图文 091、本周答疑问题汇总！

218 人次阅读 2019-10-13 07:00:00

详情 评论

本周答疑汇总

问题：

请问老师，文中 [Metaspace: 9201->9201(1058816K)] 括号中的1058816K是什么意思？也不是10M啊？

回答：那个一般不用管他，他主要根据你设置的参数触发垃圾回收

问题：

老师 我默认得 新生代 和 堆内存大小和你得不一致呢， 不过通过配置参数模拟成功了。，，不过还是想问问，这是eclips配置不同得原因吗

回答：是的，ide开发工具会有一定的影响

问题：

元数据区在1.8版本已经移到了本地内存，此时还会有gc?

回答：还是会的

问题：

如果元数据空间存放类的信息满了，然后没有进行垃圾回收的情况下， 元数据空间就会产生oom吗?

狸猫技术

进店逛

相关频道

从 0 开
战高手
已更新1

回答：是的

问题：

有个疑问，就是g1将以前gc区域化整为零，这样时间可控，更精细化，但region回收算法还是按照之前的吗？

回答：回收每个region的时候是按照复制算法来的

问题：

一个新的小对象（不会因为过大而分配到老年代的对象）是只分配到eden，还是会分配到eden+一个survivor钟

回答：仅仅分配到eden区

问题：

老师好，这课太棒了！！！这钱花得值！！！ 冒昧请教个问题，有点糊涂，就是您004课图中“内存区域运行方法使用”这块内存区域和“JAVA虚拟机栈”区别是啥呢？

您说“内存区域运行方法使用”这个内存区域是用来保存方法中的变量的，但是“JAVA虚拟机栈”不也是保存方法中变量的吗？

比如我有个m1方法，里面有个int a = 1,那么这个a是在“内存区域运行方法使用”和执行m1方法的线程所属栈帧里各存一份儿？ 这两块关系是啥呢？欺盼您的回复，谢谢？

回答：多谢支持，其实这两个东西就是一个意思

问题：

老师好。这里有个小问题，文中说基于动态年龄判断，年龄n以及之上的对象进入老年代，我理解此时最大也就是n了呢，为啥还有及以上？

回答：举个例子，年龄为1234的对象加起来就50%了，那么年龄5678的不是也会可能存在吗？

问题：

老师，有个疑问。在Minor GC之前一通检查时，假如发生了Full GC，Full GC已经包含了Minor GC，那么Full GC之后是否还会单独再执行一次Minor GC？

回答：那样一般就不会了

问题：

为什么通过cglib创建的这些代理类不能被回收？

回答：因为一直在cglib内部被引用了

问题：

老师，eden区满了，s1没有满，新来一个对象，是出发gc还是分配到s1

回答：eden区满，直接触发young gc了

问题：

老师，我在idea上执行总是会多几次分配失败的日志，这个有办法忽略吗？有时候分析会造成很大的干扰

回答：这个没事的，每个人不同的jdk版本以及idea的环境，会导致实验结果有差异，你主要关注背后的本质就可以了

问题：

第一次young gc后发现触发了动态年龄担保规则不是就会直接进入老年代了么，为啥是第二次gc后才进去的

回答：这是jvm的运行管理，第一次不会直接触发动态年龄判定，要到第二次才会触发

问题：

老师，我的jdk是1.8.0_131 版本，window环境，1.126: [GC (Allocation Failure) 1.126: [ParNew: 3326K->512K(4608K), 0.0021687 secs] 3326K->988K(9728K), 0.0023565 secs] [Times: user=0.00 sys=0.00, real=0.02 secs] 1.146: [GC (Allocation Failure) 1.146: [ParNew: 3662K->0K(4608K), 0.0012292 secs] 4138K->985K(9728K), 0.0013099 secs] [Times: user=0.06 sys=0.00, real=0.00 secs] 实验结果出现了两次gc，这个和jdk版本有关吗？

回答：是的，跟jdk版本有关系，不过没事，你重点关注本质的运行原理就可以

问题：

为什么在处理请求时会创建2个数组，这个没有太明白

回答：这个就得分析tomcat源码了，就没必要了，你只要在这里知道他对每个请求tomcat内核就是会创建2个数组存放请求和响应就可以了

问题：

需要做资源隔离（超时熔断处理，配合请求队列异步，只给几个固定的线程资源），很经典的微服务问题???

回答：是的，这个是很经典的一个问题

问题：

永久代OOM的时候异常是什么?

回答：Metaspace相关的异常

问题：

那么生产上到底要不要禁止System.gc（）？

回答：原则上要禁止，但是这个案例中特殊情况，Java NIO需要用System.gc()，所以只能放开了，只要保证你代码里别写System.gc()就可以了

问题：

线上这样的问题应该避免，rpc调用都有熔断降级的，下游系统故障不应该影响上游系统业务，否则就会引发雪崩事故。

回答：对的，就是这样的，一般应该做熔断，但是不是每个系统都会上来就做熔断的

问题：

很好奇，8g的dump是怎么打开的

回答：之前讲解MAT的时候其实说过的，你得把MAT的启动参数调整为8g，就可以本地打开8g的内存快照了

问题：

DirectByteBuffer在堆内存是引用对象吧？真正内容在堆外内存，这样堆内可以新建很多引用对象。写代码的时候要预估好，并手动释放

回答：是的

问题：

谢谢老师对于压测环境的回复，在预发布只压测一台机器得到的结果，然后做线性关联得出线上的最终效果吗？

这里仅对一台预发布机器进行压测对于最终发布后线上的性能表现有多大的参考价值，或者需要做一些什么样的措施来达到通过对预发布一台机器的压测得出最终线上的实际性能表现呢？不知道我说清楚没有，谢谢老师

回答：压测的时候一般就是一台机器，或者几台机器，然后直接线性预估线上的就可以了

问题：

老师，你之前说过复制算法是先把对象复制到另外一个空白区域，然后将这块区域回收，现在在进行Minor Gc之后发现存活区放不下存活的对象了

那么如果使用的是复制算法，在回收之前就会将存活对象放入存活区，此时jvm发现存活区放不下对象，会将存活的对象转移到老年代，我的问题是这个存活对象到底是回收之前转移到存活区还是回收之后转移到存活区

回答：回收之前，他要先把存活对象放入survivor区域，再回收eden区的垃圾对象

问题：

老师写的很好，我有个问题想问下，比如启动一个tomcat下的web应用，第一次访问页面的时候比较慢，后面就快多了，是不是就因为第一次访问时，程序中需要加载很多类的实例，这个时候这个类的.Class对象还没被加载到常量池中，堆内存中也没有写个对象，所以会关联很多类对象和实例对象，所以比较慢

第二次访问是类对象以经有了，如果是单机模式的话，那么堆内存中实例对象也创建好了，这时只需要把实例对象的引用给虚拟机栈，所以后续访问会非常快，这么理解对吗？

回答：是的，理解的基本没问题

问题：

我觉得应该是根据项目情况而定，大部分情况下，应该是要禁止掉的。 但是如果系统的类似那种文件存储系统，IM系统，大量用到NIO的场景下，就要开放System.gc() 。同时要求写代码的同学不要写System.gc() ;)

回答：是的，就是这个意思

问题：

很好奇一个问题，就是输出gc.log的时候，究竟会不会有性能问题？看日志也只有分配失败或者关键事件才会有日志输出

回答：一般是不会有性能问题的

问题：

如果一个tomcat部署两个应用，两个应用之间通过hessian调用，这两个应用就是两个进程吗？

回答：一个Tomcat就是一个进程，两个应用都在一个进程里

问题：

老师，请问你这里讲的都是基于JDK8来讲的吧，JDK8以前的JVM，堆是包含永久带，而从8开始永久带变成元数据区，并且不在堆范围内？ 另外，JDK8开始，一个什么native memory规避了内存泄漏？这个是怎么理解，谢谢

回答：都是JDK 8来讲的，其实你说的堆外内存泄漏，可以看后面的案例，后面有案例讲了

问题：

老师为什么我MAT工具分析出来的跟您不一样啊？ 我看到的是这个，并不是Demo1里面动态生成的类 java.lang.String @ 0x6e9a08a58 org.springframework.cglib.core.AbstractClassGenerator$ClassLoaderData$1

回答：这个是JDK自己的类，你没看到其他的类吗？jvm参数设置呢？

问题：

老师为什么我的是这样的 main at java.lang.OutOfMemoryError.<init>()V (OutOfMemoryError.java:48) at java.util.Arrays.copyOfRange([CII)[C (Arrays.java:3664) at java.lang.String.<init>([CII)V (String.java:201) at java.lang.StringBuilder.toString()Ljava/lang/String; (StringBuilder.java:407) at com.zhss.demo.zuul.gateway.jvm.内存泄漏.SpaceDemo.main([Ljava/lang/String;)V (SpaceDemo.java:18)

回答：这个也没问题，显示出了内存溢出的异常了

问题：老师，你们查看接口的调用耗时是怎么做的？

回答：这个一般可以用一些监控系统去做

问题：

protobuf协议里Request类的定义，如果是B定义，A有需求再通知B改，然后jar版本更新一下，今天的问题应该不会出现了吧

回答：是的，如果更新了jar就不会有问题了

问题：

老师如何知道tomcat现在有多少线程在处理 如果没有监控如何知道现在每秒多少请求

回答：之前的文章里其实讲过的，可以自己写一个简单的metrics统计框架，给一些接口加入注解，然后自动统计每个接口每秒钟内的访问

问题：

老年代不是不超过40%吗，那怎么会有超过45%的时候？

回答：年轻代和老年代的占比是动态变化的，老年代最多是可以超过40%的

问题：

如果4M的数组存满了会发生什么情况？

回答：那就只能丢弃部分数据了

问题：

如果minor gc和 full gc 都是遍历线程栈和方法区的gc roots去追踪存活对象，那么full gc的时候岂不是做了minor gc的一些活，清理了一些年轻代的短暂对象？这与full GC只负责老年代有点冲突了，还是说在遍历GC roots的时候是可以区分新生代还是老年代的？

回答：gc roots可能会遍历到年轻代和老年代，但是回收老年代的时候只会回收老年代里的垃圾对象，遍历和回收是两回事儿

问题：

请问一下线上的JVM监控是怎样实现的, 是每秒执行jstat然后进行分析和聚合得到结果在输出的方式吗？

回答：

这个你可以自己做，比如用jstat + shell脚本，自己写一个，也可以用现成的监控系统，比如zabbix，都是开源的，可以去看一下

问题：

请问一下我看Protobuf的文档上说Protobuf是对变化友好的, 也就是如果A这边的Request变更了,B那边还是原来的, 那在做反序列化的时候, 就只对B这边有的字段进行反序列化, 不会处理B不知道的字段, 这样看的话好像不会出本文的问题, 请问是这样吗?

回答：

这个还不完全如此，因为我们哪怕反序列化成功了，但是在取用字段的时候会出问题报错。比如说，服务A的Request有一个字段和服务B的Request字段名不一样，那么会导致服务B的Request反序列化完了，取不到某个字段

问题：

这个问题我们之前公司遇到过，虽然加了where条件，但是因为数据很大，每次load出来还是有几百兆，而且这块数据随着时间还会增长（当然频率很低），当时系统没做缓存，JVM也没有调优，后续就没有后续了?

回答：

是的，数据量太大会进入老年代，导致频繁full gc的

问题：

我喜欢图文，如果视频，有的内容比较冗余，而且你还不敢快进。图文更有利于阅读，我就属于那种jvm理论还算丰富，但是没什么实践，希望老师的可能能让我收获很多，辛苦了！

回答：

加油，好好学习，一定会收获很多

问题：

如果你是在idea 里面直接跑出现的问题会跟老师有很大的差异，就运行空的main 方法都会出现gc 的情况。最好的方式去打成jia 包，然后用 java -XX: ... -classpath jar包 class 这种方式去运行，出来的结果是差不多的。

回答：

是的，自己用jar包运行

学员实践反馈：

然后这是我的测试效果，机器是 mac ,jdk 版本 1.8.0_201： CommandLine flags: -XX:InitialHeapSize=10485760 -XX:MaxHeapSize=10485760 -XX:MaxNewSize=5242880 -XX:NewSize=5242880 -XX:OldPLABSize=16 -XX:PretenureSizeThreshold=10485760 -XX:+PrintGC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:SurvivorRatio=8 -XX:+UseCompressedClassPointers -XX:+UseCompressedOops -XX:+UseConcMarkSweepGC -XX:+UseParNewGC 0.065: [GC (Allocation Failure) 0.065: [ParNew: 3596K->321K(4608K), 0.0005020 secs] 3596K->321K(9728K), 0.0005608 secs] [Times: user=0.00 sys=0.00, real=0.00 secs] Heap par new generation total 4608K, used 2451K [0x00000007bf600000, 0x00000007bfb00000, 0x00000007bfb00000) eden space 4096K, 52% used [0x00000007bf600000, 0x00000007bf814930, 0x00000007bfa00000) from space 512K, 62% used [0x00000007bfa80000, 0x00000007bfad06b8, 0x00000007bfb00000) to space 512K, 0% used [0x00000007bfa00000, 0x00000007bfa00000, 0x00000007bfa80000) concurrent mark-sweep generation total 5120K, used 0K [0x00000007bfb

回答：

非常好

学员实践反馈：

在没有最后放入最后一个2M 的对象之前，前面的3个 1M 的对像，没有超过eden 的大小，从日志中可以看出来，eden 4096K, 89% used 是可以匹配的。 CommandLine flags: -XX:InitialHeapSize=10485760 -XX:MaxHeapSize=10485760 -XX:MaxNewSize=5242880 -XX:NewSize=5242880 -XX:OldPLABSize=16 -XX:PretenureSizeThreshold=10485760 -XX:+PrintGC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:SurvivorRatio=8 -XX:+UseCompressedClassPointers -XX:+UseCompressedOops -XX:+UseConcMarkSweepGC -XX:+UseParNewGC Heap par new generation total 4608K, used 3678K [0x00000007bf600000, 0x00000007bfb00000, 0x00000007bfb00000) eden space 4096K, 89% used [0x00000007bf600000, 0x00000007bf997928, 0x00000007bfa00000) from space 512K, 0% used [0x00000007bfa00000, 0x00000007bfa00000, 0x00000007bfa80000) to space 512K, 0% used [0x00000007bfa80000, 0x00000007bfa80000, 0x00000007bfb00000) concurrent mark-sweep generation total 5120K, used 0K [0x00000007bfb00000, 0x00000007c0000000, 0x00000007c0000000) Metaspace used 2698K, capacity 4486K, committed 4864K, reserved

回答：非常好

问题：

为什么没有Parallel，难道很少用吗？

回答：

很少用，一般生产不用parallel回收器

Copyright © 2015-2019 深圳小鹅网络技术有限公司 All Rights Reserved. 粤ICP备15020529号

小鹅通提供技术支持