1、你知道哪些或者你们线上使用什么 GC 策略？它有什么优势，适用于什么场景？

参考 触发 JVM 进行 Full GC 的情况及应对策略。

2、Java 类加载器包括几种？它们之间的父子关系是怎么样的？双亲委派机制是什么意思？有什么好处？

启动 Bootstrap 类加载、扩展 Extension 类加载、系统 System 类加载。

父子关系如下：

- 启动类加载器 ，由 C++ 实现，没有父类；
- 扩展类加载器，由 Java 语言实现，父类加载器为 null；
- 系统类加载器，由 Java 语言实现，父类加载器为扩展类加载器；
- 自定义类加载器，父类加载器肯定为 AppClassLoader。

双亲委派机制：类加载器收到类加载请求，自己不加载，向上委托给父类加载，父类加载不了，再自己加载。

优势避免 Java 核心 API 篡改。详细查看：深入理解 Java 类加载器(ClassLoader)

3、如何自定义一个类加载器？你使用过哪些或者你在什么场景下需要一个自定义的类加载器吗？

自定义类加载的意义：

1. 加载特定路径的 class 文件
2. 加载一个加密的网络 class 文件
3. 热部署加载 class 文件

4、堆内存设置的参数是什么？

-Xmx 设置堆的最大空间大小

-Xms 设置堆的最小空间大小

5、Perm Space 中保存什么数据？会引起 OutOfMemory 吗？

加载 class 文件。

会引起，出现异常可以设置 -XX:PermSize 的大小。JDK 1.8 后，字符串常量不存放在永久带，而是在堆内存中，JDK8 以后没有永久代概念，而是用元空间替代，元空间不存在虚拟机中，二是使用本地内存。

详细查看 Java8 内存模型—永久代(PermGen)和元空间(Metaspace)

6、做 GC 时，一个对象在内存各个 Space 中被移动的顺序是什么？

标记清除法，复制算法，标记整理、分代算法。
新生代一般采用复制算法 GC，老年代使用标记整理算法。
垃圾收集器：串行新生代收集器、串行老生代收集器、并行新生代收集器、并行老年代收集器。
CMS（Current Mark Sweep）收集器是一种以获取最短回收停顿时间为目标的收集器，它是一种并发收集器，采用的是 Mark-Sweep 算法。
详见 Java GC 机制。

7、你有没有遇到过 OutOfMemory 问题？你是怎么来处理这个问题的？处理 过程中有哪些收获？

permgen space、heap space 错误。

常见的原因

- 内存加载的数据量太大：一次性从数据库取太多数据；
- 集合类中有对对象的引用，使用后未清空，GC 不能进行回收；
- 代码中存在循环产生过多的重复对象；
- 启动参数堆内存值小。

详见 Java 内存溢出（java.lang.OutOfMemoryError）的常见情况和处理方式总结。

8、JDK 1.8 之后 Perm Space 有哪些变动? MetaSpace大小默认是无限的么? 还是你们会通过什么方式来指定大小?

JDK 1.8 后用元空间替代了 Perm Space；字符串常量存放到堆内存中。
MetaSpace 大小默认没有限制，一般根据系统内存的大小。JVM 会动态改变此值。
-XX:MetaspaceSize：分配给类元数据空间（以字节计）的初始大小（Oracle 逻辑存储上的初始高水位，the initial high-water-mark）。此值为估计值，MetaspaceSize 的值设置的过大会延长垃圾回收时间。垃圾回收过后，引起下一次垃圾回收的类元数据空间的大小可能会变大。
-XX:MaxMetaspaceSize：分配给类元数据空间的最大值，超过此值就会触发 Full GC，此值默认没有限制，但应取决于系统内存的大小。JVM 会动态地改变此值。

9、jstack 是干什么的? jstat 呢？如果线上程序周期性地出现卡顿，你怀疑可 能是 GC 导致的，你会怎么来排查这个问题？线程日志一般你会看其中的什么 部分？

jstack 用来查询 Java 进程的堆栈信息。

jvisualvm 监控内存泄露，跟踪垃圾回收、执行时内存、cpu 分析、线程分析。

详见 Java jvisualvm 简要说明，可参考 线上 FullGC 频繁的排查。

10、StackOverflow 异常有没有遇到过？一般你猜测会在什么情况下被触发？如何指定一个线程的堆栈大小？一般你们写多少？

栈内存溢出，一般由栈内存的局部变量过爆了，导致内存溢出。出现在递归方法，参数个数过多，递归过深，递归没有出口。