Redis缓存和MySQL数据一致性的解决方案

题目标签

学习时长：20分钟

题目难度：中等

知识点标签：Redis、MySQL

题目描述

Redis缓存和MySQL数据一致性的解决方案

1. 面试题分析

根据题目要求我们可以知道：

- 采用延时双删策略保证缓存和mysql数据一致
- 异步更新缓存保证缓存和mysql数据一致

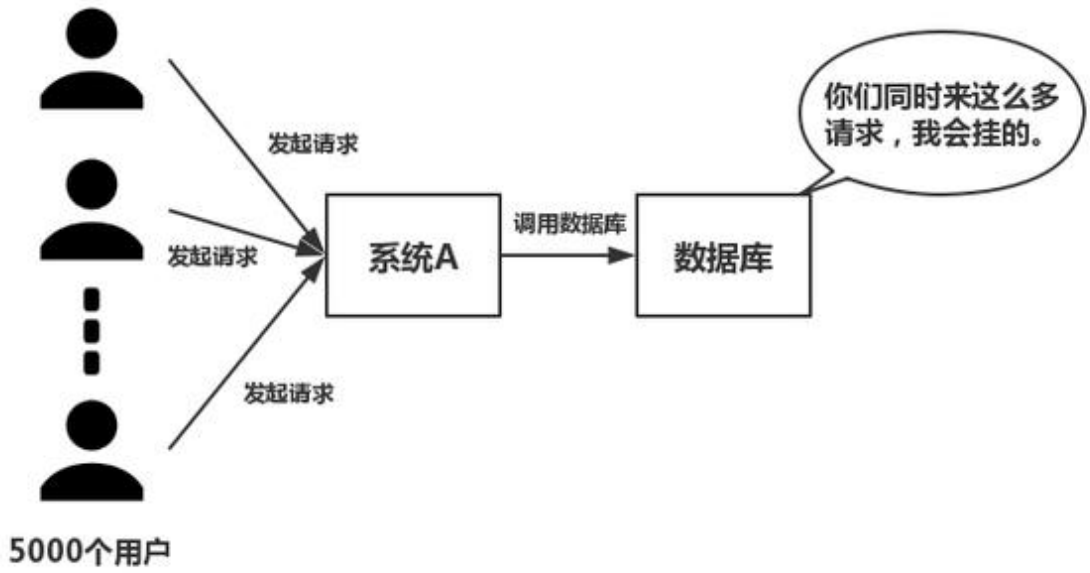
> 分析需要全面并且有深度

容易被忽略的坑

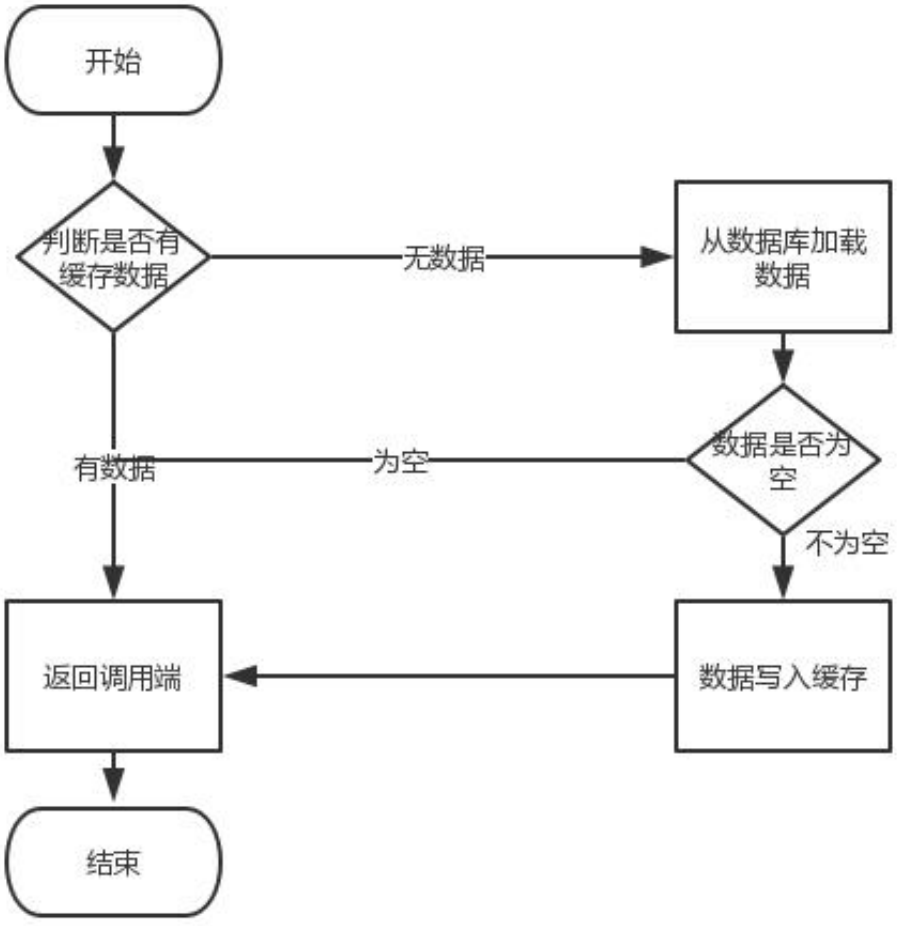
- 分析片面
- 没有深入

2. 需求分析

在高并发的业务场景下，数据库大多数情况都是用户并发访问最薄弱的环节。所以，就需要使用redis做一个缓冲操作，让请求先访问到redis，而不是直接访问MySQL等数据库。



这个业务场景，主要是解决读数据从Redis缓存，一般都是按照下图的流程来进行业务操作。



读取缓存步骤一般没有什么问题，但是一旦涉及到数据更新：数据库和缓存更新，就容易出现**缓存(Redis)和数据库（MySQL）间的数据一致性问题**。

不管是先写MySQL数据库，再删除Redis缓存；还是先删除缓存，再写库，都有可能出现数据不一致的情况。举一个例子：

1.如果删除了缓存Redis，还没有来得及写库MySQL，另一个线程就来读取，发现缓存为空，则去数据库中读取数据写入缓存，此时缓存中为脏数据。

2.如果先写了库，在删除缓存前，写库的线程宕机了，没有删除掉缓存，则也会出现数据不一致情况。

因为写和读是并发的，没法保证顺序,就会出现缓存和数据库的数据不一致的问题。

如来解决？这里给出两个解决方案，先易后难，结合业务和技术代价选择使用。

缓存和数据库一致性解决方案

第一种方案：采用延时双删策略

在写库前后都进行redis.del(key)操作，并且设定合理的超时时间。

伪代码如下

```
public void write(String key,Object data){
 redis.delKey(key);
 db.updateData(data);
 Thread.sleep(500);
 redis.delKey(key);
 }
```

1.具体的步骤

1）先删除缓存

2）再写数据库

3）休眠500毫秒

4）再次删除缓存

那么，这个500毫秒怎么确定的，具体该休眠多久呢?

需要评估自己的项目的读数据业务逻辑的耗时。这么做的目的，就是确保读请求结束，写请求可以删除读请求造成的缓存脏数据。

当然这种策略还要考虑redis和数据库主从同步的耗时。最后的的写数据的休眠时间：则在读数据业务逻辑的耗时基础上，加几百ms即可。比如：休眠1秒。

2.设置缓存过期时间

从理论上来说，给缓存设置过期时间，是保证最终一致性的解决方案。所有的写操作以数据库为准，只要到达缓存过期时间，则后面的读请求自然会从数据库中读取新值然后回填缓存。

3.该方案的弊端

结合双删策略+缓存超时设置，这样最差的情况就是在超时时间内数据存在不一致，而且又增加了写请求的耗时。

第二种方案：异步更新缓存(基于订阅binlog的同步机制)

1.技术整体思路：

MySQL binlog增量订阅消费+消息队列+增量数据更新到redis

1）读Redis：热数据基本都在Redis

2）写MySQL:增删改都是操作MySQL

3）更新Redis数据：MySQ的数据操作binlog，来更新到Redis

2.Redis更新

1）数据操作主要分为两大块：

- 一个是全量(将全部数据一次写入到redis)
- 一个是增量（实时更新）

这里说的是增量,指的是mysql的update、insert、delate变更数据。

2）读取binlog后分析，利用消息队列,推送更新各台的redis缓存数据。

这样一旦MySQL中产生了新的写入、更新、删除等操作，就可以把binlog相关的消息推送至Redis，Redis再根据binlog中的记录，对Redis进行更新。

其实这种机制，很类似MySQL的主从备份机制，因为MySQL的主备也是通过binlog来实现的数据一致性。

这里可以结合使用canal(阿里的一款开源框架)，通过该框架可以对MySQL的binlog进行订阅，而canal正是模仿了mysql的slave数据库的备份请求，使得Redis的数据更新达到了相同的效果。

当然，这里的消息推送工具你也可以采用别的第三方：kafka、rabbitMQ等来实现推送更新Redis。

3. 扩展内容

- MySQL主从同步的方案，及优劣比较
- Redis哨兵、复制、集群的设计原理及区别
- 如何解决Redis缓存雪崩、缓存穿透、缓存并发等难题
- Redis的内存回收原理，及Redis内存过期淘汰策略

- Redis并发竞争key的解决方案