**参考文档：**<https://www.jb51.net/article/139632.htm>

**Redis瞬时高并发秒杀方案总结**

 更新时间：2018年05月07日 11:39:07   作者：Brady志国   [我要评论](https://www.jb51.net/article/139632.htm#comments)

本文讲述了Redis瞬时高并发秒杀方案总结，具有很好的参考价值，感兴趣的小伙伴们可以参考一下，具体如下：

**1.Redis**

丰富的数据结构（Data Structures）

字符串（String）

    Redis字符串能包含任意类型的数据;；

     一个字符串类型的值最多能存储512M字节的内容；

    利用INCR命令簇（INCR, DECR, INCRBY）来把字符串当作原子计数器使用；

    使用APPEND命令在字符串后添加内容。

列表（List）

    Redis列表是简单的字符串列表，按照插入顺序排序；

    你可以添加一个元素到列表的头部（左边：LPUSH）或者尾部（右边：RPUSH）；

     一个列表最多可以包含232-1个元素（4294967295，每个表超过40亿个元素）；

     在社交网络中建立一个时间线模型，使用LPUSH去添加新的元素到用户时间线中，使用LRANGE去检索一些最近插入的条目；

     你可以同时使用LPUSH和LTRIM去创建一个永远不会超过指定元素数目的列表并同时记住最后的N个元素；

     列表可以用来当作消息传递的基元（primitive），例如，众所周知的用来创建后台任务的Resque Ruby库。

集合（Set）

    Redis集合是一个无序的，不允许相同成员存在的字符串合集（Uniq操作，获取某段时间所有数据排重值）；

    支持一些服务端的命令从现有的集合出发去进行集合运算，如合并（并集：union）,求交(交集：intersection)，差集, 找出不同元素的操作（共同好友、二度好友）；

    用集合跟踪一个独特的事。想要知道所有访问某个博客文章的独立IP？只要每次都用SADD来处理一个页面访问。那么你可以肯定重复的IP是不会插入的（ 利用唯一性，可以统计访问网站的所有独立IP）；

    Redis集合能很好的表示关系。你可以创建一个tagging系统，然后用集合来代表单个tag。接下来你可以用SADD命令把所有拥有tag的对象的所有ID添加进集合，这样来表示这个特定的tag。如果你想要同时有3个不同tag的所有对象的所有ID，那么你需要使用SINTER。

    使用SPOP或者SRANDMEMBER命令随机地获取元素。

哈希（Hashes）

     Redis Hashes是字符串字段和字符串值之间的映射；

     尽管Hashes主要用来表示对象，但它们也能够存储许多元素。

有序集合（Sorted Sets）

     Redis有序集合和Redis集合类似，是不包含相同字符串的合集；

    每个有序集合的成员都关联着一个评分，这个评分用于把有序集合中的成员按最低分到最高分排列（排行榜应用，取TOP N操作）；

    使用有序集合，你可以非常快地（O(log(N))）完成添加，删除和更新元素的操作；

    元素是在插入时就排好序的，所以很快地通过评分(score)或者位次(position)获得一个范围的元素（需要精准设定过期时间的应用）；

     轻易地访问任何你需要的东西: 有序的元素，快速的存在性测试，快速访问集合中间元素；

     在一个巨型在线游戏中建立一个排行榜，每当有新的记录产生时，使用ZADD 来更新它。你可以用ZRANGE轻松地获取排名靠前的用户， 你也可以提供一个用户名，然后用ZRANK获取他在排行榜中的名次。 同时使用ZRANK和ZRANGE你可以获得与指定用户有相同分数的用户名单。 所有这些操作都非常迅速；

    有序集合通常用来索引存储在Redis中的数据。 例如：如果你有很多的hash来表示用户，那么你可以使用一个有序集合，这个集合的年龄字段用来当作评分，用户ID当作值。用ZRANGEBYSCORE可以简单快速地检索到给定年龄段的所有用户。

复制（Replication, Redis复制很简单易用，它通过配置允许slave Redis Servers或者Master Servers的复制品） 一个Master可以有多个Slaves能通过接口其他slave的链接，除了可以接受同一个master下面slaves的链接以外，还可以接受同一个结构图中的其他slaves的链接redis复制是在master段是非阻塞的，这就意味着master在同一个或多个slave端执行同步的时候还可以接受查询复制在slave端也是非阻塞的，假设你在redis.conf中配置redis这个功能，当slave在执行的新的同步时，它仍可以用旧的数据信息来提供查询，否则，你可以配置当redis slaves去master失去联系是，slave会给发送一个客户端错误为了有多个slaves可以做只读查询，复制可以重复2次，甚至多次，具有可扩展性（例如：slaves对话与重复的排序操作，有多份数据冗余就相对简单了）他可以利用复制去避免在master端保存数据，只要对master端redis.conf进行配置，就可以避免保存（所有的保存操作），然后通过slave的链接，来实时的保存在slave端LRU过期处理（Eviction） EVAL 和 EVALSHA 命令是从 Redis 2.6.0 版本开始的，使用内置的 Lua 解释器，可以对 Lua 脚本进行求值Redis 使用单个 Lua 解释器去运行所有脚本，并且， Redis 也保证脚本会以原子性(atomic)的方式执行： 当某个脚本正在运行的时候，不会有其他脚本或 Redis 命令被执行。 这和使用 MULTI / EXEC 包围的事务很类似。 在其他别的客户端看来，脚本的效果(effect)要么是不可见的(not visible)，要么就是已完成的(already completed)LRU过期处理（Eviction）。

Redis允许为每一个key设置不同的过期时间，当它们到期时将自动从服务器上删除（EXPIRE）。

事务 MULTI 、 EXEC 、 DISCARD 和 WATCH 是 Redis 事务的基础事务是一个单独的隔离操作：事务中的所有命令都会序列化、按顺序地执行。事务在执行的过程中，不会被其他客户端发送来的命令请求所打断事务中的命令要么全部被执行，要么全部都不执行，EXEC 命令负责触发并执行事务中的所有命令  Redis 的 Transactions 提供的并不是严格的 ACID 的事务Transactions 还是提供了基本的命令打包执行的功能： 可以保证一连串的命令是顺序在一起执行的，中间有会有其它客户端命令插进来执行Redis 还提供了一个 Watch 功能，你可以对一个 key 进行 Watch，然后再执行 Transactions，在这过程中，如果这个 Watched 的值进行了修改，那么这个 Transactions 会发现并拒绝执行数据持久化 RDB。

特点：

RDB持久化方式能够在指定的时间间隔能对你的数据进行快照存储。

优点：

RDB是一个非常紧凑的文件,它保存了某个时间点得数据集,非常适用于数据集的备份；

RDB是一个紧凑的单一文件, 非常适用于灾难恢复；

RDB在保存RDB文件时父进程唯一需要做的就是fork出一个子进程,接下来的工作全部由子进程来做，父进程不需要再做其他IO操作，所以RDB持久化方式可以最大化redis的性能；

与AOF相比,在恢复大的数据集的时候，RDB方式会更快一些。

缺点：

如果你希望在redis意外停止工作（例如电源中断）的情况下丢失的数据最少的话，那么RDB不适合，Redis要完整的保存整个数据集是一个比较繁重的工作。

RDB 需要经常fork子进程来保存数据集到硬盘上,当数据集比较大的时候,fork的过程是非常耗时的,可能会导致Redis在一些毫秒级内不能响应客户端的请求.如果数据集巨大并且CPU性能不是很好的情况下,这种情况会持续1秒,AOF也需要fork,但是你可以调节重写日志文件的频率来提高数据集的耐久度。

AOF

特点

AOF持久化方式记录每次对服务器写的操作；

redis重启的时候会优先载入AOF文件来恢复原始的数据,因为在通常情况下AOF文件保存的数据集要比RDB文件保存的数据集要完整。

优点

     使用AOF 会让你的Redis更加耐久: 你可以使用不同的fsync策略：无fsync,每秒fsync,每次写的时候fsync；

     AOF文件是一个只进行追加的日志文件,所以不需要写入seek；

    Redis 可以在 AOF 文件体积变得过大时，自动地在后台对AOF 进行重写；

    AOF 文件有序地保存了对数据库执行的所有写入操作， 这些写入操作以 Redis 协议的格式保存， 因此 AOF 文件的内容非常容易被人读懂， 对文件进行分析（parse）也很轻松。 导出（export） AOF 文件也非常简单；

缺点

    对于相同的数据集来说，AOF 文件的体积通常要大于 RDB 文件的体积；

    根据所使用的 fsync 策略，AOF 的速度可能会慢于 RDB。

选择

     同时使用两种持久化功能。

分布式 Redis Cluster （Redis 3版本）Keepalived

     当Master挂了后，VIP漂移到Slave；Slave 上keepalived 通知redis 执行：slaveof no one ,开始提供业务；

     当Master起来后，VIP 地址不变，Master的keepalived 通知redis 执行slaveof slave IP host ，开始作为从同步数据；

    依次类推。

Twemproxy

     快、轻量级、减少后端Cache Server连接数、易配置、支持ketama、modula、random、常用hash 分片算法；

     对于客户端而言，redis集群是透明的，客户端简单，遍于动态扩容；

    Proxy为单点、处理一致性hash时，集群节点可用性检测不存在脑裂问题；

     高性能，CPU密集型，而redis节点集群多CPU资源冗余，可部署在redis节点集群上，不需要额外设备。

高可用（HA） Redis Sentinel（redis自带的集群管理工具 ）

     监控（Monitoring）： Redis Sentinel实时监控主服务器和从服务器运行状态；

    提醒（Notification）：当被监控的某个 Redis 服务器出现问题时， Redis Sentinel 可以向系统管理员发送通知， 也可以通过API 向其他程序发送通知；

     自动故障转移（Automatic failover）： 当一个主服务器不能正常工作时，Redis Sentinel 可以将一个从服务器升级为主服务器， 并对其他从服务器进行配置，让它们使用新的主服务器。当应用程序连接到 Redis 服务器时， Redis Sentinel会告之新的主服务器地址和端口。

单M-S结构

    单M-S结构特点是在Master服务器中配置Master Redis（Redis-1M）和Master Sentinel（Sentinel-1M）；

    Slave服务器中配置Slave Redis（Redis-1S）和Slave Sentinel（Sentinel-1S）；

     其中 Master Redis可以提供读写服务，但是Slave Redis只能提供只读服务。因此，在业务压力比较大的情况下，可以选择将只读业务放在Slave Redis中进行。

双M-S结构

    双M-S结构的特点是在每台服务器上配置一个Master Redis，同时部署一个Slave Redis。由两个Redis Sentinel同时对4个Redis进行监控。两个Master Redis可以同时对应用程序提供读写服务，即便其中一个服务器出现故障，另一个服务器也可以同时运行两个Master Redis提供读写服务；

    缺点是两个Master redis之间无法实现数据共享，不适合存在大量用户数据关联的应用使用。

单M-S结构和双M-S结构比较

    单M-S结构适用于不同用户数据存在关联，但应用可以实现读写分离的业务模式。Master主要提供写操作，Slave主要提供读操作，充分利用硬件资源；

    双（多）M-S结构适用于用户间不存在或者存在较少的数据关联的业务模式，读写效率是单M-S的两（多）倍，但要求故障时单台服务器能够承担两个Mater Redis的资源需求；

    发布/订阅（Pub/Sub）监控：Redis-Monitor 历史redis运行查询：CPU、内存、命中率、请求量、主从切换等实时监控曲线。

**2.数据类型Redis使用场景**

String

计数器应用

List

    取最新N个数据的操作；

    消息队列；

    删除与过滤；

    实时分析正在发生的情况，用于数据统计与防止垃圾邮件（结合Set）；

Set Uniqe操作，获取某段时间所有数据排重值实时系统，反垃圾系统共同好友、二度好友利用唯一性，可以统计访问网站的所有独立 IP好友推荐的时候，根据 tag 求交集，大于某个 threshold 就可以推荐Hashes。

    存储、读取、修改用户属性；

Sorted Set 排行榜应用，取TOP N操作需要精准设定过期时间的应用（时间戳作为Score）带有权重的元素，比如一个游戏的用户得分排行榜过期项目处理，按照时间排序。

**3.Redis解决秒杀/抢红包等高并发事务活动**

    秒杀开始前30分钟把秒杀库存从数据库同步到Redis Sorted Set；

    用户秒杀库存放入秒杀限制数长度的Sorted Set；

    秒杀到指定秒杀数后，Sorted Set不在接受秒杀请求，并显示返回标识；

    秒杀活动完全结束后，同步Redis数据到数据库，秒杀正式结束。

以上就是本文的全部内容，希望对大家的学习有所帮助，也希望大家多多支持脚本之家。