

×

 **阿里云APP**
查看产品优惠，实时监控云资源

打开 APP

探索Redis与MySQL的双写问题


2023-10-12 743 发布于浙江

版权 举报

本文涉及的产品

<div>RDS MySQL Serverless ...</div> <div>推荐场景：</div> <div>学生管理系统数据库设计</div>	<div>云数据库 Tair（兼容R...</div> <div>推荐场景：</div> <div>通过缓存加速数据库访问</div>	<div>Redis 开源版，标准版 2GB</div> <div>推荐场景：</div> <div>搭建游戏排行榜</div>
---	---	--

简介： 在实际应用Redis和MySQL过程中，如何保证Redis和MySQL双写时的数据一致性问题成为了开发者们面临的重要挑战

本文已收录至GitHub，推荐阅读  [Java随想录](#)

微信公众号：Java随想录

原创不易，注重版权。转载请注明原作者和原文链接

在日常的应用开发中，我们经常会遇到需要使用多种不同类型的数据库管理系统来满足各种业务需求。其中最典型的就Redis和MySQL的组合使用。

这两者拥有各自的优点，例如Redis为高性能的内存数据库提供了极快的读写速度，而MySQL则是非常强大的关系型数据库，支持事务处理，并且提供了很好的数据一致性。

然而，在实际应用过程中，如何保证Redis和MySQL双写时的数据一致性问题成为了开发者们面临的重要挑战。本文即将针对这个问题进行深入探讨，希望能为广大开发者们提供一些有价值的思路和解决方案。

双写一致问题

双写一致性问题主要是指当我们同时向Redis和MySQL写数据时，由于网络延迟、服务器故障等原因，可能导致数据在两个系统之间产生不一致。

例如，你可能已经更新了MySQL中的数据，但是Redis中的数据还未来得及更新，或者反过来。这样的结果就可能导致用户读到的是旧的、不正确的数据。

比如在现实生活中的购物网站场景：假设用户A在购买一件库存仅剩1件的商品，系统在接收到请求后，先将MySQL中的库存减少1，然后出现了网络延迟或系统故障，Redis中的库存没有减少。此时，用户B看到的是还有1件商品，也发起了购买请求，如果系统又首先更改了MySQL，那么就会出现超卖的情况，即实际库存已经没有，但因为缓存中的信息不准确，导致系统销售了更多的商品。

严格意义上任何非原子操作都不可能保证一致性，除非用阻塞读写实现强一致性，所以对于缓存架构我们追求的目标是最终一致性。

实际上，缓存就是通过牺牲强一致性来提高性能的。这是由CAP理论决定的。缓存系统适用的场景就是非强一致性的场景，它属于CAP中的AP。

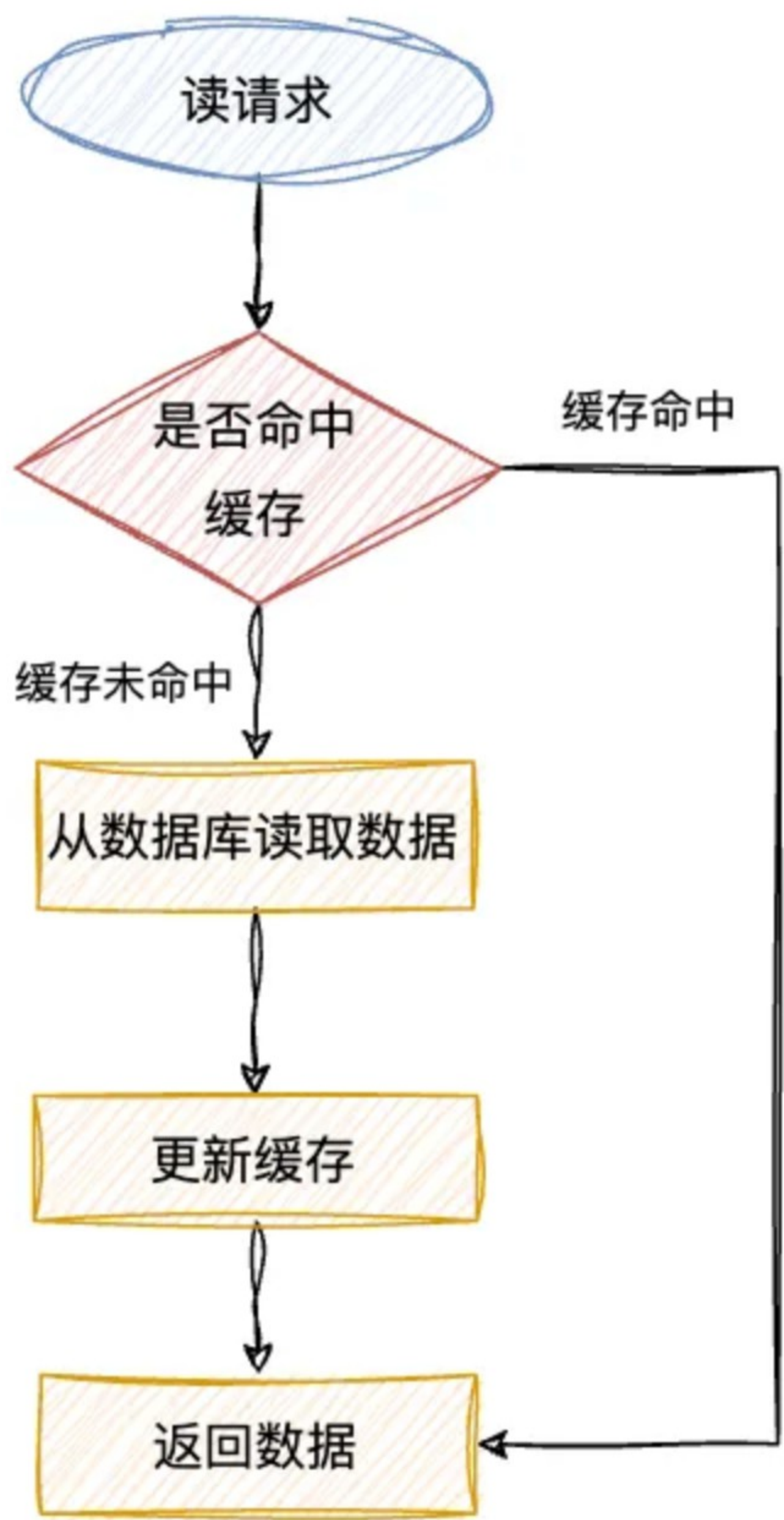
缓存读写策略

解决这种问题的常见策略就是“缓存读写策略”。这个策略用于处理先更新数据库还是先更新缓存等场景。

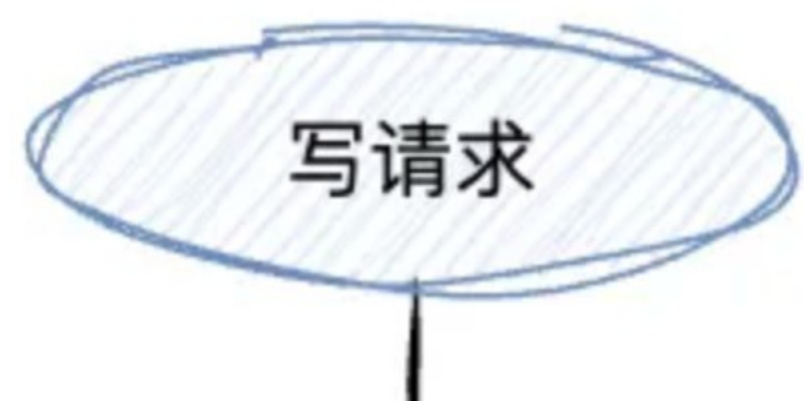
接下来，我们将探讨三种缓存读写策略。这些策略各有优劣，没有绝对的最佳选择。请根据具体的应用场景选择最合适的策略。

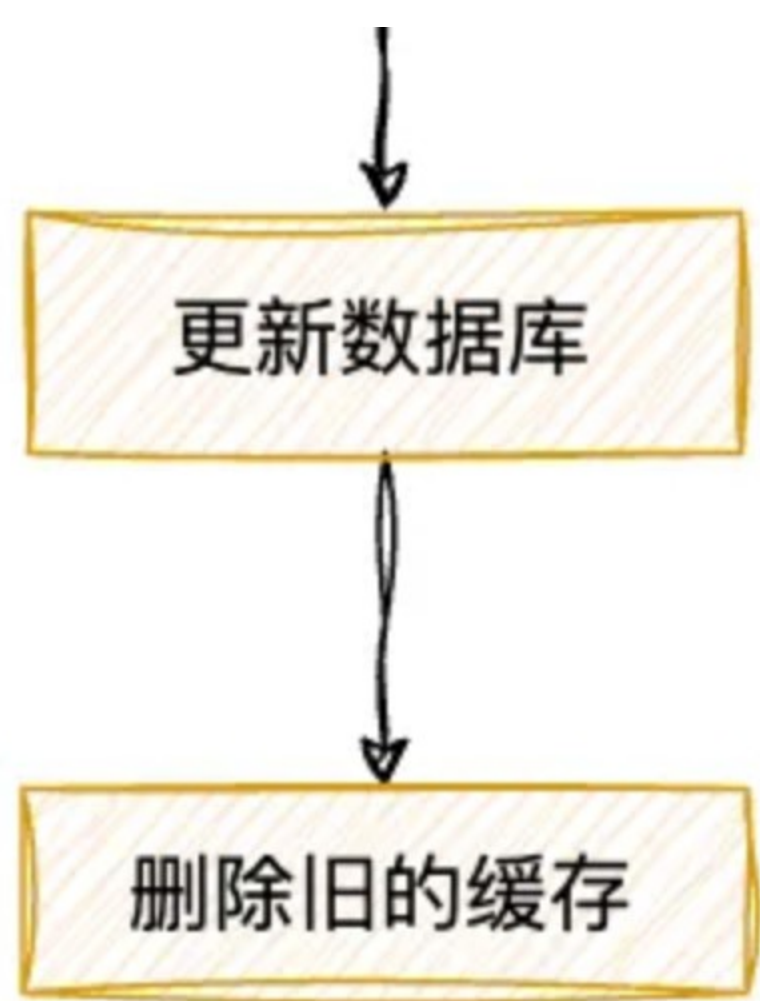
Cache-Aside Pattern（旁路缓存模式）

Cache-Aside Pattern，即旁路缓存模式，它的提出是为了尽可能地解决缓存与数据库的数据不一致问题。旁路缓存模式中服务端需要同时维护 DB 和 Cache，并且是以 DB 的结果为准。



读：从缓存读取数据，读到直接返回。如果读取不到的话，从数据库加载，写入缓存后，再返回响应。



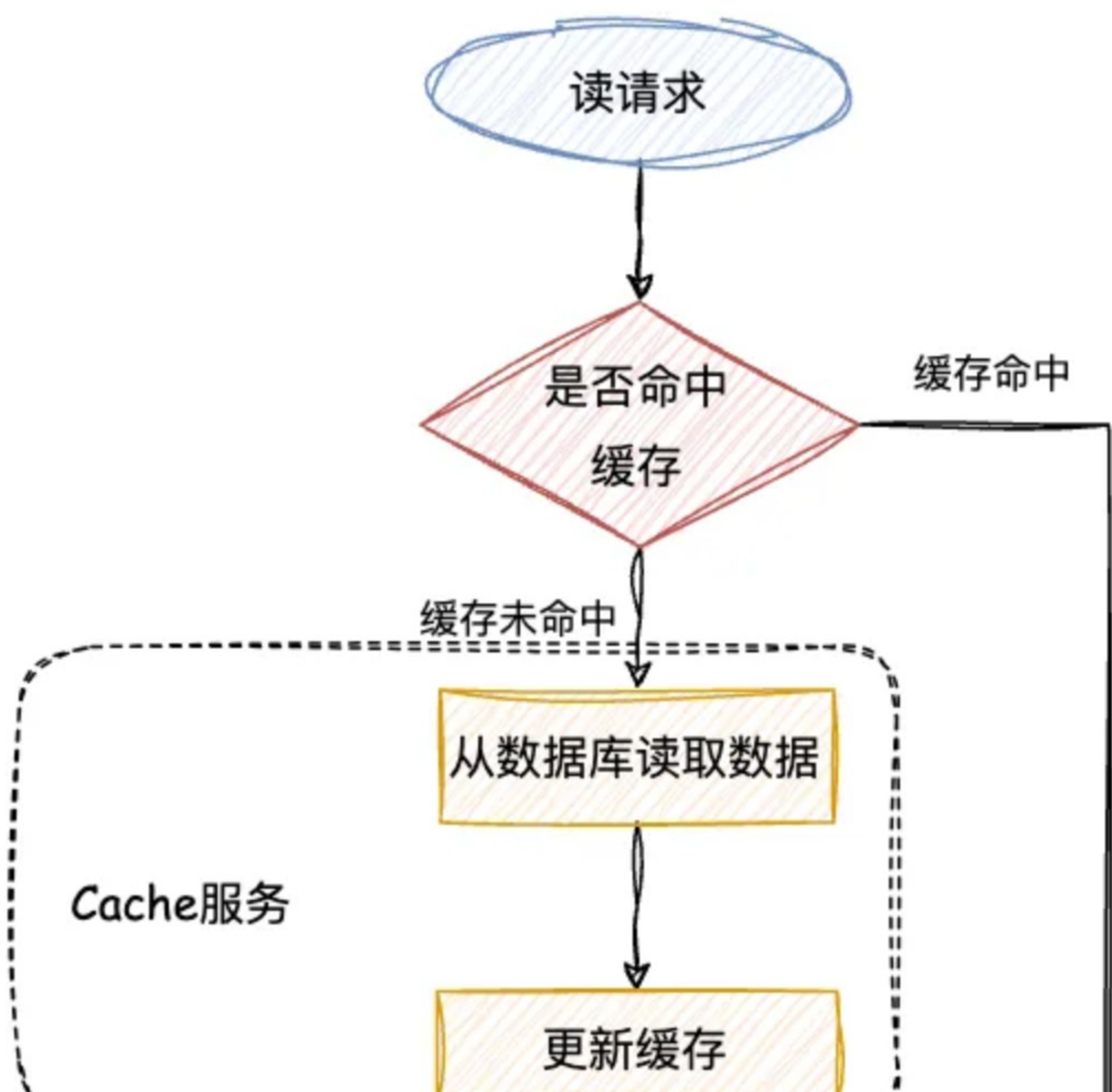


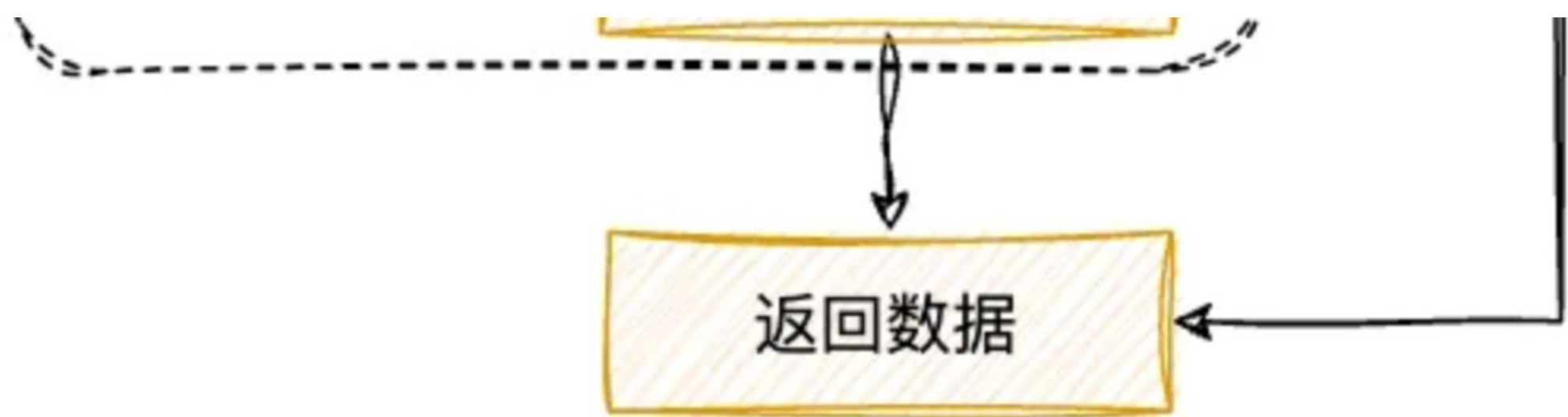
写：更新的时候，先「更新数据库，然后再删除缓存」。

Read/Write Through Pattern（读写穿透模式）

Read/Write Through Pattern 中服务端把 cache 视为主要数据存储，从中读取数据并将数据写入其中。cache 服务负责将此数据读取和写入 DB，从而减轻了应用程序的职责。

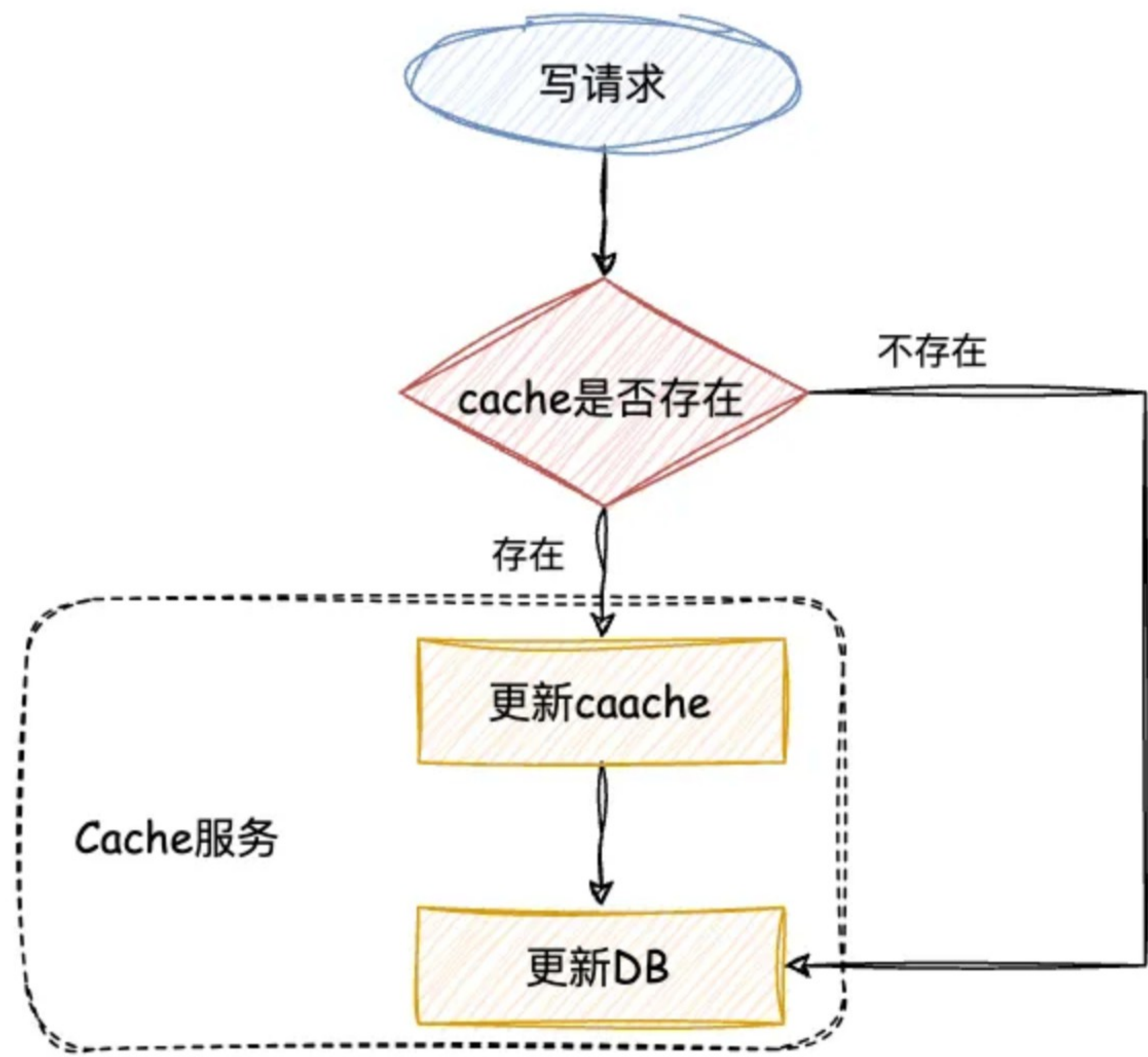
因为我们经常使用的分布式缓存 Redis 并没有提供 cache 将数据写入DB的功能，所以使用并不多。





读：从 cache 中读取数据，读取到就直接返回 。读取不到的话，先从 DB 加载，写入到 cache 后返回响应。

从流程图中可以看出，读写穿透模式和旁路缓存模式的读取流程几乎相同。不过，在旁路缓存模式中，客户端需要负责将数据写入 cache 。而在读写穿透模式中， cache 服务自行写入缓存，对客户端来说，这个过程是透明的。



写：先查 cache，cache 中不存在，直接更新 DB。cache 中存在，则先更新 cache，然后 cache 服务自己更新 DB（同步更新 cache和DB）。

Write Behind Pattern（异步缓存写入模式）

Write Behind Pattern 和 Read/Write Through Pattern 很相似，两者都是由 cache 服务来负责 cache 和 DB 的读写。

但是，两个又有很大的不同：**Read/Write Through 是同步更新 cache 和 DB，而 Write Behind Caching 则是只更新缓存，不直接更新 DB，而是改为异步批量的方式来更新 DB。**

很明显，这种方式对数据一致性带来了更大的挑战，比如cache数据可能还没异步更新DB的话，cache服务可能就挂掉了，反而会带来更大的灾难。

这种策略在我们平时开发过程中也非常非常少见，但是不代表它的应用场景少，比如消息队列中消息的异步写入磁盘、MySQL 的 InnoDB Buffer Pool 机制都用到了这种策略。

Write Behind Pattern 下 DB 的写性能非常高，非常适合一些数据经常变化又对数据一致性要求没那么高的场景，比如浏览量、点赞量等。

旁路缓存模式解析

Cache Aside Pattern 的一些疑问

旁路缓存模式是我们平时中使用最多的，根据该模式，我们可能会有以下几个疑问。

为什么写操作是删除缓存，而不是更新缓存

答：假设线程A先发起一个写操作，第一步先更新数据库。线程B再发起一个写操作，紧接着也更新了数据库。由于网络等原因，线程B比线程A先更新了缓存，然后线程A更新缓存。

这时候，缓存保存的是A的数据（老数据），而数据库保存的是B的数据（新数据），数据就不一致了，脏数据出现啦。如果是「**删除缓存取代更新缓存**」则不会出现这个脏数据问题。

实际上要写操作的时候更新缓存也是可以的，不过我们需要加一个锁/分布式锁来保证更新cache的时候不存在线程安全问题。

在写数据的过程中，为什么要先更新DB再删除缓存

答：假设请求1 是写操作，要是先删除缓存A，这时候来了请求2，请求2是读操作，先读缓存A，发现缓存被删除了(被请求1删除了)，然后去读数据库，但是此时请求1还没来得及把数据及时更新，那么请求2读的就是旧数据，并且请求2还会把读到的旧数据放到缓存中，造成了数据的不一致。

其实要先删缓存，再更新数据库也是可以，如采用「**延时双删策略**」。

休眠一段时间，再次淘汰缓存。这么做，可以将这段时间内所造成的缓存脏数据，再次删除。

注意sleep休眠的时间不能小于修改数据库数据的时间小，基本上1秒就够了。

在写数据的过程中，先更新DB，后删除cache就没有问题了么？

答：理论上来说还是可能会出现数据不一致性的问题，不过概率非常小。

假设这会有两个请求，一个请求A做查询操作，一个请求B做更新操作，那么会有如下情形产生：

1. 缓存刚好失效。
2. 请求A查询数据库，得一个旧值。
3. 请求B将新值写入数据库。
4. 请求B删除缓存。
5. 请求A将查到的旧值写入缓存 ok，如果发生上述情况，确实是会发生脏数据。

然而，发生这种情况的概率并不高

发生上述情况有一个先天性条件，就是步骤（3）的写数据库操作比步骤（2）的读数据库操作耗时更短，才有可能使得步骤（4）先于步骤（5）。

可是，仔细想想，数据库的读操作的速度远快于写操作的（不然做读写分离干嘛，做读写分离的意义就是因为读操作比较快，耗资源少），因此步骤（3）耗时比步骤（2）更短，这一情形很难出现。

还有其他造成不一致的原因么？

答：如果删除缓存过程中失败了就会造成不一致问题。可以使用Canal去订阅数据库的binlog，获得需要操作的数据。另起一个程序，获得这个订阅程序传来的信息，进行删除缓存操作。

Cache Aside Pattern 的缺陷

Cache Aside Pattern是一种常见的缓存更新策略，主要在读取数据时用于处理缓存的失效和更新。尽管它有很多优点，但也存在一些缺陷：

缺陷1：首次请求数据一定不在 cache 的问题

解决办法：可以将热点数据提前放入cache 中。

缺陷2：写操作比较频繁的话导致cache中的数据会被频繁被删除，这样会影响缓存命中率。

- 数据库和缓存数据强一致场景：更新DB的时候同样更新cache，不过我们需要加一个锁/分布式锁来保证更新cache的时候不存在线程安全问题。

- 可以短暂地允许数据库和缓存数据不一致的场景：更新DB的时候同样更新cache，但是给缓存加一个比较短的过期时间，这样的话就可以保证即使数据不一致的话影响也比较小。

延时双删

Redis的延时双删策略主要用于解决分布式系统当中的缓存与数据库数据一致性问题。以下是其基本步骤：

1. 先删除缓存。
2. 再更新数据库。
3. 最后延时再次删除缓存。

该策略的理念是：如果有其他线程在步骤1和步骤2之间查询到旧的数据并写入了缓存，那么步骤3可以保证这部分旧的数据被清除，从而尽可能维持数据库和缓存之间的数据一致性。

以下是使用Java实现的样例代码：



```
import redis.clients.jedis.Jedis;

public class RedisDoubleDelStrategy {

    private Jedis jedis;
    private static final long DELAY_MILLIS = 1000L; // 设置为你需要的延时时间

    public RedisDoubleDelStrategy(String host, int port) {

        this.jedis = new Jedis(host, port);
    }

    public void updateDBAndCache(String key, String value) {

        // Step 1: 删除缓存
        jedis.del(key);

        // Step 2: 更新数据库，此处以打印输出代替
        System.out.println("Update DB with: " + value);

        // 延迟任务来完成第二次删除
        new Thread(() -> {

            try {
```



```
        Thread.sleep(DELAY_MILLIS);
    } catch (InterruptedException e) {

        e.printStackTrace();
    }

    // Step 3: 延时后再次删除缓存
    jedis.del(key);
}).start();
}
}
```

这段代码实现了延时双删策略，但请注意它仍然不能完全保证数据库和缓存之间的一致性。

在某些情况下（比如大量并发情况下），可能仍然会出现不一致的问题。例如，在步骤3之后，如果还有其他线程查询到了旧数据并写入了缓存，那么数据库和缓存的数据就会不一致。因此，在使用该策略时，需要根据你的系统特性和一致性需求来进行权衡。

本篇文章到这就结束了，在探讨Redis与MySQL双写问题的过程中，我们分析了各种可能的场景和解决方案。双写系统不仅考验我们对数据库原理的理解，也展示了协同工作的复杂性。最终，解决这个问题的关键是理解你的用例并根据实际需求选择适当的策略和工具。

而在实际应用中，再完美的方案也可能会遇到挑战 and 困难。因此，持续监控，频繁测试和及时调整策略都至关重要。希望本文能为你在处理Redis与MySQL双写问题上提供一些思路和灵感，同时，我们也期待在未来看到更多精妙的解决方案诞生。

感谢阅读，如果本篇文章有任何错误和建议，欢迎给我留言指正。

老铁们，关注我的微信公众号「Java 随想录」，专注分享Java技术干货，文章持续更新，可以关注公众号第一时间阅读。

一起交流学习，期待与你共同进步！

文章标签：

云数据库 Tair（兼容 Redis）

云数据库 RDS MySQL 版

数据库

缓存

关系型数据库

NoSQL

MySQL

关键词：

云数据库 RDS MySQL 版redis

云数据库 Tair（兼容 Redis）MySQL

redis云数据库 RDS MySQL 版

云数据库 RDS MySQL 版双写

redis云数据库 RDS MySQL 版双写

相关实践学习

基于Redis实现在线游戏积分排行榜

本场景将介绍如何基于Redis数据库实现在线游戏中的游戏玩家积分排行榜功能。

→

云数据库 Redis 版使用教程

云数据库Redis版是兼容Redis协议标准的、提供持久化的内存数据库服务，基于高可靠双机热备架构及可无缝扩展的集群架构，满足高读写性能场景及容量需弹性变配的业务需求。 产品详情：https://www.aliyun.com/product/kvstore &nb...

→

评论

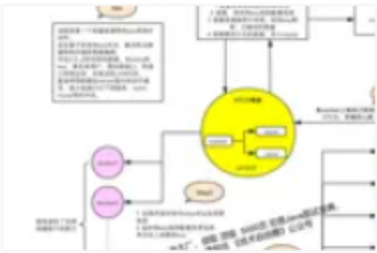
登录后可评论

相关文章

美团面试：MySQL有1000w数据，redis只存20w的数据，如何做 缓存 设计？

美团面试：MySQL有1000w数据，redis只存20w的数据，如何做 缓存 设计？

技术自由圈/原疯狂创客圈 552 阅读



Redis和Mysql如何保证数据一致？

1. 先更新Mysql，再更新Redis，如果更新Redis失败，可能仍然不一致 2. 先删除Redis缓存数据，再更新Mysql，再次...

游客bhmgxanbwhyfe 427 阅读

缓存与数据库的一致性方案，Redis与Mysql一致性方案，大厂P8的终极方案（图解+秒懂+史...

缓存与数据库的一致性方案，Redis与Mysql一致性方案，大厂P8的终极方案（图解+秒懂+史上最全）

技术自由圈/原疯狂创客圈 432 阅读

Redis与MySQL的数据一致性

在高并发环境下，保持 Redis 和 MySQL 的数据一致性是一个复杂但重要的问题。通过采用读写穿透、写穿透、分布...

蓝易云 322 阅读

《docker基础篇：8.Docker常规安装简介》包括：docker常规安装总体步骤、安装tomcat...

《docker基础篇：8.Docker常规安装简介》包括：docker常规安装总体步骤、安装tomcat、安装mysql、安装redis

刘大猫. 274 阅读

数据库运维：mysql 数据库迁移方法–mysqldump

本文介绍了MySQL数据库迁移的方法与技巧，重点探讨了数据量大小对迁移方式的影响。对于10GB以下的小型数据...

大数据文摘 412 阅读

大数据大厂之MySQL数据库课程设计：揭秘MySQL集群架构负载均衡核心...

本文聚焦 MySQL 集群架构中的负载均衡算法，阐述其重要性。详细介绍轮询、加权轮询、最少...

青云交（Java大数据AI云原生Python） 260 阅读



【YashanDB知识库】原生mysql驱动配置连接崖山数据库

【YashanDB知识库】原生mysql驱动配置连接崖山数据库

游客kufrkwrbkmpsa 171 阅读



【赵渝强老师】MySQL中的数据库对象

本教程详细介绍了MySQL数据库中的常见对象，包括表、索引、视图、事件、存储过程和存储函数的创建与管理。内...

赵渝强老师 24 阅读

比较MySQL和Oracle数据库系统，特别是在进行分页查询的方法上的不同

两者的性能差异将取决于数据量大小、索引优化、查询设计以及具体版本的数据库服务器。考虑硬件资源、数据库设...

蓝易云 58 阅读

热门文章

最新文章

- 1 MySQL 分库分表 + 平滑扩容方案 （秒懂+史上最全）23
- 2 字节面试：MySQL 百万级 导入发生的“死锁”难题如何解决？“2序4拆”，彻底攻克22
- 3 mysql事务隔离级别11
- 4 MySQL中实施排序(sorting)及分组(grouping)操作的技巧。17
- 5 如何实现MySQL百万级数据的查询？10

展开更多

相关课程

- 大数据实战项目：反爬虫系统（Lua+Spark+Redis+Hadoop框架搭建）第二阶段
- 大数据实战项目：反爬虫系统（Lua+Spark+Redis+Hadoop框架搭建）第五阶段
- 大数据实战项目 – 反爬虫系统（Lua+Spark+Redis+Hadoop框架搭建）第七阶段
- Redis入门实战演练
- 云数据库 Redis 版使用教程
- Redis数据库入门

更多

相关电子书

- Redis集群演化的心路历程——从2.x到3.0时代
- 微博的Redis定制之路
- 云数据库Redis版的开源之路

更多

推荐镜像

mysql

更多

关注我们：新浪微博

联系我们

文档 | 开发者社区 | 天池大赛 | 培训与认证



法律声明及隐私权政策 | Cookies政策

© 2009–2025 Aliyun.com 版权所有

增值电信业务经营许可证：浙B2–20080101

域名注册服务机构许可：浙D3–20210002

  浙公网安备 33010602009975号浙B2–20080101–4