# 数据中心网络测量方案研究

# 基于Counter的测量方法

基于Counter的测量方法，主要是通过有限的计数器存储有限的大流，而小流则是忽略或是只给出一个大致估计值，所以基于Counter的测量方法给出的误差界一般都是确定的。目前基于Counter的测量方法主要有Frequent， Space Saving，RAP等，这些方法的主要思路都是在有限的计数器中只记录大流的数据包，而忽略小流的数据包，只是当计数器的个数达到上限时，处理的方法不一样。这些方法能够保存大流在有限的Counter中,并且采用hash操作能够使得运行时间在常数时间范围中。但是这些方法的缺点在于只能解决Heavy Hitter或top-k问题，对于流量矩阵估计问题解决得不够好，并且由于采用hash操作，不能很好的处理hash冲突而导致运行时间变长。

## Space saving算法

### 算法步骤

Space saving算法能够保证在Counter数足够的情况下，能够保证top-k条流都被记录下来，并且算法能够达到常数时间复杂度，对于Heavy Hitter的检测有非常好的性能。主要思想如下，T保存所有Counter，每次数据包i到达时，若没有空闲的计数器，它那么这个新到达的数据包将替换所有计数器中计数值最小的那一个j，而新的数据包的计数值为j的计数值加1，同时还将替换前的计数值记录到中。具体的算法如下：

**input**(m)

1. n0;
2. ;
3. **for each** i **do**
   1. n←n+1;
   2. **if** **then**
   3. **else if** **then**
      1. ;
      2. ;
   4. **else**
      1. ;
      2. ;
      3. ;

### 实现方式

通常Space Saving可以采用链表形式的数据结构Stream Summary实现，可以达到常数时间复杂度。Stream Summary将计数值放在一个桶里面，然后每个桶用双向链表按计数值从小到大的顺序连接，记数值相同的counter用链表连接到相应计数值的桶中，每次查找最小的counter只需要在链表头中查找就行了，如图1所示为的Stream Summary实现Space Saving算法的示例：

Value=1

Value=2

ID=X

ID=Y

(a)S=X,Y

(b)S=X,Y,Y

(c)S=X,Y,Y,Z

Value=2

ID=Z

ID=Y

图1 Stream Sumary实现Space Saving

图1所示示例最大计数器个数为2，数据流为S={X,Y,Y,Z}，(a)(b)(c)分别为每个数据包到达时Stream Summary的状态。(a)当X和Y到达时，由于计数器个数没有达到上限，所以这两个包都被记录下来，并且指向了value=1的桶中；(b)当又到达一个Y时，由于Y已经记录在链表中，则增加一个value=2的桶，将Y移动到value=2的桶中；(c)当Z到来时，由于没有多余的Counter了，所以将最小的value的桶中的X替换为Z，记为X的计数值，再将这个Counter移动到value=2的桶中，删除value=1的桶。利用这种数据结构可以达到时间复杂度为O(1)，因为每次查找最小计数值时只需要在桶的链表头查找即可。

## Space Saving算法分析

显然Space Saving算法需要的空间为O(k)，如果利用Stream Summary数据结构可以达到运行时间为O(1)。对于没有记录在Space Saving的小流可以就用最小的那个Counter值来估计，但是这对于这些小流的估计值的误差是比较大的。

对于任意分布的数据流来说，要想确保检测到误差为松弛的top-k流(即假设表示第k大的流的值，表示流*i*的值，对于任意满足的流*i*，一定记录在Space Saving中)，必须保证Space Saving的Counter数为，其中表示流的个数。

对于常见的Zipf分布的数据流来说，若参数，只需要Counter数至少为，即可以保证准确地解决top-k的流的检测问题；若参数，只需要Counter数至少为，即可以保证准确地解决top-k的流的检测问题。

## 其他方法

除了Space Saving以外，经典的基于Counter的方法还有Frequent，Lossy Counting，RAP等。Frequent当Counter满的时候是通过每个计数器都减1的操作，然后将新流替换计数值为0的计数器，相比Space Saving来说测量误差更大；RAP与Space Saving不同的在于当新的流的数据包到来时，并不是所有的包都替换最小的Counter，而是按照最小计数值的倒数的概率来替换最小Counter，这样可以使得大流的计数值增长得越来越快，小流的计数值增长得越来越慢，这样可以减少大流被小流替换的概率，这样使得RAP需要的Counter数比Space Saving更少。

FlowMap

针对Space Saving算法不能够很好地估计小流的流量大小这一缺点，我们提出了FlowMap的检测方法。

现阶段我的方法是基于flowmap的网络测量方法，目前正在进行仿真实验评估。Flowmap的主要步骤是：

如图8所示，首先利用space saving算法将大流保存下来，当space saving中有流被替换时，将该流的ID进行两次hash，第一次hash决定该流应保存到哪个的hash表中，然后第二次hash是利用每个hash表中相应的hash函数将该流插入hash表中。那么对于每个hash表都可以构建一个Y=AX的线性方程组，其中Y为hash表中累加的值，A是通过每条流的ID的hash操作构建的0-1矩阵，X为每条流的大小，于是我们通过求解这个方程组的最小范数解来估计流量矩阵。

(key,value)

Space saving

剔除

一级hash

h1

h2

h3

h4

二级hash

hash函数

图8 flowmap操作步骤

**接下来的工作**

接下来的工作主要还是在流量测量方面，后面希望设计的测量方法是在网络范围的测量方法，即在网络拓扑中，各个交换机之间相互协作进行网络测量，并且在NS3这类的SDN网络仿真平台上进行实验。