什么是 Redis 持久化？Redis 有哪几种持久化方式？优缺点是什么？

答:

Redis 的持久化就是把 redis 中的数据从内存中写入到磁盘中去，防止服务宕机了内存数据丢失。

持久化策略:

1， RDB:（默认开启）

RDB 持久化是指在指定的时间间隔内将内存中的数据集快照写入磁盘，实际操作过程是 fork 一个子进程，先将数据集写入临时文件，写入成功后，再替换之前的文件，用二进制压缩存储。数据恢复的时候会将文件中的数据通过 rdbload 函数读取到内存。

2， AOF

AOF 持久化以日志的形式记录服务器所处理的每一个写、删除操作，查询操作不会记录，以文本的方式记录，可以打开文件看到详细的操作记录。

二者的比较:

RDB:

优点:

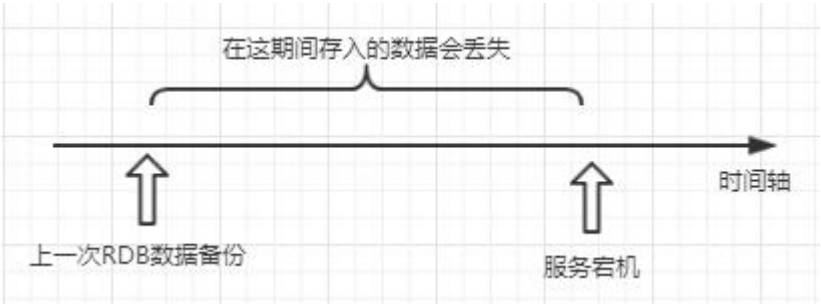
1 一旦采用该方式，那么你的整个 Redis 数据库将只包含一个文件，这对于文件备份而言是非常完美的。比如，你可能打算每个小时归档一次最近 24 小时的数据，同时还要每天归档一次最近 30 天的数据。通过这样的备份策略，一旦系统出现灾难性故障，我们可以非常容易的进行恢复。

2 性能最大化。对于 Redis 的服务进程而言，在开始持久化时，它唯一需要做的只是 fork 出子进程，之后再由子进程完成这些持久化的工作，这样就可以极大的避免服务进程执行 IO 操作了。

3 如果数据量很大，RDB 启动效率会更高

缺点:

1， 如果你想保证数据的高可用性，即最大限度的避免数据丢失，那么 RDB 将不是一个很好的选择。因为系统一旦在定时持久化之前出现宕机现象，此前没有来得及写入磁盘的数据都将丢失。如果在持久化的时候宕机，那么可能整个数据都可能丢失



2， 由于 RDB 是通过 fork 子进程来协助完成数据持久化工作的，因此，如果当数据集较大时，可能会导致整个服务器压力过大，影响到 redis

主进程的业务相应速度

AOF:

优点:

1，该机制可以带来更高的数据安全性，即数据持久性。Redis 中提供了 3 中同步策略，即每秒同步、每修改同步和不同步。事实上，每秒同步也是异步完成的，其效率也是非常高的，所差的是一旦系统出现宕机现象，那么这一秒钟之内修改的数据将会丢失。而每修改同步，我们可以将其视为同步持久化，即每次发生的数据变化都会被立即记录到磁盘中。可以预见，这种方式在效率上是最低的。

2，由于该机制对日志文件的写入操作采用的是 append 模式，因此在写入的过程中即使出现服务宕机的情况，那么也不会破坏日志文件中之前的内容。

3，AOF 包含一个格式清晰、易于理解的日志文件用于记录所有的修改操作。事实上，我们也可以通过该文件完成数据的重建。

缺点:

1，对于相同数量的数据集而言，AOF 文件 通常大于 RDB 文件，RDB 在恢复大数据集时的速度比 AOF 要快

2，根据同步策略不同，AOF 在运行效率上往往会慢于 RDB，

二者选择的标准:

二者选择的标准，就是看系统是愿意牺牲一些性能，换取更高的缓存一致性（aof），还是愿意写操作频繁的时候，不启用备份来换取更高的性能，待手动运行 save 的时候，再做备份（rdb）。rdb 这个就更有些 eventually consistent 的意思了。不过生产环境其实更多都是二者结合使用的。

Redis 支持的数据类型?

String，Hash，Set，List，Zset(SortSet)

建议了解各自的应用场景

Redis 的架构模式有哪些？讲讲各自的特点

1，单机版

缺点：1、内存容量有限 2、处理能力有限 3、无法高可用（只有一台机器，假如这台机器挂了，那么相当于整个缓存服务都挂了，所以无法高可用）

主从复制:

Redis 的复制（replication）功能允许用户根据一个 Redis 服务器来创建任意多个该服务器的复制品，其中被复制的服务器为主服务器（master），而通过复制创建

出来的服务器复制品则为从服务器（slave）

特点：

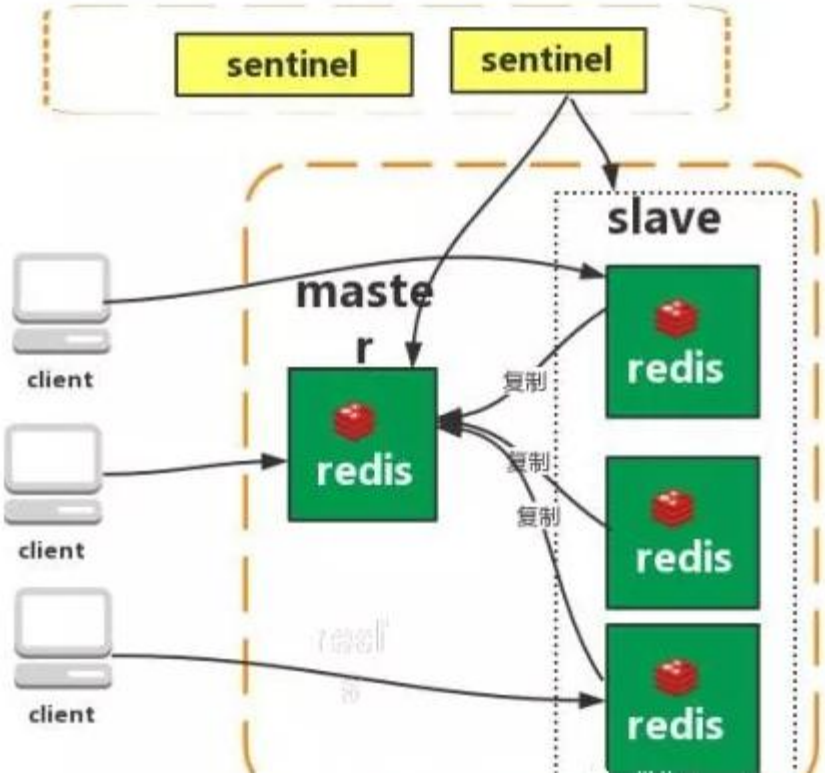
主机 master 中的内容和 slave 中的内容完全一致，那么这种模式下可以做读写分离，master 负责写，slave 负责读，这样可以缓解 master 的压力，

问题：

1， 无法高可用，一旦 master 挂了，那么整个写功能都挂了

2， 没有解决 master 写的压力

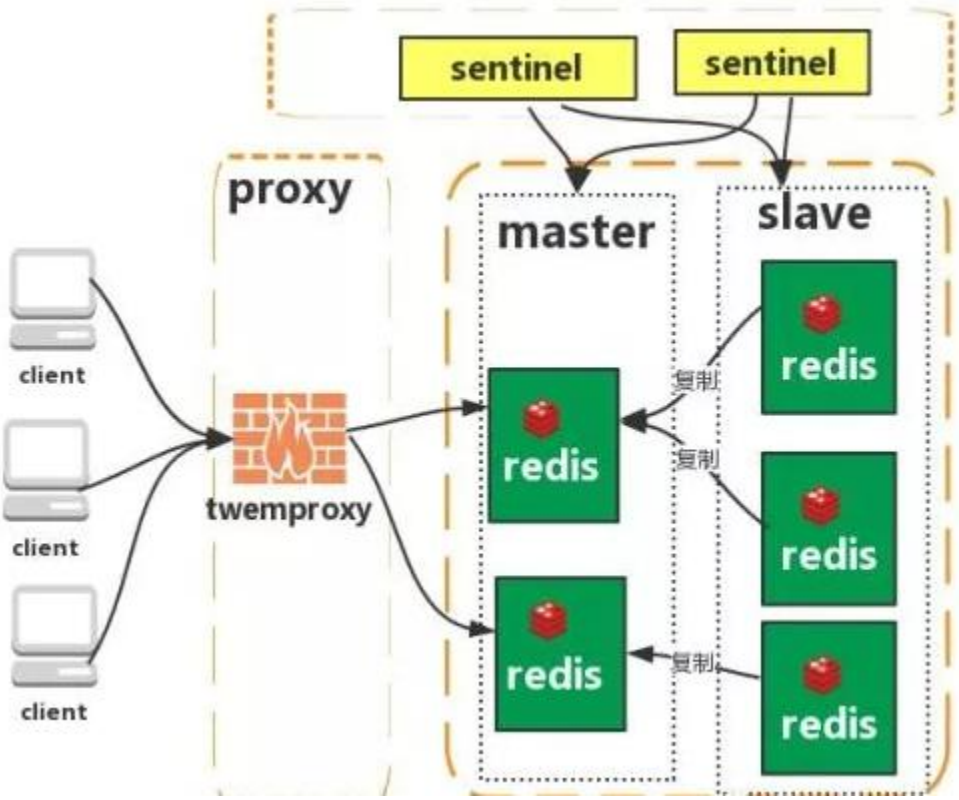
哨兵模式：如图所示



可以理解为主从架构的升级版

多了一个 sentinel 角色（哨兵），它和 redis 的主机之间保持着心跳链接机制，一旦发现 redis 的 master 服务挂了，那么会切换 master 到一台 slave 上，并且可以通过 API 向管理员或者其他应用发送通知

集群<Proxy 型>：如图所示



相当于哨兵模式的升级版，master 升级为集群，哨兵升级为集群，但是要通过 twemproxy 去映射请求，然后分发到 master 上

映射有多种 hash 算法：

MD5、CRC16、CRC32、CRC32a、hsieh、murmur、Jenkins

优点：

对比哨兵模式又有了进一步的提升，处理能力更强

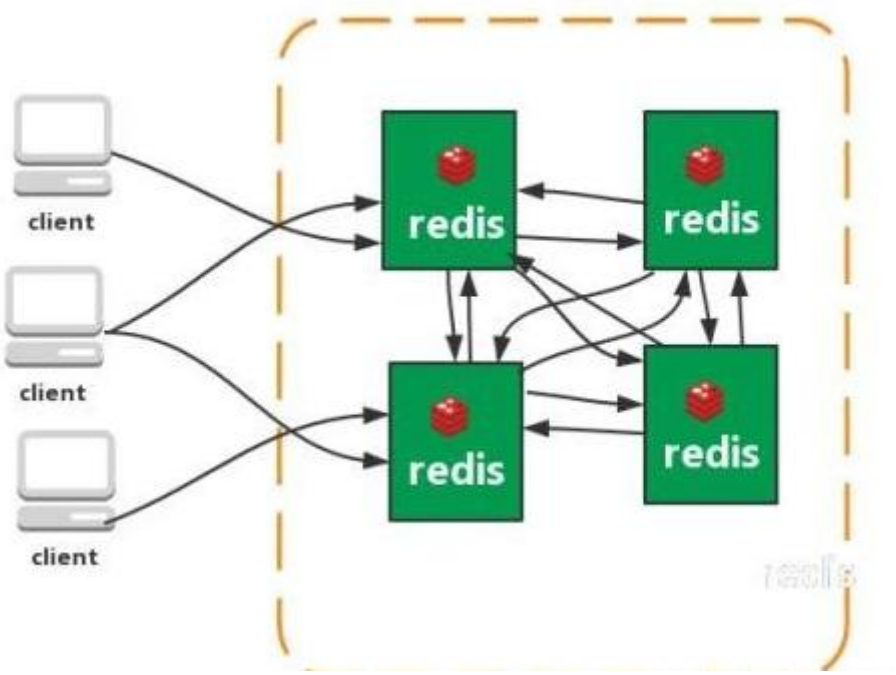
支持失败节点自动删除

后端集群存储对于开发使用人员来说透明，和操作单个 redis 基本没区别

缺点：

新增了 Proxy，维护困难

集群<直连型>



优点:

当前比较流行的“去中心化”架构，一台服务的宕机不会立马影响到整个服务

可拓展性，拓展节点很方便

高可用性，部分节点不可用时，集群仍然可用

缺点:

数据通过异步复制，所以不能保证数据的强一致性

资源隔离性差，容易互相影响

什么是缓存穿透？如何避免？什么是缓存雪崩？何如避免?

缓存穿透:

一般的缓存系统，都是按照 key 去缓存查询，如果不存在对应的 value，就应该去后端系统

查找（比如 DB）。一些恶意的请求会故意查询不存在的 key,请求量很大，就会对后端系统造成很大的压力。这就叫做缓存穿透。
如何避免：

去后面 DB 查找之后如果还是没有，那么就把这个 key 缓存起来，值可以设为空值或者一个特定的值，设置一个合理的缓存时间（5 分钟左右）那么在下一个请求过来之后，先去查找 redis，如果拿到的这个值是我们之前设置的那么值，那么就直接返回空

缓存雪崩：

缓存雪崩是指缓存中数据大批量到过期时间，而查询数据量巨大，引起数据库压力过大甚至 down 机。和缓存击穿不同的是，缓存击穿指并发查同一条数据，缓存雪崩是不同数据都过期了，很多数据都查不到从而查数据库。

1， 缓存数据的过期时间设置随机，防止同一时间大量数据过期现象发生。
2， 如果缓存数据库是分布式部署，将热点数据均匀分布在不同搞得缓存数据库中。
3， 设置热点数据永远不过期。

缓存击穿

缓存击穿是指缓存中没有但数据库中有的数据（一般是缓存时间到期），这时由于并发用户特别多，同时读缓存没读到数据，又同时去数据库去取数据，引起数据库压力瞬间增大，造成过大压力

解决方案：

1， 设置热点数据永不过期，对于秒杀这种活动应用场景，可以使用缓存预热，也就是在真正上线之前就把缓存加入到 redis 中，这样等到活动一开始的时候，就可以直接去查缓存了，从而不会造成大量的并发请求去查询数据库
2， 增加互斥锁（Mutex key）

Mysql 中有 2000w 条数据，而 redis 中只有 20w 条数据，如何保证 redis 中存的都是热点数据?

相关知识：redis 内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略（回收策略）。redis 提供 6 种数据淘汰策略：
volatile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰
volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰

volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰
allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰
allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰
no-enviction（驱逐）：禁止驱逐数据

Redis 是单线程还是单进程? Redis 如何控制并发问题?

Redis 为单进程单线程模式，采用队列模式将并发访问变为串行访问。Redis 本身没有锁的概念，Redis 对于多个客户端连接并不存在竞争，但是在 Jedis 客户端对 Redis 进行并发访问时会发生连接超时、数据转换错误、阻塞、客户端关闭连接等问题，这些问题均是由于客户端连接混乱造成。对此有 2 种解决方法：

1.客户端角度，为保证每个客户端间正常有序与 Redis 进行通信，对连接进行池化，同时对客户端读写 Redis 操作采用内部锁 synchronized。

2.服务器角度，利用 setnx 实现锁。

注：对于第一种，需要应用程序自己处理资源的同步，可以使用的方法比较通俗，可以使用 synchronized 也可以使用 lock；第二种需要用到 Redis 的 setnx 命令，但是需要注意一些问题。

Redis 有哪些常用的命令?了解 Watch 命令吗?

watch 用于在进行事务操作的最后一步也就是在执行 exec 之前对某个 key 进行监视
如果这个被监视的 key 被改动，那么事务就被取消，否则事务正常执行.
一般在 MULTI 命令前就用 watch 命令对某个 key 进行监控.如果想让 key 取消被监控，可以用 unwatch 命令

如何使用 Redis 实现分布式锁?思路是什么?

最简单的一种：

```
//可以任意给
public static String value = "xxx";
@Autowired
private RedisTemplate redisTemplate;
public Boolean getLock(String key) {
    if (redisTemplate.opsForValue().setIfAbsent(key,value)) {
        return true;
    }
    else {
        this.getLock(key);
    }
    return false;
}
```

这种是有问题的，大家可以思考一下，附件有有 demo

这里我只是给出了出现频率比较高的面试，大家也可以看下下面这个链接
redis 面试题参考链接 https://msd.misuland.com/pd/3065794831805580868