**2019年11月22日面试题总结**

**2019/11/22**

# Java基础:

1. **Spring 是如何解决循环依赖的问题的？比如A autowired B，Bautowired A，内部是如何是如何实现的？**
2. **Spring IOC的理解，原理与实现；**
3. **设计模式：代理模式与装饰者模式的区别；常用的设计模式：单例模式、代理模式、装饰着模式等，单例模式主要解决什么问题；  
   单例模式的优点：  
   单例模式的缺点：**
4. **Spring boot何时启动结束；（58、）**
5. **Spring boot 启动类上的注解 @spring boot Application说明？**

**Spring boot如何判断当前应用是否时web应用？  
spring boot启动流程：  
1）构造SpringApplication实例，在构造过程中有判断当前应用的类型，代码如下：**

**this.webApplicationType = WebApplicationType.deduceFromClasspath();**

**2）、执行run方法：  
 （1）、创建并应用监听；**

**（2）、加载配置文件；**

**（3）、准备并刷新上下文；**

1. **Bean Factory与FactoryBean有什么区别？@Bean这个注解时如何实现Bean的注入的？  
   BeanFactory是一个工厂类的接口，是spring IOC的容器的核心接口，它的职责实例化、定位、配置应用程序中的对象即建立这些对象的依赖；**

**FactoryBean是一个Bean，它是实现了FactoryBean<T> 接口的bean，根据该Bean的Id从BeanFactory中获取的实际是BeanFactory的getObject返回的对象，而不是Factory Bean本身；即BeanFactory是用来获取对象的，Bean Factory是用来管理对象的。**

1. **Java中的静态方法只有一个实例，如果想用多个实例怎么办？**
2. **深度克隆的主要实现方式？**
3. **Java面向对象的基本特征，继承、封装与多态；**
4. **重写和重载的是什么意思？**
5. **怎样生命一个类不会被继承？final**
6. **Integer是基本数据类型还是引用数据类型？**
7. **HashMap中jdk1.7与jdk1.8的区别（58）；**

**二叉树-》平衡二叉树-》红黑树；**

1. **concurrenthashMap 的底层实现原理（58）；**
2. **spring、spring boot、springcloud之间的关系；**
3. **Java种的自增是线程安全的吗，如何实现线程安全的自增？**
4. **Jdk1.8种的stream有用过吗，stream的并行操作原理，stream并行的线程池是从哪里来的；**
5. **Jdk1.8的completableFuture有用过吗；**
6. **Java种的代理有几种实现方式；**

# Java 并发及JVM相关:

1. **线上部署时出现内存溢出该如何处理？**

1、使用top命令查看当前进程对应的内存大小信息，并使用shift+M按照内存大小进行排序，查看进程中内存占用较大的进程；  
2、利用jstat查看某一进程虚拟机的GC情况；  
3、生成dump文件，通过dump文件查看当时内存对象相关的信息；生成dump文件有两种方式：  
1）、使用jmap直接生成dump文件；  
jmap -dump:format=b,file=heap.prof  17561 ；  
format=b:表示生成二进制类型的dump文件；

file=：后面写的四输出的dump文件路径；

17561：jvm进程ID；

1. **Java多线程几种状态变更的图，图中包括线程所处的状态及线程各个状态之间是如何切换的？**
2. **Java中的wait和sleep的区别与联系？**
3. **Java.untils.concurrent包主要用到了哪些知识？  
   并发容器：  
   同步锁（两种）：  
   线程池：  
   并发工具(3种):**
4. **如何在方法栈中进行数据传递？  
   使用volatile关键字**
5. **ThreadLocal的底层实现形式及实现的数据结构？**
6. **Java有返回值的多线程使用？**
7. **Java中值类型和引用类型的区别（==与equals()的区别、Java中的引用类型：值引用与）；**
8. **Java中方法参数的传递时值传递还引用传递？**
9. **Java中堆和栈的区别？**
10. **JVM 内存划分与垃圾收集算法：fullGC与youngGC；**
11. **Java JVM相关内容结合项目介绍；**
12. **线上如何排查JVM内存溢出问题？  
    1.可以通过idea进行远程debugger；  
    2.相关命令jstat、jmap、jconsole等；  
    3.相关的工具，如阿里的阿尔萨斯等；**
13. **服务器CPU占用特别高，怎么查看（58），面试重点JVM及多线程；**
14. **Array List是线程安全的吗？**
15. **ConcurrentHash Map是如何实现线程安全的；**
16. **Sychornized是否是公平锁；**
17. **JVM为什么要增加元空间；**
18. **Javam如何查看问题，并进行jvm调优；**
19. **堆G1垃圾收集器有了解么，有什么特点；**

# Mysql面试题：

1. **Mysql MVCC、MySQL为什么选择B+树作为它的存储结构，为什么不选择Hash、二叉、红黑树（58）；**
   1. **如果仅仅是select \* from table where id =45,上面三种数据结构都可以实现，但是如果是select \* from table where id <45;则它们的查找方式就类似于全表扫描。B+树只在最末端叶子节点存数据，叶子节点是以链表形式相互指向的；**
   2. **为什么不选择Hash索引：Hash索引不适合范围查找，而B+树特别适合范围查找；Hash索引每次查询都需要加载所有的索引数据到内存中，而B+树只需要根据匹配规则选择对应的叶子数据加载即可；**
2. **Mysql如何写sql才能提高sql的执行效率；**
3. **Mysql数据库的事务与锁的理解；**
4. **数据库临时表有没有用过，是怎么用的？**
5. **添加索引的缺点？**
6. **Mysql的最左原则；**
7. **Mysql的锁：两条sql语句分别执行如下：  
   sql1：更新数据的ID between 10 and 20 ；  
   sql2：更新数据的ID为11；  
   那么这个锁是如何保证11这条数据不被更新的呢？**
8. **Mysql的sql优化  
   1、Mysql使用过程中走索引的语句；**

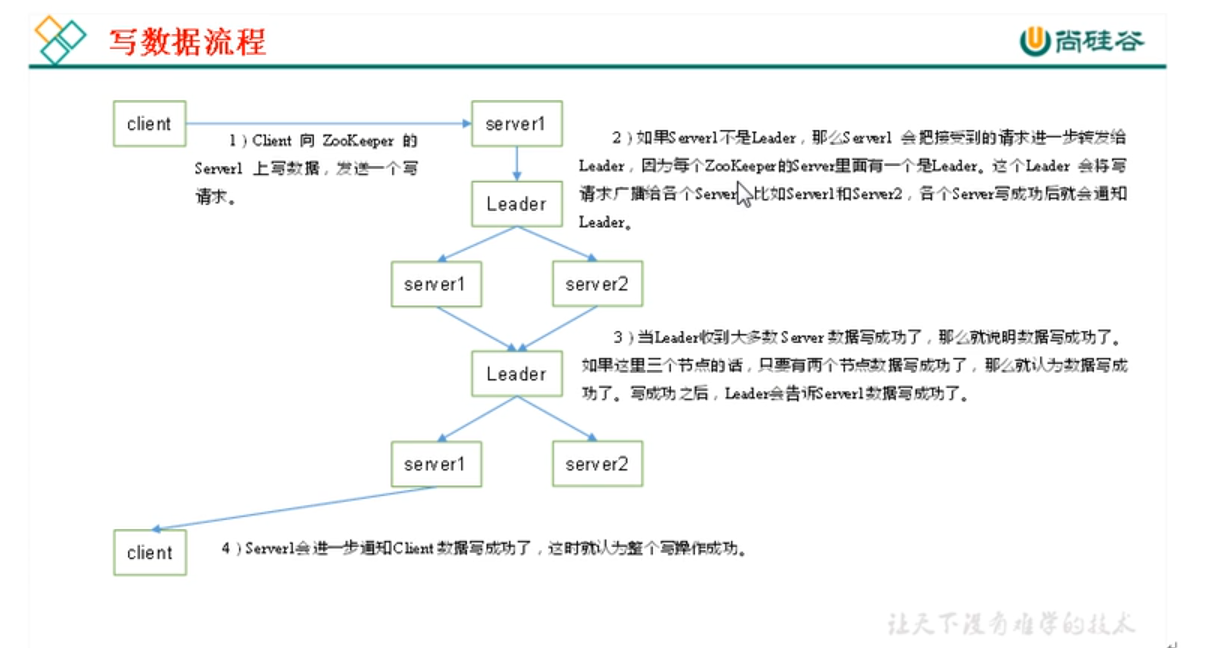
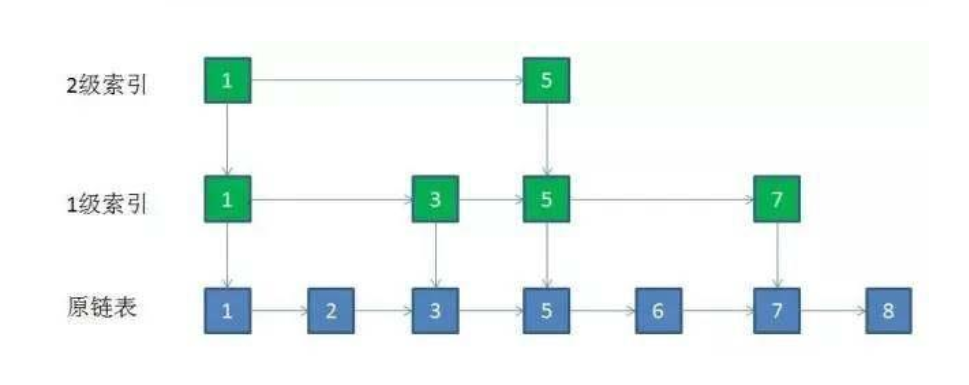
**2、索引的优化  
3、亿级的数据必须使用like进行查询，如何优化**

1. **多数据源情况下如何进行事物的管理；**
2. **Union和union all有什么区别；**
3. **dateTime和timestamp有什么区别；**
4. **mysql数据库事务的几种形式，模式事务级别，Oracle、SQL server默认事务级别；  
   读未提交；  
   读已提交；  
   不可重复读；  
   可序列化；**
5. **mysql主从模式的实现；**
6. **如何解析sql语句；即explain关键字的使用；**
   1. **Select\_type:查询类型，是简单查询还是复核查询；**
   2. **Type：对表的访问类型：All、index、range、ref、eq\_ref、const、system、null；从左到右，性能从差到好；**
   3. **Possible\_keys:指出mysql能使用按个索引在表中找到记录；**
   4. **Key：该列显示mysql实际使用的索引；**
   5. **Filtered：按表条件过滤行的百分比；**
   6. **Extra：执行情况的描述和说明；**
7. **为什么不对mysql的所有表都添加索引？**
8. **Mysql索引为什么不用hash而用B+树;Hash的时间复杂度是多少；**
9. **B+树索引可以进行范围查询吗，比如大于小于；**
10. **应用种mysql是集群还是单节点，最大表数据量是多少；**
11. **Mysql的主从同步原理，mysql主从复制主要有几种模式；**
    1. **Mysql主库在事务提交时会把数据库变更作为事件event记录在二进制文件binlog中；mysql主库上的sys\_binlog控制binlog日志刷新到磁盘；**
    2. **主库推送二进制文件binlog中的事件到从库的中继日志relay log，知乎从库根据中继日志重做数据库变更操作。通过逻辑复制，以此来达到数据一致。**
    3. **Mysql通过三个线程来完成主从库之间的数据复制：其中binLog Dump线程泡在主库上，I/O线程和SQL线程泡在从库上。当从库启动复制时，首先创建一个I/O线程连接主库，主库随后创建binlog dump线程读取数据库时间并发送给I/O线程，I/O线程获取到数据库事件更新到从库的中继日志realy log中，之后从库上的sql线程读取中继日志relay log中更新的数据库事件并应用；**

**Mysql主从复制主要有三种模式：**

* 1. **基于sql语句的复制（statement-based replication,SBR）;**
  2. **基于行的复制（row-based replication,RBR）;**
  3. **混合模式复制（mixed-based replication,MBR）;**

# Java框架相关：

1. **RabbitMQ的使用；**
2. **Zookeeper的原理，什么情况下会使用zookeeper，zookeeper如何监听生成的节点，zk内部是如何实现的；**
3. **Springcloud eureka是如何注册服务、如何监测心跳的，它注册的流程是怎么样的；**
4. **Redis在项目中是如何使用的；**
5. **在分布式环境中如何快速发现某一台服务有问题；**
6. **Mysql中如何设置一个字段为自增的；**
7. **分布式集群系统对外提供接口的时候如何验证对方的身份；**
8. **Spring boot整合jsp的流程，需要注意哪些点；**
9. **Eureka和zookeeper作为注册中心有什么区别（58）；**
   1. **CAP理论：一致性（consistency）、可用性（Aviailability）、分区容错性（Partition tolerance）三者只能满足其二；**
   2. **CA：不是分布式架构，就使用这种，关系数据库按照CA进行设计，放弃分区容错性，因为没有分区；**
   3. **AP：加强可用性和分区容错性，放弃立即一致性（强一致性），追求最终一致性。比如eureka；比如微信提现，提示两小时到账，而不是马上到账；**
   4. **CP：强调强一致性和分区容错性，放弃可用性，比如zookeeper，master在宕机后进行选举leader期间服务不饿能提供；比如夸张转账，就是立即到账，你这边转出，那边收进，才任务一个事务完成；**
   5. **由于ZK采用的时cp原则，所以其可用性降低，这是致命的问题，springcloud集成的eureka采用的是ap原则，牺牲了一致性，但是保证了可用性；**
10. **Zookeeper0、zookeeper1、zookeeper2，三个节点的集群描述一下从zk启动，到zk对外提供服务的整个过程（58）；**
    1. **Zookeeper集群leader的选举；**
       1. **服务器初始化阶段和服务器运行期间leader挂掉情况选举机制大体相同，此处仅对服务器初始化启动期间进行说明；**
       2. **先后启动两台服务器，启动后两台服务器进行通信，每台服务器都试图找到lead，于是进入leader选举过程，过程如下：**
       3. **每个server发出一个投票，由于初始概况，server1和server2都会将自己作为leader服务器来进行投票，每次投票都会包含所推举的服务器的myId和ZXID，使用（myid，ZXID）来表示，此时server1的投票为（1，0），server2的投票为（2，0），然后各自将这个投票发给集群中的其他机器。**
       4. **接受来自各个服务器的投票。集群的每个服务器收到投票后，首先判断该投票的有效性，如检查是否是本轮投票，是否来自lookiing状态的服务器。**
       5. **处理投票：针对每一个投票，服务器都需要将别人的投票和自己的投票进行PK，pk规则如下：  
          优先检查ZXID。ZXID比较大的服务器优先作为leader。  
          如果ZXID相同，那么就比较myid。Myid比较大的服务作为leader服务器。**
       6. **统计投票。每次投票后，服务器都会统计投票信息，判断是否已经有过半机器接受相同的投票信息，对于server1，server2而言，都统计出集群中已经有两台机器接受了（2，0）的投票信息，此时便认为已经选举出了leader；**
       7. **改变服务器状态。一旦确定了leader，每个服务器就会更新自己的状态，如果是follower，就会变成Followiing，如果是leader，就会变成Leading。**
    2. **Zookeeper数据同步；**
    3. **Zookeeper对外提供服务；**
11. **有一个key，往zk写入，到写入成功它的大体过程是什么样的（58）；**
    1. **此处有一个客户端client和zookeeper服务器；**
    2. **Client向zookeeper的某一个server上写数据，发送一个写请求；**
    3. **如果server1不是leader，那么server1会把接收到的请求进一步转发给leader，印美每个zookeeper集群里有一个是leader，这个leader会将写请求广播给各个server；各个server写陈工后就会通知leader；**
    4. **当leader收到大多数server（超过一半）数据写成功了，那么就说明数据写成功了，如果是三个节点的话，只要有两个节点数据写成功了，那么就默认数据写陈工l ，写成功之后，leader会告诉server1数据写成功了。**
    5. **Server1会进一步通知client数据写成功了，这时就认为整个写操作成功；**
12. **Zookeeper监听器的原理；**
    1. **首先要由一个main（）线程；**
    2. **在main线程中创建zookeeper客户端，这是就会创建两个线程，一个负责网络连接通信，一个负责监听；**
    3. **通过connect线程将注册的监听事件发送给zookeeper。**
    4. **在zookeeper的注册监听器列表中将注册的监听时间添加到列表中。**
    5. **Zookeeper监听到有数据或路径变化，就会将这个消息发送给listener线程。**
    6. **listener线程内部调用了process方法；**
13. **排行榜功能的实现：使用redis的zset；zset的底层数据结构是什么样的；除了redis的zset还有什么其他的数据结构可以实现这个功能；**
14. 
    1. **Redis的zset底层实现是跳跃表，跳跃表的底层实现是链表+索引（如上图），具体详情请点** [**https://www.jianshu.com/p/dc252b5efca6**](https://www.jianshu.com/p/dc252b5efca6)
15. **Sychronized和ReentryLock的区别（重点:底层实现的区别）;  
    区别：1、2、3、4：Sychronized底层是unsafe方法、ReentryLock底层是（AQS）；**
16. **Redis集群种类：主从模式、cluster模式及其应用；**
17. **Redis种数据类型及应用场景；**
18. **Redis的使用场景及存储格式；redis的持久化；RDB与AOF的设置，redis种哪个版本开始可以进行分布式的使用；**
    1. **Redis默认持久化方式为RDB：有三个策略：save 900 1\save 300 10\save 60 10000;**
    2. **AOF默认是关闭的；**
19. **Nginx如何做限流？令牌桶算法与漏桶算法；**
20. **Tomcat调优；**
21. **为什么使用RPC框架，什么时候会使用http请求；**
22. **什么场景下会使用MQ；MQ的优势与劣势，什么时候不能用MQ（MQ做不到同步）；**
23. **Springcloud组件介绍；**

# Linux服务器：

1. **Linux如何查看应用占用内存情况；**
2. **Linux脚本编写会么；**
3. **线上应用的debugger；  
   idea有插件，可以远程链接服务器的代码，并在本地debugger；  
   其他工具:pingpoint(分布式性能监测工具)；**
4. **服务器CPU数量及线程池数量的关系；  
   1、服务器cpu数量（n）；  
   2、是IO密集型（2n个线程）应用还是计算密集型（n+1个线程）应用；**

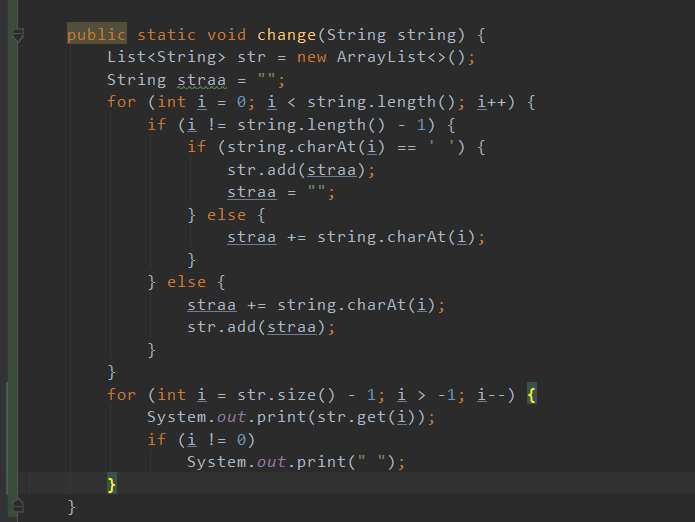
# 算法题:

1. **将一个矩阵顺时针旋转90°，请写出伪代码；  
   A11，A12，A13， A31，A21，A11，**

**A21，A22，A23， ——顺时针旋转90°——》 A32，A22，A22，**

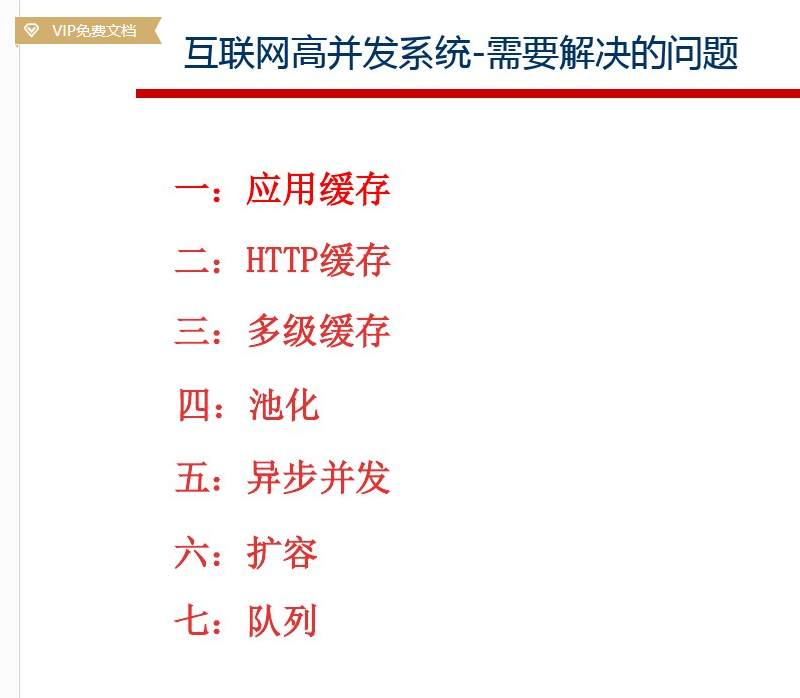
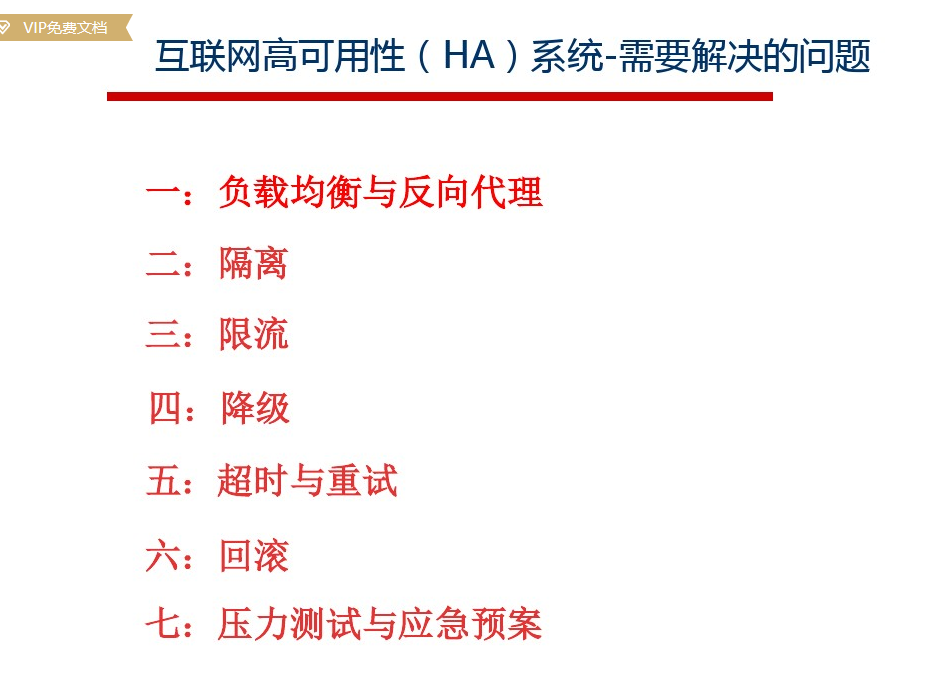
**A31，A32，A33， A33，A23，A13，  
(行,列)**

**第i行第j列——》第（j）行第（n-i）列；**

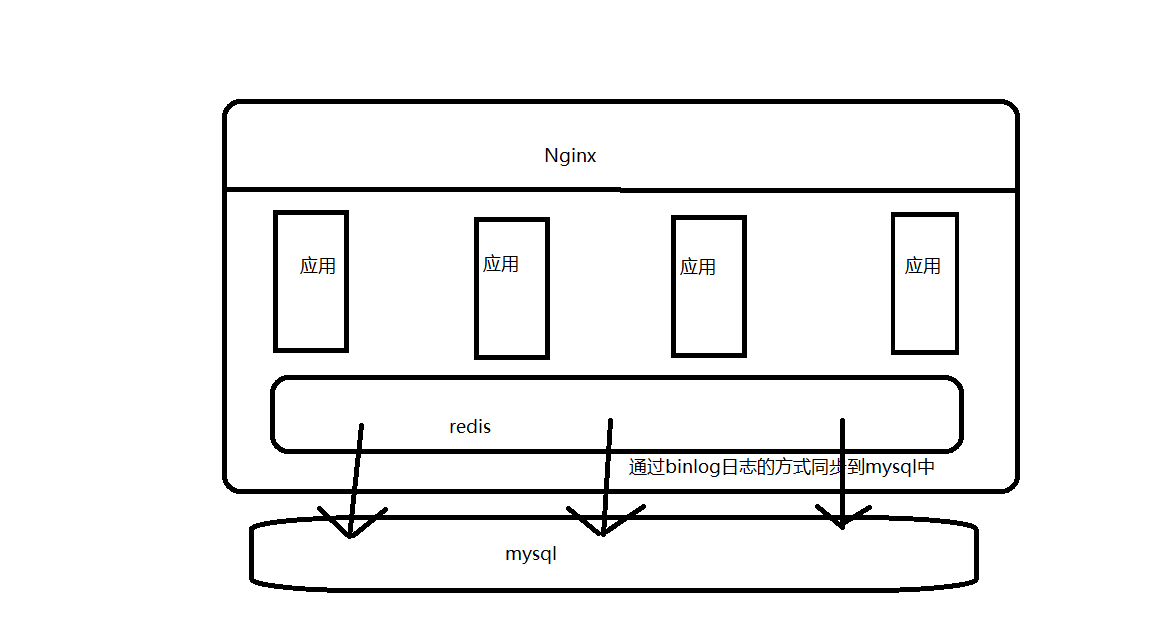
1. **输入：a boy is good ，输出：good is boy a ；**
2. **链表：1->2->3->4->5；请写一个程序，将链表反转过来：5->4->3->2->1;**

# 综合考察：

1. **如何做一个秒杀系统；  
   1、MQ做限流：保证只有1000个人可以访问系统；  
   2、使用redis做缓存；  
   3、redis往mysql中进行持久化**
2. **如何实现高可用、高并发、高吞吐的技术方案？  
   详细讲解请点*[这里](https://wenku.baidu.com/view/c405284f9a6648d7c1c708a1284ac850ac020451.html)* ，讲的非常好，需要仔细的看！（我给下载下来了，具体请看附件吧）**

**   
高并发:应用缓存、HTTP缓存、多级缓存、池化、异步并发、扩容、队列；   
高吞吐：分布式集群；  
高可用：负载均衡与反向代理（nginx）、隔离（hystrix）、限流（MQ/nginx）、降级（hytrix）、超时与重试（hystrix）、回滚（hystrix）；**

1. **A给B发请求，B执行往数据库insert的操作；设计一个架构，保证能够提供最大的并发量，架构如下：**



1. **分布式锁：**

**Redis实现方式：  
redis：为什么使用setnx，或者说set 和setNx有什么区别；  
setNx:系统在10：05 设置一个值，并给出5分钟的过期时间，系统刚刚set完之后redis集群崩溃，10：11分系统重启成功，那么redis中set的值是否还存在？**

**考察点：  
1、redis 的持久化：刚set完是否能够被持久化到快照或者binlog日志中；**

**2、假设redis被持久化，且系统重启时间超过了redis设置的过期时间，那么key是否会被清理；  
zookeeper实现方式：**

1. **缓存数据如何同DB同步更新？（mysql读取redis的binlog，进行数据的更新操作）**

# 其他：

1. **处于什么目的考PMP；**
2. **一个项目启动大会，大概是一个什么过程？  
   1.介绍项目启动；   
   2.任命项目经理；  
   3.宣布项目正式启动；**
3. **对于自己来说，最近接触的比较新的东西或者知识；JVM、python；**
4. **Python2和python3的区别（因为我上一个回答了python，所以这里会问）；**
5. **对自己开发的系统的服务器硬件的了解；**
6. **在什么应用场景下需要升级服务器配置，什么场景下需要扩充集群；**

# 附件

