

SDN大作业

TCAM表项更新



2018-12-17

目录

[一、Fast Filter Updates for Packet Classification using TCAM 3](#_Toc532810392)

[1.文章概要 3](#_Toc532810393)

[2.TCAM表项重叠带来优先级关系 3](#_Toc532810394)

[3.使用未使用位存储优先级信息 4](#_Toc532810395)

[4.使用GMR位剪枝 4](#_Toc532810396)

[Partial order theory for fast tcam updates 5](#_Toc532810397)

[1.文章概要 5](#_Toc532810398)

[2.普适TCAM更新流程 5](#_Toc532810399)

[3.TCAM更新的时间复杂度度量 8](#_Toc532810400)

[4.TCAM表项分割 11](#_Toc532810401)

[Fast Lookup Is Not Enough: Towards Efficient and Scalable Flow Entry Updates for TCAM-based OpenFlow Switches 12](#_Toc532810402)

[1.文章概要 12](#_Toc532810403)

[2.TCAM规则更新框架 13](#_Toc532810404)

[3.TCAM更新流程 14](#_Toc532810405)

[4.BIT数据结构的应用 15](#_Toc532810406)

[5.M值的定义与计算 18](#_Toc532810407)

[6.TCAM中表项的不同位置分布 19](#_Toc532810408)

[7.算法效果 21](#_Toc532810409)

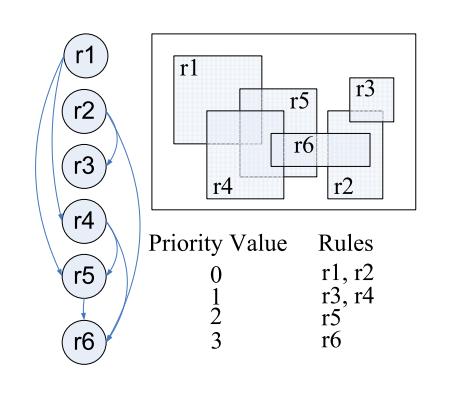
## 一、Fast Filter Updates for Packet Classification using TCAM

### 1.文章概要

本文的主要贡献是，提取出匹配项的优先级信息并附加到匹配项的物理位置上，协同TCAM的multi-match output信息进行查询和更新加速

本文针对的问题场景是，TCAM中由于需要维护匹配项之间的优先级关系，导致更新匹配项的速度慢，并且会给表项匹配带来较大延时的抖动

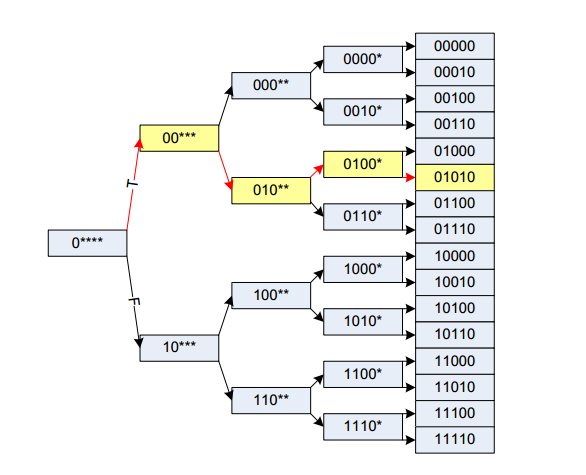
### 2.TCAM表项重叠带来优先级关系



TCAM中物理位置的先后关系代表了表项之间的优先级关系，即存储位置靠前的表项具有较高优先级。因此插入一个表项时，可能需要移动已有的表项来保证优先级关系的正确性，这会带来较大的时延，而且可能会导致转发错误。

### 3.使用未使用位存储优先级信息

TCAM可以提供144比特来与报文头部进行匹配，但是并不是全部144比特都用到了，还有一些位没有用到。匹配项的优先级关系只存在于有包含关系的匹配项之间，而且这些匹配项的有效长度一定不相同，在一般的匹配项集中，优先级不会超过64种，因此事实上优先级信息可以用较少的几个比特位来进行存储。



使用TCAM中没有用到的位来存储优先级信息，因此优先级信息不再与表项的物理存储位置相关，表项的删除、添加和更新操作都可以在任意物理位置进行，且不需要对已有的表项进行移动。结合GMR的特性，每次查询时，将可能的优先级范围进行二分，递归进行，可以在O(log n)时间内完成查询，相比于原本的线性查找O(n)有了较大提升。

### 4.使用GMR位剪枝

TCAM中有一系列可以重复配置的Global Mask Registers (GMR)，这些寄存器可以按照位信息忽略一下匹配项。multi-match output 信号指示是否有多个匹配表项，如果这个信号无效，那么表示只有一个匹配表项，那么在找到一个匹配项之后就不用继续查询了，因为不可能有更高优先级的匹配项。如果multi-match output信号无效，那么表示只有一个匹配表项，那么在找到一个匹配项之后，就不需要进行进一步的递归查询。

## Partial order theory for fast tcam updates

### 1.文章概要

本文的主要贡献是

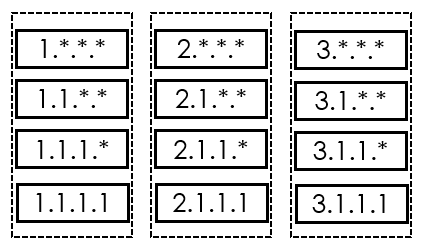
1.基于部分排序理论，提出了TCAM中增加表项的一般性过程，弄清楚了TCAM表项更新的理论下界，为今后的所有研究确立的目标；

2.提出了使用规则集分裂的方法来进一步加速TCAM更新

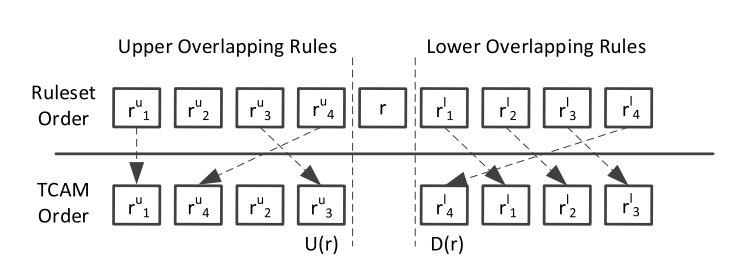
本文针对的问题场景是

TCAM更新算法的时间复杂度没有理论指导；TCAM表项更新时间代价大。

### 2.普适TCAM更新流程

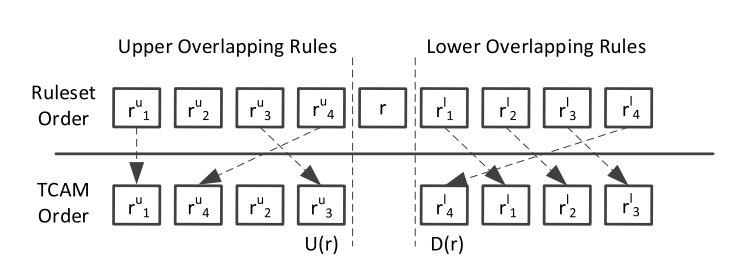


TCAM中的规则集可以分为若干组，组内的规则之间具有优先级关系，组间的规则没有优先级关系

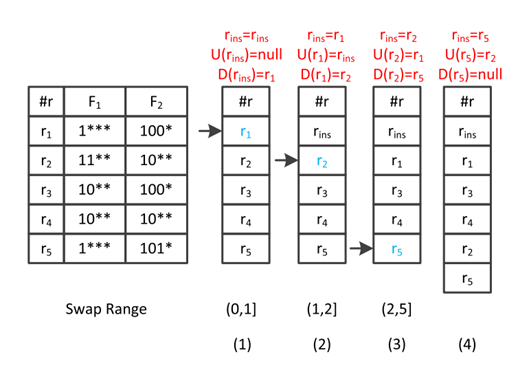


插入任何一个规则r时，都只能找到唯一一个可能与r具有优先级关系的组。

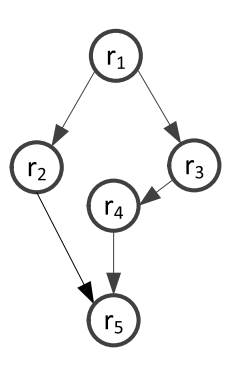
在这个组当中，按照优先级排序后，紧挨着r的两个规则分别是U(r)和D(R)



在找到U(r)和D(r)之后，这两个规则之间的所有物理位置，都可以用来存储r，将这个合适的物理位置范围成为交换区域。



如果交换区域中有空闲的位置，那么可以直接将r放在这个空闲位置上；否则需要用r替换这个区域之间的某一个规则r’，然后对r’执行插入程序。



如果交换范围中没有空闲位置，那么需要替换其中的一个规则r’，然后对r’进行插入，直到r’找到一个空闲位置。一般是不断往后找，也就是替换优先级较高的规则，最终一定会变成一个组中优先级最高的规则，然后直接放在TCAM的末尾。

### 3.TCAM更新的时间复杂度度量

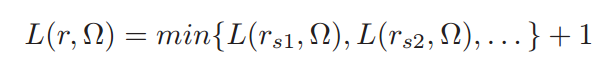
#### 1）插入规则的交换链

从以上分析可以看出，TCAM更新的主要操作，就是表项的替换，而替换的次数与表项插入循环的次数有关。

插入规则r，替换规则r’；插入r’时，可能还会继续替换并插入r’’，如此循环，最终插入过程会形成一个插入链，即r——r’——r’’——r’’’……交换链的长度就是TCAM表项替换操作的次数。也就是说，要减少TCAM的更新时延，就是要尽量使得交换链的长度减小。

因此，在交换范围中选择r’的原则，就是尽量减少替换的次数，也就是交换链的长度。如图r1——r2——r5是 插入r1的最短交换链。要选出交换链长度最短的r’，就要首先计算出每一个规则的交换链长度，然后遍历选出最短的。

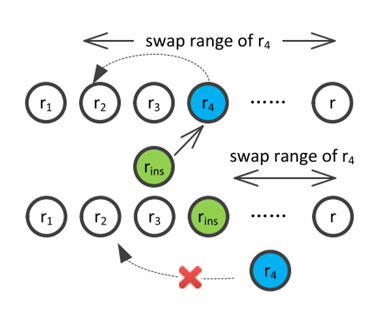
#### 2）交换链长度的计算

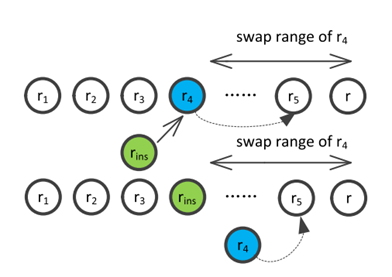


计算方法每一个规则的交换链长度，可以使用动态规划算法：

先给最末尾的（所有优先级最高的）规则赋予cost 1；其他规则，找它们交换范围中代价最小的规则+1；

#### 3）最优替换规则的选择





在交换范围中找具有最短交换链长度的r’，一共有三种策略，分别是full，bh，down：

Full策略：在整个交换范围中找合适的r’

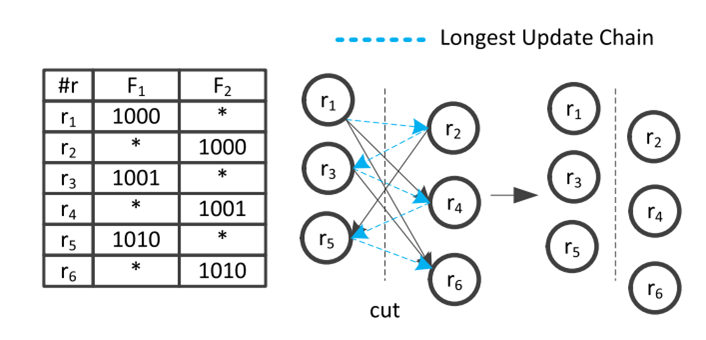
Bh策略：将交换范围分成两半，在优先级较高的一半中找r’

Down策略：直接使用D(r)，也就是交换范围中优先级最高的规则

Full策略实际上是不可行，因为会使得原有的规则顺序不稳定，规则的替换会改变现有规则的交换链长度，也就无法预先计算所有节点的交换链长度。Bh策略比目前（2018年）认为的最优算法还要高效，但是计算代价大，也无法实际运用。目前（2018年）认为最优的TreeCAM算法，就是down策略的一种实现。

### 4.TCAM表项分割

#### 1）表项分割思想

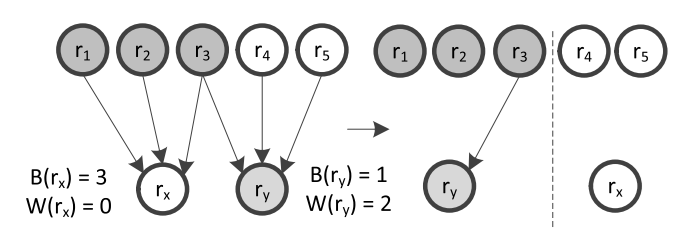


TCAM更新代价高的原因之一，是优先级约束太多，减少优先级约束的一个方法，是划分规则集。

划分规则集的主要思想是，将一个完整的规则集，划分成若干份，分别存储在不同的TCAM上，然后并行的进行查找，最后在所有查找结果中选择优先级最高的。这样可以加快表项查找时间，并且可以减少每一个TCAM中的更新时延，。

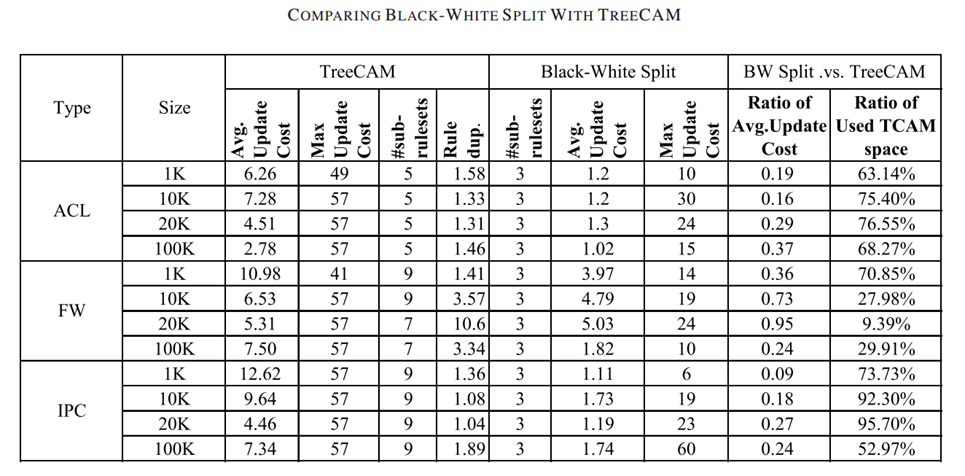
如图，如果将r1，r3，r5分为一组，r2，r4，r6分为一组，那么组内的所有规则之间就完全没有优先级关系，可以在O(1)时间内完成插入。

#### 2）表项划分算法：BW split



将规则集进行二分时，可以使用黑白划分方法：

制作一个Hasse 图，在这个图中，节点代表规则，边从低优先级节点指向紧邻的高优先级节点。首先给根节点涂黑色；对于其他节点，统计其所有父节点（指向它的边的源节点）颜色，它的颜色与大多数父亲节点（多数投票）的颜色相反。



从实际验证来看，BW split在平均更新代价和TCAM空间使用上，都比TreeCAM要好。

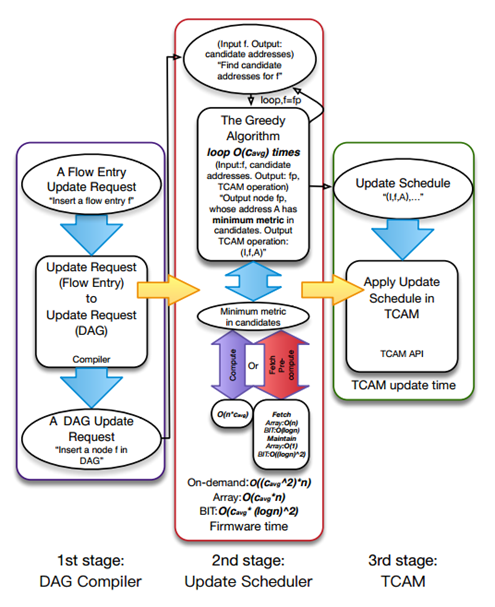
## Fast Lookup Is Not Enough: Towards Efficient and Scalable Flow Entry Updates for TCAM-based OpenFlow Switches

### 1.文章概要

本文提出了一种使用贪心算法，BIT数据结构，有向无环图的，适用于各种部署情景的更新算法。

本文的问题场景是，当前TCAM更新算法时间复杂度高。

### 2.TCAM规则更新框架



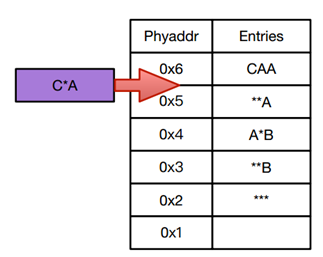
目前的TCAM更新算法的框架大体如下：

1.将更新命令转化为有向无环图中的插入操作，完成这一部分功能的模块被称为DAG编译器。

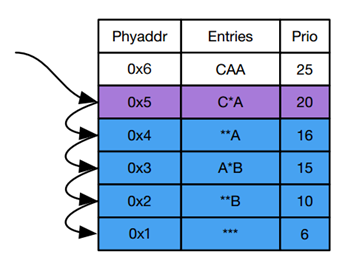
2.将有向无环图中的插入操作翻译为TCAM交换命令，完成这一部分功能的模块被称为更新调度器。

3.在TCAM中执行2中的命令序列，命令序列决定了更新时延。

### 3.TCAM更新流程



TCAM中的部分表项有优先级顺序，并且表现为物理位置上的先后关系，在执行插入f的操作时，必须要维护这种优先级关系。



具体方法是：

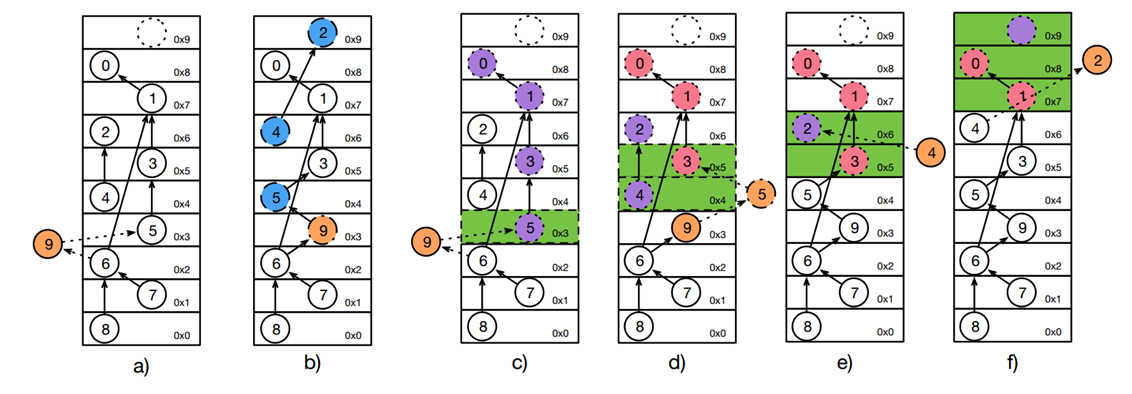
1.找到fa和fb，这是在将所有相关的规则排序之后，f两侧紧邻的规则

2.在fa和fb对应的物理位置之间的区域里，选择一个合适规则f’，和f替换

3.循环插入f’

### 4.BIT数据结构的应用

为了减少交换链的长度，在每一次选择替换规则的时候，我们都要选择交换链最短的规则。

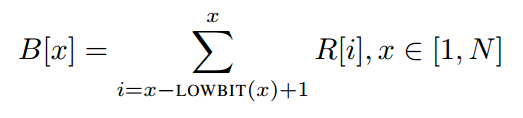


绿色区域是交换区域，蓝色节点是交换链上的节点

在交换区域中选择节点时，使用了贪心算法，即选择交换链长度（M）最短的节点



在交换区域内选择合适的规则的时候，可以使用BIT数据结构减少这一过程的时间负责度。

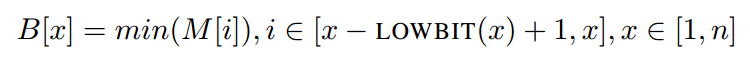


选择交换区域中M值最小的节点，如果使用朴素遍历算法，是一个耗时为O(N)的过程，如果采用BIT数据结构可以降低为O(log N)。

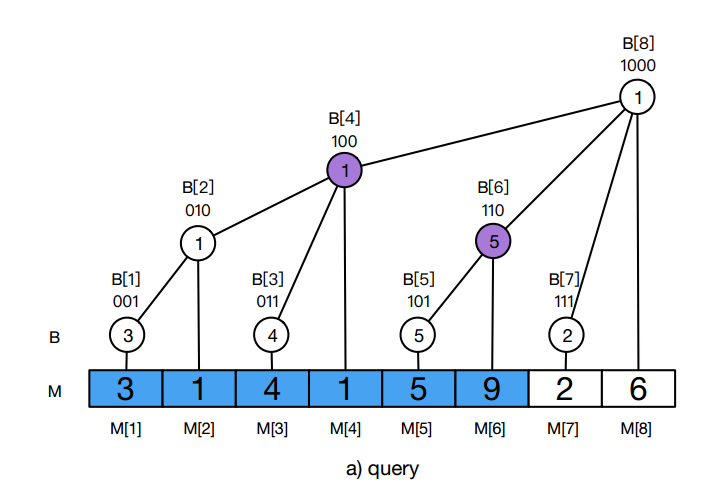
BIT数据结构可以用来在O(log N)时间内计算下标某一个范围[1,b]内所有元素的和。思想是将每一个下标按照二进制特点进行划分，使得每一个元素存储的不是自身的值，而是某一个下标范文内所有元素的和。具体来讲，B[x]存储索引在[x，(x-LOWBIT(x) + 1)]范围内的值的和。

举例：

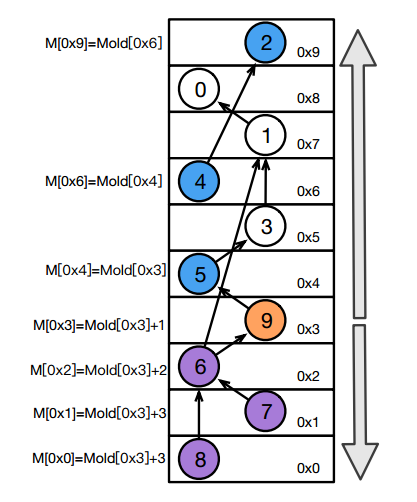
所以：



在算法的实际应用中，B[x]不再存储范围内元素的和，而是存储最小值。如下图



### 5.M值的定义与计算

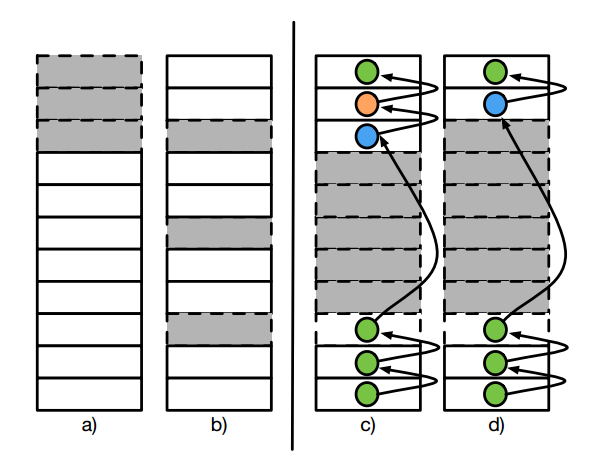


在fa和fb之间选择合适的f’时，用到了度量值M，M(A)是插入A所需的循环次数。

1.根据优先级关系制作有向无环图，有向边代表优先级关系；

2.首次计算M值时，使用DFS算法，从每一个节点出发，搜索所有邻居；

3.在之后的所有操作中，只对M值进行更新，而不进行重新计算。更新的范围是交换链上的所有节点，以及直接或者间接与交换链上节点有优先级关系的节点



### 6.TCAM中表项的不同位置分布

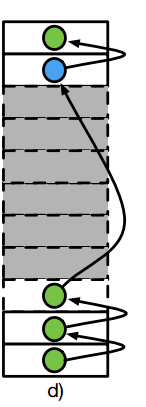
TCAM中表项的物理位置分布大致有三种：

1.全部在顶部或者全部在底部；

2.每隔一段距离留若干空位置；

3.分布在顶部和底部，且数量均衡；

1是最普通的情形，算法直接可以应用；2情形很难一直保持下去，也就是说，经过若干次插入操作，中间的空位置可能被填满了；3中的情形可以降低插入复杂度，但是增加表项删除的复杂度。



第三种物理位置分布，即表项均衡的分布在顶部和底部，增添和删除表项的操作如下：

增添表项：

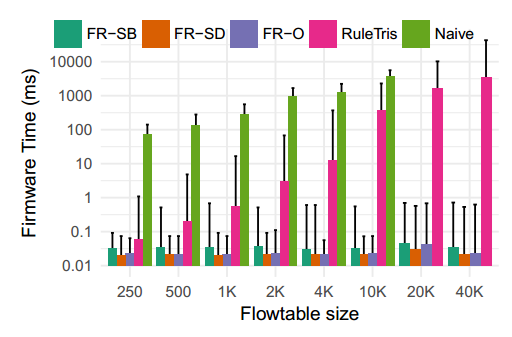
如果f两边（优先级顺序）的两个规则fa，fb，都在顶部，或者都在底部，那么就按照之前的方法插入顶部或者底部，如果fa，fb分别在顶部和底部，那么f就直接插入中间的空位置，如果顶部的规则较多，就插入靠下的位置，反之即反

删除表项：

脏删除：删除之后不进行任何处理

平衡删除：删除之后用其他表项填补空缺，维持所有表项都存储在顶部和底部的性

### 7.算法效果



FR-SB：两侧分布 + FastRule + 平衡删除策略

FR-SD：两侧分布 + FastRule + 脏删除

FR-O：一般分布

RuleTris：当前广泛使用的算法

Naïve：朴素算法