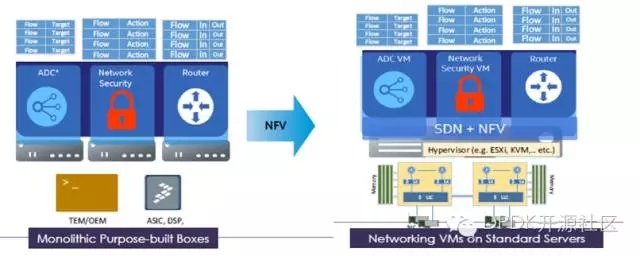
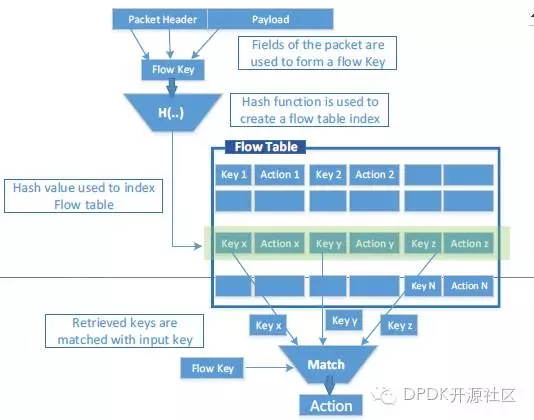
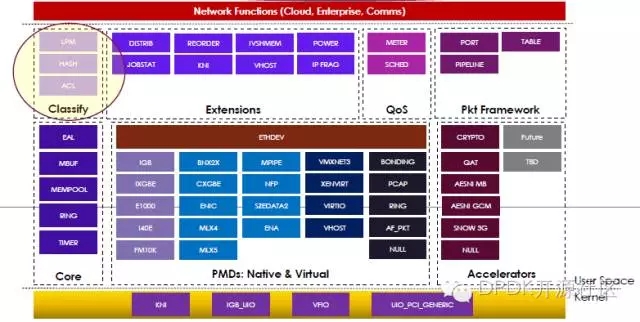
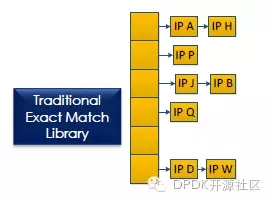
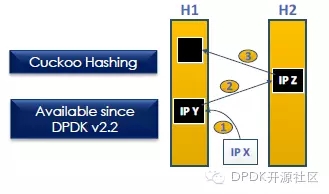
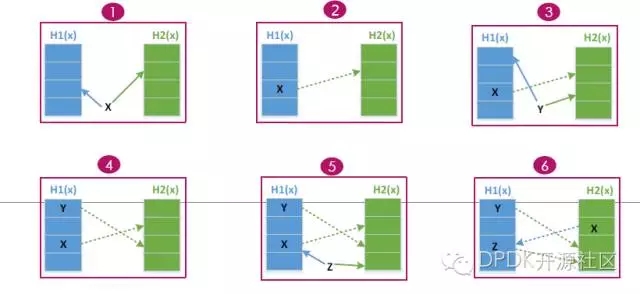
<https://www.sdnlab.com/community/article/910>

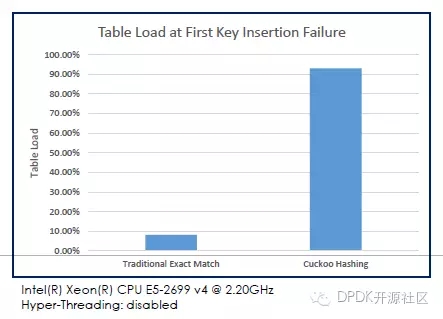
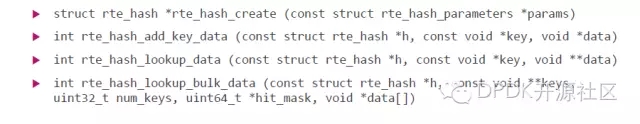
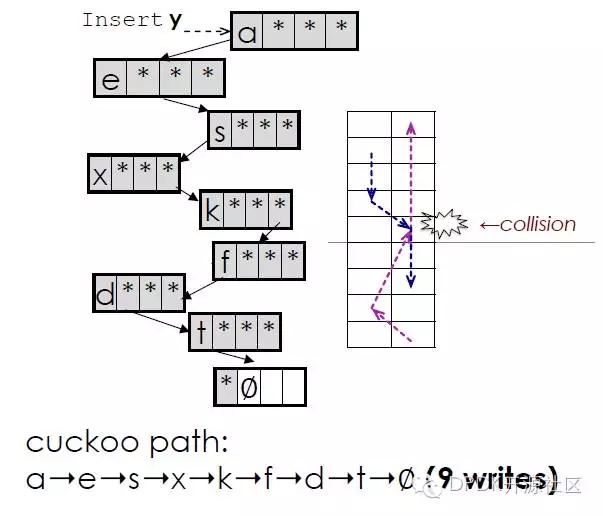
**流分类**  
流分类指的是网卡根据数据包特性将其分类的技术，是现今网卡通用的技术之一。也是网络中常用的技术，比如要实现功能卸载，需要知道要将功能卸载到哪个执行单元，在这之前需要对流进行分类。

## 传统网络设备和专用服务器硬件流分类对比

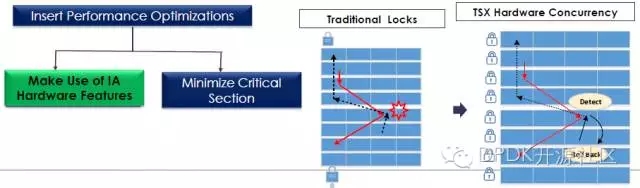
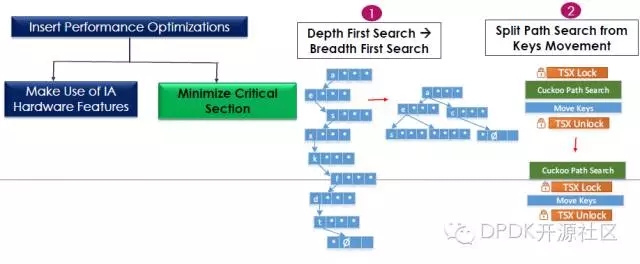
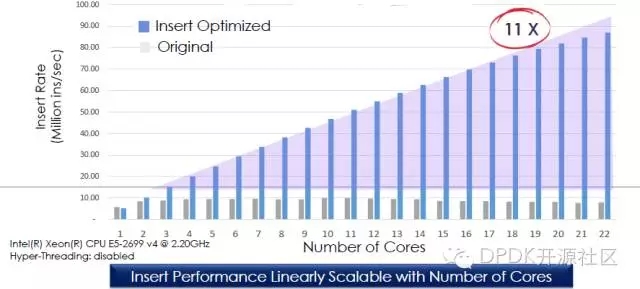
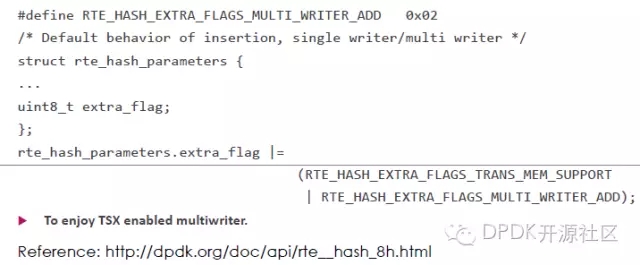
  
  
**流表运作流程**  
一般而言，流表由很多flow key 和相应的 action 构成。flow key 可能是报头中的某些字段，也可能包括有效载荷的部分内容 。对 flow key 进行 hash运算，所得到的hash值就是流表的索引。由于哈希函数不是那么完美，有时会出现冲突，这时就需要再选几个key区分action。  
度量一个流表的设计主要有三个重要的指标：

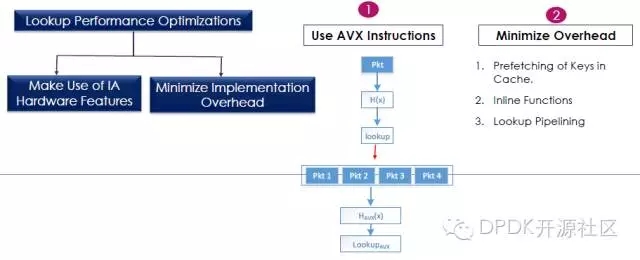
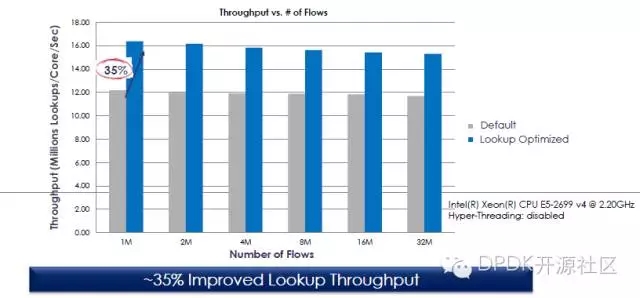
* 更高的查找率： 意味着更高的吞吐量和更低的延迟
* 更高的插入率：意味着更快的流更新速度和表初始化速度
* 高效的利用率：意味着可以添加更多的流

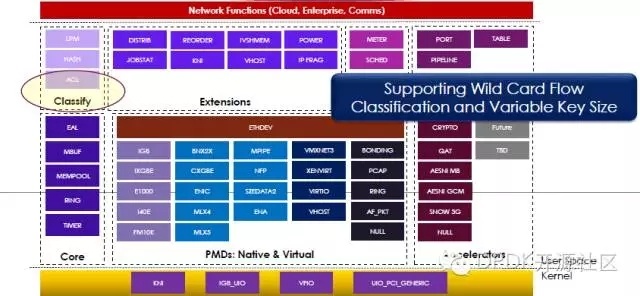
  
在DPDK中有很多库来实现流分类，我们将讨论的是精确配对。ACL，Hash以及LPM都能帮助我们进行流分类。  
  
**RTE-Hash精确配对库**  
下图是DPDK中传统的精确匹配库所用的数据结构。传统的精确匹配库基于一个”sparse”hash表实现，是对精确匹配算法的一个简单直接的实现，随着流表的增大，性能会显著降低。  
  
  
自从DPDK2.2发布以来，Cuckoo Hashing得以运用，从而能在cache中装下denser的流表，流表可扩展到百万级别的entry数目。  
  
  
**Cuckoo Hashing**  
cuckoo是一种鸟名，它们不会建立自己的鸟巢，靠侵占别的鸟巢为生，可以说是鸟界的“黑客”，但这也是Cuckoo Hashing所做的事。  
我们选取了两个HASH函数，总共6个步骤。我们从空的表格开始，我们打算插入关键字X，我们有两个函数可供使用，H1和H2，指向不同的索引，这两个都是空的，所以进入第一个位置，所以H2的位置是空的。对关键字Y来说，这两个位置也是空的，所以Y进入了最初的位置（primary location）。在第5步中，cuckoo 函数下，Z出现了，Z也指向了两个位置，其中之一是X所在的位置，Z会将X挤出来，进入X所在的位置，然后X会被存储到它所指的第二个位置。所以，为了占据空间，关键字会将别的关键字挤出。  
  
  
  
**运行Cuckoo Hashing的优点：**

* Cuckoo Hashing可以将更多的流放在流表中。
* RTE-hash 可以支持构建有着成百万级别关键字的流表，且该流表可完全放入CPU cache中。   
  ## RTE-hash API代码片断 ##  
    
  ## Cuckoo长路径和多个并发生产者 ##  
  以下图片显示的是Cuckoo的长路径。  
  流表几乎被填满时，插入关键字可能会移动很多项目。当多个并发产生者同时插入Cuckoo路径时，冲突就会产生。  
  

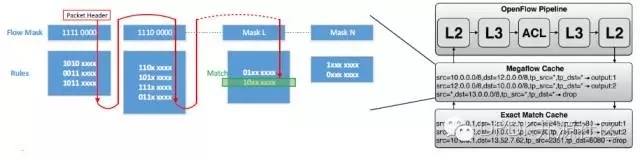
## 流表插入性能优化

插入性能优化主要基于两个方面：利用号IA 硬件功能以及临界区最小化。TSX运行方式非常简单，硬件负责监视并发情况。  
我们可以看到两幅图的临界区，在第一幅图中，当并发发生时，进口和出口会被锁住，只有1条thread可以通过，其他则无法通过。而在支持TSX的硬件中，CPU会监视每条cache line，监视谁在访问这条cache line，如果在同一条cache line 中有两个thread，该动作会被撤回。  
  
  
  
在下图中可以看到另一方面性能的提高。  
  
  
有了TSX，性能能大大提升。  
蓝色代表的是开启流表插入性能优化后的数据。  
  
  
  
使用的方式非常简单，设立相应的flag，即可使用，大家可以试一下。  
  
  
  
**流表查找优化**

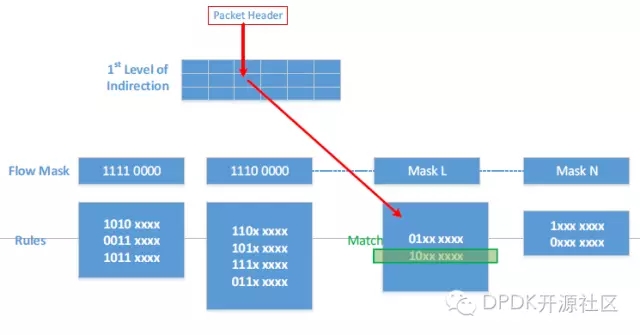
* 使用AVX 指令
* 最小化实现开销   
  在下表中可以看到优化流表查找后的数据。  
  

我们也经常会讨论如何支持通配规则的流分类以及可变的键长，虽然该内容和DPDK没有直接的联系。  


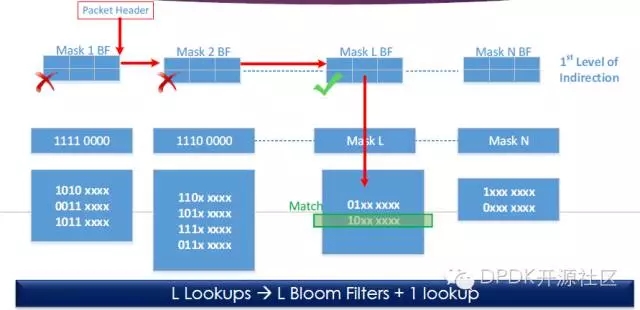
## POC: Open vSwitch 流查找

为了支持通配规则，在OvS中有多个无交集的子表，每条规则仅插入到某个子表内。查表过程是以顺序查找各个子表的方式进行的，查表过程会在遇到一个精确匹配时结束。  


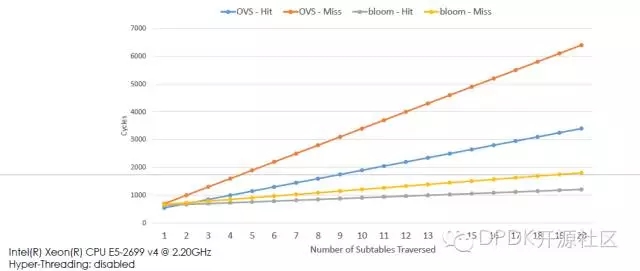
## OvS的二层查找



## Bloom Filter



## 二层查找性能显示图



[0](javascript:;)

分享 2017-09-04

## 0 个评论

s