，代码中频繁触发GC就不好了。

End