34-第23~33讲课后思考题答案及常见问题答疑

你好，我是蒋德钧。

今天，又到了我们的答疑时间，我们一起来学习下第23～33讲的课后思考题。同时，我还会给你讲解两道典型问题。

课后思考题答案

第23讲

问题：Redis的只读缓存和使用直写策略的读写缓存，都会把数据同步写到后端数据库中，你觉得它们有什么区别吗？

答案：主要的区别在于，当有缓存数据被修改时，在只读缓存中，业务应用会直接修改数据库，并把缓存中的数据标记为无效；而在读写缓存中，业务应用需要同时修改缓存和数据库。

我把这两类缓存的优劣势汇总在一张表中，如下所示：

缓存模式	优势	劣势
只读缓存	数据更新时，可以在客户端快速标记失效缓存key（例如，用HashMap数据结构记录下），避免了和数据库一起更新的一致性保证开销	数据再次被读取时，有一次缓存缺失，会影响性能
使用直写策略的读写缓存	数据再次被读取时，可以直接在缓存中命中，性能较好	数据在缓存和数据库中更新时要保证一致性，保证机制有额外开销

拼课微信：699250

第24讲

问题：Redis缓存在处理脏数据时，不仅会修改数据，还会把它写回数据库。我们在前面学过Redis的只读缓存模式和两种读写缓存模式（带同步直写的读写模式，带异步写回的读写模式）），请你思考下，Redis缓存对应哪一种或哪几种模式？

答案：如果我们在使用Redis缓存时，需要把脏数据写回数据库，这就意味着，Redis中缓存的数据可以直接被修改，这就对应了读写缓存模式。更进一步分析的话，脏数据是在被替换出缓存时写回后端数据库的，这就对应了带有异步写回策略的读写缓存模式。

第25讲

问题：在只读缓存中对数据进行删改时，需要在缓存中删除相应的缓存值。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，直接更新缓存值有什么好处和不足吗？

答案：如果我们直接在缓存中更新缓存值，等到下次数据再被访问时，业务应用可以直接从缓存中读取数据，这是它的一大好处。

不足之处在于，当有数据更新操作时，我们要保证缓存和数据库中的数据是一致的，这就可以采用我在第25讲中介绍的重试或延时双删方法。不过，这样就需要在业务应用中增加额外代码，有一定的开销。

第26讲

问题：在讲到缓存雪崩时，我提到，可以采用服务熔断、服务降级、请求限流三种方法来应对。请你思考下，这三个方法可以用来应对缓存穿透问题吗？

答案：关于这个问题，@徐培同学回答得特别好，他看到了缓存穿透的本质，也理解了穿透和缓存雪崩、击穿场景的区别，我再来回答一下这个问题。

缓存穿透这个问题的本质是查询了Redis和数据库中没有的数据，而服务熔断、服务降级和请求限流的方法，本质上是为了解决Redis实例没有起到缓存层作用的问题，缓存雪崩和缓存击穿都属于这类问题。

在缓存穿透的场景下，业务应用是要从Redis和数据库中读取不存在的数据，此时，如果没有人工介入，Redis是无法发挥缓存作用的。

一个可行的办法就是**事前拦截**，不让这种查询Redis和数据库中都没有的数据的请求发送到数据库层。

使用布隆过滤器也是一个方法，布隆过滤器在判别数据不存在时，是不会误判的，而且判断速度非常快，一旦判断数据不存在，就立即给客户端返回结果。使用布隆过滤器的好处是既降低了对Redis的查询压力，也避免了对数据库的无效访问。

另外，这里，有个地方需要注意下，对于缓存雪崩和击穿问题来说，服务熔断、服务降级和请求限流这三种方法属于有损方法，会降低业务吞吐量、拖慢系统响应、降低用户体验。不过，采用这些方法后，随着数据慢慢地重新填充回Redis，Redis还是可以逐步恢复缓存层作用的。

第27讲

问题：使用了LFU策略后，缓存还会被污染吗？

答案：在Redis中，我们使用了LFU策略后，还是有可能发生缓存污染的。@yeek回答得不错，我给你分享下他的答案。

在一些极端情况下，LFU策略使用的计数器可能会在短时间内达到一个很大值，而计数器的衰减配置项设置得较大，导致计数器值衰减很慢，在这种情况下，数据就可能在缓存中长期驻留。例如，一个数据在短时间内被高频访问，即使我们使用了LFU策略，这个数据也有可能滞留在缓存中，造成污染。

第28讲

问题：这节课，我向你介绍的是使用SSD作为内存容量的扩展，增加Redis实例的数据保存量，我想请你来聊一聊，我们可以使用机械硬盘来作为实例容量扩展吗？有什么好处或不足吗？

拼课微信：699250

答案：这道题有不少同学（例如@Lemon、@Kaito）都分析得不错，我再来总结下使用机械硬盘的优劣势。

从容量维度来看，机械硬盘的性价比更高，机械硬盘每GB的成本大约在0.1元左右，而SSD每GB的成本大约是0.4~0.6元左右。

从性能角度来看，机械硬盘（例如SAS盘）的延迟大约在3~5ms，而企业级SSD的读延迟大约是60~80us，写延迟在20us。缓存的负载特征一般是小粒度数据、高并发请求，要求访问延迟低。所以，如果使用机械硬盘作为Pika底层存储设备的话，缓存的访问性能就会降低。

所以，我的建议是，如果业务应用需要缓存大容量数据，但是对缓存的性能要求不高，就可以使用机械硬盘，否则最好是用SSD。

第29讲

问题：Redis在执行Lua脚本时，是可以保证原子性的，那么，在课程里举的Lua脚本例子（lua.script）中，你觉得是否需要把读取客户端ip的访问次数，也就是GET(ip)，以及判断访问次数是否超过20的判断逻辑，也加到Lua脚本中吗？代码如下所示：

```
local current
current = redis.call("incr",KEYS[1])
if tonumber(current) == 1 then
    redis.call("expire",KEYS[1],60)
end
```

答案：在这个例子中，要保证原子性的操作有三个，分别是INCR、判断访问次数是否为1和设置过期时间。而对于获取IP以及判断访问次数是否超过20这两个操作来说，它们只是读操作，即使客户端有多个线程并发执行这两个操作，也不会改变任何值，所以并不需要保证原子性，我们也就不用把它们放到Lua脚本中了。

第30讲

问题：在课程里，我提到，我们可以使用SET命令带上NX和EX/PX选项进行加锁操作，那么，我们是否可以用下面的方式来实现加锁操作呢？

```
// 加锁
SETNX lock_key unique_value
EXPIRE lock_key 10S
// 业务逻辑
DO THINGS
```

答案：如果使用这个方法实现加锁的话，SETNX和EXPIRE两个命令虽然分别完成了对锁变量进行原子判断和值设置，以及设置锁变量的过期时间的操作，但是这两个操作一起执行时，并没有保证原子性。

如果在执行了SETNX命令后，客户端发生了故障，但锁变量还没有设置过期时间，就无法在实例上释放了，

拼课微信：699250

第31讲

问题：在执行事务时，如果Redis实例发生故障，而Redis使用的是RDB机制，那么，事务的原子性还能得到保证吗？

答案：当Redis采用RDB机制保证数据可靠性时，Redis会按照一定的周期执行内存快照。

一个事务在执行过程中，事务操作对数据所做的修改并不会实时地记录到RDB中，而且，Redis也不会创建RDB快照。我们可以根据故障发生的时机以及RDB是否生成，分成三种情况来讨论事务的原子性保证。

1. 假设事务在执行到一半时，实例发生了故障，在这种情况下，上一次RDB快照中不会包含事务所做的修改，而下一次RDB快照还没有执行。所以，实例恢复后，事务修改的数据会丢失，事务的原子性能得到保证。
2. 假设事务执行完成后，RDB快照已经生成了，如果实例发生了故障，事务修改的数据可以从RDB中恢复，事务的原子性也就得到了保证。
3. 假设事务执行已经完成，但是RDB快照还没有生成，如果实例发生了故障，那么，事务修改的数据就会全部丢失，也就谈不上原子性了。

第32讲

问题：在主从集群中，我们把slave-read-only设置为no，让从库也能直接删除数据，以此来避免读到过期数据。你觉得，这是一个好方法吗？

答案：这道题目的重点是，假设从库也能直接删除过期数据的话（也就是执行写操作），是不是一个好方法？其实，我是想借助这道题目提醒你，主从复制中的增删改操作都需要在主库执行，即使从库能做删除，也不要在从库删除，否则会导致数据不一致。

例如，主从库上都有a:stock的键，客户端A给主库发送一个SET命令，修改a:stock的值，客户端B给从库发送了一个SET命令，也修改a:stock的值，此时，相同键的值就不一样了。所以，如果从库具备执行写操作的功能，就会导致主从数据不一致。

@Kaito同学在留言区对这道题做了分析，回答得很好，我稍微整理下，给你分享下他的留言。

即使从库可以删除过期数据，也还会有不一致的风险，有两种情况。

第一种情况是，对于已经设置了过期时间的key，主库在key快要过期时，使用expire命令重置了过期时间，例如，一个key原本设置为10s后过期，在还剩1s就要过期时，主库又用expire命令将key的过期时间设置为60s后。但是，expire命令从主库传输到从库时，由于网络延迟导致从库没有及时收到expire命令（比如延后了3s从库才收到expire命令），所以，从库按照原定的过期时间删除了过期key，这就导致主从数据不一致了。

第二种情况是，主从库的时钟不同步，导致主从库删除时间不一致。

另外，当slave-read-only设置为no时，如果在从库上写入的数据设置了过期时间，Redis 4.0前的版本不会删除过期数据，而Redis 4.0及以上版本会在数据过期后删除。但是，对于主库同步过来的带有过期时间的数据，从库仍然不会主动进行删除。

拼课微信：699250

问题：假设我们将min-slaves-to-write设置为1，min-slaves-max-lag设置为15s，哨兵的down-after-milliseconds设置为10s，哨兵主从切换需要5s，而主库因为某些原因卡住了12s。此时，还会发生脑裂吗？主从切换完成后，数据会丢失吗？

答案：主库卡住了12s，超过了哨兵的down-after-milliseconds 10s阈值，所以，哨兵会把主库判断为客观下线，开始进行主从切换。因为主从切换需要5s，在主从切换过程中，原主库恢复正常。min-slaves-max-lag设置的是15s，而原主库在卡住12s后就恢复正常了，所以没有被禁止接收请求，客户端在原主库恢复后，又可以发送请求给原主库。一旦在主从切换之后有新主库上线，就会出现脑裂。如果原主库在恢复正常后到降级为从库前的这段时间内，接收了写操作请求，那么，这些数据就会丢失了。

典型问题答疑

在第23讲中，我们学习了Redis缓存的工作原理，我提到了Redis是旁路缓存，而且可以分成只读模式和读写模式。我看到留言区有一些共性问题：如何理解Redis属于旁路缓存？Redis通常会使用哪种模式？现在，我来解释下这两个问题。

如何理解把Redis称为旁路缓存？

有同学提到，平时看到的旁路缓存是指，写请求的处理方式是直接更新数据库，并删除缓存数据；而读请求的处理方式是查询缓存，如果缓存缺失，就读取数据库，并把数据写入缓存。那么，课程中说的“Redis属于旁路缓存”是这个意思吗？

其实，这位同学说的是典型的只读缓存的特点。而我把Redis称为旁路缓存，更多的是从“业务应用程序如何使用Redis缓存”这个角度来说的。**业务应用在使用Redis缓存时，需要在业务代码中显式地增加缓存的操作逻辑**。

例如，一个基本的缓存操作就是，一旦发生缓存缺失，业务应用需要自行去读取数据库，而不是缓存自身去从数据库中读取数据再返回。

为了便于你理解，我们再来看下和旁路缓存相对应的、计算机系统中的CPU缓存和page cache。这两种缓存默认就在应用程序访问内存和磁盘的路径上，我们写的应用程序都能直接使用这两种缓存。

我之所以强调Redis是一个旁路缓存，也是希望你能够记住，在使用Redis缓存时，我们需要修改业务代码。

使用Redis缓存时，应该用哪种模式？

我提到，通用的缓存模式有三种：**只读缓存模式、采用同步直写策略的读写缓存模式、采用异步写回策略的读写缓存模式**。

一般情况下，我们会把Redis缓存用作只读缓存。只读缓存涉及的操作，包括查询缓存、缓存缺失时读数据库和回填，数据更新时删除缓存数据，这些操作都可以加到业务应用中。而且，当数据更新时，缓存直接删除数据，缓存和数据库的数据一致性较为容易保证。

当然，有时我们也会把Redis用作读写缓存，同时采用同步直写策略。在这种情况下，缓存涉及的操作也都可以加到业务应用中。而且，和只读缓存相比有一个好处，就是数据修改后的最新值可以直接从缓存中读取

拼课微信：699250

对于采用异步写回策略的读写缓存模式来说，缓存系统需要能在脏数据被淘汰时，自行把数据写回数据库，但是，Redis是无法实现这一点的，所以我们使用Redis缓存时，并不采用这个模式。

小结

好了，这次的答疑就到这里。如果你在学习的过程中遇到了什么问题，欢迎随时给我留言。

最后，我想说，“学而不思则罔，思而不学则殆”。你平时在使用Redis的时候，不要局限于你眼下的问题，你要多思考问题背后的原理，积累相应的解决方案。当然，在学习课程里的相关操作和配置时，也要有意识地亲自动手去实践。只有学思结合，才能真正提升你的Redis实战能力。

精选留言：

- 每天晒白牙 2020-11-06 02:31:41
  知识点还没掌握，要再重新学习一遍了

拼课微信：699250