Redis常见面试题精选（一）

Original Edwin 彬 星河之码 8/8

> **由经验而得的智慧，胜于书本而得的理论。**

之前的文章中针对Redis的使用，原理以及高可用都做了一个简单总结。这篇文章主要总结收集了一部分Redis的高频面试题。

1. 为什么要用缓存

用缓存，主要有两个用途：**高性能、高并发。**

- **高性能**

由于**缓存是在内存中，查询缓存会比查询数据库快很多**，在性能上能够使得系统大幅度提升。所以在长时间不会变化的数据可以放在缓存中，直接查询缓存即可。

- **高并发**

mysql 单机支撑到 `2000QPS` 就会开始报警了，不支持高并发。

Redis缓存单机支撑的并发量一秒几万十几万，支撑高并发。**单机承载并发量是 mysql 单机的几十倍。**

> 缓存是走内存的，内存天然就支撑高并发。

2. 用了缓存之后会有什么不良后果

- 缓存与数据库双写不一致
- 缓存雪崩、缓存穿透、缓存击穿
- 缓存并发竞争

3. 使用redis有哪些好处

- 速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)
- 支持丰富数据类型，支持string，list，set，sorted set，hash
- 支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行
- 丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除

4. 如何保证缓存与数据库的双写一致性

一般来说，如果系统**不是严格要求** 缓存和数据库必须保持一致性的话，最好不要做：**读请求和写请求串行化**，串到一个**内存队列**里去。

串行化可以保证一定不会出现不一致的情况，**但是它也会导致系统的吞吐量大幅度降低**，用比正常情况下多几倍的机器去支撑线上的一个请求。

一般常用的缓存方案有两种：

- **第一种**
  - 读的时候，先读缓存，缓存没有的话，读数据库，取出数据后放入缓存，同时返回响应。
  - 更新的时候，先删除缓存，在更新数据库。
- **第二种**
  - 读的时候，先读缓存，缓存没有的话，读数据库，取出数据后放入缓存，同时返回响应。
  - 更新的时候，**先更新数据库，再删除缓存。**

第二种是Cache Aside Pattern的原本思路。这两种都会造双写不一致，成会造成这两种分歧原因因在于：

- 第一种：

  a线程删除缓存 - b线程读数据库值为A并设置缓存值为A - a线程更新数据库值为B

- 第二种有两种情况：
  - a线程读数据库值为A - b线程更新数据库值为B - b线程删除缓存 - a线程设置缓存值为A；
  - a线程更新数据库值为B-由于网络延迟或宕机没有删除缓存-系统恢复后-b线程读缓存值为A。

另外还有一种导致缓存数据库不一致的原因还有读写分离，由于主从同步延迟，如果采取上面的两种方案，在极端情况下（从库读延迟），也有可能导致读请求写入缓存中的可能是旧数据。

双写不一致解决方案

一般来说，我们对缓存只要求最终一致性。前面两种方案，再来一次更新请求只要不发生同样的情况，缓存都会被再次刷成一致的。解决方案有三种：

- **缓存过期时间兜底**

  就算更新操作非常少，没有更新操作，也有一个缓存过期时间，在缓存过期之后再次刷新缓存。

- **串行化更新数据库和写缓存**

  解决这个目标的关键主要目的是保证**删除缓存、更新数据库**和**读数据库并设置缓存**两者之间要保证串行化。基于此，可能的优化有以下几种：

  - 更新数据以及更新缓存整个过程用消息队列或加锁实现。（适用于预热场景，对某些数据进行预热）。
  - 更新数据的时候发送消息队列，更新数据库并删除缓存，读数据的时候如果没命中缓存先从数据库查出来返回，在发送消息队列，读数据库并设置缓存。

- **如果引入了读写分离**

  - 通过消费binlog日志消息，再次发送消息到mq去删除缓存，读数据若没有缓存的时候也发送消息到mq读数据并设置缓存。

5. redis相比memcached有哪些优势

- Memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型
- Redis的速度比memcached快很多
- Redis可以持久化其数据

6.Memcache与Redis的区别都有哪些

- 存储方式 Memecache把数据全部存在内存之中，断电后会挂掉，数据不能超过内存大小。Redis有部份存在硬盘上，这样能保证数据的持久性。

- 数据支持类型 Memcache对数据类型支持相对简单。Redis有复杂的数据类型。
- 使用底层模型不同 它们之间底层实现方式 以及与客户端之间通信的应用协议不一样。Redis直接自己构建了VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求。

7. redis常见性能问题和解决方案

- Master写内存快照，会阻塞主线程的工作，当快照比较大时对性能影响很大，会间断性暂停服务，所以Master最好不要写内存快照。
- Master AOF持久化，**如果不重写AOF文件，这个持久化方式对性能的影响是最小的**，但是AOF文件会不断增大，AOF文件过大会影响Master重启的恢复速度。最好将数据持久化的工作交给某个Slave，某个Slave开启AOF备份数据，策略为每秒同步一次。
- Master调用BGREWRITEAOF重写AOF文件，AOF在重写的时候会占大量的CPU和内存资源，导致服务load过高，出现短暂服务暂停现象。
- Redis主从复制的性能问题，为了主从复制的速度和连接的稳定性，Slave和Master最好在同一个局域网内

8. Redis淘汰策略

> mySQL里有2000w数据，redis中只存20w的数据，如何保证redis中的数据都是热点数据

redis 内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略（回收策略）。

redis 提供 6种数据淘汰策略：

- volatile-lru：从已设置过期时间的数据集中挑选最近最少使用的数据淘汰
- volatile-ttl：从已设置过期时间的数据集中挑选将要过期的数据淘汰
- volatile-random：从已设置过期时间的数据集）中任意选择数据淘汰
- allkeys-lru：从数据集中挑选最近最少使用的数据淘汰
- allkeys-random：从数据集中任意选择数据淘汰
- no-enviction（驱逐）：禁止驱逐数据

9.为什么 redis 单线程却能支撑高并发

- 纯内存操作。
- 核心是基于非阻塞的 IO 多路复用机制。

- C 语言实现，一般来说，C 语言实现的程序“距离”操作系统更近，执行速度相对会更快。
- 单线程反而避免了多线程的频繁上下文切换问题，预防了多线程可能产生的竞争问题。

10. redis 的线程模型是什么

详细线程模型说明可以参考之前的文章**Redis线程模型**

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，将产生事件的 socket 压入内存队列中，事件分派器根据 socket 上的事件类型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5个部分：

- 多个 socket
- IO 多路复用程序
- scocket队列
- 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）

11. 缓存雪崩、缓存穿透、缓存击穿

详细缓存说明可以参考之前的文章**缓存雪崩、缓存穿透、缓存击穿**

- 穿透：不存在的key
- 雪崩：大量的key失效
- 击穿：一个key或一些key 热点key

解决了一个小问题——读源码真的只是为了应付面试?

三分恶