SDN 流表查询中的cache加速技术调查报告

小组成员及分工

|  |  |  |
| --- | --- | --- |
| 姓名 | 学号 |  |
| 张豪 | 2017213815 |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 一、SDN流表查询技术背景介绍

基于网络协议分层的思想，在传统的TCP/IP架构网络中，网络转发结点对数据包进行逐层协议处理，各个协议层次在逻辑上相对独立，但同时各层协议的处理方法也变得大相径庭。无论是在软件上还是在硬件上，各层协议的处理逻辑实现都各相迥异。随着网络协议的不断增多，网络协议的层次结构变得越来越复杂，开发新的网络协议、维护各层网络协议之间接口的兼容性等工作都变得越来越困难。因此，寻找一个逻辑上分离，但形式上统一的协议处理架构具有非凡意义。

在软件定义网络（Software Defined Network，以下简称SDN）中，掌控全局信息的控制器将根据需求制定的流表配置到各个交换机中。在收到一个数据帧后，交换机会查询其匹配的流表项，通过执行该流表项后指定的动作来完成对数据包的处理。一条流表项中包含各层协议头信息，在处理数据帧时，将其与数据帧的各层协议头进行比对从而确定此数据包是否属于该流表项对应的数据流。但在实际情况下，网络中的数据流数量是庞大的（n台主机间只考虑IP域时就能产生 (n(n-1))/2 条数据流），为了节省存储空间，流表项实际上是按照协议域拆分存储的（根据关系数据库的思想，按照协议域拆分存储在减少冗余信息的同时也便于管理）。在流表被拆分为多级流表后，查询一个数据帧所属的流需要进行多次的查询操作，这无疑降低了网络的数据帧处理性能。为解决这一问题，目前使用的最为广泛的方法就是利用cache机制进行加速。Cache中存储着完整的各层网络协议头信息，这样交换机在收到数据帧后只需要通过一次查询便可以决定该数据帧所属的流和需要执行的操作。由于cache的存储空间十分有限，不可能同时存储所有的数据流信息。因此如何维持Cache的高命中率，保证cache与多级流表的一致性，以及防止cache抖动等问题便成为了设计cache替换策略时需要综合考虑的问题。

# 二、cache替换策略

1. 性能（命中率）

2. 可靠性（缓存的一致性）

1）记录规则对应的流表项

控制器在向交换机配置规则时，为每条规则制定一个唯一的标识符。在交换机中维护着一张规则和流表项的对应表，记录着哪些流表项中包含着该规则。当网络换发生改变，控制器根据新的网络状况更新了该规则对应的动作时，此交换机可以根据这张表格查找到cache中的哪些流表项包含了此项规则，然后将这些流表项全部删除，从而保证了cache与多级流表的一致性。

2）记录流表项对应的规则

维护一张规则的状态表，当网络环境发生改变时，控制器在该状态表中给失效的规则打上无效标记。此外，在生成cache流表项时，记录下该流表项都包含了哪些规则。对于每个到达的数据帧，交换机在命中cache后都将遍历此流表项对应的所有规则的状态，如果发现只要有一项规则有无效标记，便删除此流表项，将数据帧递交给多级流表查询进程。

3）记录规则对应的最终动作集

在生成最终动作集时，计算每个最终动作集的哈希值作为该动作集的唯一标识，并记录下每项规则都对应了哪些最终动作集，当网络环境发生改变时，控制器根据该表格给失效规则的相关动作集打上无效标签。为了处理最终动作集哈希值冲突的问题，可以在动作集中维护一个时间属性，记录下该动作集的最近更新时间，在处理数据帧时，如果其匹配的流表项对应的动作集的更新时间晚于此cache流表项的生成时间，则可认为此cache流表项已经失效。

3. 稳定性（时延抖动）

# 三、对生命周期短、负载量少的流的处理方法

# 四、cache的更新问题