面试真题

【美团】讲一下你熟悉垃圾回收器？

垃圾回收器主要分为4类：

第一类 Serial GC。是单线程GC。

缺点：全程StopTheWorld，停顿时间长。

是JVM Client模式默认垃圾回收器。

第二类Parallel GC。是多线程GC。

缺点：停顿时间仍较长。

是JDK8默认垃圾回收器。

第三类 ConcurrentMarkSweep GC。四个阶段：初始标记、并发标记、重新标记、并发清除。

优点：最短停顿时间。

缺点1：内存碎片化，可能触发Serial Old做Full GC。

缺点2： CPU敏感，默认启动线程数 = (CPU数+3) / 4。

第四类 Garbage First GC。将堆划分为2048个区域，每个区域1~32MB，采用Mixed GC模式，通过-XX:MaxGCPauseMillis建立可预测停顿模型

优点1：压缩算法避免碎片

优点2：停顿时间可控

缺点：内存占用高，Remembered Set维护开销大  
JDK9+用Garbage First GC替代ConcurrentMarkSweep GC

如何优化JVM的垃圾回收？

1. 选择合适的垃圾回收器
2. 调整堆内存大小，避免频繁的Full GC  
   Full GC：JVM回收包括年轻代和老年代在内的整个堆内存，导致长停顿。
3. 调整年轻代和老年代的比例

【美团】新生代和老年代的占比有了解吗？

1. 新生代**占**堆内存的 1/3

* **Eden区**：
* **Survivor区 (S0和S1)**：存放Eden幸存对象，交替使用，避免频繁GC。

2. 老年代**占**堆内存的 2/3

新生代和老年代占比的调整

-XX:NewRatio=2

表示老年代占总堆内存的 **2/3**，新生代占 **1/3**。

如何根据应用场景调整新生代与老年代的比例？

对于短生命周期对象较多的应用（如Web应用），可以增加新生代的内存。

对于长生命周期对象较多的应用（如大数据处理），可以增加老年代的内存。

知识框架

一、JVM基础

* JVM的组成结构：解释JVM的内存模型，包括方法区、堆、栈、本地方法栈和程序计数器等。
* 类加载机制：类加载过程、类加载器（Bootstrap ClassLoader, Extension ClassLoader, System ClassLoader），双亲委派模型。
* 堆和栈的区别：堆和栈的作用、生命周期、内存管理等。
* JVM启动参数：常见的JVM启动参数（例如-Xmx, -Xms, -Xmn, -XX:PermSize等），它们的作用和设置方法。

二、内存管理与垃圾回收

* 垃圾回收（GC）机制：GC的工作原理，GC的类型（Minor GC, Major GC, Full GC）及触发条件。
* 垃圾回收算法：标记-清除、复制算法、标记-压缩等GC算法。
* 垃圾回收器：不同的垃圾回收器（Serial, Parallel, CMS, G1, ZGC, Shenandoah）的特点及使用场景。
* JVM的内存回收与内存泄漏：如何识别和解决内存泄漏问题，使用工具如VisualVM、jconsole、JProfiler进行内存分析。
* GC日志分析：如何解读GC日志，分析GC的影响及如何优化GC的性能。

三、性能调优

* JVM性能瓶颈：如何检测JVM性能瓶颈，使用工具如JProfiler、JVM自带的JVM工具（jstat、jmap、jstack等）分析性能。
* JVM调优的常见方法：如何根据具体的应用场景（如Web应用、高并发应用）进行JVM调优，例如调整堆内存大小、优化GC策略等。
* HotSpot优化：如何进行HotSpot虚拟机的调优，JIT编译优化，代码热替换等。

四、JVM与多线程

* JVM中的线程模型：线程的生命周期、状态转换等。
* JVM中的并发问题：如何避免内存可见性问题、竞态条件和死锁。
* JVM的锁优化：Synchronized关键字和锁优化（如偏向锁、轻量级锁、重量级锁等）。
* JVM如何管理线程：JVM如何管理线程的栈空间，线程的优先级和调度。

五、JVM与操作系统

* JVM与操作系统的关系：JVM如何与操作系统交互，包括内存分配、文件IO等。
* 操作系统对JVM性能的影响：操作系统的内存管理策略、进程调度对JVM性能的影响。

六、高级话题

* JVM的类卸载机制：如何理解和实现类卸载，类卸载的触发条件。
* 逃逸分析与锁消除：JVM中的逃逸分析、锁消除、内联等优化技术。
* JVM与网络通信：JVM在处理大规模分布式系统中的表现，JVM对高并发请求的优化。

七、常见面试题

* 内存溢出（OutOfMemoryError）的类型：不同类型的内存溢出（如Java Heap Space、PermGen Space、Direct Buffer Memory等）的原因及解决方法。
* JVM如何处理堆外内存：直接内存（Direct Memory）的使用、Native Memory Tracking（NMT）的启用。
* 如何进行JVM调优：如果你的应用程序性能很差，如何分析JVM并做调优？有哪些工具和方法可以用来优化JVM的性能？
* CMS和G1的区别：CMS和G1垃圾回收器的工作原理、优缺点，适用场景。
* JVM调优中的常见误区：例如，过多的堆内存设置、频繁Full GC等常见的调优错误。