图文 075、对象太多了！堆内存实在是放不下，只能内存溢出！

详情 评论

对象太多了！堆内存实在是放不下，只能内存溢出！

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

之前的文章已经分析了Metaspace和栈内存两块内存区域发生内存溢出的原理，同时给出了一些较为常见的引发他们内存溢出的场景，一般只要代码上注意一些，不太容易引发那两块区域的内存溢出。

本周的重点要来了，真正最容易引发内存溢出的，说白了就是平时我们系统创建出来的对象实在是太多了，最终就导致了系统的内存溢出！

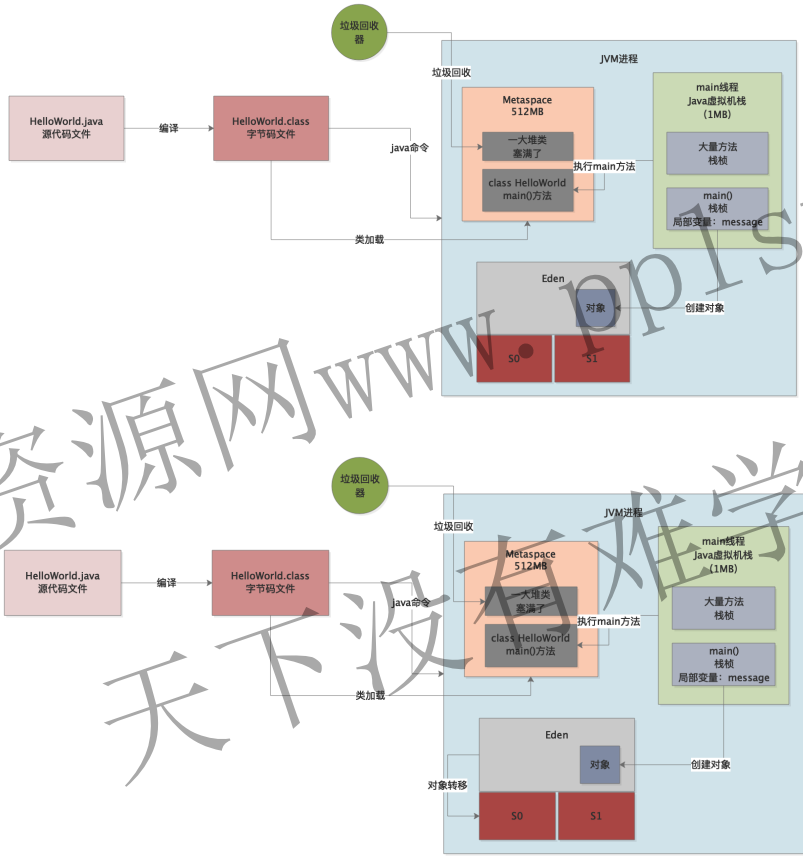
2、从对象在Eden区分配开始讲起

如果要把这大量的对象是如何导致堆内存溢出的给讲清楚，那就得从系统运行，在Eden区创建对象开始讲起了。

咱们都知道，平时系统运行的时候一直不停的创建对象，然后大量的对象会填满Eden区

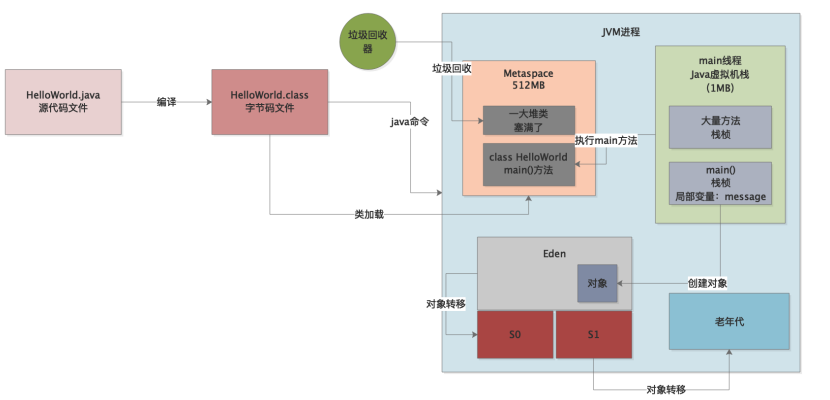
一旦Eden区满之后，就会触发一次Young GC，然后存活对象进入S区。

如下图所示

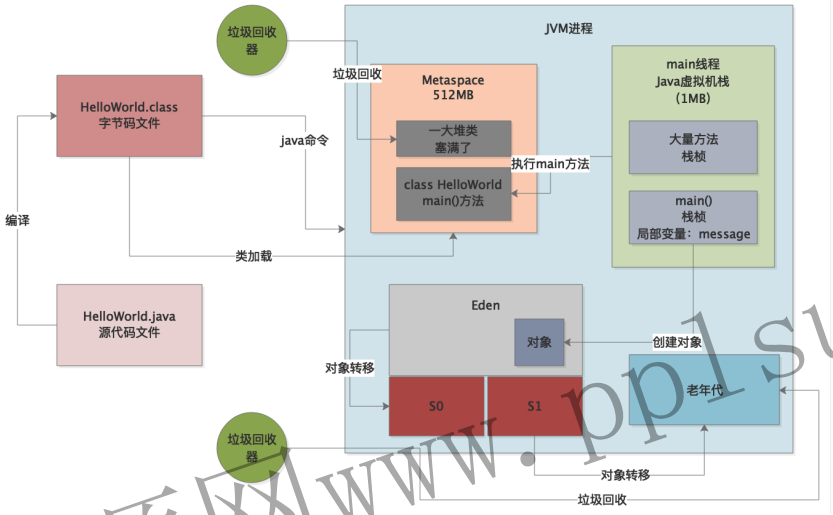


3、高并发场景下导致ygc后存活对象太多

当然因为各种各样的情况，一旦出现了高并发场景，导致ygc后很多请求还没处理完毕，存活对象太多，可能就在Survivor区域放不下了，此时就只能进入到老年代里去了，老年代很快就会放满了，如下图所示。

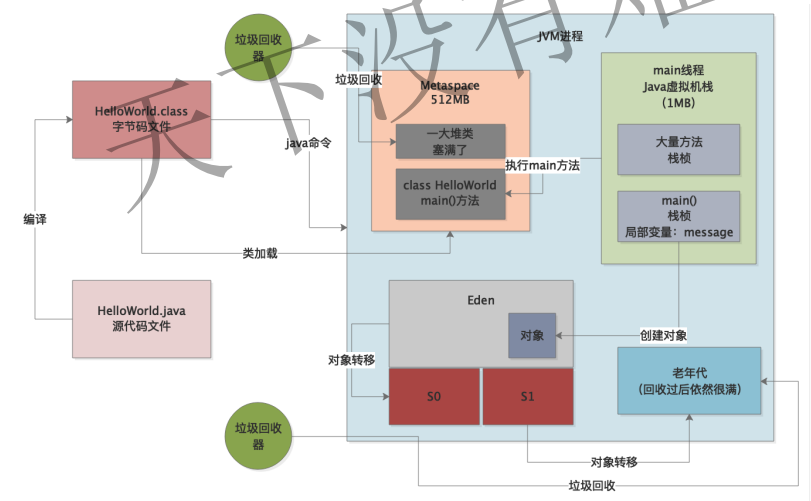


一旦老年代放满了就会触发Full GC，如下图所示。



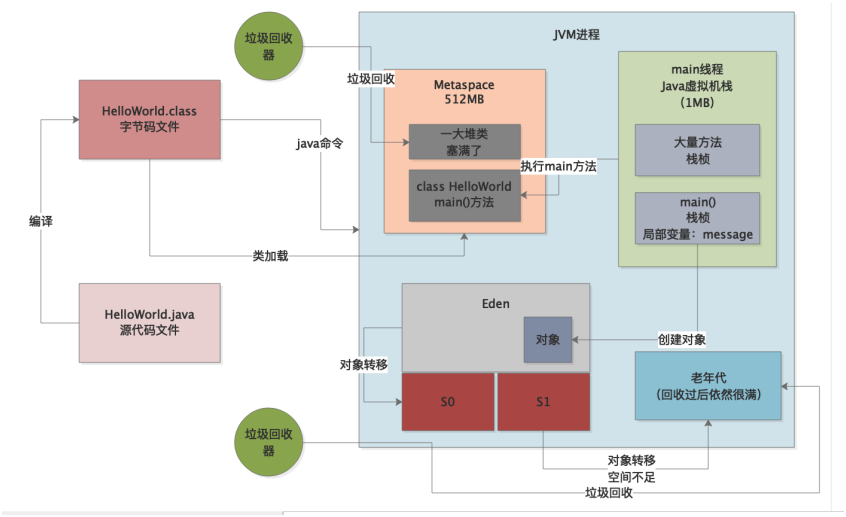
我们假设ygc过后有一批存活对象，Survivor放不了，此时就等着要进入老年代里，然后老年代也满了，那么就得等着老年代进行CMS GC，必须回收掉一批对象，才能让年轻代里存活下来的一批对象进去。

但是呢，不幸的事情发生了，老年代GC过后，依然存活下来了很多的对象！如下图所示。



这个时候如果年轻代还有一批对象等着放进老年代，人家GC过后空间还是不足怎么办？

还能怎么办！只能是内存溢出了！如下图所示！



所以这个时候，老年代都已经塞满了，你还要往里面放东西，而且触发了Full GC回收了老年代还是没有足够内存空间，你坚持要放？那只能给你一个内存溢出的异常了！JVM跑不动了，崩溃掉。

这个就是典型的堆内存实在放不下过多对象的内存溢出的一个典型范例。

4、什么时候会发生堆内存的溢出?

发生堆内存溢出的原因其实总结下来，就一句话：

有限的内存中放了过多的对象，而且大多数都是存活的，此时即使GC过后还是大部分都存活，所以要继续放入更多对象已经不可能了，此时只能引发内存溢出问题。

所以一般来说发生内存溢出有两种主要的场景：

系统承载高并发请求，因为请求量过大，导致大量对象都是存活的，所以要继续放入新的对象实在是不行了，此时就会引发OOM系统崩溃

系统有内存泄漏的问题，就是莫名其妙弄了很多的对象，结果对象都是存活的，没有及时取消对他们的引用，导致触发GC还是无法回收，此时只能引发内存溢出，因为内存实在放不下更多对象了

因此总结起来，一般引发OOM，要不然是系统负载过高，要不然就是有内存泄漏的问题

这个OOM问题，一旦你的代码写的不太好，或者设计有缺陷，还是比较容易引发的，所以这个问题也是我们后面要重点分析的。

5、本文总结

今天的文章我们分析了发生堆内存OOM的根本原因，即对象太多，且都是存活的，即使GC过后还是没有空间了，此时放不下新的对象，JVM只能选择崩溃！

希望大家好好理解本周对各种内存溢出问题的本质原理分析。

接下来我们就会用代码模拟出来各种内存溢出的场景，同时结合一些真实的案例，让大家去体验一下OOM的感觉，敬请期待

End
