**一、Java相关**

* Arraylist与LinkedList默认空间是多少；  
  10、0
* Arraylist与LinkedList区别与各自的优势List 和 Map 区别；
* 谈谈HashMap，哈希表解决hash冲突的方法；
* 为什么要重写hashcode()和equals()以及他们之间的区别与关系；
* Object的hashcode()是怎么计算的？
* 若hashcode方法永远返回1或者一个常量会产生什么结果？
* Java Collections和Arrays的sort方法默认的排序方法是什么；
* 引用计数法与GC Root可达性分析法区别；
* 浅拷贝和深拷贝的区别；
* String s="abc"和String s=new String("abc")区别；
* HashSet方法里面的hashcode存在哪，如果重写equals不重写hashcode会怎么样？
* 反射的作用与实现原理；
* Java中的回调机制；
* 模板方法模式；
* 开闭原则说一下；
* 发布/订阅使用场景；
* KMP算法（一种改进的字符串匹配算法）；
* JMM里边的原子性、可见性、有序性是如何体现出来的，JMM中内存屏障是什么意思，

<https://blog.csdn.net/lx_Frolf/article/details/82686201>

**二、多线程**

* AtomicInteger底层实现原理；
* synchronized与ReentraLock哪个是公平锁；

<https://www.jianshu.com/p/f584799f1c77>

* CAS机制会出现什么问题；
* 用过并发包下边的哪些类；
* 一个线程连着调用start两次会出现什么情况？
* wait方法能不能被重写，wait能不能被中断；
* 线程池的实现？四种线程池？重要参数及原理？任务拒接策略有哪几种？

AbortPolicy

为java线程池默认的阻塞策略，不执行此任务，而且直接抛出一个运行时异常，切记ThreadPoolExecutor.execute需要try catch，否则程序会直接退出。

DiscardPolicy

直接抛弃，任务不执行，空方法

DiscardOldestPolicy

从队列里面抛弃head的一个任务，并再次execute 此task。

CallerRunsPolicy

在调用execute的线程里面执行此command，会阻塞入口

* 线程状态以及API怎么操作会发生这种转换；
* 常用的避免死锁方法；

**三、JVM**

* Minor GC与Full GC分别在什么时候发生？什么时候触发Full GC;

<https://blog.csdn.net/weixin_38750084/article/details/83280614>

<https://blog.csdn.net/weixin_39788856/article/details/80388002>

<https://blog.csdn.net/suifeng3051/article/details/48292193>

触发MinorGC(Young GC)

虚拟机在进行minorGC之前会判断老年代最大的可用连续空间是否大于新生代的所有对象总空间

    1、如果大于的话，直接执行minorGC

    2、如果小于，判断是否开启HandlerPromotionFailure，没有开启直接FullGC

    3、如果开启了HanlerPromotionFailure, JVM会判断老年代的最大连续内存空间是否大于历次晋升的大小，如果小于直接执行FullGC

    4、如果大于的话，执行minorGC

触发FullGC

老年代空间不足

如果创建一个大对象，Eden区域当中放不下这个大对象，会直接保存在老年代当中，如果老年代空间也不足，就会触发Full GC。为了避免这种情况，最好就是不要创建太大的对象。

持久代空间不足

如果有持久代空间的话，系统当中需要加载的类，调用的方法很多，同时持久代当中没有足够的空间，就出触发一次Full GC

YGC出现promotion failure

promotion failure发生在Young GC, 如果Survivor区当中存活对象的年龄达到了设定值，会就将Survivor区当中的对象拷贝到老年代，如果老年代的空间不足，就会发生promotion failure， 接下去就会发生Full GC.

统计YGC发生时晋升到老年代的平均总大小大于老年代的空闲空间

在发生YGC是会判断，是否安全，这里的安全指的是，当前老年代空间可以容纳YGC晋升的对象的平均大小，如果不安全，就不会执行YGC,转而执行Full GC。

显示调用System.gc

* GC收集器有哪些？CMS收集器与G1收集器的特点。

<http://www.cnblogs.com/rgever/p/9534857.html>

<https://blog.csdn.net/qq_34173549/article/details/81151715>

Java在什么时候会出现内存泄漏；  
内存溢出 out of memory，是指程序在申请内存时，没有足够的内存空间供其使用，出现out of memory；比如申请了一个integer,但给它存了long才能存下的数，那就是内存溢出。

内存泄露 memory leak，是指程序在申请内存后，无法释放已申请的内存空间，一次内存泄露危害可以忽略，但内存泄露堆积后果很严重，无论多少内存,迟早会被占光。

* Java中的大对象如何进行存储；
* rt.jar被什么类加载器加载，什么时间加载；
* 自己写的类被什么加载，什么时间加载；
* 自己写的两个不同的类是被同一个类加载器加载的吗？为什么？
* 为什么新生代内存需要有两个Survivor区？

<https://blog.csdn.net/antony9118/article/details/51425581>

**设置两个Survivor区最大的好处就是解决了碎片化**，，刚刚新建的对象在Eden中，经历一次Minor GC，Eden中的存活对象就会被移动到第一块survivor space S0，Eden被清空；等Eden区再满了，就再触发一次Minor GC，Eden和S0中的存活对象又会被复制送入第二块survivor space S1（**这个过程非常重要，因为这种复制算法保证了S1中来自S0和Eden两部分的存活对象占用连续的内存空间，避免了碎片化的发生**）。S0和Eden被清空，然后下一轮S0与S1交换角色，如此循环往复。如果对象的复制次数达到16次，该对象就会被送到老年代中

* 几种常用的内存调试工具：jmap、jstack、jconsole；
* 类加载的五个过程：加载、验证、准备、解析、初始化；
* G1停顿吗，CMS回收步骤，CMS为什么会停顿，停顿时间；
* 栈主要存的数据是什么，堆呢？
* 堆分为哪几块，比如说新生代老生代，那么新生代又分为什么？
* 软引用和弱引用的使用场景（软引用可以实现缓存，弱引用可以用来在回调函数中防止内存泄露）；

软引用：

描述一些有用但并不是必需的对象，在Java中用java.lang.ref.SoftReference类来表示。   
**用途**：对于软引用关联着的对象，只有在内存不足的时候JVM才会回收该对象。因此，这一点可以很好地用来解决OOM的问题，并且这个特性很适合用来实现缓存：比如网页缓存、图片缓存等。

弱引用

是用来描述非必需对象的，当JVM进行垃圾回收时，无论内存是否充足，都会回收被弱引用关联的对象（**与软引用的区别**）。在java中，用java.lang.ref.WeakReference类来表示

**四、数据库**

* 数据库索引，什么是全文索引，全文索引中的倒排索引是什么原理；  
  普通索引的结构主要以B+树和哈希索引为主，用于实现对字段中数据的精确查找，比如查找某个字段值等于给定值的记录，A=10这种查询，因此适合数值型字段和短文本字段。
* 全文索引是用于检索字段中是否包含或不包含指定的关键字，有点像搜索引擎的功能，其内部的索引结构采用的是与搜索引擎相同的[倒排索引](https://www.baidu.com/s?wd=%E5%80%92%E6%8E%92%E7%B4%A2%E5%BC%95&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)结构，其原理是对字段中的文本进行分词，然后为每一个出现的单词记录一个索引项，这个索引项中保存了所有出现过该单词的记录的信息，也就是说在索引中找到这个单词后，就知道哪些记录的字段中包含这个单词了。因此适合用大文本字段的查找。
* 数据库最佳左前缀原则是什么？

解释一下最左前缀原则： 2.当b+树的数据项是复合的数据结构，比如(name,age,sex)的时候，b+数是按照从左到右的顺序来建立搜索树的，比如当(张三,20,F)这样的数据来检索的时候，b+树会优先比较name来确定下一步的所搜方向，如果name相同再依次比较age和sex，最后得到检索的数据；但当(20,F)这样的没有name的数据来的时候，b+树就不知道下一步该查哪个节点，因为建立搜索树的时候name就是第一个比较因子，必须要先根据name来搜索才能知道下一步去哪里查询。比如当(张三,F)这样的数据来检索时，b+树可以用name来指定搜索方向，但下一个字段age的缺失，所以只能把名字等于张三的数据都找到，然后再匹配性别是F的数据了， 这个是非常重要的性质，即索引的最左匹配特性

* 数据库的三大范式；
* 悲观锁和乐观锁的原理和应用场景；
* 左连接、右连接、内连接、外连接、交叉连接、笛卡儿积等；
* 一般情况下数据库宕机了如何进行恢复（什么是Write Ahead Log机制，什么是Double Write机制，什么是Check Point）；
* 什么是redo日志、什么是undo日志；  
  <https://blog.csdn.net/yu757371316/article/details/81081669>
* 数据库中的隔离性是怎样实现的；原子性、一致性、持久性又是如何实现的；
* 什么是组合索引，组合索引什么时候会失效；
* 关系型数据库和非关系型数据库区别；

<https://blog.csdn.net/qq_31536117/article/details/78179646>

* 数据库死锁如何解决；
* MySQL并发情况下怎么解决（通过事务、隔离级别、锁）；
* MySQL中的MVCC机制是什么意思，根据具体场景，MVCC是否有问题；
* MySQL数据库的隔离级别，以及如何解决幻读；

**五、缓存服务器**

* Redis中zSet跳跃表问题；
* Redis的set的应用场合？
* Redis高级特性了解吗？
* Redis的pipeline有什么用处？
* Redis集群宕机如何处理，怎么样进行数据的迁移；
* Redis的集群方案；
* Redis原子操作怎么用比较好；
* Redis过期策略是怎么实现的呢？

**六、SSM相关**

* Spring中@Autowired和@Resource注解的区别？
* Spring声明一个 bean 如何对其进行个性化定制；
* MyBatis有什么优势；
* MyBatis如何做事务管理；

**七、操作系统**

* Linux静态链接和动态链接；
* 什么是IO多路复用模型（select、poll、epoll）；
* Linux中的grep管道用处？Linux的常用命令？
* 操作系统中虚拟地址、逻辑地址、线性地址、物理地址的概念及区别；
* 内存的页面置换算法；
* 内存的页面置换算法；
* 进程调度算法，操作系统是如何调度进程的；
* 父子进程、孤儿进程、僵死进程等概念；
* fork进程时的操作；
* kill用法，某个进程杀不掉的原因（僵死进程；进入内核态，忽略kill信号）；
* 系统管理命令（如查看内存使用、网络情况）；
* find命令、awk使用；
* Linux下排查某个死循环的线程；

**八、网络相关**

* 数据链路层是做什么的?
* 数据链路层的流量控制？
* 网络模型的分层、IP和Mac地址在那个层、TCP和HTTP分别在那个层；
* TCP滑动窗口；
* TCP为什么可靠；
* TCP的同传，拆包与组装包是什么意思；
* Https和Http有什么区别；
* Http 为什么是无状态的；
* TCP三次握手，为什么不是三次，为什么不是四次；
* TCP的拥塞控制、流量控制详细说明？
* Http1.0和Http2.0的区别；
* 两个不同ip地址的计算机之间如何通信；
* 地址解析协议ARP；
* OSI七层模型分别对应着五层模型的哪一部分；
* TCP三次握手数据丢失了怎么办？那如果后面又找到了呢？

**九、分布式相关**

* 消息队列使用的场景介绍和作用（应用耦合、异步消息、流量削锋等）；
* 如何解决消息队列丢失消息和重复消费问题；
* Kafka使用过吗，什么是幂等性？怎么保证一致性，持久化怎么做，分区partition的理解，LEO是什么意思，如何保证多个partition之间数据一致性的（ISR机制），为什么Kafka可以这么快（基于磁盘的顺序读写）；
* 异步队列怎么实现；
* 你项目的并发是多少？怎么解决高并发问题？单机情况下Tomcat的并发大概是多少，MySQL的并发大致是多少？
* 什么是C10K问题；
* 高并发情况下怎么办；
* 分布式理论，什么是CAP理论，什么是Base理论，什么是Paxos理论；
* 分布式协议的选举算法；
* 说一下你对微服务的理解，与SOA的区别；
* Dubbo的基本原理，RPC，支持哪些通信方式，服务的调用过程；
* Dubbo如果有一个服务挂掉了怎么办；
* 分布式事务，操作两个表不在一个库，如何保证一致性。
* 分布式系统中，每台机器如何产生一个唯一的随机值；
* 系统的量级、pv、uv等；
* 什么是Hash一致性算法？分布式缓存的一致性，服务器如何扩容（哈希环）；
* 正向代理、反向代理；
* 什么是客户端负载均衡策略、什么是服务器端负载均衡策略；
* 如何优化Tomcat，常见的优化方式有哪些；
* Nginx的Master和Worker，Nginx是如何处理请求的；

**十、系统设计相关**

* 如何防止表单重复提交（Token令牌环等方式）；
* 有一个url白名单，需要使用正则表达式进行过滤，但是url量级很大，大概亿级，那么如何优化正则表达式？如何优化亿级的url匹配呢？
* 常见的Nginx负载均衡策略；已有两台Nginx服务器了，倘若这时候再增加一台服务器，采用什么负载均衡算法比较好？
* 扫描二维码登录的过程解析；
* 如何设计一个生成唯一UUID的算法？
* 实现一个负载均衡的算法，服务器资源分配为70%、20%、10%；
* 有三个线程T1 T2 T3，如何保证他们按顺序执行；
* 三个线程循环输出ABCABCABC....

**十一、安全相关**

* 什么是XSS攻击，XSS攻击的一般表现形式有哪些？如何防止XSS攻击；