Redis项目使用总结

项目背景

基金行情及走势数据，新闻等等数据，接口数据量大，响应慢，为减轻数据库压力，保证服务的可用性，需引入分布式缓存

缓存中间件选型

现在常用的分布式缓存主要有redis与Memecache两种，但是redis功能更为强大，更为主流，对比如下

对比参数	Redis	Memcached
类型	1、支持内存 2、非关系型数据库	1、支持内存 2、key-value键值对形式 3、缓存系统
数据存储类型	1、String 2、List 3、Set 4、Hash 5、Sort Set 【俗称ZSet】	1、文本型 2、二进制类型【新版增加】
查询【操作】类型	1、批量操作 2、事务支持【虽然是假的事务】 3、每个类型不同的CRUD	1、CRUD 2、少量的其他命令
附加功能	1、发布/订阅模式 2、主从分区 3、序列化支持 4、脚本支持【Lua脚本】	1、多线程服务支持
网络IO模型	1、单进程模式	2、多线程、非阻塞IO模式
事件库	自封装简易事件库AeEvent	贵族血统的LibEvent事件库
持久化支持	1、RDB 2、AOF	不支持

集群方案选择

https://blog.csdn.net/fenglei0415/article/details/83757634
由于服务器资源限制，故采用哨兵模式部署

哨兵模式原理学习

https://www.cnblogs.com/Eugene-Jin/p/10819601.html

Sentinel的工作原理总结

1. 每个Sentinel以每秒钟一次的频率向它所知的Master，Slave以及其他 Sentinel 实例发送一个 PING 命令。
2. 如果一个实例（instance）距离最后一次有效回复 PING 命令的时间超过 down-after-milliseconds 选项所指定的值， 则这个实例会被 Sentinel 标记为主观下线。
3. 如果一个Master被标记为主观下线，则正在监视这个Master的所有 Sentinel 要以每秒一次的频率确认Master的确进入了主观下线状态。
4. 当有足够数量的 Sentinel（大于等于配置文件指定的值）在指定的时间范围内确认Master的确进入了主观下线状态， 则Master会被标记为

客观下线 。

5. 在一般情况下， 每个 Sentinel 会以每 10 秒一次的频率向它已知的所有Master，Slave发送 INFO 命令 。
6. 当Master被 Sentinel 标记为客观下线时，Sentinel 向下线的 Master 的所有 Slave 发送 INFO 命令的频率会从 10 秒一次改为每秒一次 。
7. 若没有足够数量的 Sentinel 同意 Master 已经下线， Master 的客观下线状态就会被移除。
8. 若 Master 重新向 Sentinel 的 PING 命令返回有效回复， Master 的主观下线状态就会被移除。

Redis常用命令参照

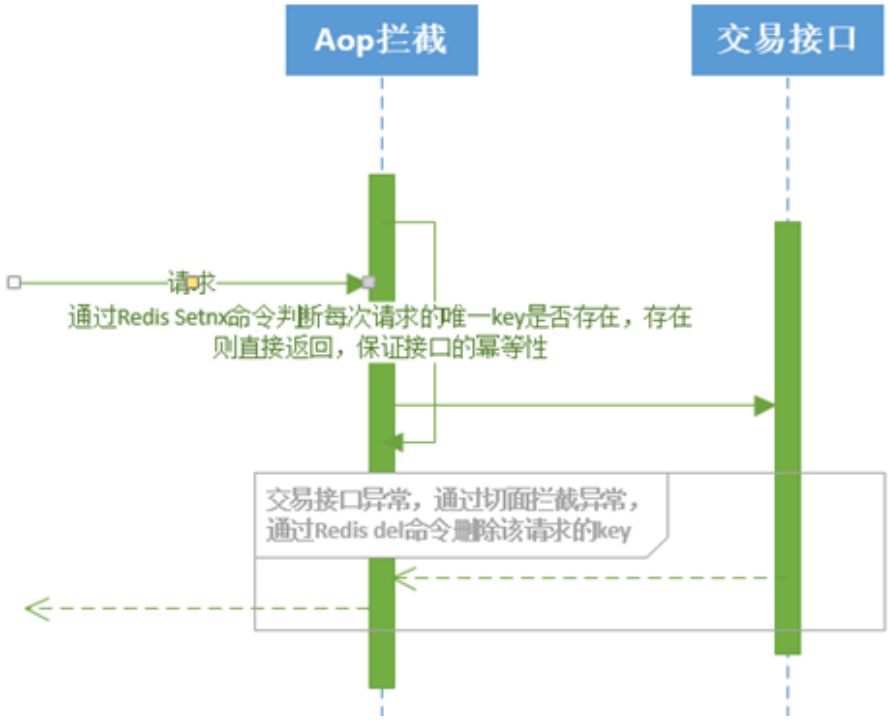
https://www.runoob.com/redis/redis-commands.html

项目使用案例

防重检测

基金项目的交易操作至关重要，需进行防重检测，通过redis分布式锁加aop技术，前端每个交易相关的请求都传递一个唯一的字符串用于后端校验，执行setnx命令时需要通过expire命令加一个短的过期时间，防止锁一直存在导致业务方法不可用，如果接口异常会直接在切面里删除锁

流程图

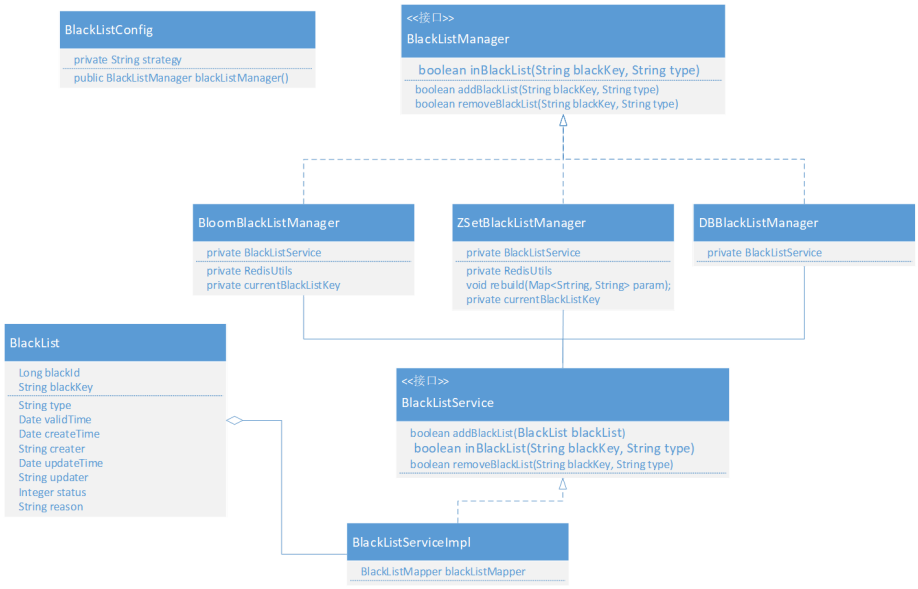


黑名单

黑名单功能，提供数据库、缓存、布隆过滤器三种实现方式，其中缓存方式基于Redis Zset数据格式实现，布隆过滤器通过基于guava改造，数据存储也通过redis实现，可用于分布式项目。

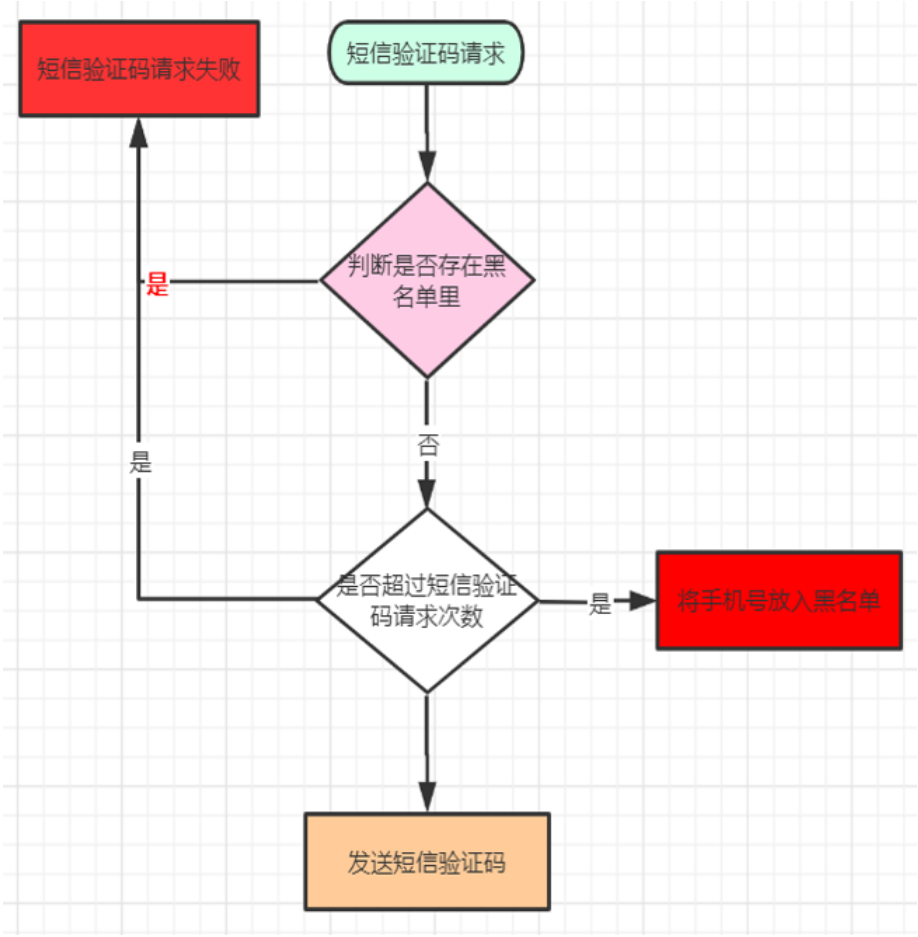
目前基金项目的短信验证码恶意防刷功能采用黑名单功能的缓存方式实现，因为目前数量级没有特别大，所以没有采用布隆。

功能实现类图



短信防刷处理流程

Redis 有序集合**(sorted set)，简称zset，**每个元素都会关联一个double类型的分数，通过分数来为集合中的成员进行从小到大的排序，黑名单功能zset实现中，以手机号为一个成员，过期日期时间戳作为score，添加到zset中，ZSCORE指令返回有序集中成员的分数值score,和当前时间戳对比即可判定。



黑名单失效数据处理：

1.数据库实现：直接批量查询数据，对比数据的过期时间

2.zset实现：批量查询redis，score为过期时间，score小于当前时间的批量删除

3.布隆过滤器实现：bitmap不支持元素删除，通过数据库数据重新构造新的bitmap，通过key删除之前的bitmap

新闻数据处理

新闻数据都放到redis中，用zset数据格式存储，新闻内容作为member,新闻发布日期作为score
用zset的好处：
1.Redis Zrevrangebyscore 返回有序集中指定分数区间内的所有的成员。从大到小，因为可以指定区间，所以分页查询比较方便
2. Redis Zremrangebyscore 命令用于移除有序集中，指定分数（score）区间内的所有成员，方便移除历史新闻数据

通用解决方案

基金相关接口众多，返回数据格式一样，单独实现每个接口的缓存实现效率低，成本大，易出错，故采用注解+AOP方式统一处理，spring已自带了缓存注解，但是存在一定局限性，所以采用自定义注解方案处理。

对比图

	SPRING	自定义
并发处理	并发处理依赖redis锁，redis挂掉后无法实现降级	并发同步锁处理，保证单个服务高并发请求不落库
库指定	不能	能
键值压缩	不支持	支持
自动降级	直接抛出异常	自动降级

长K压缩的必要性

内存浪费，在数据集中搜索对比的时候需要耗费更多的成本，可采用md5进行压缩

大V压缩的必要性

1. 内容越大需要的持久化时间就越长，需要挂起的时间越长，Redis 的性能就会越低；
2. 内容越大在网络上传输的内容就越多，需要的时间就越长，整体的运行速度就越低；
3. 内容越大占用的内存就越多，就会更频繁的触发内存淘汰机制，从而给 Redis 带来了更多的运行负担

压缩方案选择Gzip,压缩比大，性能也不错

总结

1. redis分布式锁需要把setnx和expire合成一条指令来用，防止setnx之后执行expire之前进程意外crash导致锁永远不释放
2. 一些高频数据或是不会变化的数据可以在系统启动时或者用定时任务直接加载到缓存里
3. 批量性的操作指令使用pipeline，可以将多次IO往返的时间缩减为一次，大幅提升性能
4. 缓存穿透解决方案：
    a. 对查询结果为空的情况也进行缓存，设置一个默认值，缓存时间设置短一点
    b. 布隆过滤器过滤
5. 缓存雪崩解决方案：
    a. 缓存时间过期时间加一个随机值，避免同时失效
    b. 保证redis集群的高可用性，发现机器宕机尽快修复
    c. 利用本地ehcache做二级缓存
    d. hystrix或sentinel限流+降级

问题：

1.业务背景：数据量大有很多种方案解决，静态化、CDN伪静态化、缓存都可以解决，缓存反而不是最优方案；

2.数据更新频率不一样导致缓存的使用方式是不一样的。比如在售基金现在是定时任务刷新缓存，也可以做缓存预热；

3.redis事务为什么是假事务，原理是什么，与redis的管道有什么区别；

假事务：redis的事务并不保证原子性，只能保证每个操作的隔离性，如果中间的操作失败了，也不影响后的操作，不支持回滚，所以redis事务为假事务，如果事务的指令较多可以和管道一起使用，可以将多次IO压缩为单次IO。

原理：只有当被调用的Redis命令有语法错误时，这条命令才会执行失败（在将这个命令放入事务队列期间，Redis能够发现此类问题），或者对某个键执行不符合其数据类型的操作：实际上，这就意味着只有程序错误才会导致Redis命令执行失败，这种错误很有可能在程序开发期间发现，一般很少在生产环境发现。Redis已经在系统内部进行功能简化，这样可以确保更快的运行速度，因此Redis不需要事务回滚的能力。

事务与管道的区别：事务是服务端的行为，一次批处理多条命令，而管道是客户端的行为，客户端向服务端发送一个请求，并监听Socket返回，通常是以阻塞模式，等待服务端响应。

4.单进程模式是什么意思？

redis 内部使用文件事件处理器 `file event handler`，这个文件事件处理器是单线程的，所以 redis 才叫做单线程的模型。它采用 IO 多路复用机制同时监听多个 socket，根据 socket 上的事件来选择对应的事件处理器进行处理。

文件事件处理器的结构包含 4 个部分：

- 多个 socket
- IO 多路复用程序
- 文件事件分派器
- 事件处理器（连接应答处理器、命令请求处理器、命令回复处理器）

多个 socket 可能会并发产生不同的操作，每个操作对应不同的文件事件，但是 IO 多路复用程序会监听多个 socket，会将 socket 产生的事件放入队列中排队，事件分派器每次从队列中取出一个事件，把该事件交给对应的事件处理器进行处理。

**5.集群：什么是主观下线，什么是客观下线

主观下线：所谓主观下线，就是单个sentinel认为某个服务下线（有可能是接收不到订阅，之间的网络不通等等原因）。

sentinel会以每秒一次的频率向所有与其建立了命令连接的实例（master，从服务，其他sentinel）发ping命令，通过判断ping回复是有效回复，还是无效回复来判断实例时候在线（对该sentinel来说是“主观在线”）。

sentinel配置文件中的down-after-milliseconds设置了判断主观下线的时间长度，如果实例在down-after-milliseconds毫秒内，返回的都是无效回复，那么sentinel回认为该实例已（主观）下线，修改其flags状态为SRI_S_DOWN。如果多个sentinel监视一个服务，有可能存在多个sentinel的down-after-milliseconds配置不同，这个在实际生产中要注意。

客观下线：当主观下线的节点是主节点时，此时该哨兵3节点会通过指令sentinel is-masterdown-by-addr寻求其它哨兵节点对主节点的判断，如果其他的哨兵也认为主节点主观线下了，则当认为主观下线的票数超过了quorum（选举）个数，此时哨兵节点则认为该主节点确实有问题，这样就客观下线了，大部分哨兵节点都同意下线操作，也就说是客观下线

6.黑名单功能如何实现黑名单的过期处理（zset，布隆过滤器），布隆过滤器写了就要知道原理

https://github.com/Snailclimb/JavaGuide/blob/master/docs/dataStructures-algorithms/data-structure/bloom-filter.md

7.redis的长key为什么压缩，长key为什么比对时间长

1. redis对比查找key时会进行hash，如果key过长，会让hash过程变慢，如果有一个Hash表满足特定条件需要扩容，则会申请另一个Hash表，然后把元素ReHash过来，ReHash的意思就是**重新计算每个Key的Hash值**，然后把它存放在第二个Hash表合适的位置

8.redis如何存储key value，查找key的时候做了什么https://blog.csdn.net/yangbodong22011/article/details/78467583
https://www.jianshu.com/p/7f53f5e683cf
9.如何保证redis集群的高可用性，发现机器宕机尽快修复，如何发现宕机

Redis的哨兵机制是官方推荐的一种高可用（HA）方案，我们在使用Redis的主从结构时，如果主节点挂掉，这时是不能自动进行主备切换和通知客户端主节点下线的。

Redis-Sentinel机制主要用三个功能：

(1)监控：不停监控Redis主从节点是否安装预期运行

(2)提醒：如果Redis运行出现问题可以 按照配置文件中的配置项 通知客户端或者集群管理员及时处理

(3)自动故障转移：当主节点下线之后，哨兵可以从主节点的多个从节点中选出一个为主节点，并更新配置文件和其他从节点的主节点信息。

开源redis监控工具：

https://mp.weixin.qq.com/s?__biz=MzU3NDE0NjMwNw==&mid=2247483964&idx=1&sn=eec054f008eca3b0c3d82fa8b5f7a852&chksm=fd379050ca4019461f68a5df9d1f4e1eea131f1a57dbfb4bc9d21056dcef08cb3cd031aa3395&mpshare=1&scene=1&srcid=0426MM77vhlDiw4LYfF85xHh&sharer_sharetime=1587900904254&sharer_shareid=d5b9b6e453b39732b4abc3d5dc30ef6a&rd2werd=1#wechat_redirect