3/28/2020

Stone Sun

Redis

Redis technical doc

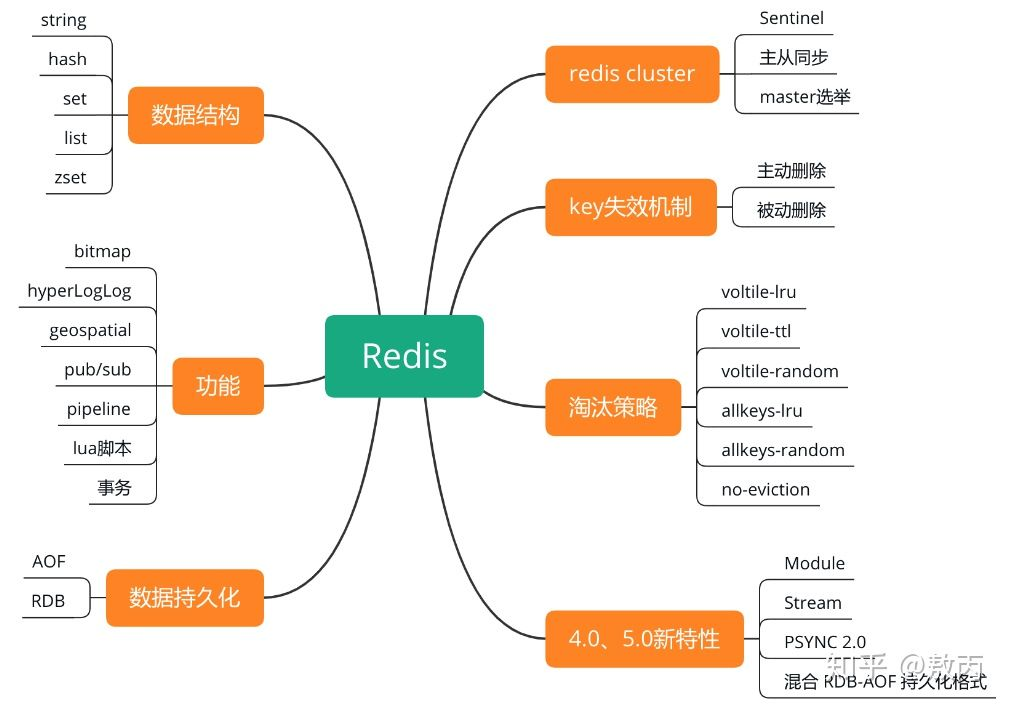
|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Comments |
| 2020/03/28 | 0.1 | Stone Sun | Internal Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Version RecordRedis概述

1. NoSQL内存数据库
2. 持久化机制把内存中的数据同步到硬盘，重启之后，会加载到内存中

实现：单独创建fork()一个子进程，将当前父进程的数据库数据复制到子进程的内存中，然后由子进程写入到临时文件中，持久化的过程结束了，再用这个临时文件替换上次的快照文件，然后子进程退出，内存释放。

1. 分片来扩展写性能。



# 缓存类型

缓存的类型分为：本地缓存、分布式缓存和多级缓存。

**本地缓存**：

本地缓存就是在进程的内存中进行缓存，比如我们的 JVM 堆中，可以用 LRUMap 来实现，也可以使用 Ehcache 这样的工具来实现。

本地缓存是内存访问，没有远程交互开销，性能最好，但是受限于单机容量，一般缓存较小且无法扩展。

**分布式缓存**：

分布式缓存可以很好得解决这个问题。

分布式缓存一般都具有良好的水平扩展能力，对较大数据量的场景也能应付自如。缺点就是需要进行远程请求，性能不如本地缓存。

**多级缓存**：

为了平衡这种情况，实际业务中一般采用多级缓存，本地缓存只保存访问频率最高的部分热点数据，其他的热点数据放在分布式缓存中。

在目前的一线大厂中，这也是最常用的缓存方案，单考单一的缓存方案往往难以撑住很多高并发的场景。

# 为什么快

(一)纯内存操作

(二)单线程操作，避免了频繁的上下文切换

(三)采用了非阻塞I/O多路复用机制

# Redis结构内部实现

<https://mp.weixin.qq.com/s?__biz=MzA4NTg1MjM0Mg==&mid=2657261203&idx=1&sn=f7ff61ce42e29b874a8026683875bbb1&scene=21#wechat_redirect>



# 缓存类型

# 缓存类型

# 缓存淘汰策略

不管是本地缓存还是分布式缓存，为了保证较高性能，都是使用内存来保存数据，由于成本和内存限制，当存储的数据超过缓存容量时，需要对缓存的数据进行剔除。

一般的剔除策略有 **FIFO** 淘汰最早数据、**LRU(Least Recently Used)** 剔除最近最少使用、和 **LFU(Least Frequently Used)** 剔除最近使用频率最低的数据几种策略。

noeviction:返回错误当内存限制达到并且客户端尝试执行会让更多内存被使用的命令（大部分的写入指令，但DEL和几个例外）

allkeys-lru: 尝试回收最少使用的键（LRU），使得新添加的数据有空间存放。

volatile-lru: 尝试回收最少使用的键（LRU），但仅限于在过期集合的键,使得新添加的数据有空间存放。

allkeys-random: 回收随机的键使得新添加的数据有空间存放。

volatile-random: 回收随机的键使得新添加的数据有空间存放，但仅限于在过期集合的键。

volatile-ttl: 回收在过期集合的键，并且优先回收存活时间（TTL）较短的键,使得新添加的数据有空间存放。

如果没有键满足回收的前提条件的话，策略volatile-lru, volatile-random以及volatile-ttl就和noeviction 差不多了。

# 使用场景

1. 计数器，string类型
2. 缓存
3. 会话缓存

可以使用 Redis 来统一存储多台应用服务器的会话信息，当应用服务器不再存储用户的会话信息，也就不再具有状态，一个用户可以请求任意一个应用服务器，从而更容易实现高可用性以及可伸缩性。

1. 分布式锁实现
2. Top 10：sorted set
3. 就是利用交集、并集、差集等操作，可以计算共同喜好，全部的喜好，自己独有的喜好等功能

# Memcached与Redis的区别都有哪些？

两者都是非关系型内存键值数据库，主要有以下不同：

1. 数据类型

Memcached 仅支持字符串类型，而 Redis 支持五种不同的数据类型，可以更灵活地解决问题

1. 数据持久化

Redis 支持两种持久化策略：RDB 快照和 AOF 日志，而 Memcached 不支持持久化。

1. 分布式

Memcached不支持

Redis Cluster

1. 内存管理机制

在 Redis 中，并不是所有数据都一直存储在内存中，可以将一些很久没用的 value 交换到磁盘，而 Memcached 的数据则会一直在内存中。

Memcached 将内存分割成特定长度的块来存储数据，以完全解决内存碎片的问题。但是这种方式会使得内存的利用率不高，例如块的大小为 128 bytes，只存储 100 bytes 的数据，那么剩下的 28 bytes 就浪费掉了。

# 一致性问题

写的时候先写入数据库，在写入Redis，

读取的时候，先从Redis读取，没读到在从数据库读取，并写入Redis

# 使用考虑

1. 业务数据常用吗？命中率如何？如果命中率很低，就没有必要写入缓存；
2. 该业务数据是读操作多，还是写操作多？如果写操作多，频繁需要写入数据库，也没有必要使用缓存；
3. 业务数据大小如何？如果要存储几百兆字节的文件，会给缓存带来很大的压力，这样也没有必要；

# 持久化

1. RDB 持久化

将某个时间点的所有数据都存放到硬盘上。

可以将快照复制到其它服务器从而创建具有相同数据的服务器副本。

如果系统发生故障，将会丢失最后一次创建快照之后的数据。

如果数据量很大，保存快照的时间会很长。

1. AOF持久化

使用 AOF 持久化需要设置同步选项，从而确保写命令同步到磁盘文件上的时机。这是因为对文件进行写入并不会马上将内容同步到磁盘上，而是先存储到缓冲区，然后由操作系统决定什么时候同步到磁盘。有以下同步选项：



* always 选项会严重减低服务器的性能；
* everysec 选项比较合适，可以保证系统崩溃时只会丢失一秒左右的数据，并且 Redis 每秒执行一次同步对服务器性能几乎没有任何影响；
* no 选项并不能给服务器性能带来多大的提升，而且也会增加系统崩溃时数据丢失的数量。

随着服务器写请求的增多，AOF 文件会越来越大。Redis 提供了一种将 AOF 重写的特性，能够去除 AOF 文件中的冗余写命令。

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型

# 缓存类型