25-缓存异常（上）：如何解决缓存和数据库的数据不一致问题？

你好，我是蒋德钧。

在实际应用Redis缓存时，我们经常会遇到一些异常问题，概括来说有4个方面：缓存中的数据和数据库中的不一致；缓存雪崩；缓存击穿和缓存穿透。

只要我们使用Redis缓存，就必然会面对缓存和数据库间的一致性保证问题，这也算是Redis缓存应用中的“必答题”了。最重要的是，如果数据不一致，那么业务应用从缓存中读取的数据就不是最新数据，这会导致严重的错误。比如说，我们把电商商品的库存信息缓存在Redis中，如果库存信息不对，那么业务层下单操作就可能出错，这当然是不能接受的。所以，这节课我就重点和你聊聊这个问题。关于缓存雪崩、穿透和击穿等问题，我会在下一节课向你介绍。

接下来，我们就来看看，缓存和数据库之间的数据不一致是怎么引起的。

缓存和数据库的数据不一致是如何发生的？

首先，我们得清楚“数据的一致性”具体是啥意思。其实，这里的“一致性”包含了两种情况：

- 缓存中有数据，那么，缓存的数据值需要和数据库中的值相同；
- 缓存中本身没有数据，那么，数据库中的值必须是最新值。

不符合这两种情况的，就属于缓存和数据库的数据不一致问题了。不过，当缓存的读写模式不同时，缓存数据不一致的发生情况不一样，我们的应对方法也会有所不同，所以，我们先按照缓存读写模式，来分别了解下不同模式下的缓存不一致情况。我在第23讲中讲过，根据是否接收写请求，我们可以把缓存分成读写缓存和只读缓存。

对于读写缓存来说，如果要对数据进行增删改，就需要在缓存中进行，同时还要根据采取的写回策略，决定是否同步写回到数据库中。

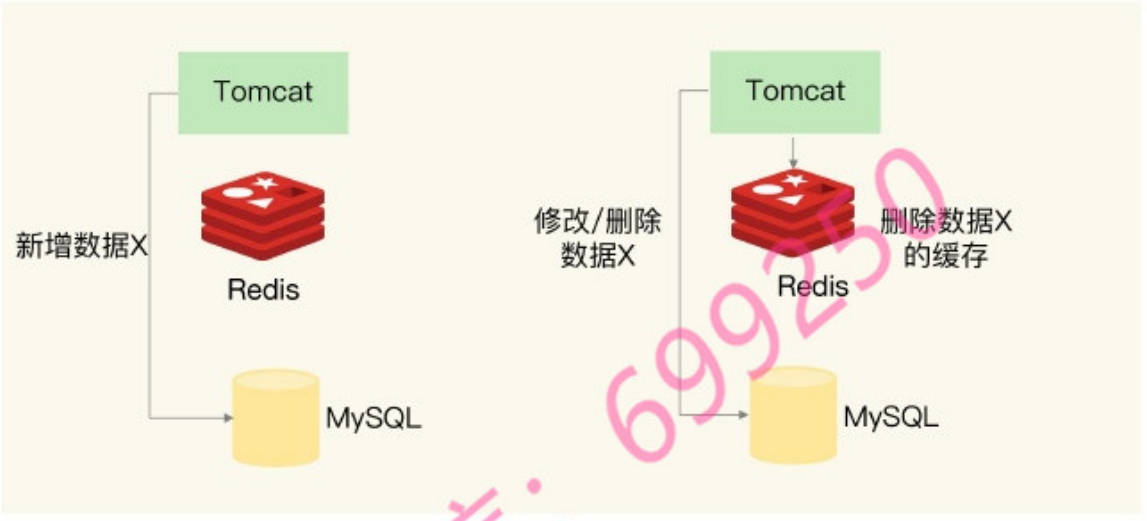
- 同步直写策略：写缓存时，也同步写数据库，缓存和数据库中的数据一致；
- 异步写回策略：写缓存时不同步写数据库，等到数据从缓存中淘汰时，再写回数据库。使用这种策略时，如果数据还没有写回数据库，缓存就发生了故障，那么，此时，数据库就没有最新的数据了。

所以，对于读写缓存来说，要想保证缓存和数据库中的数据一致，就要采用同步直写策略。不过，需要注意的是，如果采用这种策略，就需要同时更新缓存和数据库。所以，我们要在业务应用中使用事务机制，来保证缓存和数据库的更新具有原子性，也就是说，两者要不一起更新，要不都不更新，返回错误信息，进行重试。否则，我们就无法实现同步直写。

当然，在有些场景下，我们对数据一致性的要求可能不是那么高，比如说缓存的是电商商品的非关键属性或者短视频的创建或修改时间等，那么，我们可以使用异步写回策略。

下面我们再来说说只读缓存。对于只读缓存来说，如果有数据新增，会直接写入数据库；而有数据删改时，就需要把只读缓存中的数据标记为无效。这样一来，应用后续再访问这些增删改的数据时，因为缓存中没有相应的数据，就会发生缓存缺失。此时，应用再从数据库中把数据读入缓存，这样后续再访问数据时，就能够直接从缓存中读取了。

接下来，我以Tomcat向MySQL中写入和删改数据为例，来给你解释一下，数据的增删改操作具体是如何进行的，如下图所示：



从图中可以看到，Tomcat上运行的应用，无论是新增（Insert操作）、修改（Update操作）、还是删除（Delete操作）数据X，都会直接在数据库中增改删。当然，如果应用执行的是修改或删除操作，还会删除缓存的数据X。

那么，这个过程中会不会出现数据不一致的情况呢？考虑到新增数据和删改数据的情况不一样，所以我们分开来看。

1.新增数据

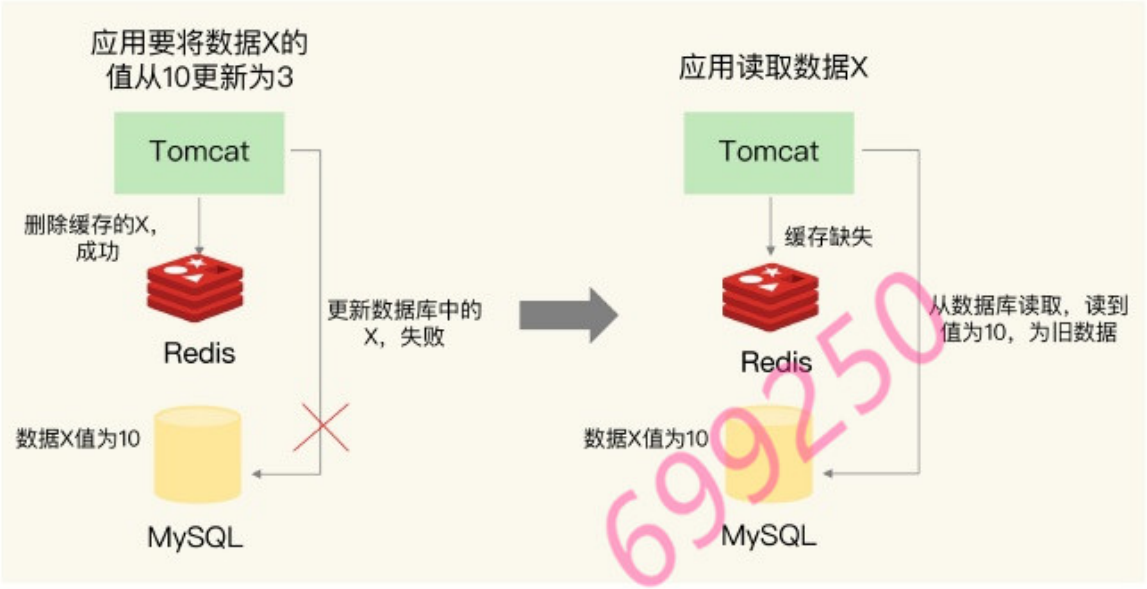
如果是新增数据，数据会直接写到数据库中，不用对缓存做任何操作，此时，缓存中本身就没有新增数据，而数据库中是最新值，这种情况符合我们刚刚所说的一致性的第2种情况，所以，此时，缓存和数据库的数据是一致的。

2.删改数据

如果发生删改操作，应用既要更新数据库，也要在缓存中删除数据。这两个操作如果无法保证原子性，也就是说，要不都完成，要不都没完成，此时，就会出现数据不一致问题了。这个问题比较复杂，我们来分析一下。

我们假设应用先删除缓存，再更新数据库，如果缓存删除成功，但是数据库更新失败，那么，应用再访问数据时，缓存中没有数据，就会发生缓存缺失。然后，应用再访问数据库，但是数据库中的值为旧值，应用就访问到旧值了。

我来举个例子说明一下，可以先看看下面的图片。

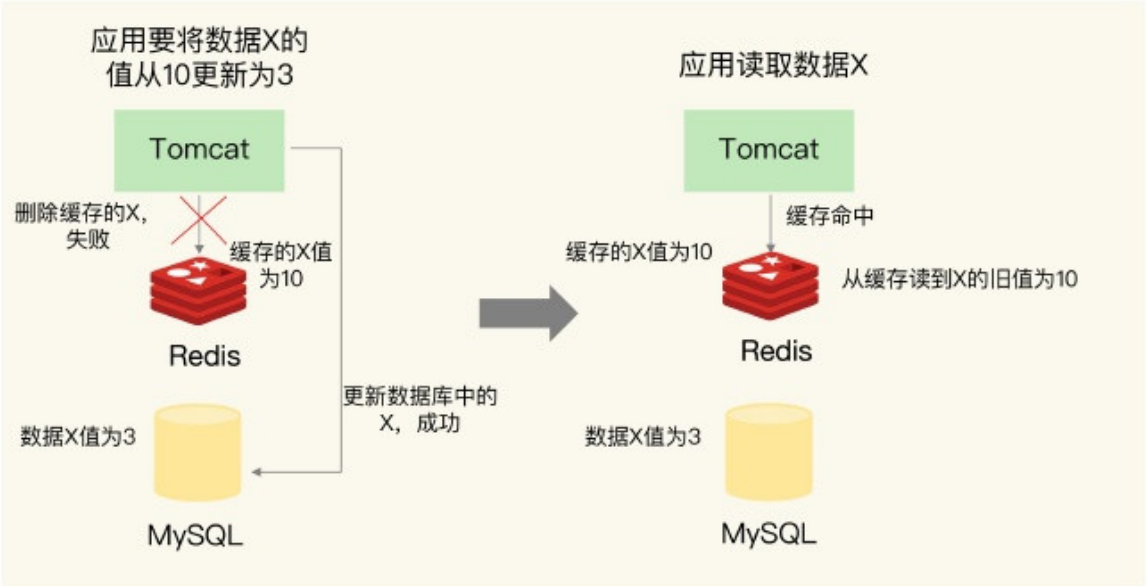


应用要把数据X的值从10更新为3，先在Redis缓存中删除了X的缓存值，但是更新数据库却失败了。如果此时有其他并发的请求访问X，会发现Redis中缓存缺失，紧接着，请求就会访问数据库，读到的却是旧值10。

你可能会问，如果我们先更新数据库，再删除缓存中的值，是不是就可以解决这个问题呢？我们再来分析下。

如果应用先完成了数据库的更新，但是，在删除缓存时失败了，那么，数据库中的值是新值，而缓存中的是旧值，这肯定是不一致的。这个时候，如果有其他的并发请求来访问数据，按照正常的缓存访问流程，就会先在缓存中查询，但此时，就会读到旧值了。

我还是借助一个例子来说明一下。



应用要把数据X的值从10更新为3，先成功更新了数据库，然后在Redis缓存中删除X的缓存，但是这个操作却失败了，这个时候，数据库中X的新值为3，Redis中的X的缓存值为10，这肯定是不一致的。如果刚好此时有其他客户端也发送请求访问X，会先在Redis中查询，该客户端会发现缓存命中，但是读到的却是旧值

10。

好了，到这里，我们可以看到，在更新数据库和删除缓存值的过程中，无论这两个操作的执行顺序谁先谁后，只要有一个操作失败了，就会导致客户端读取到旧值。我画了下面这张表，总结了刚刚所说的这两种情况。

不同情况	潜在问题
先删除缓存值，后更新数据库值	数据库更新失败，导致请求再次访问缓存时，发现缓存缺失，再读数据库时，从数据库中读到旧值
先更新数据库值，后删除缓存值	缓存删除失败，导致请求再次访问缓存时，发现缓存命中，并从缓存中读取到旧值

问题发生的原因我们知道了，那该怎么解决呢？

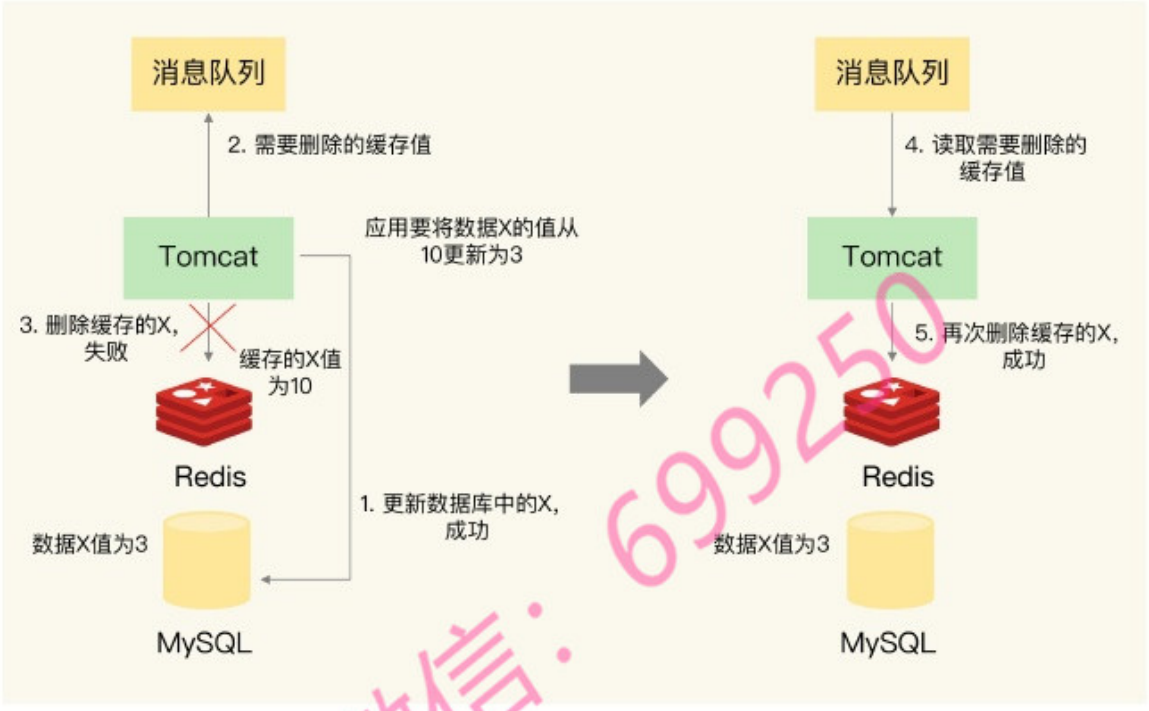
如何解决数据不一致问题？

首先，我给你介绍一种方法：重试机制。

具体来说，可以把要删除的缓存值或者是要更新的数据库值暂存到消息队列中（例如使用Kafka消息队列）。当应用没有能够成功地删除缓存值或者是更新数据库值时，可以从消息队列中重新读取这些值，然后再次进行删除或更新。

如果能够成功地删除或更新，我们就要把这些值从消息队列中去除，以免重复操作，此时，我们也可以保证数据库和缓存的数据一致了。否则的话，我们还需要再次进行重试。如果重试超过的一定次数，还是没有成功，我们就需要向业务层发送报错信息了。

下图显示了先更新数据库，再删除缓存值时，如果缓存删除失败，再次重试后删除成功的情况，你可以看下。



刚刚说的是在更新数据库和删除缓存值的过程中，其中一个操作失败的情况，实际上，即使这两个操作第一次执行时都没有失败，当有大量并发请求时，应用还是有可能读到不一致的数据。

同样，我们按照不同的删除和更新顺序，分成两种情况来看。在这两种情况下，我们的解决方法也有所不同。

情况一：先删除缓存，再更新数据库。

假设线程A删除缓存值后，还没有来得及更新数据库（比如说有网络延迟），线程B就开始读取数据了，那么这个时候，线程B会发现缓存缺失，就只能去数据库读取。这会带来两个问题：

1. 线程B读取到了旧值；
2. 线程B是在缓存缺失的情况下读取的数据库，所以，它还会把旧值写入缓存，这可能会导致其他线程从缓存中读到旧值。

等到线程B从数据库读取完数据、更新了缓存后，线程A才开始更新数据库，此时，缓存中的数据是旧值，而数据库中的是最新值，两者就不一致了。

我用一张表来汇总下这种情况。

时间	线程A	线程B	问题
t1	删除数据X的缓存值		
t2		1. 读取数据X，缓存缺失，从数据库读取X，读到旧值 2. 把数据X写入缓存	1. 线程A尚未更新数据库的值，导致线程B读到旧值 2. 线程B把旧值写入缓存，导致其他线程可能读到旧值
t3	更新数据库中的X		缓存中是旧值，数据库中是新值，两者不一致

这该怎么办呢？我来给你提供一种解决方案。

在线程A更新完数据库值以后，我们可以让它先sleep一小段时间，再进行一次缓存删除操作。

之所以要加上sleep的这段时间，就是为了让线程B能够先从数据库读取数据，再把缺失的数据写入缓存，然后，线程A再进行删除。所以，线程A sleep的时间，就需要大于线程B读取数据再写入缓存的时间。这个时间怎么确定呢？建议你在业务程序运行的时候，统计下线程读数据和写缓存的操作时间，以此为基础来进行估算。

这样一来，其它线程读取数据时，会发现缓存缺失，所以会从数据库中读取最新值。因为这个方案会在第一次删除缓存值后，延迟一段时间再次进行删除，所以我们也把它叫做“延迟双删”。

下面的这段伪代码就是“延迟双删”方案的示例，你可以看下。

```
redis.delKey(X)
db.update(X)
Thread.sleep(N)
redis.delKey(X)
```

情况二：先更新数据库值，再删除缓存值。

如果线程A删除了数据库中的值，但还没来得及删除缓存值，线程B就开始读取数据了，那么此时，线程B查询缓存时，发现缓存命中，就会直接从缓存中读取旧值。不过，在这种情况下，如果其他线程并发读缓存的请求不多，那么，就不会有很多请求读取到旧值。而且，线程A一般也会很快删除缓存值，这样一来，其他线程再次读取时，就会发生缓存缺失，进而从数据库中读取最新值。所以，这种情况对业务的影响较小。

我再画一张表，带你总结下先更新数据库、再删除缓存值的情况。

时间	线程A	线程B	问题
t1	删除数据库中的数据X		
t2		读取数据X，缓存命中，从缓存读取X，读到旧值	线程A尚未删除缓存值，导致线程B读到缓存的旧值
t3	删除缓存的数据X		

好了，到这里，我们了解到了，缓存和数据库的数据不一致一般是由两个原因导致的，我给你提供了相应的解决方案。

- 删除缓存值或更新数据库失败而导致数据不一致，你可以使用重试机制确保删除或更新操作成功。
- 在删除缓存值、更新数据库的这两步操作中，有其他线程的并发读操作，导致其他线程读取到旧值，应对方案是延迟双删。

小结

在这节课，我们学习了在使用Redis缓存时，最常遇见的一个问题，也就是缓存和数据库不一致的问题。针对这个问题，我们可以分成读写缓存和只读缓存两种情况进行分析。

对于读写缓存来说，如果我们采用同步写回策略，那么可以保证缓存和数据库中的数据一致。只读缓存的情况比较复杂，我总结了一张表，以便于你更加清晰地了解数据不一致的问题原因、现象和应对方案。

操作顺序	是否有并发请求	潜在问题	现象	应对方案
先删除缓存值，再更新数据库	无	缓存删除成功，但数据库更新失败	应用从数据库读到旧数据	重试数据库更新
	有	缓存删除后，尚未更新数据库，有并发读请求	并发请求从数据库读到旧值，并且更新到缓存，导致后续请求都读取旧值	延迟双删
先更新数据库，再删除缓存	无	数据库更新成功，但缓存删除失败	应用从缓存读到旧数据	重试缓存删除
	有	数据库更新成功后，尚未删除缓存，有并发读请求	并发请求从缓存中读到旧值	等待缓存删除完成，期间会有不一致数据短暂存在

希望你能把我总结的这张表格放入到你的学习笔记中，时不时复习一下。

最后，我还想再多说几句。在大多数业务场景下，我们会把Redis作为只读缓存使用。针对只读缓存来说，我们既可以先删除缓存值再更新数据库，也可以先更新数据库再删除缓存。我的建议是，优先使用先更新数据库再删除缓存的方法，原因主要有两个：

1. 先删除缓存值再更新数据库，有可能导致请求因缓存缺失而访问数据库，给数据库带来压力；

2. 如果业务应用中读取数据库和写缓存的时间不好估算，那么，延迟双删中的等待时间就不好设置。

不过，当使用先更新数据库再删除缓存时，也有个地方需要注意，如果业务层要求必须读取一致的数据，那么，我们就需要在更新数据库时，先在Redis缓存客户端暂存并发读请求，等数据库更新完、缓存值删除后，再读取数据，从而保证数据一致性。

每课一问

按照惯例，我给你提个小问题。这节课，我提到，在只读缓存中进行数据的删改操作时，需要在缓存中删除相应的缓存值。我想请你思考一下，如果在这个过程中，我们不是删除缓存值，而是直接更新缓存的值，你觉得和删除缓存值相比，有什么好处和不足吗？

欢迎在留言区写下你的思考和答案，我们一起交流讨论。如果你觉得今天的内容对你有所帮助，也欢迎你分享给你的朋友或同事。我们下节课见。

精选留言：

- Kaito 2020-10-14 00:07:54

  数据在删改操作时，如果不是删除缓存值，而是直接更新缓存的值，你觉得和删除缓存值相比，有什么好处和不足？

  这种情况相当于把Redis当做读写缓存使用，删改操作同时操作数据库和缓存。

  1、先更新数据库，再更新缓存：如果更新数据库成功，但缓存更新失败，此时数据库中是最新值，但缓存中是旧值，后续的读请求会直接命中缓存，得到的是旧值。

  2、先更新缓存，再更新数据库：如果更新缓存成功，但数据库更新失败，此时缓存中是最新值，数据库中是旧值，后续读请求会直接命中缓存，但得到的是最新值，短期对业务影响不大。但是，一旦缓存过期或者满容后被淘汰，读请求就会从数据库中重新加载旧值到缓存中，之后的读请求会从缓存中得到旧值，对业务产生影响。

  同样地，针对这种其中一个操作可能失败的情况，也可以使用重试机制解决，把第二步操作放入到消息队列中，消费者从消息队列取出消息，再更新缓存或数据库，成功后把消息从消息队列删除，否则进行重试，以此达到数据库和缓存的最终一致。

  以上是没有并发请求的情况。如果存在并发读写，也会产生不一致，分为以下4种场景。

  1、先更新数据库，再更新缓存，写+读并发：线程A先更新数据库，之后线程B读取数据，此时线程B会命中缓存，读取到旧值，之后线程A更新缓存成功，后续的读请求会命中缓存得到最新值。这种场景下，线程A未更新完缓存之前，在这期间的读请求会短暂读到旧值，对业务短暂影响。

  2、先更新缓存，再更新数据库，写+读并发：线程A先更新缓存成功，之后线程B读取数据，此时线程B命中缓存，读取到最新值后返回，之后线程A更新数据库成功。这种场景下，虽然线程A还未更新完数据库，数据库会与缓存存在短暂不一致，但在这之前进来的读请求都能直接命中缓存，获取到最新值，所以对业务没影响。

  3、先更新数据库，再更新缓存，写+写并发：线程A和线程B同时更新同一条数据，更新数据库的顺序是先A后B，但更新缓存时顺序是先B后A，这会导致数据库和缓存的不一致。

  4、先更新缓存，再更新数据库，写+写并发：与场景3类似，线程A和线程B同时更新同一条数据，更新缓

存的顺序是先A后B，但是更新数据库的顺序是先B后A，这也会导致数据库和缓存的不一致。

场景1和2对业务影响较小，场景3和4会造成数据库和缓存不一致，影响较大。也就是说，在读写缓存模式下，写+读并发对业务的影响较小，而写+写并发时，会造成数据库和缓存的不一致。

针对场景3和4的解决方案是，对于写请求，需要配合分布式锁使用。写请求进来时，针对同一个资源的修改操作，先加分布式锁，这样同一时间只允许一个线程去更新数据库和缓存，没有拿到锁的线程把操作放入到队列中，延时处理。用这种方式保证多个线程操作同一资源的顺序性，以此保证一致性。

综上，使用读写缓存同时操作数据库和缓存时，因为其中一个操作失败导致不一致的问题，同样可以通过消息队列重试来解决。而在并发的场景下，读+写并发对业务没有影响或者影响较小，而写+写并发时需要配合分布式锁的使用，才能保证缓存和数据库的一致性。

另外，读写缓存模式由于会同时更新数据库和缓存，优点是，缓存中一直会有数据，如果更新操作后会立即再次访问，可以直接命中缓存，能够降低读请求对于数据库的压力（没有了只读缓存的删除缓存导致缓存缺失和再加载的过程）。缺点是，如果更新后的数据，之后很少再被访问到，会导致缓存中保留的不是最热的数据，缓存利用率不高（只读缓存中保留的都是热数据），所以读写缓存比较适合用于读写相当的业务场景。 [29赞]

- 与路同飞 2020-10-14 06:54:47
  如果业务层要求必须读取一致的数据，那么，我们就需要在更新数据库时，先在 Redis 缓存客户端暂存并发读请求，等数据库更新完，再读取数据，从而保证数据一致性。这个redis客户端暂存并发读请求咋弄 [1赞]

- williamcai 2020-10-16 08:15:41
  老师你好，用事务保证数据库和redis一致，不可行呀

- yeek 2020-10-15 08:39:27
  在并发下，更新缓存可能造成缓存中部分数据不一致，删除重建缓存，则不会

- Dovelol 2020-10-14 20:57:38
  “应用要把数据 X 的值从 10 更新为 3，先在 Redis 缓存中删除了 X 的缓存值，但是更新数据库却失败了。如果此时有其他并发的请求访问 X，会发现 Redis 中缓存缺失，紧接着，请求就会访问数据库，读到的却是旧值 10”。老师好，这一段更新数据库失败说明数据库的值就是10，缓存删除了再从数据库读到的值就是10，这个怎么能说是旧值呢？这个流程就相当于是缓存被删了，数据库没修改那对数据一致性没有印象吧？

- 早起不吃虫 2020-10-14 09:57:59
  老师老师
  在删除缓存更新DB的过程中有大量请求落到数据库上面，这个过程可以使用分布式锁控制访问量么
  以及请求未获取到锁未获取到请求结果，这些请求要怎么操作么

- 那时刻 2020-10-14 09:30:21
  请问老师，在采用基于消息队列的重试机制来解决数据不一致问题时，在数据删除或更新，我们就要把这些值从消息队列中去除。如果数据已经成功的删除或者更新，但是在从消息队列删除过程失败，导致已经处理的消息依然在消息队列中，这种情况怎么处理呢？

  我目前想到的是通过增加消息版本号来实现幂等操作。不知道有木有其它方法？