Redis系列

Redis的数据结构

- String字符串
- list列表
- hash散列
- sets集合
- sorted set有序集合

Redis持久化

- **AOP持久化**

  将Redis的操作记录以追加的形式写入文件。

- **RDB持久化（默认方式）**

  指定的时间间隔内将内存中的数据集快照写入磁盘的RDB快照文件中。

Redis为什么快

- redis是基于内存的，内存的读写十分快
- redis是单线程的，可以省去很多上下文切换时间
- 等

Redis为什么使用单线程

回答这个问题，首先要明确多线程是怎么来的。多线程的定义本身就是来源于CPU，也就是多线程是在CPU的基础上模拟出来的概念，一个单核的CPU可以拥有N个线程，每个线程实际上是在模拟一个核心（指CPU核心单元组）进行工作。换句话来讲，多线程的使用就是为了加大CPU的利用率。

但是，我们的主角Redis本身就在CPU内存里面了，单线程又还容易实现，已经是最完美的了，还做多线程干嘛？？？还平白无故增加线程上下文切换的时间。多不划算！

当然，还有其他原因使得最好的解决方案就是单线程。

a. 无需考虑锁带来的性能损耗

　多线程模式下，对于redis中大部分存储结构的数据进行操作时，需要使用锁处理高并发的问题，还有可能造成死锁等。而使用单线程完全不必考虑这样的开销。

b. 等等。了解更多自行搜索

redis内存回收机制

redis内存回收主要围绕这两个方面：

- Redis过期策略：删除过期时间的key值
- Redis淘汰策略：当内存使用达到maxmemory上线时触发淘汰策略

Redis过期策略

过期策略通常是三种：

- **定时过期**

每设置一个过期时间就创建一个定时器，到过期时间就立即清除。该策略对内存十分友好，能够精确地清理掉过期的key，但是该策略需要维护定时器，同时还要不断对过期的数据进行处理，会占用大量的CPU资源，从而影响到响应时间和吞吐量。

- **惰性过期**

  每一个设置了过期时间的key只有当再一次被访问时，才会触发删除。该策略对CPU十分友好，因为无需维护任何计时器，但是一旦出现大量的过期key没有被二次访问时，就会在内存中存在大量的过期key，对内存不友好。

- **定期删除**

  每隔一段时间就会对一定数量的key进行扫描，并且清理已经过期的key。该策略属于前面两个策略的折中方案。通过设置扫描间隔时间和每次扫描的范围使得在不同情况下对内存和CPU都友好。

在redis中，使用的是后两者也就是惰性过期和定时删除结合。

Redis淘汰策略

Redis淘汰策略，是指当内存到达maxmemory极限时，需要使用LAU淘汰算法来决定清理掉哪些数据，以保证新数据能够被存入。

- **LAU淘汰算法**

  是redis默认使用的淘汰算法。

  LAU淘汰算法就是最近最少使用算法，Redis使用该算法的结果是删除最近最少使用的key。

  值得注意的是，在Redis中，不会准确地删除所有key中最近最少使用的。而是随机抽取n条数据，删除这n条数据中最近最少使用的。 n是可配置的，但是n越高，则对CPU造成的压力就越大，一般情况n设置在10以内，对应的字段为：maxmeory-samples

  也就是说，每一次触发淘汰策略只会删除一条key。

- **Redis缓存清理流程**

  客户端执行写入操作。

  redis接收到写入操作后检查maxmemory的限制，如果超过限制，就根据对应的策略进行数据清理。

  执行数据写入。

Redis分布式锁的应用

当系统存在并发竞争时，也就是多个客户端对同一对象进行操作时。现在假设一个情况：存在一个key，对应的值是1，此时有三个客户端分别修改key为2,3,4同时对该key进行赋值，本意是想要将这个key值从小到大进行赋值的，最后结果是4。但是如果并发不加控制，完全有可能出现3-4-2这样的结果，导致最终的key值为2。

传统的synchronized和Lock仅仅能对同一线程进行加锁，但是对于多个客户端的请求来说，各个操作处于不同的线程，因此传统的锁起不到任何效果。

而分布式锁不同，分布式锁让所有客户端中只会有一个客户端持有锁，在他释放之前，其他客户端不能持有。因此，只需要引入Redis分布式锁，同时借助CAS原理即可实现对key从小到大的赋值。

当然，分布式锁除了Redis分布式锁，还有数据库的乐观锁、Zookeeper分布式锁等。

> 本案例还有另外一种解决方案，那就是队列。把每个客户端对key值的更改依次放到队列中，队列中的操作必须是一个个进行的，这样也可以有效避免并发下导致的数据错误问题。

Redis的高可用

哨兵　哨兵可以管理多个Redis服务器，提供监控、提醒以及故障转移等功能。

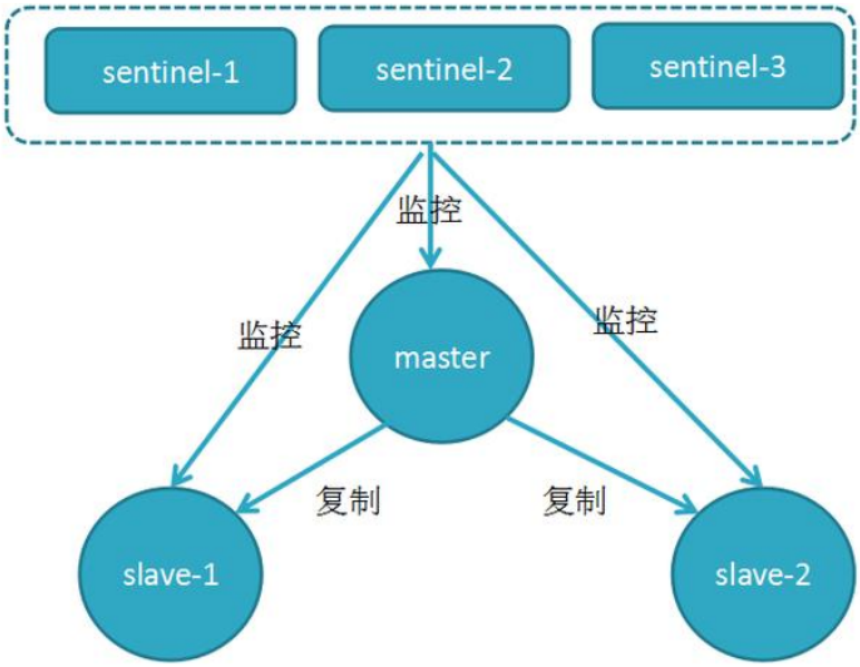
复制　负责让一个Redis服务器可以配备多个备份的服务器

Redis主要是使用这两个功能实现Redis的高可用。

哨兵

哨兵是Redis集群架构中非常重要的组件，哨兵的出现是为了解决主从复制出现问题时需要人为干预的问题。

- **主要功能**
  - 集群监控 监控master和slave节点服务是否正常工作。
  - 消息通知 当某个Redis发生故障，哨兵负责将警报发送给管理员。
  - 故障转移 如果master node挂掉了，会自动将主节点转移到slave node上
  - 配置中心 当故障转移发生后，通知client客户端新的master地址
- **哨兵的高可用**
  - 哨兵高可用的原理主要是当哨兵检测到某个节点发生故障时，能够自动有效地完成节点转移，并且通知应用方。
  - 哨兵机制建立了多个哨兵节点（进程），共同监控redis服务节点。
  - 多个哨兵互相通信。
  - 每隔一秒每个哨兵都会向整个redis集群（Master主服务器+Slave从服务器+其他哨兵进程）发送ping命令做心跳检测。



- **主观判断**

  主观判断是哨兵中的一个新概念，相对的，还存在一个客观判断在下文提到。主观判断就是只有一个哨兵发现redis主节点挂了。

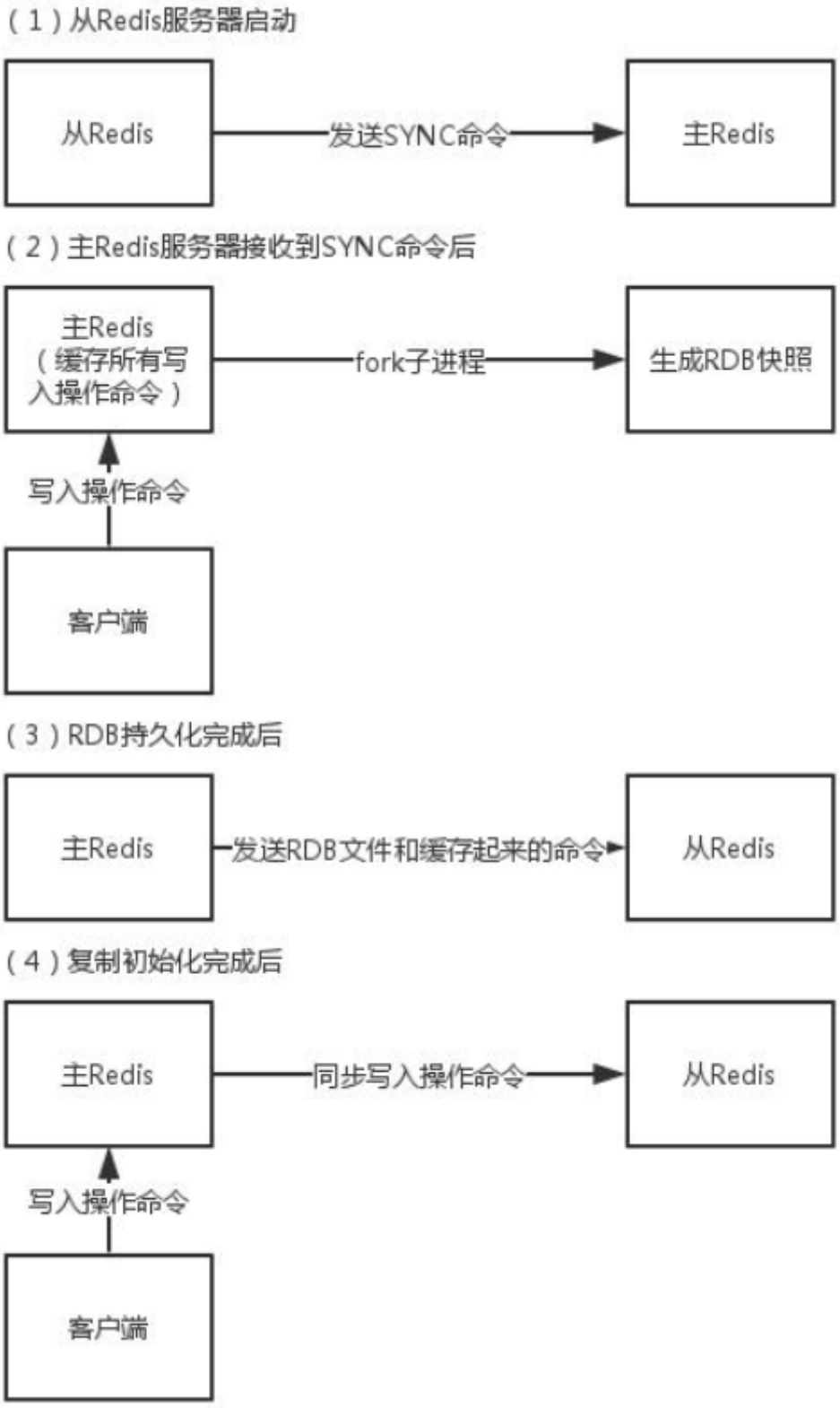
- **客观判断**

  与主观判断一样，客观判断也是哨兵中的新概念，客观判断强调的是当至少半数的哨兵节点都发现redis主节点挂掉时，则客观判断该主节点挂掉了。也就是首先存在主观判断，然后才会导致客观判断的结果。此时，最先发现主节点挂掉的哨兵就会发起投票，推选一个领导者哨兵节点完成主从切换以及通知到应用方的任务。

复制

Redis为解决单点数据库的问题，会把数据库复制到多个副本部署到其他节点。通过复制，实现Redis的高可用。

复制流程



Redis故障

缓存雪崩

缓存雪崩是指redis中大量数据同时失效（redis宕机、很多key同时过期等）导致所有请求都去查数据库，导致数据库CPU和内存负载过高而宕机。

预防雪崩

- 实现缓存高可用 例如：redis集群、redis哨兵
- 设置不同的key过期时间
- 等，自行搜索

缓存穿透

缓存穿透是指请求查询一个不存在的数据。在redis缓存中没有命中，然后前往数据库查询依然没有数据而不会写入缓存，这将导致同类请求都会穿过redis对数据库造成压力。

预防穿透

- 当查询数据库也为空时，直接设置一个默认值到缓存中。但是记得设置过期时间，或者当数据库有值之后能够通知到缓存清除原本为null的数据。

缓存并发

缓存并发是指多个redis客户端同时对一个key进行set操作。

由于redis本身就是单线程的，所以每个客户端的命令都会排着队一个个的执行，只是无法保证执行顺序。可以通过其他手段实现。如：乐观锁，或者是队列（排好序放进队列）

缓存预热

数据上线前通过各种手段执行一遍数据同步操作（如：当前数据缓存刷新），将数据加载到缓存中，然后再发布到生产环境。