系统设计实例讲解：如何设计 post 系统

作者：东邪老师，九章算法版权所有

一、题目描述：

实现前 5 分钟，1 小时，24 小时内分享最多的 post 的系统

这个题是一个很好的面试题，因为可以从算法和系统两个角度进行考察。

二、从算法的角度分析

从算法的角度分析，可以简单的称之为 **Top K Frequent Elements in Recent X mins.**

算法的角度，本质就是设计一个**数据结构**，支持给某个 key 的 count+1（有一个 post 被分享了），给某个 key 的 count-1（有一个分享的计数已经过期了），然后查询 Top k。

做法是维护一个有序序列（用链表来维护），每个链表的节点的 key 是 count，value 是 list of elements that has this count ,也用 linked list 串起来。 比如 a 出现 2 次 ,b 出现 3 次，c 出现 2 次，d 出现 1 次。那么这个链表就是：

```
{3: [b]} --> {2: [a ->c]} --> {1: [d]}
```

然后另外还需要一个 hashmap，key 是 element，value 是这个 element 在链表上的具体位置。

因为每一次的操作都是 count + 1 和 count - 1，那么每次你通过 hashmap 找到对应的 element 在数据结构中的位置，+1 的话，就是往头移动一格，-1 的话，就是往尾巴移动一格。总而言之复杂度都是 O(1)。当你需要找 Top K 的时候，也是 O(k)的时间 可以解决的。

这个数据结构来自 LFU 的论文：

http://dhruvbird.com/lfu.pdf

LintCode 上有 LFU 的题， 可以做一下：

www.lintcode.com/problem/lfu-cache

三、从系统设计的角度分析

所以一般来说，你可能首先需要按照 LFU 的思路答出上述的方法。这个就过了第一关，**算法关**。但是还没结束，这个题还有第二关，那就是**系统设计关**。上面的算法从算法的角度没办法更优了，每个分享操作都是 O(1)的代价，每个求 Top K 都是 O(k)的代价。已经很棒了。但是系统的角度出发，会存在这样一些问题：

1. 如果 QPS 比较高，比如 1m，这个数据结构因为要加锁才能处理，所以会很慢。

2. 分享的数据本身是分布式的，而不是中心化的，也就是说，比如有 1000 台 web 服务器，那么这 1000 台 web 服务器的是最先获得哪个帖子被分享的数据的，但是这些数据又都分布在这 1000 台 web 服务器中，如果用一个中心化的节点来做这个数据结构的服务，那么很显然这个中心节点会成为瓶颈。
3. 比如这个系统用在 twitter 这样的服务中，根据长尾理论，有 80%或者更多的帖子连 Top 20% 都排不进去。而通常来说，从产品的角度，我们可能只需要知道 Top 20 最多是 Top 100 的数据就可以了。整个系统浪费了很多时间去统计那些永远不会成为 Top 100 的数据。
4. 题目的要求是“5 分钟，1 小时，24 小时”，而不是“最近 2 分零 30 秒”，“最近 31 秒”，也存在较大的优化空间
5. 真实产品实时性要求和准确性没有那么高。你需要查询最近 5 分钟的 Top K，结果得出的是最近 5 分 02 秒的 Top K 在产品中是没有太大问题的。
6. 查询 Top k 的次数远低于 count + 1 和 count -1 的次数。

综上所述我们给出一些针对性优化策略：

01 分布式统计 Distributed:

每隔 5~10 秒向中心节点汇报数据

也就是说，哪些帖子被分享了多少次这些数据，首先在 web server 中进行一次缓存，也就是说 web server 的一个进程接收到一个分享的请求之后，比如 tweet_id=100 的 tweet 被

分享了。那么他把这个数据先汇报给 web server 上跑着的 agent 进程，这个 agent 进程在机器刚启动的时候，就会一直运行着，他接受在台 web server 上跑着的若干个 web 进程 (process) 发过来的 count +1 请求。

这个 agent 整理好这些数据之后，每隔 5~10 秒汇报给中心节点。这样子通过 5~10s 的数据延迟，解决了中心节点访问频率过高的问题。这个设计的思路在业界是非常常用的（做这种数据统计服务的都是这么做的），我们在《系统设计班》的 datadog 一节的课中，就讲到过用这种思路来统计每一个 event 发生了多少次。

02 分阶段统计 Level

在《系统设计班》的 ratelimiter 一节课中，我们也提到了这种分阶段统计的思想。

即如果我要去算最近 5 分钟的数据，我就按照 1 秒钟为一个 bucket 的单位，收集最近 300 个 buckets 里的数据。如果是统计最近 1 小时的数据，那么就以 1 分钟为单位，收集最近 60 个 Buckets 的数据，如果是最近 1 天，那么就以小时为单位，收集最近 24 小时的数据。那么也就是说，当来了一个某个帖子被分享了 1 次的数据的时候，这条数据被会分别存放在当前时间(以秒为单位），当前分钟，当前小时的三个 buckets 里，用于服务之后最近 5 分钟，最近 1 小时和最近 24 小时的数据统计。

你可能会疑惑，为什么要这么做呢？这么做有什么好处呢？

这样做的好处是，比如你统计最近 1 小时的数据的时候，就可以随着时间的推移，每次增加当前分钟的所有数据的统计，然后扔掉一小时里最早的 1 分钟里的所有数据。这样子就不用真的一个一个的+1 或者-1 了，而是整体的 +X 和 -X。当然，这样做之后，前面的算法部分提出

来的数据结构就不 work 了，但是可以结合下面提到的数据抽样的方法，来减小所有候选 key 的数目，然后用普通的 Top K 的算法来解决问题。

参考练习题：http://www.lintcode.com/en/problem/top-k-frequent-words/

03 数据抽样 Sample

可以进行一定程度的抽样，因为那些 Top K 的 post，一定是被分享了很多很多次的，所以可以进行抽样记录。

如果是 5 分钟以内的数据，就不抽样，全记录。如果是最近 1 小时，就可以按照比如 1/100 的概率进行 sample。

这个思想我们在 Web Crawler 的那节课中提到过。

04 缓存 Cache

对于最近 5 分钟的结果，每隔 5s 才更新一次。

对于最近 1 小时的结果，每隔 1 分钟更新一次。

对于最近 24 小时的结果，每隔 10 分钟才更新一次。

用户需要看结果的时候，永远看的是 Cache 里的结果。另外用一个进程按照上面的更新频率去逐渐更新 Cache。

四、总结

以上的这些优化方法，基本都基于一个基本原则：**在很多的系统设计问题中，不需要做到绝对精确和绝对实时。**

特别是这种统计类的问题。如果你刷算法题刷很多，很容易陷入设计一个绝对精确和绝对实时的系统的误区。**一般来说面试中不会这么要求**，如果这么要求了，那说明考你的是算法，算法才需要绝对准确和实时。

这道题考了算法，**大数据**和**系统设计**。Top K 的算法我们在九章算法班中讲过，Top K 的大数据算法我们在大数据班中讲过，系统设计中用到的各类思想基本也都在课上讲过。所以**好好听课很重要**。

想了解更多关于系统设计的面试要求和准备建议，**欢迎参加《系统设计班》免费试听课**。

报名网址：www.jiuzhang.com