redis缓存与数据库一致性问题

redis与数据库数据一致性问题是个老生常谈的问题了，这里也没啥新鲜玩意，就是总结一下。

不一致产生的原因

我们在使用redis过程中，或者网上一些资料介绍，通常会这样做：先读取缓存，如果缓存不存在，则读取数据库。伪代码如下：

Object stuObj = new Object();

public Stu getStuFromCache(String key){

Stu stu = (Stu) redis.get(key);

if(stu == null){

synchronized (stuObj) {

stu = (Stu) redis.get(key);

if(stu == null){

Stu stuDb = db.query();

redis.set(key, stuDb);

}

}

}

return stu;

}

上面加锁是为了防止过多的查询走到数据库层。

写数据库伪代码

public void setStu(){

redis.del(key);

db.write(obj);

}

不管是先写库、再删除缓存，还是先删缓存、再写库，都有可能出现数据不一致的情况。因为写和读是并发的，没法保证顺序，如果删了缓存，还没有来得及写库，另一个线程就来读取，发现缓存为空，则去数据库中读取数据写入缓存，此时缓存中为脏数据。如果先写了库，再删除缓存前，写库的线程宕机了，没有删除掉缓存，则也会出现数据不一致情况。

如果是redis集群，或者主从模式，写主读从，由于redis复制存在一定的时间延迟，也有可能导致数据不一致。

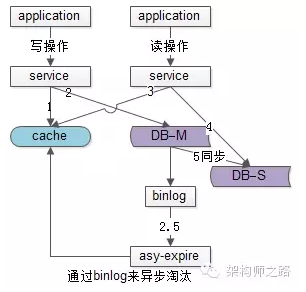
优化思路

双删 + 超时

在写库前后都进行redis.del(key)操作，并且设定合理的超时时间。这样最差的情况是在超时时间内存在不一致，当然这种情况及其少见，可能的原因就是服务宕机。此种情况可以满足绝大多数需求。当然这种策略要考虑redis和数据库主从同步的耗时，所以在第二次删除前最好休眠一定时间，比如500毫秒，这样毫无疑问又增加了写请求的耗时。

异步淘汰缓存

通过读取binlong的方式，异步淘汰缓存。



好处：业务代码侵入性低，将缓存与数据库不一致的时间尽可能缩小。