课时 08

哈希表在 Facebook 和Pinterest 中的应用

1. 均摊时间复杂度
2. 哈希表的应用——缓存
3. 哈希表在Facebook中的应用

均摊时间复杂度

哈希表是一个可以根据键来直接访问在内存中存储位置的值的数据结构

虽然哈希表无法对存储在自身的数据进行排序

但是它的插入和删除操作的均摊时间复杂度都属于均摊 O(1)（Amortized O(1)）

均摊时间复杂度

均摊时间复杂度可以这样来理解：

如果说一个数据结构的均摊时间复杂度是 X

那么这个数据结构的时间复杂度在大部分情况下都可以达到 X

只有当在极少数的情况下出现时间复杂度不是 X

均摊时间复杂度

为什么在分析哈希表的时候我们会用到均摊时间复杂度呢？

均摊时间复杂度

主要是因为在处理哈希碰撞的时候

需要花费额外的时间去寻找下一个可用空间，这样造成的时间复杂度并不是 O(1)

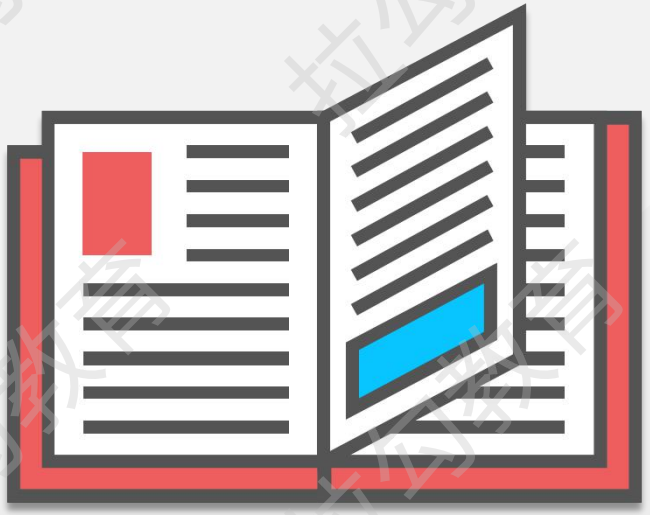
极端情况下，所有插入的数据如果都产生了哈希碰撞

而我们采用的是分离链接法来解决哈希碰撞，那时间复杂度就变成了 O(N)

均摊时间复杂度

因为这种特性，使得哈希表的应用十分广泛，很常见的一种应用就是缓存（Cache）

缓存这个概念其实不单单只是针对于内存来说的，可以抽象地把缓存看作是一种读取速度更快的媒介



均摊时间复杂度

Memcached 和 Redis 这两个框架是现在应用得最广泛的两种缓存系统

它们的底层数据结构本质都是哈希表

Memcached 缓存

Memcache 是一种分布式的键值对存储系统，它的值可以存储多种文件格式，比如图片、视频等

很大特点就是数据完全保存在内存中

可以把保存在 Memcache 中的数据看作是 Memcache 维护了一个超级大的哈希表数据结构

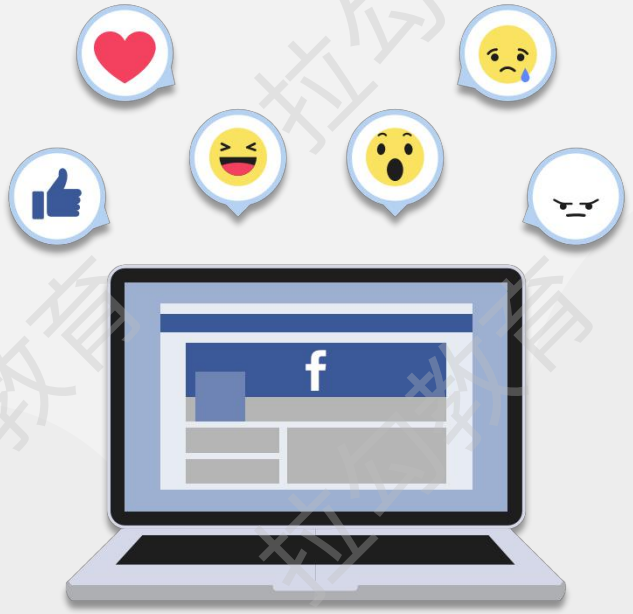
并没有任何内容保存在硬盘中

Memcached 缓存

Facebook 会把每个用户发布过的文字和视频、去过的地方、点过的赞、喜欢的东西等内容都保存下来

自己做了一个名叫 mcrouter 的服务器集群出来，可以将不同的数据请求导向不同的 Memcache

而他们还开源了这个管理 mcrouter 的代码（https://github.com/facebook/mcrouter）



Memcached 缓存

拉勾
LAGOU.COM

很多数据不从数据库读取的话是拿不到最新数据的，怎么办呢？

Memcached 缓存

解决的方案是在第一次读取数据之后

将这些通过数据库算出的结果存放在 Memcache 中并设定一个过期时间

只要数据没有超过设置的过期时间

后续的所有读取都不需要通过数据库计算，而是直接从 Memcache 中读取

Memcached 缓存

Facebook 好友生日提醒

其做法是将用户 ID 和用户的生日日期作为键值对存放在 Memcache 中

每个用户在当天登录的时候，会先以所有的好友 ID 作为键，去 Memcache 中寻找是否有他们的数据存在

存在则判断当天的日期是否是好友生日的日期，然后决定是否发送生日提醒

不存在则先去数据库中拿出所有好友的生日日期，然后存在 Memcache 中，最后返回给用户判断

Memcached 缓存

Facebook Live（直播应用）

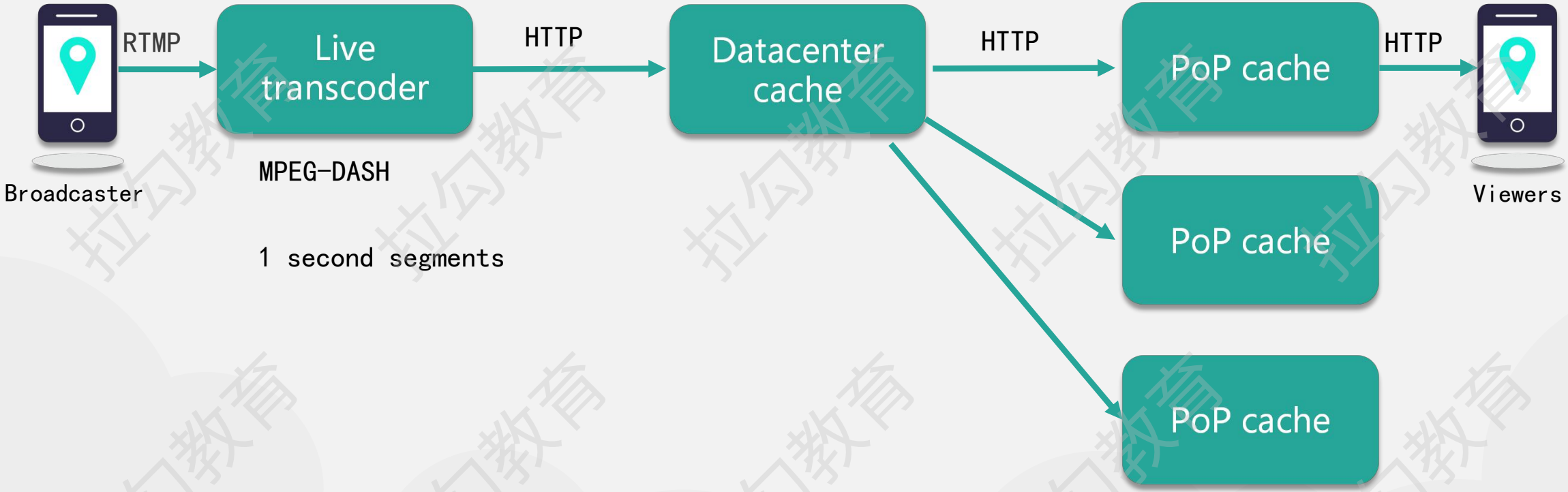
它的一个特点是即使用户错过了直播时间，后面也可以通过访问直播链接来观看回放

Facebook 把每一个直播的视频流数据按照每一秒钟的时间分割成一个块（Segment）

每一个视频流块都会被存放在 Memcache 中

当用户读取直播视频流的时候，会以直播 ID 加上这个时间进度作为哈希表的键，来读取每一秒的直播视频

Memcached 缓存



2016 年 Facebook 技术讲座的整体架构

Redis 缓存

Redis 与 Memcache 一样，同样是一个保存键值对的存储系统

与 Memcache 的一个很大不同是，保存在 Redis 上的数据会每间隔一段时间写入到磁盘中

以防止当机器宕机后可以重新恢复数据

所支持的数据类型也十分简单

包括 Strings、Lists、Sets、Sorted Sets、Hashes 和之前介绍的 Bitmaps 等

Redis 缓存

在 Pinterest 的应用里，每个用户都可以发布一个叫 Pin 的东西

Pin 可以是自己原创的一些想法，也可以是物品，还可以是图片视频等

不同的 Pin 可以被归类到一个 Board 里面

Redis 缓存

可以通过关注 Board 和 Pin 来获取推荐系统给用户发布的内容

例如

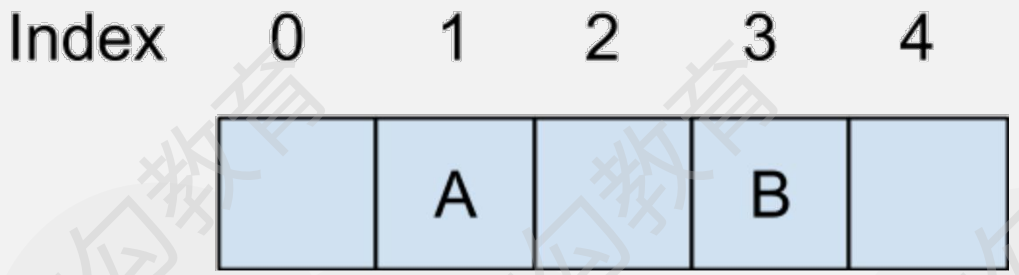
如果一个用户关注了 Board A，那每当 Board A 有新的 Pin 加进去的时候都会推送给这个用户

用户 A 也可以通过关注用户 B，那以后用户 B 所发布的每一个 Pin 和 Board 都会推送给用户 A

假设这里的哈希函数是 H(X)，键 A 和键 B 都已经插入到哈希表中了，而C并没有插入

所以我们判断出A和B是在这个集合里的，而C并不存在集合里

H(A) = 1
H(B) = 3
H(C) = 4



Redis 缓存

Sorted Sets 这个类型其实就是在 Set 外的基础上加上了一个 Score 的概念

Redis 内部会根据 Score 的大小对插入的键进行排序

比如说，Pinterest 会把一个用户所关注的其他用户按照以关注时间戳为 Score，关注的用户 ID 作为键存放在 Sorted Sets 里

好处

当一个用户在查看自己所有关注过的用户时，可以读取所有存储在这个 Sorted Sets 里的数据，而因为 Score 的值是关注这个用户的时间戳，所以读取数据出来的时候，会按照自己关注他们的时间顺序读取出来，而不是乱序地读取关注过的用户