1. redis如何做内存优化？
   1. 缩减键值对象
   2. 共享对象池
   3. 字符串优化
   4. 编码优化
   5. 控制key的数量
   6. 配置内存回收策略
2. redis中的管道（pipeline）有什么用？

当大量的数据需要一次性操作完成的时候（这些操作是无关的），使用管道可以进行批处理，将这些操作合并成一次操作（也就是无需等待旧请求响应就可以写入新请求），减少网络交互，提升性能。

1. redis有哪些合适的应用场景？

缓存、队列、分布式锁、共享session、排行榜/计数器

1. redis相比memcache有哪些优势?
2. redis支持更丰富的数据类型
3. redis速度更快
4. redis可持久化数据

5、redis为什么那么快？

完全基于内存操作、采用单线程（6.0采用了多线程io，不过命令的执行还是单线程的）、避免了多线程频繁的上下文切换问题、非阻塞io模型，epoll。

6、缓存击穿、缓存穿透、缓存雪崩

缓存击穿：缓存击穿是指一个Key非常热点，在不停地扛着大量的请求，大并发集中对这一个点进行访问，当这个Key在失效的瞬间，持续的大并发直接落到了数据库上，就在这个Key的点上击穿了缓存。

解决：设置热点数据永不过期，或者加上个锁就搞定了。

缓存穿透：查询一条数据库和缓存都没有的一条数据，就会一直查询数据库，对数据 库的访问压力就会增大；

解决方案：最简单粗暴的方法如果一个查询返回的数据为空（不管是数据不存在，还是系统故障），我们就把这个空结果进行缓存，但它的过期时间会很短，最长不超过五分钟。

\*\*布隆过滤器（Bloom Filter）\*\*这个也能很好的预防缓存穿透的发生，就是利用高效的数据结构和算法快速判断出你这个Key是否在数据库中存在，不存在你return就好了，存在你就去查DB刷新KV再return

缓存雪崩：在一个较短的时间内，缓存中较多的key集中过期或者缓存挂了，导致了数据库服务器崩溃

缓存雪崩的事前事中事后的解决方案如下：

在批量往Redis存数据的时候，把每个Key的失效时间都加个随机值就好了，这样可以保证数据不会再同一时间大面积失效。如果 Redis 是集群部署，将热点数据均匀分布在不同的 Redis 库中也能避免全部失效。或者设置热点数据永不过期，有更新操作就更新缓存就好了

7、如何保障redis与mysql数据一致性？

采用延时双删策略：1）先删除缓存

2）再写数据库

3）休眠500毫秒

4）再次删除缓存

那么，这个500毫秒怎么确定的，具体该休眠多久呢？

需要评估自己的项目的读数据业务逻辑的耗时。这么做的目的，就是确保读请求结束，写请求可以删除读请求造成的缓存脏数据。

异步更新缓存(基于订阅binlog的同步机制)：

MySQL binlog增量订阅消费+消息队列+增量数据更新到redis

1）读Redis：热数据基本都在Redis

2）写MySQL:增删改都是操作MySQL

3）更新Redis数据：MySQ的数据操作binlog，来更新到Redis

8、用redis做延时队列，具体如何实现？

第一步将需要发放的消息发送到延迟队列中。

延迟队列将数据存入 Redis 的 ZSet 有序集合中 score 为当前时间戳，member 存入需要发送的数据。

添加一个 schedule 来进行对 Redis 有序队列的轮询。

如果到达达到消息的执行时间，那么就进行业务的执行。

如果没有达到消息的执行，那么消息等待下轮执行。

9、redis的事务

Redis事务就是一次性、顺序性、排他性的执行一个队列中的一系列命令；没有隔离性、不保证原子性