图文 052、从测试到上线：如何分析JVM运行状况及合理优化?

详情 评论

从测试到上线：

如何分析JVM运行状况及合理优化?

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

前面两篇文章，已经给大家介绍了jstat、jmap、jhat等工具，可以非常轻松的分析出系统运行时的JVM状况，包括内存使用压力还有GC压力，包括内存中的对象分布情况。

这篇文章，我们结合之前介绍过的两个工具，给大家做一个实际开发、测试到上线的一个整体JVM优化的梳理。

2、开发好系统之后的预估性优化

大家平时如果在开发一个新系统的时候，完成开发之后，是不是就要经历测试以及上线的过程?

此时在系统开发完毕之后，实际上各位同学就应该参照之前我们多个案例中介绍的思路，对系统进行预估性的优化。

那什么叫做预估性的优化呢?

就是跟之前案例中说的一样，自行估算系统每秒大概多少请求，每个请求会创建多少对象，占用多少内存，机器应该选用什么样的配置，年轻代应该给多少内存，Young GC触发的频率，对象进入老年代的速率，老年代应该给多少内存，Full GC触发的频率。

这些东西其实是可以根据你自己写的代码，大致合理的预估一下的。

在预估完成之后，就可以采用之前多个案例介绍的优化思路，先给自己的系统设置一些初始性的JVM参数

比如堆内存大小，年轻代大小，Eden和Survivor的比例，老年代的大小，大对象的阈值，大龄对象进入老年代的阈值，等等。

优化思路其实简单来说就一句话：**尽量让每次Young GC后的存活对象小于Survivor区域的50%，都留存在年轻代里。尽量别让对象进入老年代。尽量减少Full GC的频率，避免频繁Full GC对JVM性能的影响。**

这个之前几周内容都在围绕这个核心在讲述，相信大家现在都理解的很清楚了。实际上这个过程应该是在新系统开发完毕之后必须有的一个环节。

3、系统压测时的JVM优化

通常一个新系统开发完毕之后，就会经过一连串的测试

从本地的单元测试，到系统集成测试，再到测试环境的功能测试，预发布环境的压力测试，要保证系统的功能全部正常

而且在一定压力下性能、稳定性和并发能力都正常，最后才会部署到生产环境运行。

这里非常关键的一个环节就是预发布环境的压力测试，通常在这个环节，会使用一些压力测试工具模拟比如1000个用户同时访问系统，造成每秒500个请求的压力，然后看系统能否支撑住每秒500请求的压力。同时看系统各个接口的响应延时是否在比如200ms之内，也就是接口性能不能太慢，或者是在数据库中模拟出来百万级单表数据，然后看系统是否还能稳定运行。

具体如何进行系统压测，不是我们这里要讲述的内容，大家自行百度一下“Java压力测试”，就会看到很多开源的工具，可以轻松模拟出N个用户同时访问你系统的场景，还能给你一份压力测试报告，告诉你系统可以支撑每秒多少请求，包括系统接口的响应延时。

在这个环节，通常压测工具会对系统发起持续不断的请求，持续很长时间，比如几个小时，甚至几天时间。

所以此时，大家完全就可以在这个环节，对测试机器运行的系统，采用jstat工具来分析在模拟真实环境的压力下，JVM的整体运行状态。

具体如何使用jstat来进行分析，之前都讲的很详细了，包括如何借助jstat的各种功能分析出来以下JVM的关键运行指标：新生代对象增长的速率，Young GC的触发频率，Young GC的耗时，每次Young GC后有多少对象是存活下来的，每次Young GC过后有多少对象进入了老年代，老年代对象增长的速率，Full GC的触发频率，Full GC的耗时。

然后根据压测环境中的JVM运行状况，如果发现对象过快进入老年代，可能是因为年轻代太小导致频繁Young GC，然后Young GC的时候很多对象还是存活的，结果Survivor也太小，导致很多对象频繁进入老年代。当然也可能是别的什么原因。

此时就需要采用之前介绍的优化思路，合理调整新生代、老年代、Eden、Survivor各个区域的内存大小，保证对象尽量留在年轻代，不要过快进入老年代中。

之前很多人网上会胡乱搜索JMV优化的博客，看到里面人家怎么优化，你就怎么优化

比如很多博客说年轻代和老年代的占比一般是3:8，其实完全是片面的。每个系统都是不一样的，特点不同，复杂度不同。

大家记住一点：真正的优化，必须是你根据自己的系统，实际观察之后，然后合理调整内存分布，根本没什么固定的JVM优化模板。

当你对压测环境下的系统优化好JVM参数之后，观察Young GC和Full GC频率都很低，此时就可以部署系统上线了。

4、对线上系统进行JVM监控

当你的系统上线之后，你就需要对线上系统的JVM进行监控，这个监控通常来说有两种办法。

第一种方法会“low”一些，其实就是每天在高峰期和低峰期都用jstat、jmap、jhat等工具去看看线上系统的JVM运行是否正常，有没有频繁Full GC的问题。

如果有就优化，没有的话，平时每天都定时去看看，或者每周都去看看即可。

第二种方法在中大型公司里会多一些，大家都知道，很多中大型公司都会部署专门的监控系统，比较常见的有Zabbix、OpenFalcon、Ganglia，等等。

然后你部署的系统都可以把JVM统计项发送到这些监控系统里去。

此时你就可以在这些监控系统可视化的界面里，看到你需要的所有指标，包括你的各个内存区域的对象占用变化曲线，直接可以看到Eden区的对象增速，还会告诉你Young GC发生的频率以及耗时，包括老年代的对象增速以及Full GC的频率和耗时。

而且这些工具还允许你设置监控。也就是说，你可以指定一个监控规则，比如线上系统的JVM，如果10分钟之内发生5次以上Full GC，就需要发送报警给你。比如发送到你的邮箱、短信里，这样你就不用自己每天去看着了。

但是这些监控工具的使用不在我们专栏范畴里，因为这些内容并不一定每个公司都一样，也不一定每个公司都有

大家如果有兴趣，完全可以自行百度学习，比如“OpenFalcon监控JVM”，会看到很多资料。

对于我们而言，主要会带大家使用的就是JDK自身提供的命令行工具，包括jstat、jmap和jhat

其实把这些命令行用好了，基本线上系统的JVM监控和优化都能搞定了。而且我本人而言，还是非常推崇工程师平时除了要会用图形化工具，还必须得熟练使用命令行的工具，这才像一个“工程师”应该有的样子。

简单一句话总结：对线上运行的系统，要不然用命令行工具手动监控，发现问题就优化，要不然就是依托公司的监控系统进行自动监控，可视化查看日常系统的运行状态。

5、今日思考题

你们公司的系统开发流程里有压测环节吗?

如果有，你能否在自己的工作流程中加入一项，在开发之后先进行预估性JVM优化，然后再在压测环境进行测试性JVM优化。

此外，你们线上生产环境有没有JVM监控方案?

如果没有，建议在工作内容中引入一项，每天日常工作可以在固定时间段去线上机器里用命令行工具观察一下JVM运行状态，作为日常线上系统巡查的一个工作内容。

如果你们公司有类似Zabbix、OpenFalcon之类的监控系统，是否对线上系统都指定了JVM GC监控？如果JVM发生频繁Full GC能否及时通知到你?

专栏学习到这里，大家完全可以对自己负责的生产系统“**下手**”了。引入规范化、流程化的JVM监控和优化的工作内容，保障自己负责的系统的JVM性能是绝对良好的。

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）