没人告诉过你更复杂的缓存穿透怎么解决

原创 艾小仙 艾小仙 今天

收录于话题
#技术架构

5个

你应该从网上看过太多的文章说缓存穿透怎么解决？无非就是布隆过滤器，缓存空值什么的。

但是，更深入的一个问题，缓存空值有没有问题？如果缓存的空值太多怎么办？

如果用的redis，那么太多的空值会不会打爆你的redis？如果用的本地缓存，会不会打爆你的内存？继而引发的问题就是还是会打爆你的数据库。

从线上问题说起

前不久，我们线上环境压测，在QPS压倒2W之后RT达到了几十秒，排查后发现是redis的连接数不够导致大量的连接超时。

经过考虑之后，我们最终决定弃用redis缓存的方案，改为本地缓存，因为我们缓存的都是一些配置信息，实际上几个月都不太可能修改，而redis配置的连接数是200，5分钟超时，数据量实际上也就只有几千条而已，实际上来说并没有很大的必要，本地缓存完全就可以解决问题了。

本地缓存使用Guava的LoadingCache实现。

```
public class CacheManager {
    private static LoadingCache<String, Object> cache = null;

    static {
        cache = CacheBuilder.newBuilder()
                //缓存池大小
                .maximumSize(10000)
                //最后一次访问之后多少时间后移除
                .expireAfterAccess(1, TimeUnit.DAYS)
                //被创建或值被替换后多少时间移除
                .expireAfterWrite(1, TimeUnit.DAYS)
```

```
                .recordStats()
                .build(new CacheLoader<String, Object>() {
                    @Override
                    public Object load(String s) throws Exception {
                        // 处理缓存键不存在缓存值时的处理逻辑
                        return "";
                    }
                });
    }

    public static void put(String key, Object value) {
        cache.put(key, value);
    }

    public static Object get(String key) {
        Object value = null;
        try {
            value = cache.getIfPresent(key);
        } catch (Exception e) {
            //---
        }
        return value;
    }
```

但是修改完之后，压测之后还是发现有接口全部走到数据库查询，先排查代码，是否是代码的BUG导致实际没有生效，后来发现实际上发生了**缓存穿透**，压测使用了一些数据库中不存在的记录，导致了穿透的问题，实际上这个问题在使用redis的时候也一直存在，只是由于连接数的问题一直没有发现而已。

接下来就是考虑怎么解决的问题？

由于我们都是缓存的一些配置信息，几千条数据而已，最终考虑简单解决的办法。直接把所有的key全部从数据库查出来缓存下来，查数据库之前直接根据key过滤一把，如果不存在就直接返回，不要走数据库查询了。

当然，这是由于我们的场景比较简单，这样直接处理就行了，那么，如果再复杂一点，比如上亿的缓存数据呢？

解决方案

前置过滤

如果说类似我这种比较简单的一些缓存，使用我上面说的解决方案也可以，还有一些缓存的key是比如ID之类，也可以根据一定的范围规则去提前过滤，比如缓存的key明确知道在1-10万的范围之后，那么过滤掉在这个范围之外的请求直接返回就可以了。

当然，很明显这种简单的规则过滤适用于数据量不是很大，并且数据不会频繁发生改变的情况。

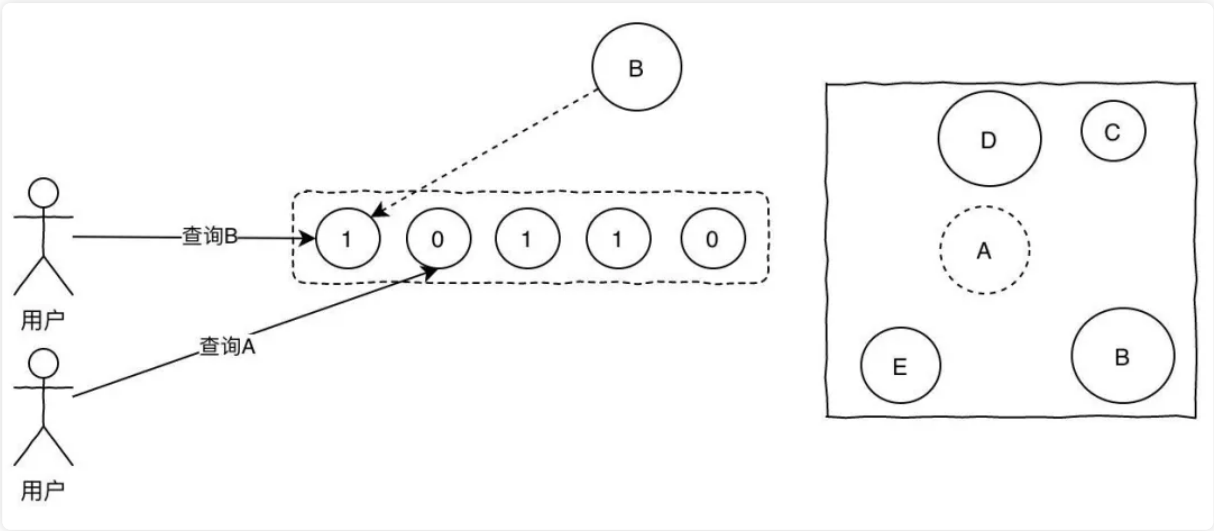
布隆过滤器

对于上述场景，因为数据量很小，简单的代码实现缓存即可，如果说数据量很大的话，比如有一亿个key，使用布隆过滤器就是个更优解。

我们可以每天定时把所有的配置信息从数据库中查询出来构建成bitmap。

关于布隆过滤器我前面的文章也有写过，贴上之前的图一张，如果查询的位置都是1的话说明key存在，反之只要有一个0则说明肯定不存在。

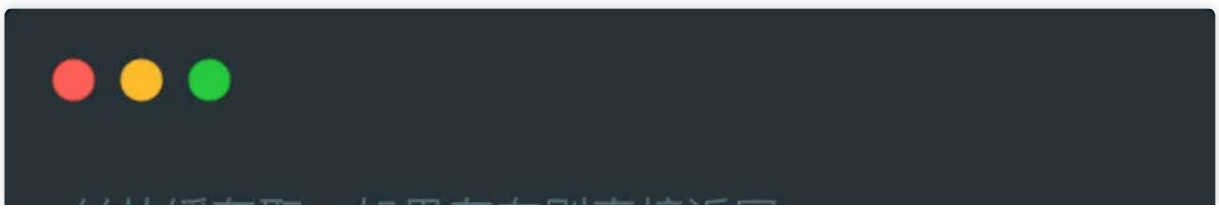
使用布隆过滤器的缺点也很明显，存在一定概率的误判。当然，既然用了，对于误判比例、内存占用等等问题应该事先评估好。



缓存空值

这个是网上说烂的问题，但是缓存空值的空值太多明显也是有问题的，再进一步解决方案就是快速过期。

一般来说，普通的缓存的写法如下，先查缓存，如果缓存存在则直接返回，如果缓存没有则去数据库查询，结果不是空就保存到缓存中。



```
//从缓存取，如果存在则直接返回
Object value = Cache.get(key);
if (value != null) {
  return value;
}
//缓存没有则去查数据库,如果不是空则放入缓存中
value = xxMapper.get(key);
if (value != null) {
    Cache.put(key, value);
}
return value;
```

改进版的写法就是缓存空对象，针对空的数据，设置过期时间，比如10分钟，快速过期，防止太多的空值问题。

```
//从缓存取，如果存在则直接返回
Object value = Cache.get(key);
if (value != null) {
  return value;
}
//缓存没有则去查数据库,如果不是空则放入缓存中
value = xxMapper.get(key);
Cache.put(key, value);
if (value == null) {
  Cache.put(key, new Object());
  Cache.expire(key,10);
}
return value;
```

```
return value;
```

但是这个解决方案仍然有点小问题，就是短暂的数据不一致的问题。

想象一下如果缓存的空值这时候实际上已经有值了，那么在过期时间的这段时间内就可能存在短暂的数据不一致。

总结

缓存穿透的问题总结下来就是三点，这三个方式不是说是隔离的解决方案，他们可以结合在一起使用。

首先看数据量，如果数据量很小并且没有频繁变更的话，选择前置过滤的方式，根据具体的业务规则来处理就可以。

如果数据量大的话，可以选择使用布隆过滤器，但是存在一定概率的误判。

通过前置的拦截，应该拦截住大部分的流量，避免直接打爆数据库。

最后，可以使用缓存空值并且设置快速过期的方式来作为一个兜底的方案。

如果还有问题，那么就是限流、降级了。

- END -

往期推荐

《我想进大厂》之网络篇夺命连环12问

面试官：缓存一致性问题怎么解决？

我摊牌了，大厂面试Linux就这5个问题

淘宝|蚂蚁|菜鸟|盒马|嘀嘀|饿了么面经