面向全可编程网络数据平面的资源优化方法

申 涓,段 通,兰巨龙
(国家数字交换系统工程技术研究中心,河南郑州 450002)

摘 要: 现有数据平面无法在整体上支持网络功能的创新和演进,因此面向用户可编程的新型网络数据平面技术发展迅速.针对现有可编程数据平面硬件开销过大且缺乏资源优化的问题,从解析器、匹配表、动作执行器三个方面建立了资源开销模型,并分别提出了类型域合并、匹配域偏移量合并、"域-字"拆分合并映射等资源优化方法.基于 NetFPGA 的仿真结果表明,与现有机制相比,所提方法减小了 38% 左右的资源开销.

关键词: 网络数据平面;可编程;硬件结构;资源优化
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2018)10-2423-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.10.016

The Resource Optimization Towards Fully-Programmable Network Dataplane

SHEN Juan, DUAN Tong, LAN Ju-long
(*National Digital Switching System Engineering & Technological Research Center, Zhengzhou, Henan* 450002, *China*)

Abstract: The existing network data plane could not support the evolution and innovation of network functions, thus the new data plane technology for user-programmability is developing rapidly. For the huge amount of hardware resource costs and the lack of responsible resource optimization, a resource cost model combining parser, match table, and action processor is built, and the corresponding resource optimization methods like type field combination, match field offset combination and "field-to-word" mapping are proposed. The NetFPGA-based experimental results show that the proposed mechanism can reduce hardware resource cost by 38% compared to existing mechanisms.

Key words: network dataplane; programmable; hardware structure; resource optimization

1 引言

随着未来网络研究的开展,新型网络技术如网络虚拟化[1]、网络功能虚拟化(Network Function Virtualization, NFV)[2]等,新型网络体系如软件定义网络(Software-Defined Networking, SDN)[3]、命名数据网络(Named Data Networking, NDN)[4]等随之涌现.为支持这些新型技术的实现,基于可编程技术的数据平面设计成为业界关注的重点.然而,现有可编程数据平面的研究均着重研究如何实现更灵活的、甚至达到逐比特编程的可编程能力,却均未考虑可编程粒度过细导致的硬件资源开销急速膨胀问题.在资源有限的硬件中,如何在保证可编程能力的基础上尽可能减小硬件资源开销,对未来可编程网络设备的开发和实现具有重要的实用价值.

在数据平面的可编程结构方面,大多数研究如 OpenFlow[5]、Forwarding Metamorphosis[6]、P4[7]、POF[8](Protocol Oblivious Forwarding)、dRMT[9]等,均从不同方面提出了可编程数据处理流水线结构,但基本思想都是对解析器、匹配单元、处理器这三个主要模块进行编程.其中,包头解析模块负责将数据包中的匹配域提取出来,以便后续匹配模块进行查找,如果包头解析模块可以支持用户自定义的解析流程,则可以支持新型网络协议包头的添加. Kangroo[10]、CAFE[11]等包头解析器均采用自定义偏移的方式进行灵活包头解析,用户可通过配置包头解析器内的存储单元实现对解析状态机的控制.这种解析方式可以精确到字节级甚至比特级的提取,但这种任意比特域抽取的方式在灵活度高的

收稿日期:2017-02-21;修回日期:2018-02-27;责任编辑:李勇锋
基金项目:国家网络空间安全专项课题(No. 2017YFB0803204);国家 863 高技术研究发展计划(No. 2015AA016102);国家自然科学基金群体创新项目(No. 61521003)

情况下也带来了资源开销过大等问题. 在匹配和处理模块的灵活性提升方面,OpenFlow 针对传统 TCP/IP 协议包进行灵活匹配和处理,通过对数据包中十元组的匹配来识别各种类型的数据流. OpenFlow 的局限是缺乏较强的动作处理可编程能力,即无法支持除 TCP/IP 协议以外的新型网络协议数据包的处理. 在 POF、P4 和 ClickNP[12] 中,用户可以定义每个匹配表所需匹配的匹配域,并且配置精确到比特级的动作指令,可实现对新型网络协议包头的修改、添加/删除等操作. 这种方案的灵活度极高,也带来了硬件层面较高的硬件资源开销.

在资源优化方面,现有针对可编程数据平面结构的资源优化方法大都集中在用户配置层面,即:如何通过最优的用户配置,减少对硬件可编程单元的使用. 如在配置解析器时,通过压缩匹配域偏移量来减小解析器内部可编程单元的存储开销[13,14];在配置匹配表时,通过多匹配表的组合[15,16]和单匹配表内匹配表项的组合[17]来减小表项的存储开销;在配置动作执行器时,通过元动作单元之间的组合来[6]减小动作单元的使用. 以上研究主要存在两方面的问题:一是大都集中在用户配置层面如何更好地利用可编程单元,而未考虑其硬件结构本身的资源开销优化;二是仅考虑单个模块的资源开销优化,而数据处理流水线的开销是由各个模块的可编程性协同作用的结果.

基于以上分析,如何从整体上对可编程数据平面的硬件开销进行优化,是亟待完善的问题. 为不失一般性,本文选取业界通用的全可编程数据平面结构,从解析器、匹配表、动作执行器三个方面分析了影响其资源开销的因素,建立了资源开销模型;其次,基于该模型分别提出了解析、匹配、处理模块的资源优化方法;最后,通过在 NetFPGA-10G 平台上的资源仿真,验证了所提方法的优化效果,与现有方案相比,降低了近 38% 的硬件资源开销. 所提资源优化方法对可编程数据平面的硬件实现具有一定实用价值.

2 面向全可编程结构的资源开销模型

为不失一般性,本文以目前可编程能力最强的交换机硬件结构[6]为基础,分析可编程数据平面的资源开销并提出优化方法. 该结构因其全可编程的、灵活度达比特级的多级流表流水线设计,并引起了学术界和产业界的巨大反响.

该结构如图 1 所示,主要由包头解析器和多级流表组成. 其中,包头解析器用于识别数据包的协议类型,同时根据数据包的协议类型得到相应所需的匹配域并将其组合成包头域,向多级流表输出;各级流表是最基本的数据包处理单元,用来实现“匹配 + 查找 + 动作”的操作. 流表内的匹配域选择器和动作执行器支持用户的动态配置,可以实现流表匹配域的选取以及协议无关的动作处理. 流表之间通过元数据进行信息传递,实现匹配表之间的组合.

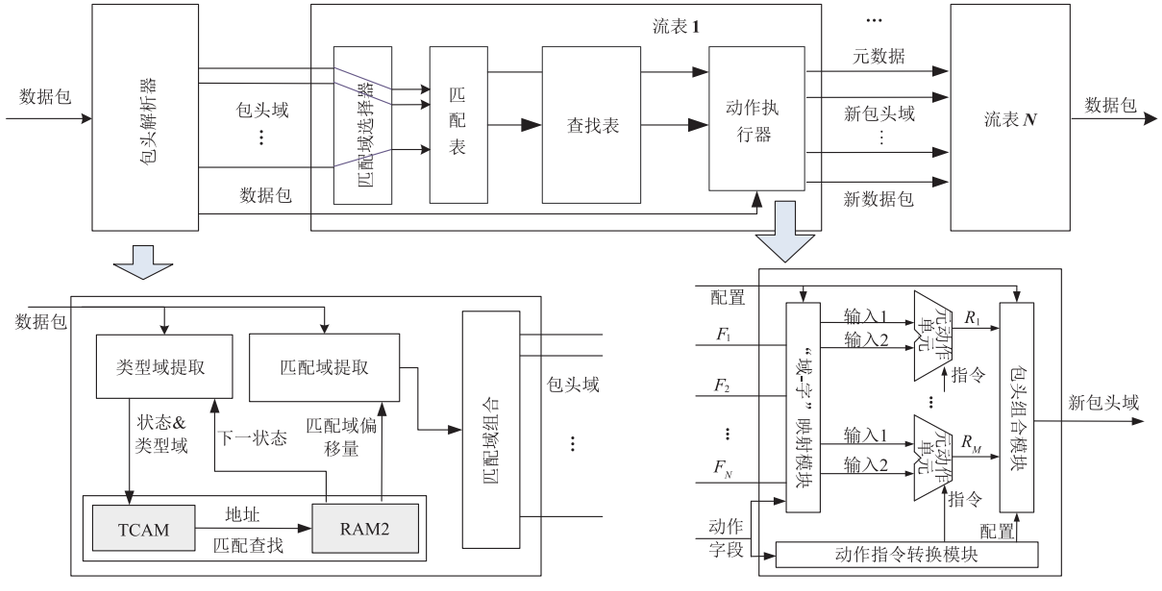


图1 全可编程的交换机硬件结构

本章首先给出本文所用相关概念的定义;其次,分别对包头解析器、匹配单元、动作执行器三个模块的资源开销进行建模.

定义 1 提取. 是指从一个比特向量中摘取出相应长度比特向量的操作,一般为连续提取方式,即摘取连续的比特组成相应长度的比特向量. 提取过程需要三

个参数,一是原始比特向量的长度;二是开始提取比特的位置,也称偏移量;三是所要提取比特向量的长度.

定义 2　类型域. 是指数据包包头中标识下一协议包头类型的比特向量. 数据包包头由多层协议包头叠加而成,如“MAC + VLAN + IPv4 + TCP”的协议包头组合,各层协议包头内包含标识下一协议包头类型的比特向量,如 MAC 协议包头内的 protocol 字段,用于标识下一协议包头是 VLAN、IPv4 还是 MPLS 等.

定义 3　匹配域. 是指数据包包头内所需要提取并用于后续流表匹配的比特向量. 如 OpenFlow1. 0 规定的包括 MAC 源地址、MAC 目的地址、IP 源地址、IP 目的地址等在内的 12 元组.

2.1　包头解析器资源开销

包头解析的流程可用状态转移图表示,如图 2 左图所示,状态个数即所有可能的协议包头个数;状态转移图的边,表示所有可能的包头叠加关系. 设数据包包头长度为 L_P,可能的状态转移边个数为 K_M,单个协议包头内所要匹配域个数最大为 N_M. 类型域识别模块内的数据包缓冲区宽度为 W_T,匹配域提取模块内的数据包缓冲区宽度为 W_M,则包头解析器各部分的资源开销可表示为:

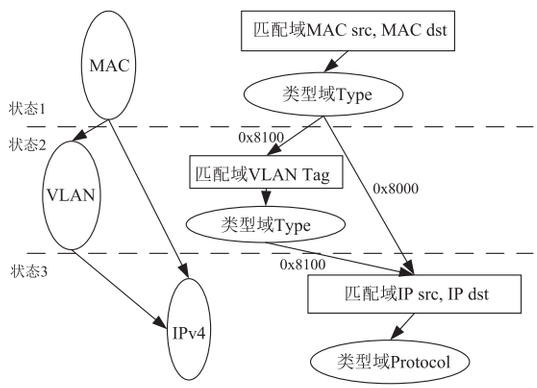


图2　包头解析流程示例

(1)类型域提取模块开销主要由包头缓冲区开销决定,可表示为:$C_{\text{TypeF}} \approx \partial_t \times L_P$,其中,$\partial_t$ 为线性系数;

(2)匹配查找模块的开销主要由 TCAM 开销和 RAM 开销决定,可表示为:$C_{\text{MatchF}} = f_{\text{TCAM}}(W_T, K_M) + f_{RAM}\{N_M \times 2\log_2(W_M), K_M\}$,其中,$f_{TCAM}(width, depth)$ 和 $f_{RAM}(width, depth)$ 分别是 TCAM 和 RAM 资源开销随存储宽度和深度变化的函数;

(3)匹配域提取模块的开销主要由提取模块和数据包缓冲区大小决定,可表示为:$C_{\text{Extraction}} \approx \partial_e \times L_P + \beta_e \times N_M \times W_M$,其中,$\partial_e$,$\beta_e$ 为线性系数;

(4)匹配域组合模块仅将各个匹配域的输出放置到同一个输出向量中,因此不占用硬件逻辑资源.

因此,包头解析器的总资源开销可表示为:

$$C_P = C_{\text{TypeF}} + C_{\text{MatchF}} + C_{\text{Extraction}} \tag{1}$$

2.2　匹配单元资源开销

解析器生成的包头域要经过各级流表,用于实现“匹配 + 查找 + 动作”的处理抽象. 匹配单元包括“匹配 + 查找”,其中,匹配表多用 TCAM 或哈希表实现,查找表多用 RAM/SRAM 实现. 在全可编程结构中,匹配单元所要匹配和处理的匹配域是可以用户配置的,通过匹配域选择器来实现用户定制.

设解析器生成的包头域长度为 L_H,匹配表宽度为 W_{MT},查找表宽度为 W_{ST},一共 N_T 级流表,则各部分的资源开销可表示为:

(1)匹配域选择器开销主要由包头域长度和匹配表宽度决定,可表示为:$C_{\text{Selector}} \approx f_S(L_H, W_{MT})$,其中,$f_S(A,B)$ 是从 A 长度比特中提取出 B 长度比特的资源开销;

(2)匹配查找模块的开销主要由匹配表和查找表开销决定,可表示为:$C_{\text{MatchT}} = f_{\text{Match}}(W_{MT}) + f_{\text{Search}}\{W_{ST}\}$,其中,$f_{\text{Match}}(width)$ 和 $f_{\text{Search}}(width)$ 分别是在存储深度一定时匹配表和查找表资源开销随存储宽度变化的函数;

因此,匹配单元的总资源开销可表示为:

$$C_{\text{FlowT}} = N_T \times (C_{\text{Selector}} + C_{\text{MatchT}}) \tag{2}$$

2.3　动作执行器资源开销

动作执行器主要由“域-字”映射模块、动作指令转换模块、包头组合模块以及多个并行的元动作单元组成. 其中,元动作单元的输入长度是固定的,一般分为 8 比特,16 比特和 32 比特三种. 包头域中 8 比特,16 比特,32 比特的长度叫做“字”(word),字作为各个元动作单元的输入,是从包头域中的匹配域(field)拆分或合并而成的. “域-字”映射模块根据用户的配置将包头域中相应的匹配域以及动作字段内容拆分或合并成字,输入至元动作单元. 动作指令转换模块将表项匹配得到的动作指令转化为各个元动作单元的输入,使得各个元动作单元同时执行处理操作. 包头组合模块将处理后的“字”再反映射至“域”,得到新包头域并输出.

动作执行器的开销可分为三部分,即两个映射模块的开销(包括“域-字”映射模块和包头组合模块),元动作单元的开销,以及指令转换模块的开销,可形式化表述为:

$$C_{\text{ActionP}} = 2 \times C_{\text{Mapping}} + C_{\text{units}} + C_{\text{others}} \tag{3}$$

其中,C_{Mapping} 代表单个映射模块的开销,C_{units} 表示元动作单元的开销,C_{others} 表示包含指令转换模块在内的其他开销.

(1)映射模块开销. 由于控制映射模块的配置信息计算被移动至软件层面,应此“域-字”映射和“字-域”映射仅需要一个 crossbar,即可根据配置信息将匹配域

映射至目的动作单元. 因此,映射模块的开销即可转化为 crossbar 的开销. 由于 crossbar 可以由多个双输入的 demux 实现,而每一个 demux 连接一个动作单元的输入,因此 crossbar 的开销由动作单元的数量决定:

$$C_{\text{Mapping}} = (N_{\text{field8}} + N_{\text{field16}} + N_{\text{field32}}) (N_{\text{unit8}} + N_{\text{unit16}} + N_{\text{unit32}}) \times C_{\text{demux2}} \tag{4}$$

其中,N_{unit8},N_{unit16} 和 N_{unit32} 分别表示 8 比特,16 比特,32 比特输入的元动作单元数量;N_{field8},N_{field16} 和 N_{field32} 分别表示 8 比特,16 比特,32 比特长度的匹配域数量.

(2)元动作单元开销. 元动作单元采用协议无关动作指令集,包括逻辑运算、数学运算、域的复制/移动/删除等指令. 这些指令的操作复杂度与输入长度成线性关系,并不涉及复杂度较高的运算(如乘法运算等). 由于元动作单元的指令集是线性操作,因此其资源开销与输入长度几乎成线性关系. 定义 C_m 为每比特操作的资源开销,则元动作单元的总开销为:

$$C_{\text{units}} = N_{\text{unit8}} \times C_{\text{unit8}} + N_{\text{unit8}} \times C_{\text{unit8}} + N_{\text{unit8}} \times C_{\text{unit8}} \tag{5}$$

$$C_{\text{unit8}} = 8 \times C_m, C_{\text{unit16}} = 16 \times C_m, C_{\text{unit32}} = 32 \times C_m \tag{6}$$

(3)其他开销. 上面两部分开销已经涵盖了动作执行器的大部分资源开销,剩下的开销主要由指令转换模块占用. 指令转换模块使用动作字段内的指针将动作字段内的动作指令发送至相应的元动作单元,其他不需进行处理的动作单元则不进行功能处理,因此其资源开销与元动作单元的数量成正比. 附加开销则消耗在一些数据的存储上,与包头域长度成线性关系. 定义 α 和 β 分别为线性系数,则其他开销可表示为:

$$C_{\text{others}} \approx \alpha_H \times L_H + \beta_H \times (N_{\text{unit8}} + N_{\text{unit16}} + N_{\text{unit32}}) \times L_{\text{instruction}} \tag{7}$$

3 资源开销优化方法

在数据处理流水线上,包头解析器和动作执行器的设计加入了大量支持用户配置的存储单元,这部分开销是相比于传统数据平面增加的额外开销,因此,对该部分的开销优化是解析器和动作单元优化的重点;而对于匹配表,由于其采用固定的 TCAM 或 SRAM,无法进行优化,但由于匹配表的宽度和流表级数会影响匹配单元的开销,因此对合适宽度级数的选取是匹配单元优化的重点. 本节重点分别讨论了解析器、匹配单元和动作执行器的资源开销优化算法.

3.1 解析器资源优化方法

由文献[13]分析可知,式(1)中,匹配查找模块的开销 C_{MatchF} 和匹配域提取模块的开销 $C_{\text{Extraction}}$ 占据了解析器总开销的 90% 左右,而这两部分的开销与状态转移边个数 K_M 和单个协议包头内所要匹配域个数 N_M 有关. 因此,解析器资源开销优化的重点是减少类型域和匹配域的数量. 可通过类型域和匹配域合并的方法降低开销:如图 3 所示,当有连续的几个类型域是固定联系时,可以仅匹配一次类型域的值,例如,当 VLAN 标签后直接跟 IPv4 包头时,可以将两者看成一个协议包头,从而降低类型域占据的 TCAM 表项;当有连续的几个匹配域均需提取时,可一并取出.

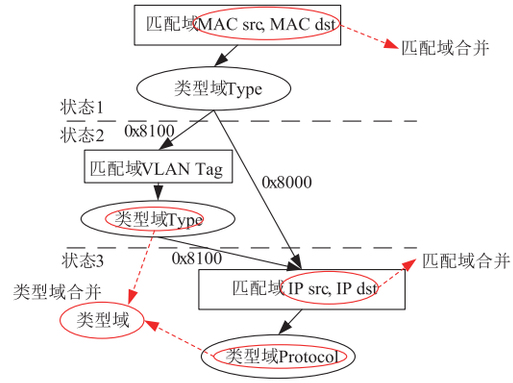


图3 包头解析优化方法示例

具体地,包头解析器的优化方法可形式化表述为如下算法,其中,以图 2 所示的协议状态转移图为输入,状态节点分别为各个协议包头,状态节点内部又包含匹配域和类型域.

算法 1 解析优化算法

```
0. i = 1;//初始化
1. while i ≤ N_P   //N_P是状态节点个数
2.    if node(i). out = 1 //合并连续的状态节点
3.       j = 0;
4.       while node(i + j). in = 1 & node(i + j). out ≤1 //可合并的前提是状态节点是链式相连,没有分叉
5.          j = j + 1;
6.       end while
7.       combine-t(node(i)-node(i + j));//合并状态节点
8.       i = i + j; N_P = N_P - j;
9.    end if
10.    else i = i + 1;
11. end while //完成类型域合并
12. for i = 1, i ≤ N_P, i = i + 1 //进行匹配域合并
13.    if M(node(i). m) //判断匹配域是否连续
14.       combine-m(node(i). m);//连续则合并
15. end for
```

其中,为方便表述,combine-t 表示类型域合并操作,以减少 TCAM 开销;combine-m 表示匹配域合并操作,$M(node(i).m)$ 则表示判断状态节点 i 中是否存在连续的匹配域,如果存在,则使用 combine-m 进行合并操作,以减少 RAM 开销.

3.2　匹配单元资源优化方法

假设所需的匹配域数目和长度一定,那么当流表的匹配宽度较小时,需要多级流表组合实现,由于各级流表内还包含动作执行器,因此其资源开销较大;但由于表项粒度较小,因此表项利用率较高. 相反,当匹配宽度较大时,资源开销较小,但表项利用率较低. 因此,匹配单元的资源优化可通过计算选取最优的流表宽度来实现:

$$\min\ C(N_T, W_{MT}) = N_T \times (C_{\mathrm{FlowT}}(W_{MT}) + C_{\mathrm{ActionP}}) \quad (8)$$

$$N_T \geq \sum_{i=1}^{K} \langle w_i / W_{MT} \rangle \quad (9)$$

其中,式(9)是优化方程的约束条件,表示假设有 K 个处理需求,每个处理需求的匹配域长度为 w_i;$\langle x/y \rangle$ 函数表示 x/y 的向上取整. 由于 $C_{\mathrm{FlowT}}(N_T, W_{MT})$ 是 W_{MT} 的非线性函数,且其变化规律由 TCAM 和 RAM 的具体实现方法决定,因此在理论上难以得到最优的流表匹配宽度值. 具体确定方法可根据实现平台(如 FPGA、ASCI 等)的不同,带入该实现平台上的测试数据,计算出最优解.

3.3　动作执行器资源优化方法

由式(3 ~ 7)可知,三类元动作单元的个数 N_{unit8},N_{unit16} 和 N_{unit32} 是决定动作执行器开销的关键因素,因此,最小化动作执行器的开销可通过设定三种元动作单元的最优个数来实现. 该问题可描述为:给定包头域长度,通过选择合适的元动作单元个数,最小化动作执行器的资源开销. 公式化表述为:

$$\min f(N_{\mathrm{unit8}}, N_{\mathrm{unit16}}, N_{\mathrm{unit32}}) = C_{\mathrm{ActionP}}(N_{\mathrm{unit8}}, N_{\mathrm{unit16}}, N_{\mathrm{unit32}}) \quad (10)$$

$$\text{s. t. } 8 \times N_{\mathrm{unit8}} + 16 \times N_{\mathrm{unit16}} + 32 \times N_{\mathrm{unit32}} = L_H \quad (11)$$

$$(N_{\mathrm{unit8}}, N_{\mathrm{unit16}}, N_{\mathrm{unit32}}) \geq (N_{\mathrm{min8}}, N_{\mathrm{min16}}, N_{\mathrm{min32}}) \quad (12)$$

其中,式(11)是 N_{unit8},N_{unit16} 和 N_{unit32} 所需要满足的数量要求,即其处理长度相加等于包头域的长度 L_H. 不等式(12)表示每类元动作单元都有一个最小值,所选的元动作单元数量不得低于其最小值,这是由于有一些匹配域无法拆分或合并后再进行处理,比如对 IP 数据包的 TTL 减 1 处理;在满足最小值约束的情况下,可选择合适的元动作单元数量以最小化开销. 更进一步地,从式(3 ~ 7)推导可知,为实现式(10)的优化目标,尽量减少 8 比特元动作单元的数量,增加 32 比特元动作单元的数量. 为实现该目标,可通过如图 4 所示的“域-字”拆分合并映射等方法,将所要处理的匹配域合并成一个元动作单元的输入,从而映射到多数 32 比特元动作单元中;同时,通过匹配域拆分,尽量充分利用各个元动作单元的处理能力,减少动作单元数量.

4　仿真结果

本文在 Nick 所提结构的基础上,基于本文所提优化方法,在 NetFPGA-10G 板卡[18]上完成了原型实现,包括收发单元、处理单元以及配置单元,其中收发单元包含 4 个 10G 物理端口和 1 个虚拟端口,物理端口与外部网络相连,虚拟端口通过 DMA 与主机虚拟网卡相连;处理单元包含包头解析器和 2 级流表;配置单元接收上层用户的配置信息并将表项下发到处理单元中去.

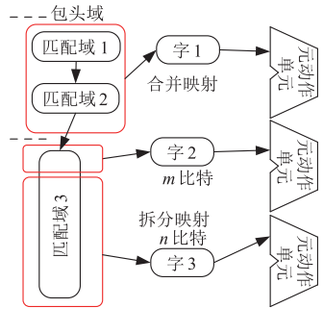


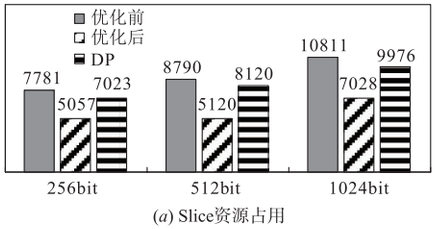
图4　“域–字”拆分合并映射示例

本节以64 比特为各级流表的匹配宽度,首先,分析了优化前后包头解析器和动作执行器的资源开销;然后,分析了整体资源开销.

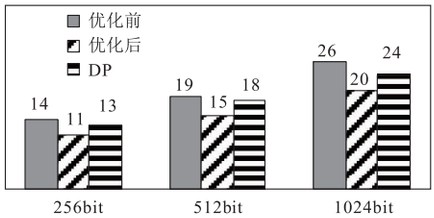
4.1　优化前后资源开销对比

(1)包头解析器优化前后资源开销

将数据总线位宽设为 1024 比特,为验证资源优化的效果,在包头域长度为 256 比特、512 比特和 1024 比特时,分别对包头解析器的资源开销进行了仿真. 图 5 对比了资源优化前后包头解析器开销,其中,DP 代表采用文献[13]中基于匹配域合并的优化方法.



(*a*) Slice资源占用



(*b*) RAM资源占用

图5　包头解析器资源与性能对比

由仿真结果可知,在同一包头域长度下,优化后的资源开销比优化前的资源开销降低了约 37%. 相比于 DP 的匹配域合并优化方法,本文针对每一个状态节点均进行匹配域合并,扩大了匹配域合并的范围;此外,还增加了类型域合并的优化方法,因此所带来的资源

优化效果更为显著.

(2)动作执行器优化前后资源开销

为验证动作执行器的资源优化方法,在包头域长度为256比特、512比特和1024比特时,分别对单个动作执行器的资源开销进行了仿真,布局布线的结果如图6所示.

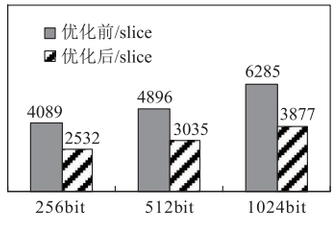


图6 优化前后的动作执行器资源开销

由仿真结果可知,在同一包头域长度下,优化后的资源开销比优化前的资源开销降低了约38%;若元处理单元个数增多,那么节省的硬件资源将更多.

4.2 整体资源开销分析

考虑到现有数据平面的研究尚未有基于NetFPGA-10G的原型系统实现,为对比整体资源开销,本文对NetFPGA开源社区中成员贡献的OpenFlow1.0项目[19]的代码进行了小量修改,将1级流表扩展到2级流表(片内资源限制,无法扩展更多级),包头域长度扩展到512比特,记作OF2.通过分模块布局布线和整体布局布线仿真,对片内资源较为紧张的分布式Slice资源耗费进行了对比,如图7所示,其中,本文所采用的结构记作FM,用本文所提方法进行优化后的FM记作FMO.

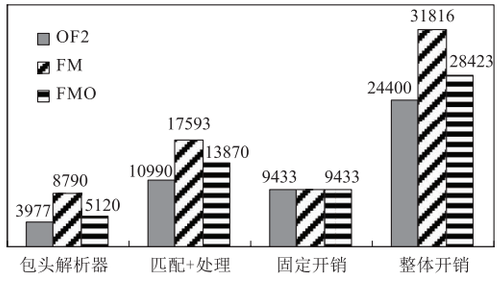


图7 OF2和FM/FMO硬件Slice资源开销对比

由硬件布局布线结果可知,FMO的整体资源开销比FM降低了约10.7%.由于硬件资源限制,仅实现了2级流表,但由结果可以看出,当在商用板卡上实现更多级流表时,其降低的资源将会更多.FMO一级流表的开销是OpenFlow的一级流表的1.26倍,包头解析器开销是OpenFlow的1.3倍,可见FMO虽然增加了许多用于配置的存储单元,但与OpenFlow处理相比,减少了大量的状态机使用,因此其额外开销不大;从整体开销上讲,不考虑PHY数据包转换模块、出入队列模块、以及DMA和寄存器系统等固定开销的情况下,FMO是OF2的1.16倍,在提供全可编程能力的基础上仅比OpenFlow多出约4000个Slice开销,其额外开销远小于数据平面中存放功能规则的TCAM开销.

因此,由实验结果可知,通过使用本文所提的资源优化方法,可以降低全可编程的设计架构所带来的额外开销.

5 结论

本文研究了面向全可编程硬件处理结构的资源优化方法,首先,分别对包头解析器和各级流表的资源开销进行了分析,明确了资源开销的可优化目标;其次,通过类型域合并、匹配域偏移量合并、"域-字"拆分合并映射等方法,降低了约38%的硬件资源开销.所提资源优化方法对可编程数据平面的硬件实现具有一定实用价值.

参考文献

[1] SCHAFFRATH G,WERLE C,PAPADIMITRIOU P,et al. Network virtualization architecture:proposal and initial prototype[A]. Proceedings of ACM Workshop on Virtualized Infrastructure Systems and Architectures[C]. Barcelona:ACM,2009. 63-72.

[2] CHIOSI M,CLARKE D,et al. Network Functions Virtualisation-Introductory White Paper[OL]. http://portal. etsi. org/NFV/NFV_White_Paper. pdf,2012-12-22.

[3] MCKEOWN N. Keynote talk:software-defined networking[A]. IEEE International Conference on Computer Communications[C]. Rio de Janeiro:IEEE,2009. 1-11.

[4] JACOBSON V,SMETTERS D K,THORNTON J D,et al. Networking named content[J]. Communications of the ACM,2012,55(1):117-124.

[5] MCKEOWN N,ANDERSON T,BALAKRISHNAN H N,et al. Openflow:enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review,2008,38(2):69-74.

[6] BOSSHART P,GIBB G,KIM H S,et al. Forwarding metamorphosis:fast programmable match-action processing in hardware for SDN[J]. ACM SIGCOMM Computer Communication Review,2013,43(4):99-110.

[7] BOSSHART P,DALY D,GIBB G,et al. P4:programming protocol-independent packet processors[J]. ACM SIGCOMM Computer Communication Review,2014,44(3):87-95.

[8] SONG H. Protocol oblivious forwarding:unleash the power of SDN through a future-proof forwarding plane[A]. Proceedings of ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)[C]. Hong Kong:ACM,2013. 127-132.

[9] CHOLE S, FINGERHUT A, MA S, et al. dRMT: disaggregated programmable switching [A]. Proceedings of ACM SIGCOMM'17 [C]. Los Angeles: ACM, 2017. 1 – 14.

[10] KOZANITIS K, HUBER J, SINGH S, et al. Leaping multiple headers in a single bound: wire-speed parsing using the Kangaroo system [A]. Proceedings of IEEE International Conference on Computer Communications [C]. San Diego: IEEE, 2010. 1 – 9.

[11] LU G, SHI Y, GUO C, et al. CAFE: a configurable packet forwarding engine for data center networks [A]. Proceedings of ACM SIGCOMM workshop on programmable routers for extensible services of tomorrow (PRESTO) [C]. Barcelona: ACM, 2009. 25 – 30.

[12] LI B, TAN K, LUO L, et al. ClickNP: highly flexible and high performance network processing with reconfigurable hardware [A]. Proceedings of ACM SIGCOMM [C]. Florianopolis: ACM, 2016. 1 – 14.

[13] GIBB G, VARGHESE G, HOROWITZ M, et al. Design principles for packet parsers [A]. Proceedings of the 9th ACM/IEEE Symposium on Architectures for Networking and Communications Systems [C]. NJ: ACM, 2013. 13 – 24.

[14] 段通,兰巨龙,胡宇翔,等. 一种支持网络功能演进的可重构数据平面 [J]. 电子学报, 2016, 44 (7): 1721 – 1727.
DUAN Tong, LAN Ju-long, HU Yu-xiang, et al. A reconfigurable dataplane enabling network function evolution [J]. Acta Electronica Sinica, 2016, 44 (7): 1721 – 1727. (in Chinese)

[15] JOSE L, YAN L, VARGHESE G, et al. Compiling packet programs to reconfigurable switches [A]. Proceedings ofUsenix Conference on Networked Systems Design and Implementation [C]. Oakland: USENIX Association, 2015. 103 – 115.

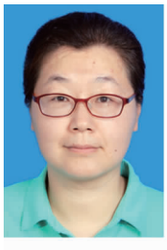
[16] MONSANTO C, REICH J, FOSTER N, et al. Composing software-defined networks [A]. Proceedings of Usenix Conference on Networked Systems Design and Implementation [C]. LOMBARD: USENIX Association, 2013. 1 – 14.

[17] 段通,兰巨龙,胡宇翔,等. SDN 中一种基于多级流表的功能组合方法 [J]. 电子学报, 2016, 44 (11): 2682 – 2687.
DUAN Tong, LAN Ju-long, HU Yu-xiang, et al. A function composition method of software-defined network based on multiple tables [J]. Acta Electronica Sinica, 2016, 44 (11): 2682 – 2687. (in Chinese)

[18] Stanford University. NetFPGA-10G project [OL]. https://github.com/NetFPGA/NetFPGA-public/wiki/Home_NetFPGA-10G, 2014 – 11 – 09.

[19] YABE T. OpenFlow implementation on NetFPGA-10G Design Document [OL]. https://docs.google.com/document/d/1ZwHXQZocKwQls6Ted8VZO8h9MjBtu9WxV2fAY44eOgE/edit, 2014 – 12 – 09.

作者简介



申　涓　女. 1976 年 10 月出生,河南方城人. 现为国家数字交换系统工程技术研究中心博士研究生. 主要研究方向为新型网络体系结构.
E-mail: sjcheng1@163.com

段　通　男. 1992 年 3 月出生,河南泌阳人. 2013 年毕业于清华大学电子工程系,现为国家数字交换系统工程技术研究中心博士研究生. 主要研究方向为可编程网络数据平面、网络功能虚拟化.
E-mail: duantong21@126.com