## Redis6(:6379)

运行：在安装目录：/usr/local/bin下输入：redis-server /etc/redis.conf

## **什么是Redis？它主要用来什么的？**

Redis是一个基于内存所以读写速度很快，并且可以将数据持久化的非关系数据库，它既不遵循SQL标准也不支持ACID。它被广泛应用于缓存，也经常被用来做分布式锁。适用于对数据高并发的读写，海量数据的读写，对数据高可扩展性的读写

## **Redis是怎么用来优化的**

## **为什么Redis采用单线程？（单线程+IO多路复用）**

因为Redis是基于内存操作的，CPU不是Redis的瓶颈，当所有的数据都在内存中使用单线程去操作速度已经足够快了，当我们换成多线程时，线程上下文的切换会浪费CPU资源，而且使用多线程是存在并发安全的，当我们使用各种锁去控制的话又会导致性能的消耗。所以采用单线程，多进程集群的方案。

但是由于现在业务非常庞大的原因，Redis6引入多线程（但默认关闭需手动配置），可以充分利用多核CPU，多个线程任务分担Redis同步IO读写负荷，由主线程来管理IO线程的读写，所以不需要担心并发安全问题

## **字符串String**

字符串的底层数据结构为简单动态字符串SDS，它包含一个字符数组buf，以及这个数组的长度len和剩余空间free。我们可以直接通过len来获取长度，避免对字符串的遍历，也可以通过free判断添加数据时是否需要重新分配空间，避免了溢出的情况。

二进制安全：buf以二进制形式存储，所以任何可以转化为二进制的都可以存储

扩容：当字符串长度小于1M时，扩容都是加倍现有的空间，如果超过1M，扩容时一次只会多扩1M的空间。字符串最大长度为512M。

## **列表List**

List的底层数据结构为快速链表，当元素比较少时还只是一个连续分配的压缩链表只存放数据不存放指针，当元素变多后，将多个压缩链表作为节点使用双向链表的方式连接起来就形成了一个快速链表

## **哈希Hash**

Hash的底层数据结构为压缩链表或者Hashtable；当所有元素的key和value的大小都不超过64字节且不超过512个元素时使用压缩链表，否则使用Hashtable；

## **集合Set**

Set的底层是一个整数集合intset或者哈希表hashtable；全部是整数且不超过512个是使用intset否则使用hashtable，所有的value都指向同一个内部值；

## **有序集合Zset**

当Zset中存储的元素都小于64字节且不超过128个时使用压缩链表从小到大存储，否则同时使用hash表和跳跃表存储，hash表实现了快速查找功能而跳跃表则实现了范围查找和排序功能

## **三种特殊数据类型**

Geo：用于存储地理位置信息。

HyperLogLog：用来做基数统计算法的数据结构，如统计网站的UV。

Bitmaps ：它的底层是基于字符串类型实现的，可以看成是一个以比特位为单位的数组。用这些比特位来映射其它元素的状态。

## **举例各个类型的应用场景**

String：访问量统计，buf是二进制数组，访问时可加1

List：两端存放，可用于存放消息等比如新闻。或者和时间有关的排序

Hash：存储用户属性等，比如一个用户名对应这他的一些个人信息

Set：集合的操作，比如“共同好友”和“猜你喜欢”等交集功能

Zset：排行榜

## **事务**

Redis的事务本质上是一次性地执行一组命令，由multi开启，将命令入队后输入exec执行事务，它没有隔离级别的概念也不保证原子性。入队时有错误命令则全都不执行。执行后出现错误命令其它命令正常执行。可以使用discard取消事务；也可以使用watch在开启事务之前监测元素，当在命令队列中发现对watch监测元素有改动时则这个事务不会执行（类似于乐观锁）。

悲观锁：每次在拿数据的时候都会上锁

乐观锁：使用版本号控制。在更新的时候判断是否有人修改版本（适用于有多读）。

## **RDB持久化（Redis Database）**

RDB是将内存中的数据以快照的形式写入磁盘，有手动和自动触发两种方式，手动触发又分为save和bgsave。save会由主线程进行持久化，性能较低；bgsave会先用fork创建一个子线程，主线程继续进行客户端的交互，由这个子线程采用写时复制技术将数据写入到 一个临时文件中，待持久化过程都结束了，再用这个临时文件替换磁盘中上次持久化好的文件。 整个过程中，主线程是不进行任何IO操作的，这就确保了极高的性能。自动触发是由设置触发bgsave的条件实现的，比如一秒内多少个key发生改变就自动触发bgsave持久化。 如果需要进行大规模数据的恢复，且对于数据恢复的完整性不是非常敏感，那RDB方式要比AOF方式更加的高效。**RDB的缺点是最后一次持久化后的数据可能丢失**。

## **AOF持久化**

AOF是以**日志**的形式来记录每个写操作，redis 重启的话就根据日志文件的内容将写指令从前到后执行一遍以完成数据的恢复工作。有三种持久化的策略，每条写操作追加一次（always）、每秒一次（everysec）或者交给操作系统决定（no）。当AOF文件过大时Redis还会对AOF文件进行压缩重写（重写方式与RDB一样）。AOF记录的数据比较精确不容易丢失但是文件比较大，恢复备份的过程也比较慢

## **应该选择哪个持久化**

如果对数据完整性有很高的要求，推荐两个都启用；对数据完整性要求不高可用使用RDB；不建议单独使用AOF，因为可能出现bug；如果只是作为缓存使用可以都不用。

## **什么是写时复制技术**

如果有多个调用者同时请求相同资源，他们会共同获取相同的指针指向相同的资源，直到某个调用者试图修改资源的内容时，系统才会真正复制一份专用副本给该调用者

## **主从复制的原理**

从服务器连接到主服务器后会发送一个请求命令，主服务器接到命令后会进行RDB持久化，然后将RDB文件发送给从服务器（全量复制）。之后每次主服务器进行写操作都会主动和从服务器进行同步（增量复制）。

## **一主二仆**

从服务器宕机，如果是在配置文件中配置主从模式则重启后仍是从服务器。主服务器宕机从服务器不会上位，仍会等着主服务器重启。

## **薪火相传**

从服务器之下还有它自己的从服务器，主服务器或者中间的从服务器宕机了后面的从服务器都无法备份

## **反客为主**

**当主服务器宕机后，后面的从服务器可以立刻升级为主服务器，且后面的从服务器不用做任何修改。需手动执行slaveof no one**

## **哨兵模式**

也就是自动完成反客为主的模式，设置哨兵节点，定时向主服务器、从服务器和其它哨兵发送信号当达到一定时间没有回应时认定主服务器主观下线，多个哨兵认定主观下线就会变成客观下线，然后哨兵通过投票算法选出新的主服务器。当原来的主服务器再次上线时会变成从服务器。选择条件为优先级高的到复制数据最多的再到runid小的。

## **集群**

集群主要是为了解决写并发问题和海量数据的存储压力，redis的集群采用了去中心化的思想，集群至少有三个节点，然后将不同的数据分到集群中不同的服务器中，但是对于客户端来说，整个集群可以看成一个整体，可以连接任意一个节点进行操作，就像操作单一Redis实例一样，集群节点下还可以有从节点，当集群中的一个节点挂掉后，集群会提升它的某个从节点作为新的主节点加入集群

## **缓存穿透**

指访问一个缓存和数据库中都不存在的key，由于这个key在缓存中不存在，则会到数据库中查询，数据库中也不存在该key，无法将数据添加到缓存中，所以每次都会访问数据库导致数据库压力增大

解决：

1. 将空key添加到缓存中
2. 使用布隆过滤器过滤空key。
3. 设置可访问的白名单

## **缓存击穿**

指大量请求访问缓存中的一个key时，该key过期了，导致这些请求都去直接访问数据库，短时间大量的请求可能会将数据库击垮。

解决：

1. 预先将热门数据移入缓存并实时监控和调整key的过期时长
2. 添加互斥锁或分布式锁，只能有一个线程去访问数据库，将数据添加到缓存中后，其他线程直接从缓存中获取。

## **缓存雪崩**

 指在系统运行过程中，缓存服务宕机或大量的key值同时过期，导致所有请求都直接访问数据库导致数据库压力增大

解决：

1. 构建**多级缓存架构**比如同时使用nginx缓存 和redis缓存
2. 加互斥锁，同一key值只允许一个线程去访问数据库，其余线程等待写入后直接从缓存中获取
3. **将缓存key失效的时间分散开**

## **分布式锁**

首先是分布式锁的应用场景：当多个并发的进程同一时刻要去修改某个资源时，我们希望该资源同一时刻只能一个进程修改。否则会发生混乱。

redis分布式锁的基本原理：利用setnx 每个key只能设置一次vlue的特性来实现锁的获取，后边给相同的key赋值的进程被阻塞，直到前面的进程完成操作后删掉key，也就是释放锁，后边的进程才可以成功赋值，然后进行他的工作。

但在实际的应用中往往会有很多情况出现：

1. 假如客户端1在获取到锁资源后宕机了，那么这个key会一直存在redis中，那么其他服务就永远无法获取到这个锁了。就发生了死锁。我们可以通过设置key的过期时间解决这个问题。即使当前客户端宕机了，key也会在过期时间到后被删除。
2. 但锁的过期时间要到了，但是我的工作还没有完成怎么办呢？redis已经帮我们实现了这个自动续时的功能。会给锁重新设置一个过期时间。
3. 如何防止释放其它服务的锁呢？设置的value带上自己的标识，要删某个key之前先get出来看一下这个value是不是自己添加的
4. 防止主服务器宕机：RedLock

## **Redis和MongoDB的区别**

MongoDB是一个基于分布式文件存储的数据库。是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

Redis和MongoDB是两种不同的数据库类型，它们在设计目标、数据模型、性能和使用场景上有很大的差异。

Redis是一种基于内存的键值存储数据库，支持高性能读写和支持数据持久化。它通常用于缓存、消息队列、实时分析等应用场景。

MongoDB是一种文档型数据库，支持高性能读写和分布式部署。它通常用于大规模数据存储、查询和分析。

总的来说，Redis更适合高性能场景，MongoDB更适合大规模数据存储和分析。