链滴

数据库缓存最终一致性的方案

作者：AutisticV5

原文链接：https://ld246.com/article/1577432723223

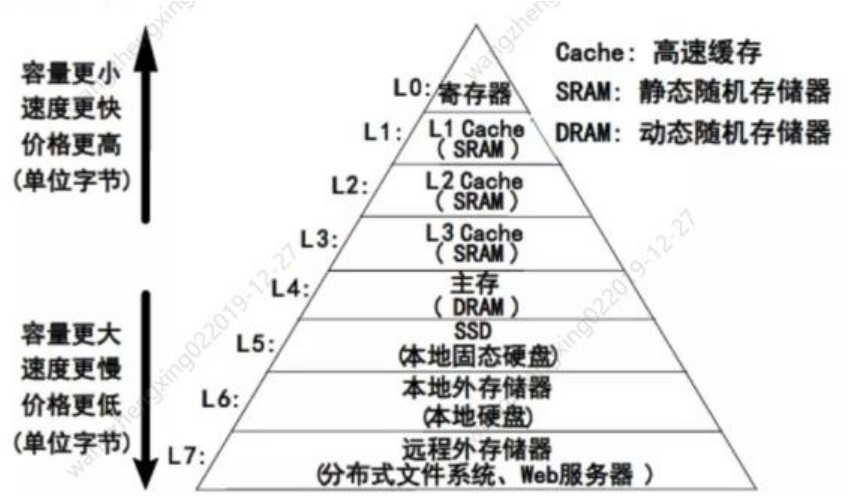
来源网站：链滴

背景

缓存是软件开发中一个非常有用的概念，数据库缓存更是在项目中必然会遇到的场景。

缓存是什么

存储的速度是有区别的。缓存就是把低速存储的结果，临时保存在高速的存储地技术。



为什么需要缓存

存储如mysql通常支持完整的ACID特性，因为可靠性，持久性等因素，性能普遍不高，高并发查询会mysql带来压力，造成数据库系统的不稳定。同时也容易产生延迟。根据局部原理，80%请求会落到2%的热点数据上，在读多写少场景，增加一层缓存非常有助提升系统吞吐量和健壮性。

原文链接：数据库缓存最终一致性的方案

存在问题

存储的数据随着时间可能会发生变化，而缓存中的数据就会不一致。具体能容忍的不一致时间，需要体业务具体分析，但是通常的业务都需要做到最终一致。

redis作为mysql缓存

通常的开发模式中，都会使用mysql作为存储，而redis作为缓存，加速和保护mysql。但是当mysql据更新之后，redis怎么保持同步呢。

强一致性同步成本太高，如果追求强一致，那么没必要用缓存了，直接用mysql即可。通常考虑的都最终一致性。

解决方案

方案一

通过key的过期时间，mysql更新时，redis不更新。这种方式实现简单，但不一致的时间会很长。如读请求非常频繁，且过期时间比较长，则会产生很多长期的脏数据。

优点：

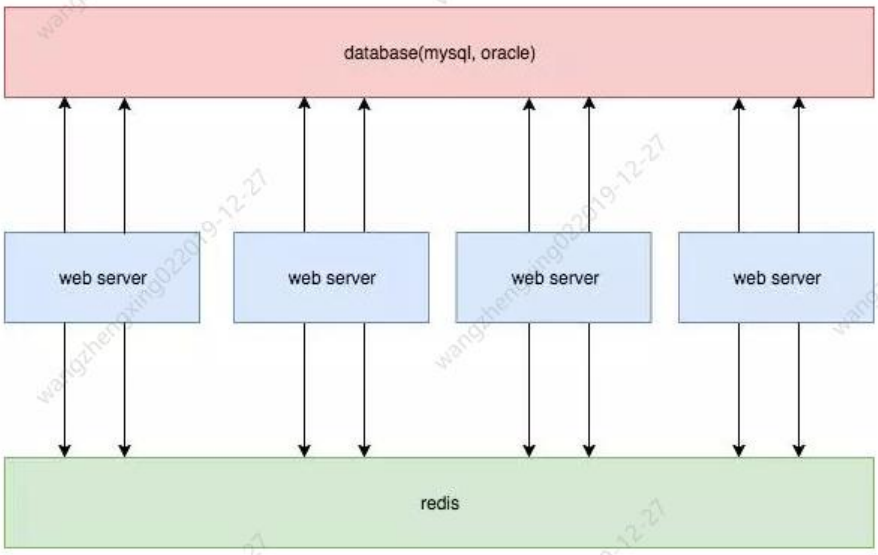
- 开发成本低，易于实现；
- 管理成本低，出问题的概率会比较小

缺点：

- 完全依赖过期时间，时间太短容易导致缓存频繁失效，太长容易有长时间更新延迟(不一致)

方案二

在方案一的基础上拓展，通过key的过期时间兜底，并且在更新mysql时，同时更新redis。



优点：

● 相对方案一，更新延迟更小

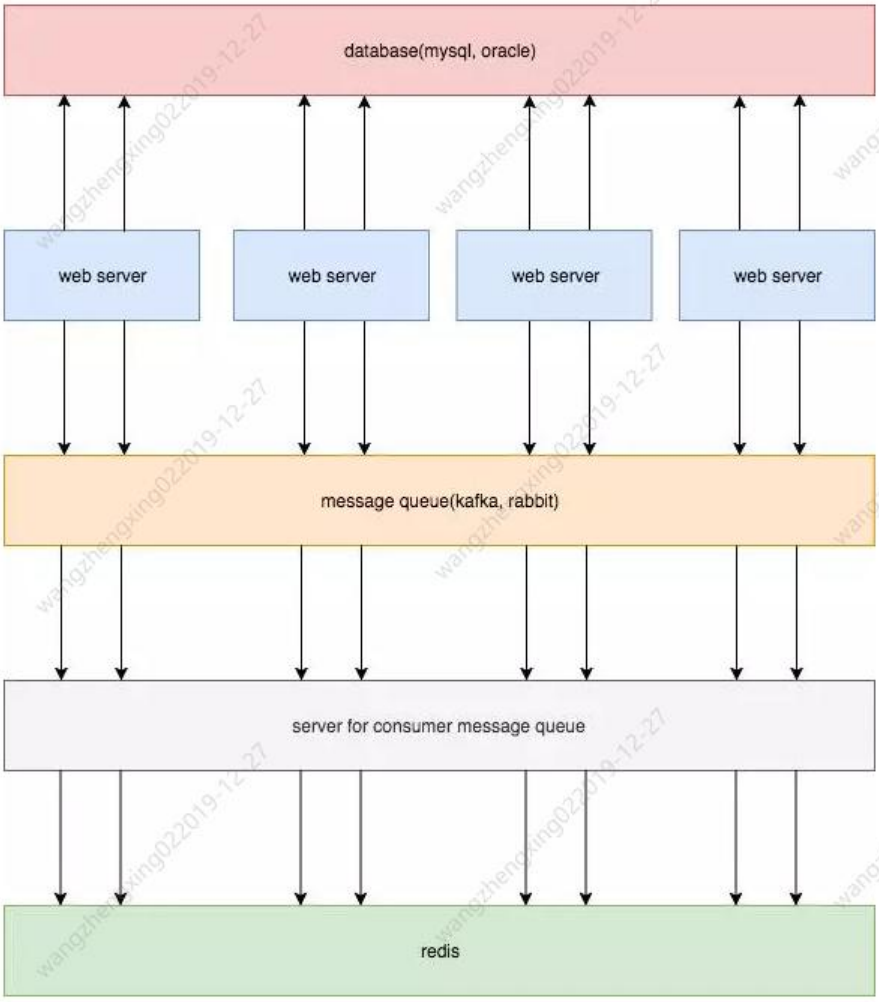
缺点：

● 如果更新mysql成功，更新redis失败，就退化到了方案一；

● 在高并发场景，业务server需要和mysql，redis同时进行连接。这样是损耗双倍的连接资源，容易成连接数过多的问题。

方案三

针对方案二的同步写redis进行优化，增加消息队列，将redis更新操作交给kafka，由消息队列保证可性，再搭建一个消费服务，来异步更新redis。



优点：

● 消息队列可以用一个句柄，很多消息队列客户端还支持本地缓存发送，有效解决了方案二连接数过的问题；

● 使用消息队列，实现了逻辑上的解耦；

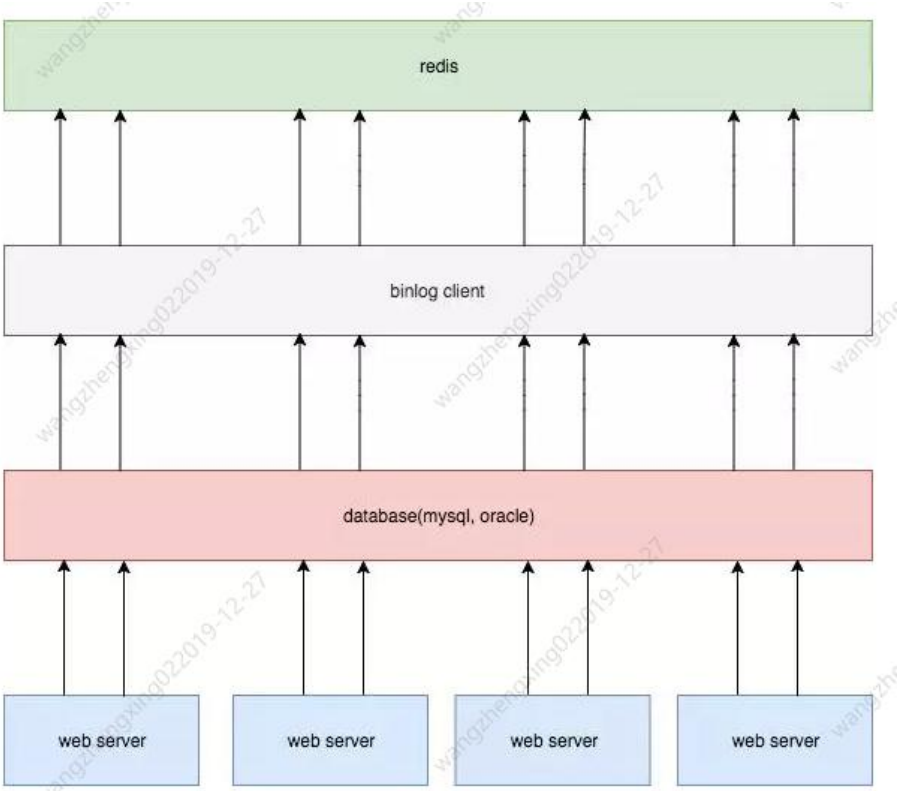
● 消息队列本身具有可靠性，通过手动提交等手段，可以至少一次消费到redis。

缺点：

- 依旧解决不了时序性问题，如果多台业务服务器分别处理针对同一行数据的两条请求，比如：a = 1; a = 5;如果mysql中是第一条先执行，而进入kafka的顺序是第二条先执行，那么数据就会产生不一致。
- 引入了消息队列，同时要增加服务消费消息，成本较高。

方案四

通过订阅binlog来更新redis，把我们搭建的消费服务，作为mysql的一个slave，订阅binlog，解析更新内容，再更新到redis。



优点：

- 在mysql压力不大情况下，延迟较低；
- 和业务完全解耦；
- 解决了时序性问题。

缺点：

- 要单独搭建一个同步服务，并且引入binlog同步机制，成本较大。

总结

方案选型

1. 首先确认产品上对延迟性的要求，如果要求极高，且数据有可能变化，别用缓存。

2. 通常来说，方案一就够了，因为能用缓存方案，通常是读多写少的场景，同时业务上对延迟具有一的包容性。方案一没有开发成本，比较使用。

3. 如果想增加更新时的及时性，就选择方案二，不过没必要做重试保证之类的。

4. 方案三，方案四针对于对延时要求比较高的业务，一个是推模式，一个是拉模式，而方案四具有更的可靠性，既然都愿意花功夫做处理消息的逻辑，不如一步到位，用方案四。

结论

一般情况，方案一够用。若延时要求高，直接选择方案四。