Redis 面试必备知识点

作者：jingqueyimu
原文链接：https://ld246.com/article/1620219480387
来源网站：链滴

1、Redis 简介

- 是一个用 C 语言开发的，高性能的键值对数据库。
- 数据存在于内存，读写速度快。
- 可用来做缓存、分布式锁、消息队列。
- 提供多种数据类型来支持不同的业务场景。
- 支持事务、持久化、Lua 脚本、多种集群方案。

2、Redis 与 Memcached 对比

共同点：

- 都是基于内存的数据库，常用来做缓存。
- 都有过期策略。
- 性能都非常高。

区别：

- Redis 支持多种数据类型；而 Memcached 只支持 string。
- Redis 支持数据持久化；而 Memcached 不支持。
- Redis 有灾难恢复机制（通过加载持久化数据实现）；而 Memcached 没有。
- Redis 内存不足时，可以将不用的数据放到磁盘上；而 Memcached 会直接报异常。
- Redis 原生支持集群模式；而 Memcached 没有原生的集群模式，需要客户端实现。
- Redis 使用单线程、IO 多路复用的网络模型（Redis 6.0 引入了多线程 IO）；而 Memcached 使多线程、非阻塞 IO 复用的网络模式。
- Redis 支持发布订阅模型、Lua 脚本、事务等功能；而 Memcached 不支持。
- Redis 过期数据的删除策略使用了惰性删除和定期删除；而 Memcached 只用了惰性删除。

3、为什么要用 Redis

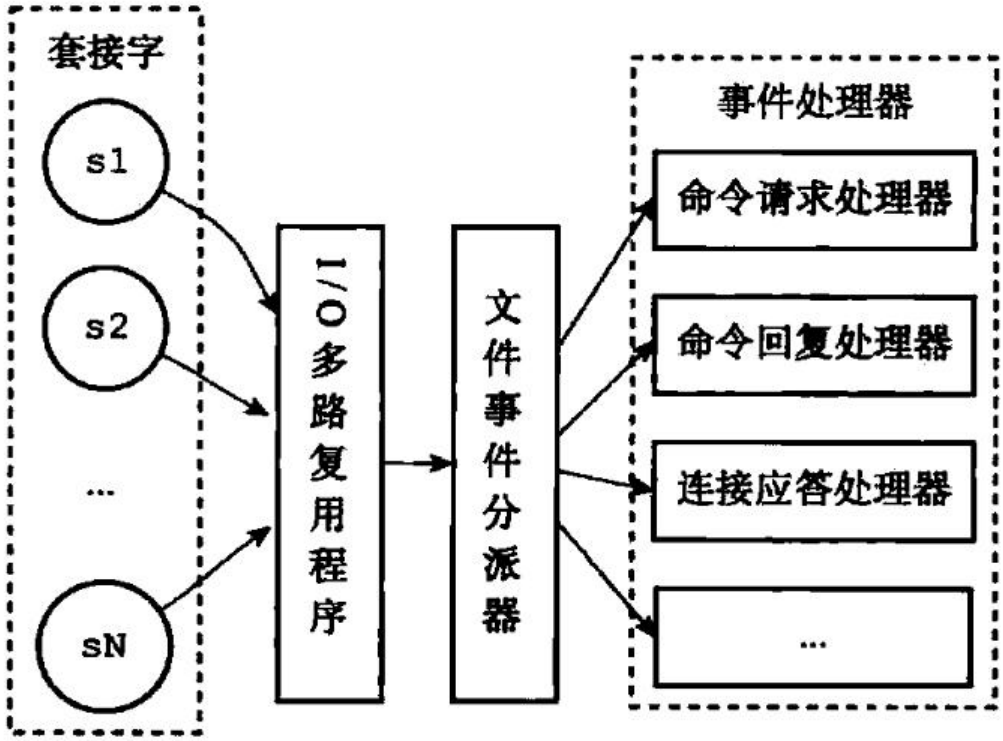
- 高性能：Redis 的数据存在于内存，读写速度快。
- 高并发：单机情况下，MySQL 这类数据库 QPS 在 1w 左右（4 核 8G），而 Redis 可达到 10w+ 甚至 30w+。

4、Redis 数据类型

- string：简单动态字符串（SDS）。
  - 常用命令：set、get、strlen、decr、incr、setex 等。
  - 应用场景：需要计数的场景，比如用户访问量、文章转发量等。
- list：双向链表。
  - 常用命令：lpush、rpop、rpush、lpop、lrange、llen 等。

- 应用场景：发布订阅（即消息队列）。
- hash：类似 Java 的 HashMap，底层为数组 + 链表。
  - 常用命令：hset、hget、hexists、hkeys、hvals 等
  - 应用场景：存储对象数据。
- set：类似 Java 的 HashSet，无序、去重，可实现交集、并集、差集的操作。
  - 常用命令：sadd、spop、smembers、sismember、scard、sunion 等。
  - 应用场景：需要存放的数据不能重复、需要获取多个数据源的交集、并集、差集。
- sorted set（zset）：和 set 相比，增加了一个权重参数 score，能够按 score 进行排序。类似 Java 的 TreeSet，有序、去重。
  - 常用命令：zadd、zcard、zscore、zrange、zrevrange 等。
  - 应用场景：需要根据某个权重进行排序的场景，比如直播系统中的礼物排行榜、消息弹幕等。

5、Redis 单线程模型



- Redis 基于 Reactor 模式来开发自己的网络事件处理器，这个处理器被称为文件事件处理器。
- 由于文件事件处理器是单线程的，所以我们一般都说 Redis 是单线程模型。
- Redis 通过 IO 多路复用来监听多个客户端连接（或者说监听多个 socket）。
- 文件事件处理器主要包含四个部分：
  - 多个 socket：客户端连接。
  - IO 多路复用程序：支持多个客户端连接的关键。

- 文件事件分派器：将 socket 关联到相应的事件处理器。
  - 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。

6、Redis 单线程为什么那么快

- 数据存在于内存，读写速度快。
- 单线程操作，避免了死锁、上下文切换带来的性能开销。
- 采用非阻塞 IO 多路复用机制。

7、Redis 6.0 之前为什么使用单线程

- 单线程编程容易，并且更容易维护。
- Redis 的性能瓶颈不在 CPU，主要在内存和网络。
- 多线程存在死锁、线程上下文切换等问题，甚至会影响性能。

8、Redis 6.0 之后为何引入了多线程

- 为了提高网络 IO 读写性能。
- Redis 只在网络数据读写这类耗时操作上使用了多线程，执行命令仍然是单线程顺序执行，因此不要担心线程安全问题。

9、Redis 如何判断数据是否过期

- 通过过期字典（hash 表）保存数据过期时间。
- 过期字典的键指向 Redis 中的某个键，过期字典的值保存了数据的过期时间（毫秒精度的时间戳）。

10、Redis 过期数据的删除策略

- 定时删除：设置过期时间时，同时创建一个定时器，定时器会在过期时间到来时自动删除过期数据节约内存，但 CPU 压力大。
- 定期删除：每隔一段时间抽取一批数据进行过期检查。内存、CPU 压力都不是很大。
- 惰性删除：在取出数据时进行过期检查。节约 CPU 性能、但内存压力大。

Redis 采用定期删除 + 惰性删除。但仍会漏掉某些过期数据，可能导致大量过期数据堆积在内存，从导致内存溢出，Redis 通过内存淘汰机制来解决这个问题。

11、Redis 内存淘汰机制

- volatile-lru（least recently used）：从已设置过期时间的数据中，淘汰最近最少使用的数据。
- volatile-ttl：从已设置过期时间的数据中，淘汰将要过期的数据。
- volatile-random：从已设置过期时间的数据中，随机淘汰数据。
- allkeys-lru（least recently used）：从所有数据中，淘汰最近最少使用的数据。
- allkeys-random：从所有数据中，随机淘汰数据。

- noeviction：禁止淘汰数据，内存不足时直接报错。

4.0 版本后增加两种：

- volatile-lfu（least frequently used）：从已设置过期时间的数据中，淘汰最不经常使用的数据。
- allkeys-lfu（least frequently used）：从所有数据中，淘汰最不经常使用的数据。

12、Redis 持久化机制

- RDB 持久化：默认方式，将某个时刻内存中的数据以快照的形式保存在磁盘上。文件小、恢复速度，对性能影响小，但是实时性差。
- AOF 持久化：以日志的方式将每一条写命令保存到磁盘的文件上。文件大、恢复速度慢，对性能影大，但是实时性高。目前 Redis 持久化的主流方式。
    - 一般设置为每秒同步一次 AOF 文件：appendfsync 选项设置为 everysec。
    - AOF 重写（bgrewriteaof 命令）：将内存中的数据以命令的方式保存到新的 AOF 文件中，然用新的 AOF 文件替换旧的 AOF 文件。

13、Redis 事务

- 相关命令：
    - MULTI：开启事务。
    - EXEC：执行事务中的所有命令。
    - DISCARD：取消事务，放弃执行事务中的所有命令。
    - WATCH：监听一个或多个 key，如果事务在执行前，监听的 key 被其他命令修改，则中断事务不会执行事务中的任何命令。
    - UNWATCH：取消 WATCH 对所有 key 的监听。
- 使用 MULTI 命令后可输入多个命令，Redis 不会立即执行这些命令，而是会放到队列中，等调用 EXEC 命令后再原子化执行这些命令。
- Redis 不支持回滚，因此不满足原子性。
- 当 Redis 运行在 AOF 持久化模式下，并且 appendfsync 选项设为 always 时，具有持久性。

14、Redis 集群（多机）

- 主从模式：主库（master）负责读写，从库（slave）负责读。不具备高可用。
    - master 挂了，不影响 slave，只是不再提供写服务。
    - master 挂了，不会在 slave 中选一个 master。
- 哨兵（Sentinel）模式：通过 sentinel 进程监视 master、slave 的运行状态。
    - 哨兵模式是建立在主从模式的基础上。
    - 为了避免 sentinel 挂掉，一般会做 sentinel 集群。并且不和 master、slave 部署在同一台机器。
    - sentinel 会每秒向 master、slave 以及其他 sentinel 实例发送一个 PING 命令，以监视其运行态。

- master 挂了，sentinel 会在 slave 中选一个 master，并修改所有实例的配置。
  - master 重启后，将作为 slave 运行。

- 集群（Cluster）模式：Redis 提供的分布式数据库方案。该模式是为了解决单机 Redis 容量上限的题，它通过分片来进行数据共享。
  - 由多个节点组成，每个节点包含一主一从（或一主多从），从库仅作为备用。
  - 客户端可以连接任一节点的主库进行读写。
  - 集群的整个数据库被分为 16384 个槽（slot），当存储一个 key-value 时，会被分配在某个槽

。

15、缓存穿透

- 大量请求不存在缓存中的 key，导致请求落在数据库上。
- 解决办法：
  - 拦截非法 key：拦截所有一定不存在数据的 key。一般采用布隆过滤器。
  - 缓存空值：即使查询结果为空，也将这个空结果缓存，但设置一个较短的过期时间。

16、缓存雪崩

- 大量缓存在同一时间失效，或者一些被大量访问的缓存（热点缓存）在某一时刻失效，导致大量请落在数据库上。
- 解决办法：
  - 设置不同的过期时间。
  - 限流，避免同时处理大量请求。
  - 设置热点缓存永不失效。

Reference

[1] https://snailclimb.gitee.io/javaguide-interview/#/./docs/d-2-redis

[2] https://blog.csdn.net/miss1181248983/article/details/90056960

[3] 《Redis 设计与实现》