链滴

使用 redis 缓存的经验

作者：Changer0914

原文链接：https://ld246.com/article/1520822628400

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

问题：怎么保持缓存与数据库一致?

要解答这个问题，我们首先来看不一致的几种情况。我将不一致分为三种情况：

1. 数据库有数据，缓存没有数据；

2. 数据库有数据，缓存也有数据，数据不相等；

3. 数据库没有数据，缓存有数据。

在讨论这三种情况之前，先说明一下我使用缓存的策略，也是大多数人使用的策略，叫做 Cache Asid Pattern。酷壳里的 缓存更新的套路 一文，很值得一读，我的策略也是从他那学来的。

简而言之，就是

1. 首先尝试从缓存读取，读到数据则直接返回；如果读不到，就读数据库，并将数据会写到缓存，返回。

2. 需要更新数据时，先更新数据库，然后把缓存里对应的数据失效掉（删掉）。

读的逻辑大家都很容易理解，谈谈更新。如果不采取我提到的这种更新方法，你还能想到什么更新方呢？大概会是：先删除缓存，然后再更新数据库。这么做引发的问题是，如果A,B两个线程同时要更数据，并且A,B已经都做完了删除缓存这一步，接下来，A先更新了数据库，C线程读取数据，由于缓没有，则查数据库，并把A更新的数据，写入了缓存，最后B更新数据库。那么缓存和数据库的值就不致了。

另外有人会问，如果采用你提到的方法，为什么最后是把缓存的数据删掉，而不是把更新的数据写到存里。这么做引发的问题是，如果A,B两个线程同时做数据更新，A先更新了数据库，B后更新数据库则此时数据库里存的是B的数据。而更新缓存的时候，是B先更新了缓存，而A后更新了缓存，则缓存是A的数据。这样缓存和数据库的数据也不一致。

按照我提到的这种更新缓存的策略，理论上也是有不一致的风险的，酷壳的文章有提到，只不过概率小，我们暂时可以不考虑，后面我们有其他手段来补救。

讨论完使用缓存的策略，我们再来看这三种不一致的情况。

1. 对于第一种，在读数据的时候，会自动把数据库的数据写到缓存，因此不一致自动消除

2. 对于第二种，数据最终变成了不相等，但他们之前在某一个时间点一定是相等的（不管你使用懒载还是预加载的方式，在缓存加载的那一刻，它一定和数据库一致）。这种不一致，一定是由于你更数据所引发的。前面我们讲了更新数据的策略，先更新数据库，然后删除缓存。因此，不一致的原因一定是数据库更新了，但是删除缓存失败了。

3. 对于第三种，情况和第二种类似，你把数据库的数据删了，但是删除缓存的时候失败了。

因此，最终的结论是，需要解决的不一致，产生的原因是更新数据库成功，但是删除缓存失败。

我想出的解决方案大概有以下几种：

1. 对删除缓存进行重试，数据的一致性要求越高，我越是重试得快。

2. 定期全量更新，简单地说，就是我定期把缓存全部清掉，然后再全量加载。

3. 给所有的缓存一个失效期。

第三种方案可以说是一个大杀器，任何不一致，都可以靠失效期解决，失效期越短，数据一致性越高但是失效期越短，查数据库就会越频繁。因此失效期应该根据业务来定。

哪些数据需要放在缓存中?

首先，缓存的对象有三种：

1. 数据库中单条的的数据（以表名跟id作为key永久保存到redis），在有更新的地方都要更新缓存不适用于需要经常更新的数据）；

2. 对于一些不分页，不需要实时（需要多表查询）的列表，我们可以将列表结果缓存到redis中，设一定缓存时间作为该数据的存活时间。用获取该列表的方法名作为key，列表结果为value；这种情况试用于不经常更新且不需要实时的情况下。

3. 不需要实时的，需要分页的列表：可以把分页的结果列表放到一个map（key为分页标识，value分页结果）中，然后将该map存到redis的list中（用该方法名为key）。然后给该list设置一个缓存存时间（用expire）。这样通过方法名lrange出来就能获取存有分页列表的数据，遍历该list，通过遍历list中map的key判断该分页数据是否在缓存内，是则返回，不存在则rpush进去。这种做法能解决比如15页的数据已经重新加载，而6-10页的数据依然是缓存的数据而导致脏数据的情况。

本人走过的一些弯路：

1. 对于数据缓存不是所有东西都缓存到redis就是好的，而是要针对一些改动不大或者访问率大的数进行缓存来减少关系型数据库的压力。

2. 不要试图在拦截器或者过滤器中判断是否有缓存的存在，因为每个请求（不管该请求对应的方法否做了缓存）它都会去redis中请求数据并判断，这样会浪费一定的内存资源跟响应时间。所以应该针需要缓存的方法进行判断。

3. 一个方法中使用多个get或者set的方法，我们需要尽可能的减少去jedispool中获取jedis对象，所在一个方法中应该只获取一次jedis对象，在方法结束的时候把该对象return还给连接池，这样才能做尽可能的高效。

4. 在设置连接池中参数的时候要考虑到自身系统需求，不然会经常出现连接池中无可用对象获取，sping时不时发起连接请求到redis等不必要的错误和资源浪费。