面试真题

【美团】讲一下你熟悉垃圾回收器？

**1. Serial GC：**Serial GC使用单线程进行垃圾回收，适用于单核环境或资源受限的场景。它的回收过程会在应用线程停顿时进行，可能导致较长的GC停顿时间。

**2. Parallel GC：**Parallel GC使用多个线程并行进行垃圾回收，主要通过分代回收的方式优化回收过程。回收时，应用线程会暂停，称为Stop-the-World。

**3. CMS GC (Concurrent Mark-Sweep GC)：**CMS GC旨在减少垃圾回收的停顿时间，它将标记阶段与清理阶段并行进行，应用线程在此期间可以继续执行，减少了停顿时间。回收时会有多个线程并行执行，优先考虑响应时间。

**缺点**：可能出现碎片问题（Full GC的频率较高，特别是老年代的内存碎片化严重时）。

**适用场景**：对响应时间要求较高的应用，如Web服务器、实时系统。

**4. G1 GC (Garbage First GC)：**采用了区域划分（Region）的方式，不同的区域大小可以进行独立回收。在回收过程中，G1会尝试优先回收那些垃圾最多的区域。

**缺点**：较为复杂，调优较困难，且初期性能可能不如Parallel GC。

**适用场景**：大内存、大型Web应用、长时间运行的服务器等。

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回收整个堆内存（包括年轻代和老年代），导致较长的停顿。
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 (Young Generation)：**存放新创建的对象，**占**堆内存的 **1/3**

* **Eden区**：新对象大部分会在此分配。
* **Survivor区 (S0和S1)**：存放从Eden区中幸存下来的对象。两块Survivor区交替使用，避免频繁的GC。

**2. 老年代 (Old Generation)：**存放经过多次GC仍然存活对象，**占**堆内存的 **2/3**

新生代和老年代占比的调整

**-XX:NewRatio=<ratio>**：老年代：新生代。

比如设置为 -XX:NewRatio=2 会让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