

Redis和数据库 数据同步问题

Redis和数据库同步问题

缓存充当数据库

比如说Session这种访问非常频繁的数据，就适合采用这种方案；当然了，既然没有涉及到数据库，那么也就不会存在一致性问题；

缓存充当数据库热点缓存

读操作

目前的读操作有个固定的套路，如下：

1. 客户端请求服务器的时候，发现如果服务器的缓存中存在，则直接取服务器的；
2. 如果缓存中不存在，则去请求数据库，并且将数据库计算出来的数据回填给缓存；
3. 返回数据给客户端；

写操作

各种情况会导致数据库和缓存出现不一致的情况，这就是缓存和数据库的双写一致性问题；

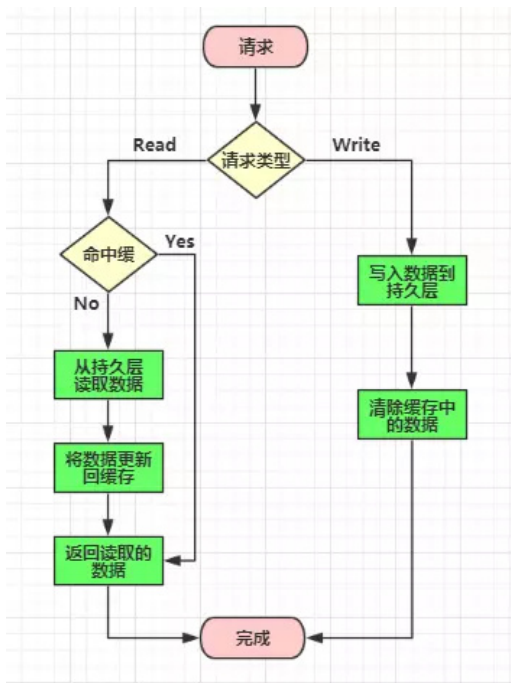
目前缓存存在三种策略，分别是

- Cache Aside 更新策略：同时更新缓存和数据库；
- Read/Write Through 更新策略：先更新缓存，缓存负责同步更新数据库；
- Write Behind Caching 更新策略：先更新缓存，缓存定时异步更新数据库；

三种策略各有优缺点，可以根据业务场景使用；

Cache Aside 更新策略

该策略大概的流程就是请求过来时先从缓存中取，如果命中缓存的话，则直接返回读取的数据；相反如果没有命中的话，接着会从数据库中成功获取到数据后，再去清除缓存中的数据；具体流程图如下：



公告



关注公众号：数据结构与算法那些事儿

昵称： banananana
园龄： 3年1个月
粉丝： 102
关注： 77
[+加关注](#)

2019年10月						
日	一	二	三	四	五	六
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索

<input type="text"/>	<input type="button" value="找查看"/>
<input type="text"/>	<input type="button" value="谷歌搜索"/>

积分与排名

积分 - 142144
排名 - 3436

随笔分类

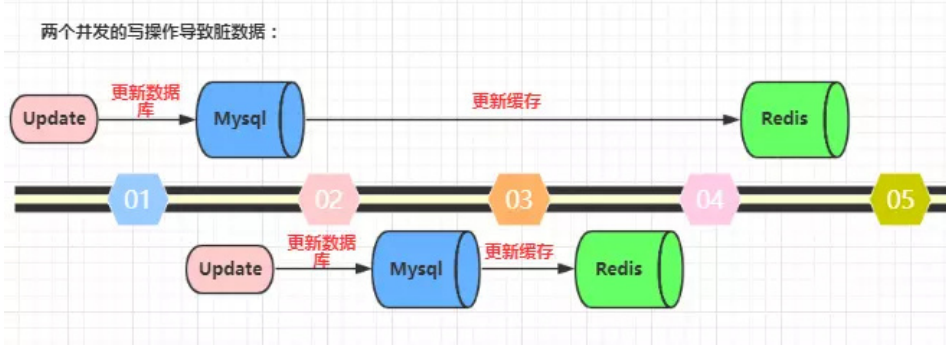
ACM Coding(2)
AL(35)
AL_ Memoization(5)
AL_Array(30)
AL_AVL
AL_Backtracking(18)
AL_BFS(5)
AL_Binary Indexed Tree(2)
AL_Binary Search(11)
AL_Bit Manipulation
AL_Combination(1)
AL_DFS(13)
AL_Divide and Conquer(1)
AL_DP(23)
AL_Graph(1)
AL_Greedy(3)
AL_Hash Table(2)
AL_Heap
AL_LCA(1)
AL_Linked List(19)
AL_Math(2)
AL_Matrix(3)
AL_Permutation(1)
AL_Queue
AL_Recursion
AL_Segment Tree
AL_Sliding Window(4)
AL_Sort(3)
AL_Stack(7)

但是以上在某些特殊的情况下是存在问题：

问题1：先更新数据库，后更新缓存

两个线程在高并发的情况下就会出现数据脏读的情况：

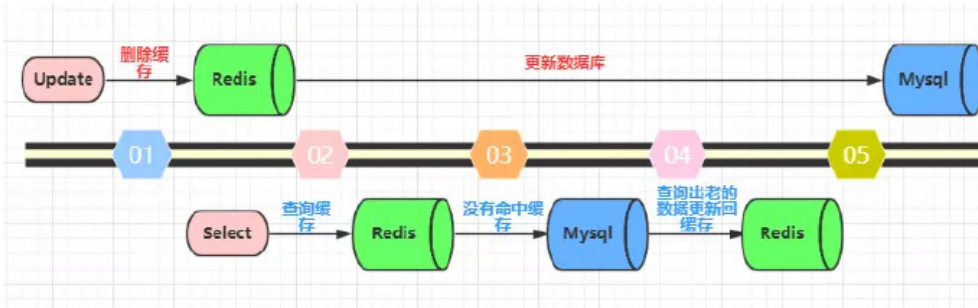
1. 线程A执行写操作，成功更新数据库；
2. 线程B同样执行和线程A一样的操作，但是在线程A执行更新缓存的过程中，线程B更新了新的数据库数据到缓存中；
3. 线程A在线程B全部操作完成以后才将相对老的数据又更新到了缓存中；



问题2：先删除缓存，后更新数据库

同样的，在高并发场景下同样会出现脏读的情况：

1. 线程A成功删除了缓存，等待更新数据库；
2. 线程B进行读操作，由于此时缓存已经被删除了，因此线程B重新从数据库中获取老的数据并且更新到了缓存中；
3. 线程A在线程B完成了整个的读操作以后，才更新数据库，此时缓存中的数据依旧是老的数据；

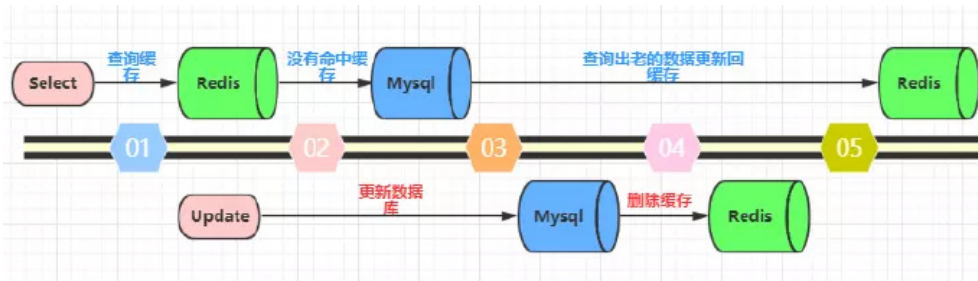


问题3：先更新数据库，后删除缓存

目前这是比较普遍的操作，即使它还是有可能出现脏读的情况：

1. 线程A进行读操作，此时正好没有命中缓存，接着请求数据库；
2. 线程B进行写操作，在线程A没有从数据库中获取到数据之前，把数据写入到数据库中，并且还成功删除了缓存；
3. 线程A在线程B完成了整个的写操作以后，才将相对老的数据更新到缓存中；

但是以上的情况比较不会出现，这是因为上述情况需要满足线程A的读操作要慢于线程B的写操作，但是在现实过程中，读操作通常都是要快于写操作得多的，但是为了避免发生以上的情况，通常都是要给缓存加上一个过期的时间；



但是设想一下，如果上面的删除缓存失败了怎么办呢，这样显然会导致数据脏读的情况，我觉得方案如下：

AL_String(9)
AL_Topological Sort
AL_Tree(16)
AL_Trie
AL_Two Pointers(5)
AL_Union Find
ALCompetition(2)
C++(7)
C++ Primer(9)
Celery(1)
Cocos2dx(6)
Design Pattern(21)
Django(6)
Experience(4)
Git
Golang
Hadoop
JavaScript(5)
LeetCode(139)
Linux(1)
Linux Shell(3)
Machine Learning(1)
MySQL(1)
Network(7)
Nginx(2)
Offers(5)
OpenGL(6)
OpenGL ES(4)
OpenSource Project(1)
Operating System(2)
Photography(1)
Process
Python(24)
RabbitMQ(2)
Redis(9)
Review of Year(1)
Shell(6)
WebGL(1)
WeiXin(1)

最新评论

1. Re:树链剖分原理和实现
orz
太棒了
--黄偶hj
2. Re:Redis和数据库 数据同步问题
我喜欢把客户当成小白鼠去更新数据
--张子浩
3. Re:从0到后端工程师
请问转行花了多长时间呢
--洋葱YY
4. Re:B树和B+树的总结
@ 方家小白关注前后两个图就好。...
--banananana
5. Re:阻塞和非阻塞，同步和异步 总结
谢谢指点
--cyhdmj

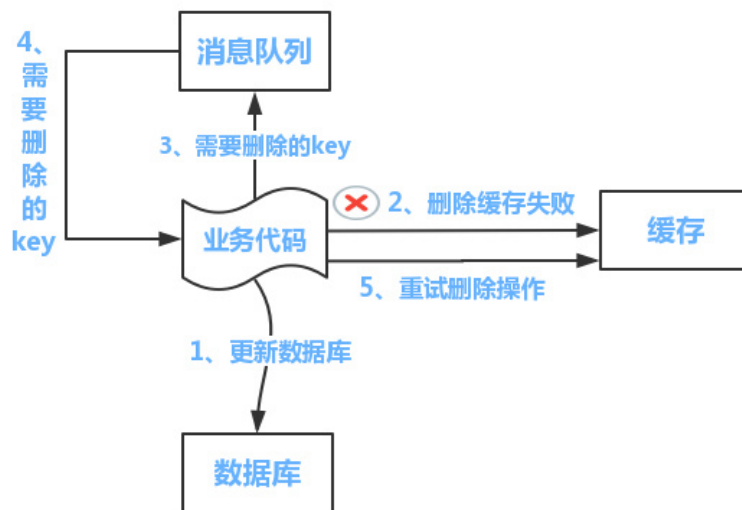
阅读排行榜

1. B树和B+树的总结(30866)
2. 阻塞和非阻塞，同步和异步 总结(27876)
3. 图的深度优先遍历和广度优先遍历理解(25841)
4. 树状数组的原理和实现(17799)
5. 关于access_token过期的解决办法(14362)

评论排行榜

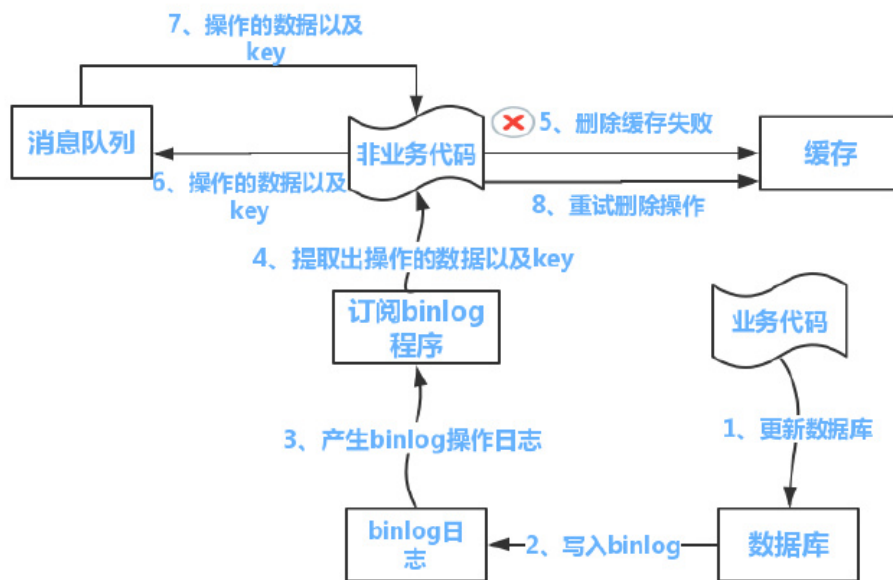
1. 对排名前3000位博主进行数据分析(14)
2. 阻塞和非阻塞，同步和异步 总结(10)
3. 树链剖分原理和实现(8)
4. How Django works? (8)

1. 设置缓存的过期时间（必须要做）；
2. 提供一个保障重试机制，将哪些删除失败的key提供给消息队列去消费；
3. 从消息队列取出这些key再次进行删除，失败再次加入到消息队列中，超过一定次数以上则人工介入；



但是以上情况需要在业务代码中进行操作，显然得需要进行解耦；

目前我们公司就是使用该方案，具体过程为在更新数据库数据的时候，数据库会以binlog日志的形式保存下来，通过 **canal** 开源软件将binlog解析成程序语言可以解析的地步，接着订阅程序获取到这些数据以后，尝试删除缓存操作，如果操作失败的话，则将其加入到消息队列中，重复消费，当删除操作的失败次数到达一定的次数以后，还是得人工介入。

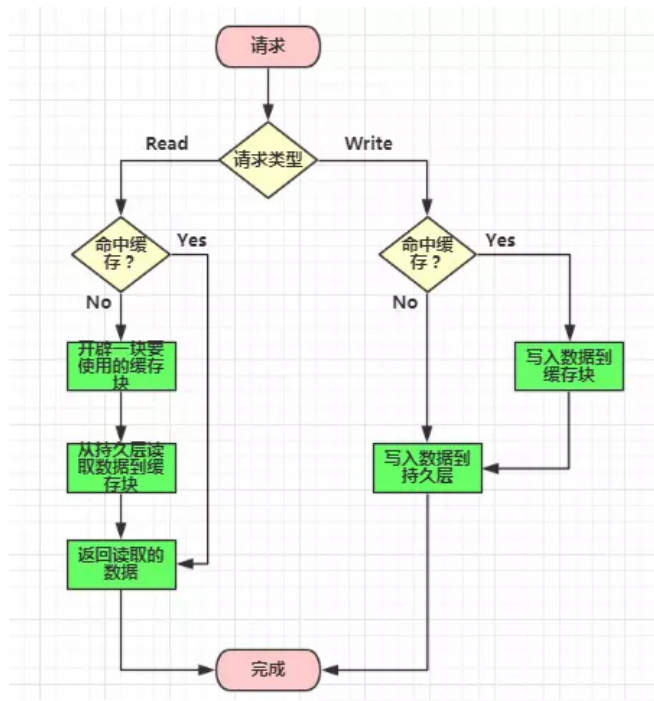


Read/Write Through 更新策略

该模式下，程序只需要维护缓存即可，数据库的同步工作交由缓存来同步更新；

该策略具体又分为两种：

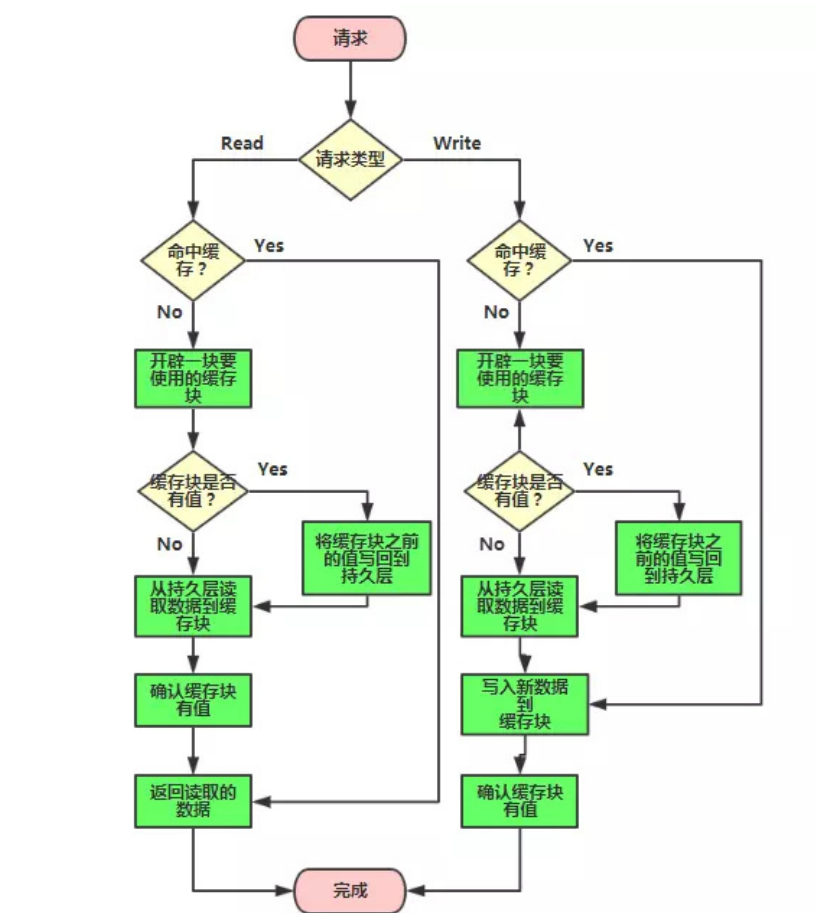
1. Read Through：在查询的过程中更新缓存；
2. Write Through：在写操作的过程中如果命中缓存，则直接更新缓存，数据库则由缓存自己同步去更新；



Write Behind Caching 更新策略

该策略只更新缓存，不会立马更新数据库，只会在一定的时间异步的批量去操作数据库；这样的好处在于直接操作缓存，效率极高，并且操作数据是异步的，还可以将多次的操作数据库语句合并到一个事务中一起提交，因此效率很客观；

但是，该策略没有办法做到数据强一致性，并且实现逻辑相对是比较复杂的，因为它需要确认哪些是需要更新到数据库的，哪些是仅仅想要存储在缓存中的；



比较

目前通常使用的是第一种策略中的先更新数据库，后更新缓存；其他的相比较起来实现都比较复杂；

最后想说的是，缓存本来就是为了牺牲强一致性来提高性能的，所以肯定会存在一定的延迟时间，我们只需要保证最终的数据一致性即可；

最后

以上是我在学习过程中的总结（其中很多内容都用了其他博客的内容），感恩～

【原创】分布式之数据库和缓存双写一致性方案解析

面试前必须要知道的Redis面试题

使用缓存的正确姿势



关注公众号：数据结构与算法那些事儿，每天一篇数
据结构与算法

分类: [Experience](#)

好文要顶

关注我

收藏该文

banananana

关注 - 77
粉丝 - 102

1

0

+加关注

« 上一篇: [203. 移除链表元素](#)
» 下一篇: [94. 二叉树的中序遍历](#)

posted @ 2019-03-26 16:50 [banananana](#) 阅读(1894) 评论(1) [编辑](#) [收藏](#)

评论列表

#1楼 2019-03-26 20:30 [ZaraNet](#)

我喜欢把客户当成小白鼠去更新数据

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#) 网站首页。

【推荐】超50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【活动】京东云服务器_云主机低于1折，低价高性能产品备战双11

【推荐】天翼云双十一提前开抢，1核1G云主机3个月仅需59元

【活动】魔程社区技术沙龙训练营—大数据主题专场等你来报名

【优惠】腾讯云 11.1 1智惠上云，爆款提前购与双11活动同价

【福利】个推四大热门移动开发SDK全部免费用一年，限时抢！

相关博文：

- [Redis和数据库数据同步问题](#)
 - [Redis和mysql数据怎么保持数据一致的？](#)
 - [redis缓存和mysql数据库同步](#)
 - [Redis如何保持和MySQL数据一致【二】](#)
 - [redis缓存和mysql数据库同步](#)
- » [更多推荐...](#)

最新 IT 新闻：

- [1024特别版：向“程序媛们”致敬！](#)
 - [发布一年后 Android Pie 普及率突破五分之一](#)
 - [WeWork新任董事长邮件曝光：必须裁员，现金流转正是当务之急](#)
 - [科大讯飞发布家电行业专用语音芯片CSK400X系列，算力128GOPS/s](#)
 - [谷歌宣称首次实现量子优越性，IBM“不服”，中国同行咋看？](#)
- » [更多新闻...](#)

历史上的今天：

2019-03-26 [203. 移除链表元素](#)

2019-03-26 [19. 删除链表的倒数第N个节点](#)