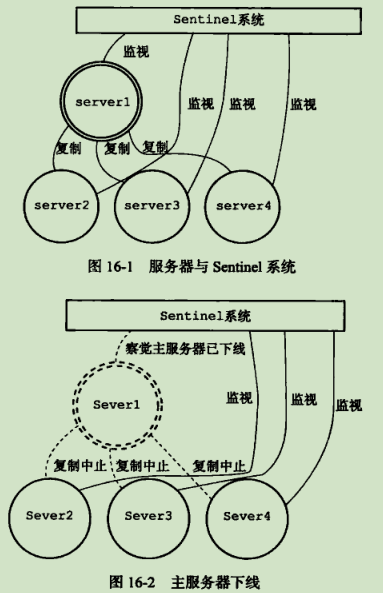
一、redis五种数据结构

Redis支持五种数据类型：string（字符串），hash（哈希），list（列表），set（集合）及zset(sorted set：有序集合)。

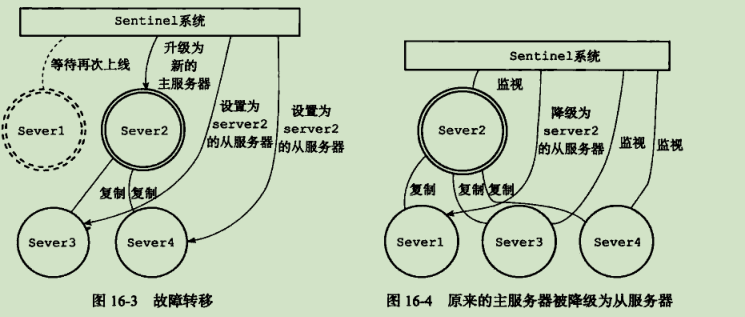
二、redis哨兵机制

　Sentinel（哨兵）是Redis 的高可用性解决方案：由一个或多个Sentinel 实例 组成的Sentinel 系统可以监视任意多个主服务器，以及这些主服务器属下的所有从服务器，并在被监视的主服务器进入下线状态时，自动将下线主服务器属下的某个从服务器升级为新的主服务器，然后由新的主服务器代替已下线的主服务器继续处理命令请求。



当server1的下线时长超过用户设定的下线时长上限时，Sentinel系统就会对 server1执行故障转移操作：

* 首先，Sentinel系统会挑选server1属下的其中一个从服务器，并将这个被选中的 从服务器升级为新的主服务器。
* 之后，Sentinel系统会向server1属下的所有从服务器发送新的复制指令，让它们成为新的主服务器的从服务器，当所有从服务器都开始复制新的主服务器时，故障转移操作执行完毕。
* 另外，Sentinel还会继续监视已下线的server1,并在它重新上线时，将它设置为新的主服务器的从服务器。



2.1 启动并初始化Sentinel

|  |
| --- |
| $ redis-sentinel /path/to/your/sentinel.conf  或  $ redis-sentinel /path/to/your/sentinel.conf --sentinel |

这两个命令的效果完全相同。

当一个sentinel启动时，它需要执行以下步隳：

1).初始化服务器

2).将普通Redis服务器使用的代码替换成sentinel专用代码。

3).初始化sentinel状态的master属性。

4).根据给定的配置文件，初始化sentinel的监视主服务器列表。

5)创建连向主服务器的网络连接。

总结：

1）Sentinel只是一个运行在特殊模式下的Redis服务器，它使用了和普通模式不同的命令表，所以Sentinel模式能够使用的命令和普通Redis服务器能够使用的命令不同。

2）Sentinel会读入用户指定的配置文件，为每个要被监视的主版务器创建相应的实例结构，并创建连向主服务器的命令连接和订阅连接，其中命令连接用于向主服务器发 送命令请求，而订阅连接则用于接收指定频道的消息。

3） Sentinel通过向主服务器发送INFO命令来获得主服务器属下所有从服务器的地址信息，并为这些从服务器创建相应的实例结构，以及连向这些从服务器的命令连接和 订阅连接。

4）在一般情况下，Sentinel以每十秒一次的频率向被监视的主服务器和从服务器发送命令，当主服务器处于下线状态，或者Sentinel正在对主服务器进行故障转移 操作时.Sentinel向从服务器发送INFO命令的频率会改为每秒一次。

5）对于监视同一个主服务器和从服务器的多个Sentinel来说.它们会以每两秒一次 的 频 率 ， 通过向被监视服务器的\_\_ sentinel \_ :hello频道发送消息来向其他 Sentinel宣告自己的存在。

6）每个Sentinel也会从\_sentinel\_: hello频道中接收其他Sentinel发来的信息， 并根据这些侑息为其他Sentinel创建相应的实例结构，以及命令连接。

7） Sentinel只会与主服务器和从服务器创建命令连接和订阅连接，Sentinel与Sentinel 之间则只创建命令连接。

8) Sentinel以每秒一次的频率向实例（包括主服务器、从服务器、其他Sentinel）发送 PING命令，并根据实例对PING命令的回复来判断实例是否在线，当一个实例在栴 定的时长中连续向Sentinel发送无效回复时，Sentinel会将这个实例判断为主观下线。

9) 当Sentinel将一个主服务器判断为主观下线时，它会向同样监视这个主服务器的其他 Sentinel进行询问，看它们是否同意这个主服务器已经进入主观下线状态。

10）当Sentinel收集到足够多的主观下线投票之后，它会将主服务器判断为客观下线，并发起一次针对主服务器的故障转移操作。

三、RDB持久化

总结：

1、RDB文件用于保存和还原Redis服务器所有数据库中的所有键值对数据。

2、SAVE命令由服务器进程直接执行保存操作，所以该命令会阻塞服务器。

3、BGSAVE令由子进程执行保存操作，所以该命令不会阻塞服务器。

4、服务器状态中会保存所有用save选项设置的保存条件，当任意一个保存条件被满 足时，服务器会自动执行命令。

5、 RDB文件是一个经过压缩的二进制文件，由多个部分组成。

6、对于不同类型的键值对，RDB文件会使用不同的方式来保存它们。

四、AOF持久化

AOF 持久化的实现可以分为追加（append）、文件写入、文件同步（sync）三个步骤。

总结：

1、AOF文件通过保存所有修改数据库的写命令请求来记录服务器的数据库状态。

2、AOF文件中的所有命令都以Redis命令请求协议的格式保存。

3、命令请求会先保存到AOF缓冲区里面，之后再定期写人并同步到AOF文件

4、appendfsync选项的不同值对AOF持久化功能的安全性以及Redis服务器的性能 有很大的影响。

5、服务器只要载入并重新执行保存在AOF文件中的命令，就可以还原数据库本来的状态。 6、 AOF重写可以产生一个新的AOF文件，这个新的AOF文件和原有的AOF文件所保存的数据库状态一样，但体积更小。

7、 AOF重写是一个有歧义的名字，该功能是通过读取数据库中的键值对来实现的，程 序无须对现有AOF文件进行任何读入、分析或者写入操作。

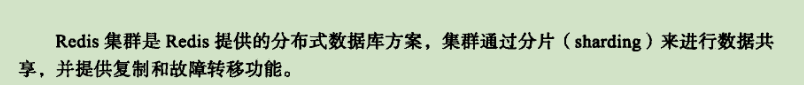
8、在执行BGREWRITEAOF命令时 ，Redis服务器会维护一个AOF重写缓冲区，该缓冲区会在子进程创建新AOF文件期间，记录服务器执行的所有写命令。当子进程完成创建新AOF文件的工作之后，服务器会将重写缓冲区中的所有内容追加到新AOF 文件的末尾，使得新旧两个AOF文件所保存的数据库状态一致。最后，服务器用新 的AOF文件替换旧的AOF文件，以此来完成AOF文件重写操作。

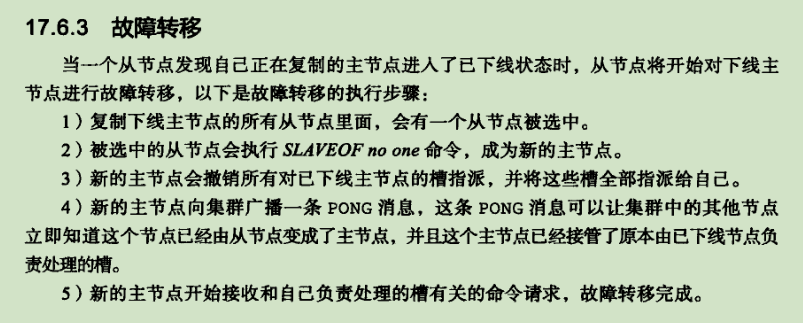
AOF 与RDB 的区别：

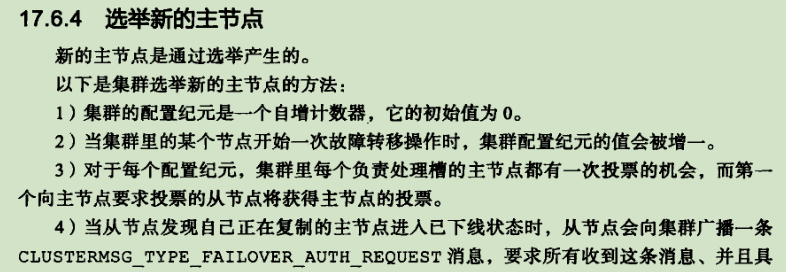
RDB 持久化通过保存数据库中的键值对来记录数据库状态不同；

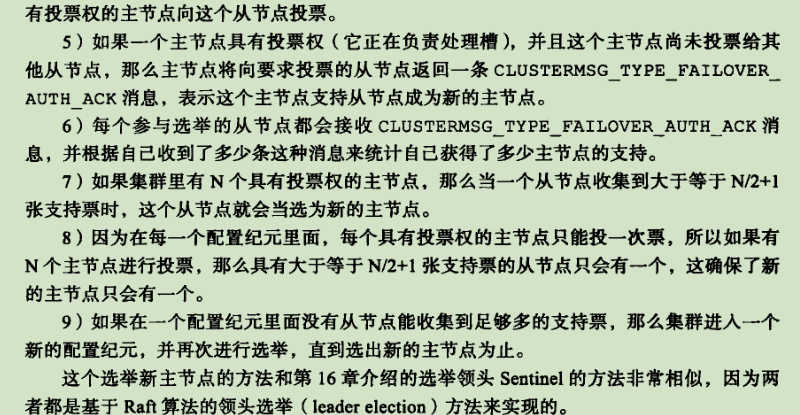
AOF持久化是通过保存Redis服务器所知晓的写命令来记录数据库状态的。

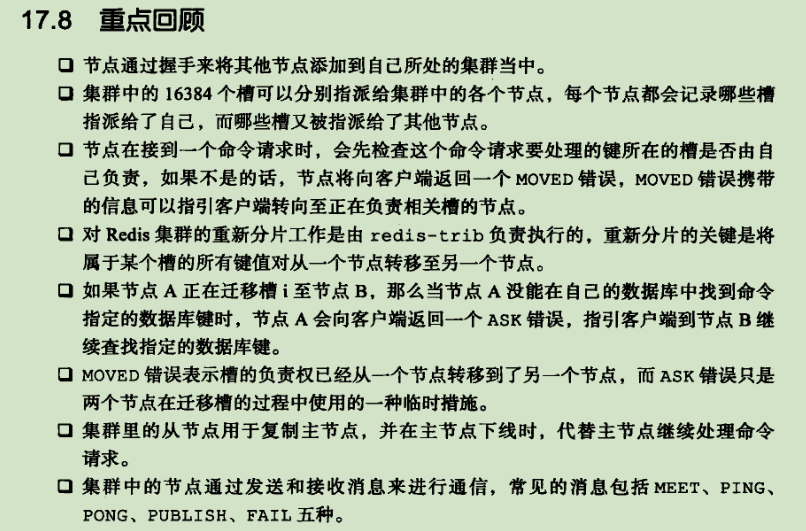
五、redis集群



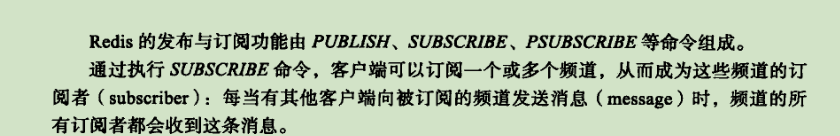


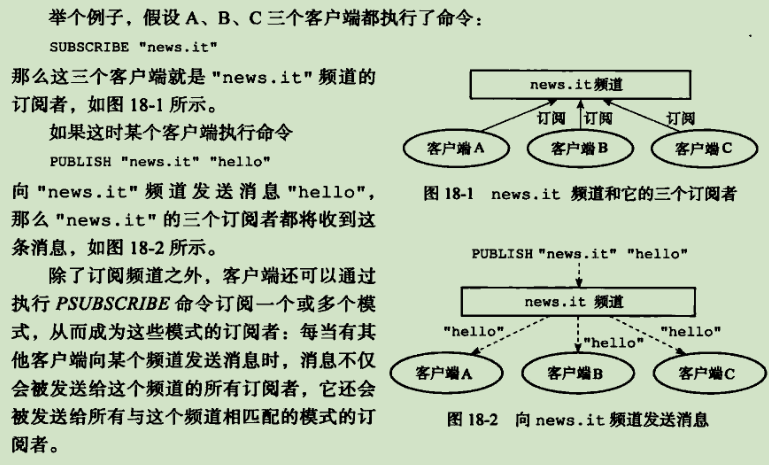


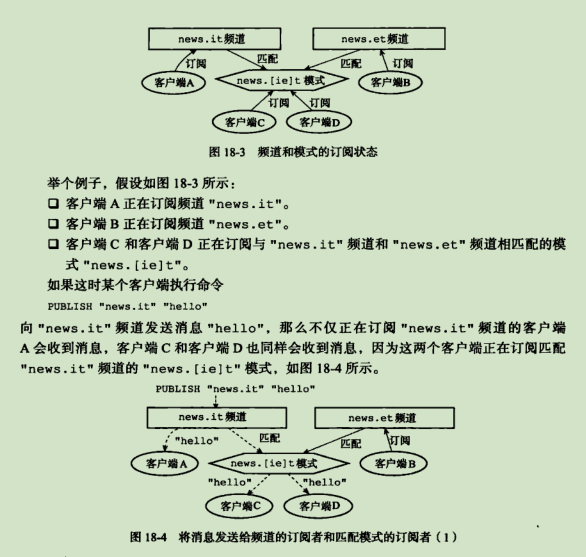


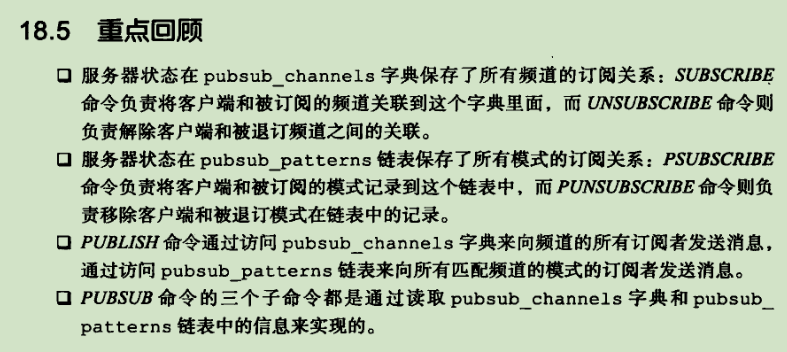


六、pub/sub









七、redis雪崩，缓存击穿

1、缓存击穿问题

缓存击穿表示恶意用户模拟请求很多缓存中不存在的数据，由于缓存中都没有，导致这些请求短时间内直接落在了数据库上，导致数据库异常。这个我们在实际项目就遇到了，有些抢购活动、秒杀活动的接口API被大量的恶意用户刷，导致短时间内数据库宕机了，好在数据库是主从结构，同时也有进行接口限流，hold的住。

解决方案的话：

方案1、使用互斥锁排队

方案2、布隆过滤器（推荐）

2、缓存雪崩问题

缓存在同一时间内大量键过期（失效），接着来的一大波请求瞬间都落在了数据库中导致连接异常。

解决方案：

方案1、也是像解决缓存穿透一样加锁排队，实现同上;

方案2、建立备份缓存，缓存A和缓存B，A设置超时时间，B不设值超时时间，先从A读缓存，A没有读B，并且更新A缓存和B缓存;

方案3、设置缓存超时时间的时候加上一个随机的时间长度，比如这个缓存key的超时时间是固定的5分钟加上随机的2分钟，酱紫可从一定程度上避免雪崩问题；