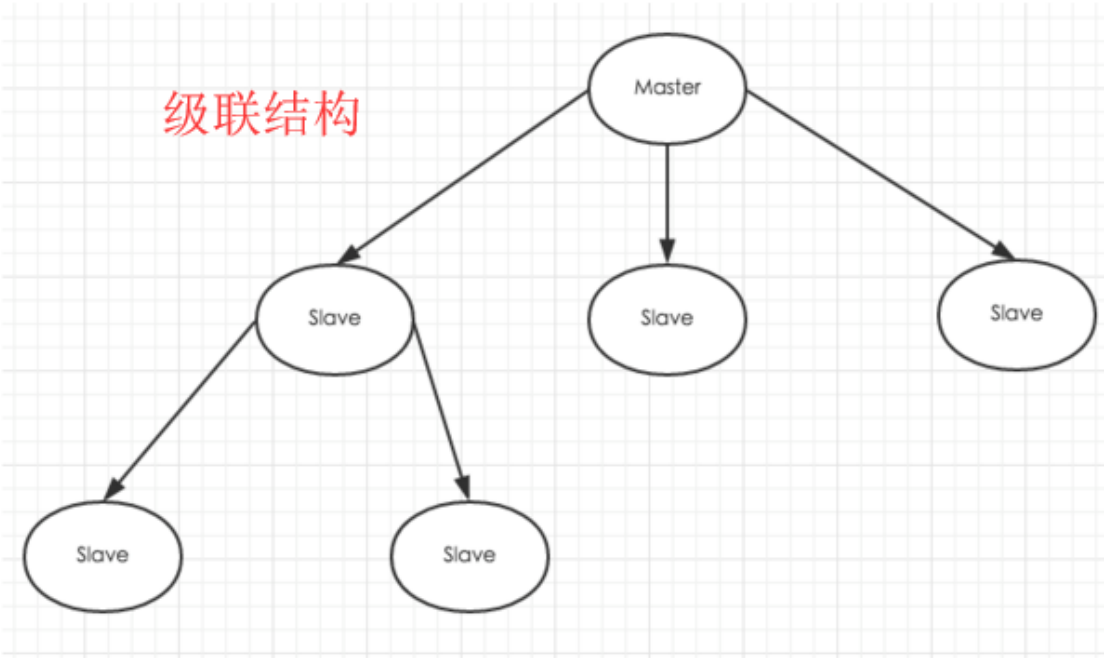
腾讯面经

1. 看你项目介绍中大量使用了 Redis，那能不能介绍下 Redis 的主从同步机制呢？

关于这道题，因为我在之前的文章也分析过 Redis 主从同步的机制，所以我从 完整重同步 和 部分重同步 两个阶段去分析的，结果也得到了面试官的认可。详细的完整重同步和部分重同步机制原理是什么样的，在这里就不展开介绍了。

Redis 主从复制

和 Mysql 主从复制的原因一样，Redis 虽然读取写入的速度都特别快，但是也会产生读压力特别大的情况。为了分担读压力，Redis 支持主从复制，Redis 的主从结构可以采用一主多从或者级联结构，Redis 主从复制可以根据是否是全量分为全量同步和增量同步。



全量同步

Redis 全量复制一般发生在 Slave 初始化阶段，这时 Slave 需要将 Master 上的所有数据都复制一份。具体步骤如下：

- 从服务器连接主服务器，发送 SYNC 命令；
- 主服务器接收到 SYNC 命名后，开始执行 BGSAVE 命令生成 RDB 文件并使用缓冲区记录此后执行的所有写命令；
- 主服务器 BGSAVE 执行完后，向所有从服务器发送快照文件，并在发送期间继续记录被执行的写命令；
- 从服务器收到快照文件后丢弃所有旧数据，载入收到的快照；
- 主服务器快照发送完毕后开始向从服务器发送缓冲区中的写命令；
- 从服务器完成对快照的载入，开始接收命令请求，并执行来自主服务器缓冲区的写命令；

完成上面几个步骤后就完成了从服务器数据初始化的所有操作，从服务器此时可以接收来自用户的读请求。

增量同步

Redis 增量复制是指 Slave 初始化后开始正常工作时主服务器发生的写操作同步到从服务器的过程。

增量复制的过程主要是主服务器每执行一个写命令就会向从服务器发送相同的写命令，从服务器接收并执行收到的写命令。

Redis 主从同步策略

主从刚刚连接的时候，进行全量同步；全同步结束后，进行增量同步。当然，如果有需要，slave 在任何时候都可以发起全量同步。redis 策略是，无论如何，首先会尝试进行增量同步，如不成功，要求从机进行全量同步

2. 你们项目中 Redis 的内存使用和过期数据是怎么做的？

这是两道题，先来说说内存是怎么设计的，首先我们是按照业务维度去评估资源内存的使用量和 QPS 两个维度去评估 Redis 该部署多少资源，然后提到了业务中 Redis 的部署方式及原因。然后在聊到过期数据的剔除策略，这些在之前的文章也有介绍过，附上链接朋友们自行查看吧。缓存过期剔除策略

1.内存使用：(增加内存；使用内存淘汰策略；Redis 集群。)

https://blog.csdn.net/u014590757/article/details/79788076

2.过期数据：Redis 中有个设置时间过期的功能。

https://www.cnblogs.com/xuliangxing/p/7151812.html

3. 看你在实际项目中使用 Streaming 做了一些实时推荐的项目，能介绍下吗？

我们的部分实时推荐项目使用的是 Steaming，其通过对 Kafka 的 Consumer 进行了二次开发封装，屏蔽了 Kafka 不同版本对于业务的影响，同时也可以用 Steaming 去对接不同的数据源，比如现在也支持 xx 数据源的消费;

SparkStreaming 的 Receiver 方式和直连 direct 方式：

https://www.cnblogs.com/hdfs/p/9971761.html

4. 既然使用了 Kafka，你知道为什么 Kafka 那么快吗？

这道题之前在 Kafka 常见面试题中有提到过，这里需要补充一点是，Kafka 的“零拷贝”机制需要朋友在深入研究下，在之前的文章里面没有深入解释过这个点，剩余的其他方面附上链接朋友们自行查看吧：Kafka 为什么那么快

5. 使用 Kafka 有遇到过重复消费的情况吗？你们是怎么解决的？

有遇到过，这里补充一点，因为业务形态所决定消息重复消费产生的影响不是特别严重，所以在一定程度上可以暂时接受，后续通过一些手段去解决了，解决办法之前介绍过如何处理，懒得在这里继续重复了，接看链接吧：Kafka 重复消费怎么办

6. 关于项目咱们先聊这么多，出个题吧，怎么实现一个阻塞队列？

消费线程等到非空了消费，生产线程等到非满了生产；阻塞队列（BlockingQueue）是一个支持两个附加操作的队列。这两个附加的操作是：在队列为空时，获取元素的线程会等待队列变为非空。当队列满时，存储元素的线程会等待队列可用。阻塞队列常用于生产者和消费者的场景，生产者是往队列里添加元素的线程，消费者是从队列里拿元素的线程。阻塞队列就是生产者存放元素的容器，而消费者也只从容器里拿元素。

7. MySQL 熟悉吗？MySQL 事务是什么？

ACID 准备下，时间不多了，直接略过，准备面试的朋友们记得准备一下哈。简单聊了下读未提交、读已提交、重复读、序列化几种情况。需要额外再准备下业务中使用的事务隔离级别是怎么用的（我在项目中使用的 MySQL 不多，面试官也没有多问，主要是一个小时的面试时间快到了，哈哈哈）

Transaction

- 事务：一个最小的不可再分的工作单元；通常一个事务对应一个完整的业务(例如银行账户转账业务，该业务就是一个最小的工作单元)
- 一个完整的业务需要批量的 DML(insert、update、delete)语句共同联合完成
- 事务只和 DML 语句有关，或者说 DML 语句才有事务。这个和业务逻辑有关，业务逻辑不同，DML 语句的个数不同。

https://blog.csdn.net/w_linux/article/details/79666086

8. 知道 MySQL 的 MVCC 机制吗？

平时基本不用 MySQL，这个没答出来，之后了解了下 InnoDB 大概是通过 undo log 和版本号机制来实现的（怪我没有充分准备就面试了，哈哈，自己活该）

多版本控制（Multiversion Concurrency Control）：指的是一种提高并发的技术。最早的数据库系统，只有读读之间可以并发，读写，写读，写写都要阻塞。引入多版本之后，只有写写之间相互阻塞，其他三种操作都可以并行，这样大幅度提高了 InnoDB 的并发度。在内部实现中，InnoDB 通过 undo log 保存每条数据的多个版本，并且能够找回数据历史版本提供给用户读，每个事务读到的数据版本可能是不一样的。在同一个事务中，用户只能看到该事务创建快照之前已经提交的修改和该事务本身做的修改。

9. JVM 的垃圾回收机制了解吗？

标记-清楚算法、复制算法、标记-压缩算法、分代收集算法的实现原理。

一、 技术背景

二、 哪些内存需要回收？

2.1 引用计数算法

2.1.1 算法分析

2.1.2 优缺点

2.1.3 是不是很无趣，来段代码压压惊

2.2 可达性分析算法

2.3 Java 中的引用你了解多少

2.4 对象死亡（被回收）前的最后一次挣扎

2.5 方法区如何判断是否需要回收

三、常用的垃圾收集算法

3.1 标记-清除算法

3.2 复制算法

3.3 标记-整理算法

3.4 分代收集算法

3.4.1 年轻代（Young Generation）的回收算法

3.4.2 年老代（Old Generation）的回收算法

3.4.3 持久代（Permanent Generation）的回收算法

四、常见的垃圾收集器

五、GC 是什么时候触发的（面试最常见的问题之一）

5.1 Scavenge GC

5.2 Full GC

https://www.cnblogs.com/1024Community/p/honery.html

10. 项目中用的是 CMS 垃圾回收器吗？为什么要分为四个阶段？

初始标记，并发标记，重新标记，并发清理四阶段，系统要求快速响应低延迟，对于多核 CPU 性能更佳等方面去回答即可，网上文章很多，这里不多赘述了。

https://www.jianshu.com/p/86e358afdf17

https://zhuanlan.zhihu.com/p/150696908

面经资料来自网络