OpenFlow网络数据流路径建立开销的量化分析

吴 洁[1,2] 付斌章[1] 陈明宇[1,2] 张立新[1,2]
(中国科学院计算技术研究所先进计算机系统研究中心 北京 100190)[1]
(中国科学院大学 北京 100190)[2]

摘 要 OpenFlow采用数据平面与控制平面分离的架构,以软件实现的OpenFlow控制器作为控制平面对网络进行集中控制。在这种分离架构中,由于交换机需要与控制器进行交互,因此必然会产生一定的时间开销。经实验,数据流建立过程中的信息交互导致数据包传输时延至少增长2倍,严重降低了网络性能。因此,量化分析OpenFlow网络中流建立开销具有重要意义。分析流建立开销产生的原因,对导致数据包传输时延增长的影响因素进行量化分析。评估流建立开销对网络性能的影响是量化分析数据流路径建立开销的重点。

关键词 OpenFlow网络,数据流建立开销,影响评估
中图法分类号 TP393.0 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.11.011

Quantitative Analysis of Flow-setup Cost in OpenFlow Network

WU Jie[1,2] FU Bin-zhang[1] CHEN Ming-yu[1,2] ZHANG Li-xin[1,2]
(Research Center for Advanced Computer Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[1]
(University of Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract OpenFlow introduces a new network architecture by separating control plane from data plane and handing it over to a centralized software application named OpenFlow controller. Due to the devolving of two planes, the process of flow-setup will involve an interaction between the controller and the OpenFlow switches which shall bring ignorable cost. Experiments show that the flow-setup cost will lead to at least twice increment in packet transmission time. So it is meaningful to give a quantitative analysis of flow-setup cost and find the factors that influence packet transmission latency. Additionally, building an OpenFlow network to evaluate how serious the affection is can be necessary.

Keywords OpenFlow network, Flow-setup cost, Impact assessment

1 OpenFlow网络机制

OpenFlow由斯坦福大学的Nick Mckeown等人提出[6],其采用控制平面与数据平面分离的机制,由控制平面对数据平面进行集中控制。OpenFlow网络由两类基本元件构成:OpenFlow控制器和OpenFlow交换机。前者负责控制平面,后者负责数据平面,两者间通过OpenFlow协议实现信息交互。OpenFlow协议定义了多种OpenFlow消息,如packet-in、flow-mod、packet-out等。

OpenFlow采用的两平面分离架构为其带来诸多优势:由于控制平面交由软件(OpenFlow控制器)处理,从而可对网络进行可编程且细粒度的控制;交换机不再承担控制功能,故可足够简单,能使用廉价的商用交换机代替专用交换机;再者提供了一个开放的标准协议(OpenFlow协议),避免了对供应商的依赖[10]。

OpenFlow控制器与交换机间的交互过程如下:1)数据包到达交换机并发生流表项匹配失败时,其发送packet-in消息给控制器;2)控制器接收、处理packet-in消息,并向交换机下发添加流表项信息;3)交换机安装指定流表项,并将数据包从指定端口发出。

2 相关工作

OpenFlow数据平面与控制平面分离的架构虽有诸多优势,但两平面的分离使控制器与交换机间的交互易成为OpenFlow网络的瓶颈,影响网络性能[2,11]。虽然文献[2-4]也指出造成包传输时延的原因是数据流路径建立时的流表不命中,但并未就造成包传输时延增加的影响因素进行分析,而这正是本文的重点。

现在与数据流路径建立相关的研究主要集中在3方面:1)评估数据流建立速度[2,8],即OpenFlow网络中每秒可以建多少条流。但相关研究并未评估流建立开销对网络性能造成的影响,如对于流建立过程中发生流表不命中的数据包,其传

到稿日期:2014-10-20 返修日期:2014-12-30 本文受国家自然科学基金资助项目(61202056,60921002),中国科学院战略性先导科技专项(XDA06010401),华为A类高通量服务器项目(YBCB2011030)资助。

吴 洁(1991—),女,硕士生,主要研究领域为软件定义网络、网络虚拟化,E-mail:wujie@ict.ac.cn;付斌章(1982—),男,博士,副研究员,主要研究领域为高性能、高可靠系统互联、数据中心网络和片上网络,E-mail:fubinzhang@ict.ac.cn;陈明宇(1972—),男,博士,研究员,博士生导师,主要研究领域为计算机体系结构、操作系统、高性能计算机算法优化,E-mail:cmy@ict.ac.cn;张立新(1971—),男,博士,研究员,主要研究领域为高性能计算、微处理器、内存系统和数据中心计算系统等,E-mail:zhanglixin@ict.ac.cn。

输时延与流建立之后传输时延进行比较以及评估流建立开销占包传输时间的比例，这些正是本文要评估的内容。2）管理OpenFlow交换机流表，以减少流表不命中，从而减少流建立请求的数量，包括合理设置流表项过期时间[3,9]，使用合适的流表项替换算法[3,4]，使用通配的（wild-carded）OpenFlow规则[2,13]等。这些研究的目的在于如何避免开销的产生，本文旨在分析当开销产生时（每个数据流的建立都会产生开销，无可避免）造成开销的因素。3）缓解数据流建立带来的控制器高负载问题[2,11,12]，包括将控制器的一些控制功能下放到交换机、对packet-in消息进行过滤等。

本文旨在分析数据流建立开销产生的原因及数据流建立过程中造成数据包传输时延增长的影响因素，评估数据流建立开销对网络性能的影响，希望本结论能为控制器和交换机的优化提供参考。

3 数据流建立开销描述

数据流建立过程中，数据包由发送端经网络传输到接收端。在数据包传输过程中，当经过的交换机没有与该数据包匹配的流表项时则产生本地流表表项不匹配，即流表不命中（table-miss）。此时，交换机向控制器请求转发决定。该过程涉及交换机与控制器通过OpenFlow协议交互，这必然会产生一定的时间开销。由于在数据中心中2%～13%的流的流间隔时间小于10μs以及每个服务器机架每秒可产生约10000条流[5]，因此存在极大的流建立开销。本节对数据流建立开销进行量化分析。

3.1 数据包传输时延增长的影响因素

本节分析数据流建立过程中由OpenFlow架构导致的数据包传输延迟增长的影响因素。分析中以由1个控制器、2个终端和3个交换机组成的简单拓扑为例，如图1所示。

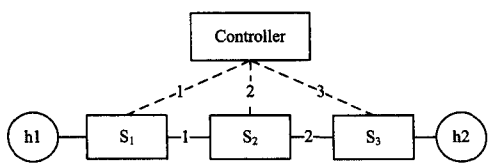

图1 示例拓扑

本节所用符号定义如下：

$TSL(i)$：数据包在交换机之间链路i上的传输时间；

$TCL(i)$：数据包在交换机与控制器之间链路i上的传输时间；

$TP(C)$：控制器处理时间；

$TP(S_i)$：交换机S_i安装流表项的时间；

T_{pre}：packet-in消息由第一台交换机发出到控制器处理完成所需的时间；

$T(i)$：流表项从控制器发出并安装入交换机S_i到缓存的数据包离开交换机S_i的时间；

T_{trans}：数据包从发送端到接收端的传输时间。

$h1$向$h2$发送数据包的过程如下：$h1$发出的数据包到达第一跳交换机S_1，由于此时S_1的流表中没有与该数据包匹配的流表项，导致流表查找不命中（table-miss）。其后，S_1向控制器发送packet-in消息请求流表项信息。消息由交换机发出且到达控制器的传输时间为$TCL(1)$。控制器接收并处理该消息，计算从$h1$到$h2$的路由路径。控制器的处理时间记为$TP(C)$。自packet-in消息由交换机发出到控制器处理完成的时间定义为预处理时间：$T_{pre}=TCL(1)+TP(C)$。控制器处理完成后将流表项下发到相关交换机S_1、S_2和S_3。上述处理过程中，因为$h1$持续向$h2$发送数据包，所以此时仍有大量数据包到达S_1。这些数据包都因为流表查找不命中而被缓存在本地，并发送packet-in消息到控制器（控制器对重复packet-in消息的处理由控制器具体实现决定）。控制器添加流表项的flow-mod消息由控制器发出并到达S_1。该消息的传输时间为$TCL(1)$。之后，S_1安装流表项，并将缓存的数据包从与S_2连接的端口转发，该过程需要$TP(S_1)$时间。流表项自控制器发出到数据包离开S_1共需时间为$T(1)=TCL(1)+TP(S_1)$。

数据包由$S1$发出并被$S2$接收，需时间$TSL(1)$。此时有两种情况：1）控制器发给S_2的流表项已到达并安装，即$TCL(2)+TP(S_2)\leqslant T(1)+TSL(1)=TCL(1)+TP(S_1)+TSL(1)$。假设所有交换机安装流表项的时间同为$TP(S)$，则上式可简化为$TCL(2)\leqslant TCL(1)+TSL(1)$（下面做同样简化）。此时$S_2$不会出现table-miss。2）控制器发给$S_2$的流表项未到达，即$TCL(2)>TCL(1)+TSL(1)$，这时$S_2$出现table-miss并发送packet-in消息到控制器，同时该数据包被缓存在本地，直到时间$T_{pre}+TCL(2)+TP(S)$才能将数据包发往S_3。综合上述两种情况，$T(2)=\max\{TCL(2),TCL(1)+TSL(1)\}+TP(S)$。

之后，数据包从S_2发出，经过$TSL(2)$时间到达S_3。同样地，在S_3处也存在两种情况：1）控制器发给S_3的流表项已到达并安装，即$TCL(3)+TP(S)\leqslant T2+TSL(2)=\max\{TCL(2),TCL(1)+TSL(1)\}+TP(S)+TSL(2)$，即$TCL(3)\leqslant\max\{TCL(2),TCL(1)+TSL(1)\}+TSL(2)$；2）控制器发给$S_3$的流表项尚未到达，即$TCL(3)>\max\{TCL(2),TCL(1)+TSL(1)\}+TSL(2)$。综上所述：

$$T(3)=\max\{TCL(3)+TP(S),T2+TSL(2)\}=\max\{TCL(3),TCL(2)+TSL(2),TCL(1)+TSL(1)+TSL(2)\}+TP(S)$$

。由此可以得出：

$$T(i)=\max\{TCL(i)+TP(S),T(i-1)+TSL(i-1)\} \tag{1}$$

则最终数据包传输时间为：

$$T_{trans}=T_{pre}+T(i)=TCL(1)+TP(C)+\max\{TCL(i)+TP(S),T(i-1)+TSL(i-1)\} \tag{2}$$

展开即得：

$$T_{trans}=TCL(1)+TP(C)+TP(S)+\max\{TCL(i),TCL(i-1)+TSL(i-1),TCL(i-2)+TSL(i-2)+TSL(i-1),\cdots,TCL(1)+TSL(1)+TSL(2)+\cdots+TSL(i-1)\} \tag{3}$$

由此式可见，数据流建立过程中，数据包的传输由OpenFlow架构所引入的额外时间受以下因素影响：

1）第一跳交换机与控制器间的传输延时（即式（3）中的$TCL(1)$）；

2）控制器处理packet-in消息的时间和交换机安装流表项的时间（分别对应式（3）中的$TP(C)$和$TP(S)$，分别表征控制器和交换机的处理能力）。

同时，由式（3）可以得出如下结论：数据包传输路径上离发送端越近的交换机，其到控制器间的传输延时对整个传输时间的影响越大（从上式可以看到，离发送端越近，在max式子中其加的TSL项越多）。

3.2 数据流建立开销产生原因

由 3.1 节 OpenFlow 网络中数据流建立过程的描述得出，数据流建立的额外开销是由数据包在交换机处进行流表匹配时流表查找不命中造成的。位于 OpenFlow 网络中不同位置的交换机造成流表不命中的原因略有区别。

第一跳交换机：造成 table-miss 的原因有两个：1)新流的到达。新流到达时交换机的流表中没有匹配该流的流表项，因此会出现 table-miss。2)流表项过期。这使得同一条流的数据包再次到达时产生 table-miss。这涉及到如何合理设置流表项过期时间(timeout)，以在流表尺寸和流表命中率间取得折中[3]。

后续交换机：数据包到达下一跳交换机时，控制器发出的流表项尚未安装到该交换机。以交换机 S_2 为例，假设该交换机到控制器的链路延迟 $TCL(2)$ 较大，如 $TCL(2)>TCL(1)+TSL(1)$，当数据包到达 S_2 时，控制器发往 S_2 的流表项尚未到达，这时会造成 S_2 匹配流表发生 table-miss，导致其向控制器发送 packet-in 消息。这种后续交换机上发生的 table-miss 会给控制器造成额外的处理 packet-in 消息的负载。

table-miss 的产生增加了交换机与控制器的交互，从而增加了数据包的传输时延。增加的传输时延由式(3)中 max 项结果决定。由于数据包传输时延是控制器到沿途交换机的链路延迟及交换机间链路延迟的多元函数。因此，是否发生 table-miss 以及 table-miss 对时延的影响与具体的网络环境配置有关。

另外，由于发生 table-miss 时，交换机会产生 packet-in 消息发往控制器，且每个新流都会造成一个或多个 packet-in，显著增加了控制器负载。第一个发生 table-miss 的数据包，由交换机产生 packet-in 消息发往控制器，经由控制器响应，产生用于添加新流表项的 flow-mod 消息发往交换机，填充流表项。在对第一个 table-miss 处理的整段时间内，后续到达的该流的其它数据包在此交换机上依然会因为 table-miss 而产生 packet-in 消息。多个重复的 packet-in 消息必然会极大地增加控制器的处理开销。

4 实验评估

本实验采用 Open vSwitch(OVS)[1](使用 1.9.3 版本)作为 OpenFlow 交换机。实验中，通过修改 Open v-Switch代码，在沿途的每个 OVS 交换机上精确地得到每个数据包从发生 table-miss 到其发送到网卡驱动的时间(精确到微秒[1])，这段时间即该数据包在此交换机上的时延)，经过多个交换机上时延的累加，从而精确评估数据流建立开销。实验中采用 Beacon[7]控制器(使用 1.0.4 版本)作为 OpenFlow 控制器。实验中数据包的发送端和接受端为 kvm 虚拟机。实验中的 OVS 和 Beacon 均安装在华为 RH2258 服务器上，每台服务器配备 32GB 内存，CPU 型号为 Intel Xeon E5645，主频2.4 GHz，网卡为 Intel Corporation 82580 千兆网卡，服务器操作系统为 CentOS 6.3，内核版本为 2.6.32。实验拓扑如图 2 所示。

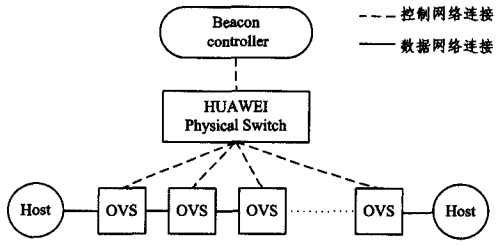

图 2 实验拓扑示意

Open vSwitch 交换机与控制器之间通过一台外部华为交换机连接，采用 out-of-band 方式通信。实验结果采用从发送端向接收端发 ping 测试包[2]的形式获得，沿途 OVS 交换机将时延累计入 ping 测试包，从最后一跳交换机处读出最终的数据流路径建立开销。整个评测包含 5 组实验，分别对应发送端与接收端间有 2 跳、3 跳、4 跳、5 跳、6 跳的情形。每组实验重复进行 10 次，并取平均值作为最终结果。实验中计算了因流表项建立而带来的开销占总传输时间的比例，并计算了在数据流建立过程中(产生 table-miss 的情况下)与数据流建立后(流表命中，table-hit)相比包传输时间增长的倍数：

$$\text{包传输时间增长倍数} = \frac{\text{数据流建立过程中包传输时间}-\text{数据流建立后包传输时间}}{\text{数据流建立后包传输时间}} \quad (4)$$

实验结果如图 3 和图 4 所示。

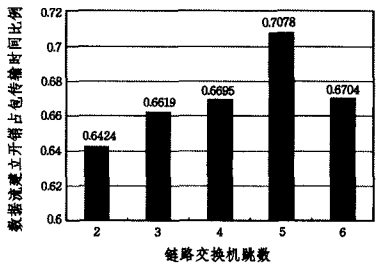

图 3 数据流建立开销占包传输时间的比例

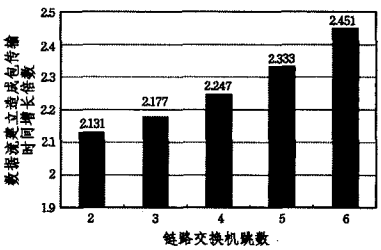

图 4 数据流建立开销造成包传输时间增长倍数

从上面实验得出：数据流路径建立开销对网络延迟造成的影响很严重。图 4 显示数据流路径建立导致包传输时间至少增长 2 倍，也就是在数据流路径未建立情况下包传输时间至少是数据流建立后的 3 倍。图 3 表明建立流表造成的开销占包总传输时间的 60%～70% 。而且，在实际网络环境中，网络中的流量不仅只有测试包，因而建立数据流路径的过程带来的开销将会更加严重。

结束语 本文就 OpenFlow 网络中数据流路径建立开销问题进行了详细的量化分析。数据流建立开销是由于在新流

---

[1] 在实验过程中，交换机上的时延通常不足 1 毫秒，因此取微秒作为计时单位。

[2] 此处 ping 测试包并非由标准 Linux ping 程序产生，而是由我们另写的程序生成的特定 ping 测试包，能够被修改过的 OVS 识别并对其进行特殊处理。

到来时，交换机中无该流相关的转发信息而导致流表不命中造成的。通过分析发现，产生流表不命中的原因与交换机在网络中的位置有关；数据流建立开销对包延时的影响是控制器到沿途交换机的链路延迟及交换机间链路延迟的多元函数；是否会发生 table-miss、发生 table-miss 对时延产生的影响与具体的网络环境配置有关。同时，网络产生的延时与以下因素相关：1)第一跳交换机到控制器的延时；2)控制器及交换机的处理能力；3)交换机在路由路径中的位置。最后，我们通过实验评估了流建立开销对传输延迟造成的影响：数据流的建立致使包传输延迟至少增加 2 倍，对网络性能产生显著影响。

参 考 文 献

[1] Open vSwitch[OL]. http://openvswitch.org/

[2] Curtis A R, Mogul J C, Tourrilhes J, et al. DevoFlow: Scaling flow management for high-performance networks[J]. ACM SIGCOMM Computer Communication Review. ACM, 2011, 41(4): 254-265

[3] Zarek A, Ganjali Y, Lie D. Openflow timeouts demystified[D]. Toronto, Ontario, Canada: Univ. of Toronto, 2012

[4] Kim E D, Lee S I, Choi Y, et al. A flow entry management scheme for reducing controller overhead[C]// 2014 16th International Conference on Advanced Communication Technology (ICACT). IEEE, 2014: 754-757

[5] Benson T, Akella A, Maltz D A. Network traffic characteristics of data centers in the wild[C]// Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. ACM, 2010: 267-280

[6] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74

[7] Beacon controller [OL]. https://openflow.stanford.edu/display/Beacon/Home

[8] Huang D Y, Yocum K, Snoeren A C. High-fidelity switch models for software-defined network emulation[C]// Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, 2013: 43-48

[9] Vishnoi A, Poddar R, Mann V, et al. Effective switch memory management in OpenFlow networks[C]// Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems. ACM, 2014: 177-188

[10] Fundation O N. Software-defined networking: The new norm for networks[R]. ONF White Paper, 2012

[11] Wang A, Guo Y, Hao F, et al. Scotch: Elastically Scaling up SDN Control-Plane using vSwitch based Overlay[C]// Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies. ACM, 2014: 403-414

[12] Yu M, Rexford J, Freedman M J, et al. Scalable flow-based networking with DIFANE[J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 351-362

[13] Yan B, Xu Y, Xing H, et al. CAB: a reactive wildcard rule caching system for software-defined networks[C]// Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, 2014: 163-168

(上接第 58 页)

面的应用提供了十分契合的细化并行方案，且对代码修改量和修改难度的要求较低，最大限度地保证了代码移植效率。然而在具体的技术细节方面，特别是对 C++语言的支持方面，Intel 编译器对 CPU/MIC 混合架构尚未做到软件上的完美无缺。除了在原理上无法保证 C++原有的良好的封装性外，甚至还存在不少的漏洞。目前完整的大规模并行有限元模拟程序一般仅采用 MPI 进程并行模型。正如引言中指出的那样，每个进程包含的单元计算可以并且应当得到进一步细化的并行。当计算节点采用 CPU+MIC 构型时，应当考虑利用 MIC 设备将每个进程中的单元操作并行化。本文中的应用相当于多进程任务中的某一个进程与 MIC 进行结合的测试。后续我们还将考虑将一个成熟的有限元计算流体程序[6]完整地移植到大规模多 CPU 多 MIC 的硬件平台上。

参 考 文 献

[1] 沈铂，张广勇，吴韶华，等. 基于 MIC 平台的 offload 并行方法研究[J]. 计算机科学，2014，41(6A)：477-480
Shen Bo, Zhang Guang-yong, Wu Shao-hua, et al. Research of Offload Parallel Method Based on MIC Platform[J]. Computer Science, 2014, 41(6A): 477-480

[2] 刘跃进，薛孟君. LDLT 分块求解计算方法在有限元分析中的编程实现[J]. 计算机科学，2014，41(11A)：408-409
Liu Yue-jin, Xue Meng-jun. Program of Blocks Combining with LDLT Method for Finite Element Analysis[J]. Computer Science, 2014, 41(11A): 408-409

[3] 刘建华，王朝尉，任江勇，等. 面向异构架构的混合精度有限元算法及其 CUDA 实现[J]. 计算机科学，2012，39(6)：293-296
Liu Jian-hua, Wang Chao-wei, Ren Jiang-yong, et al. Mixed Precision Finite Element Algorithm on Heterogeneous Architecture [J]. Computer Science, 2012, 39(6): 293-296

[4] 王迎瑞，任江勇，田荣. 基于 GPU 的高性能稀疏矩阵向量乘及 CG 求解器优化[J]. 计算机科学，2013，40(3)：46-49
Wang Ying-rui, Ren Jiang-yong, Tian Rong. Efficient Sparse Matrix-vector Multiplication and CG Solver Optimization on GPU[J]. Computer Science, 2013, 40(3): 46-49

[5] Luo Li, Yang Chao, Zhao Yu-bo, et al. A scalable hybrid algorithm based on domain decomposition and algebraic multigrid for solving partial differential equations on a cluster of CPU/GPUs [J/OL]. http://www.cs.colorado.edu/~cai/papers/lyzc2011.pdf

[6] Chan K, Zhang Ke-ke, Li Li-gang, et al. A new generation of convection-driven spherical dynamos using EBE finite element method[J]. Physics of the Earth and Planetary Interiors, 2007, 163(1-4): 251-265

[7] Kong Da-li, Zhang Ke-ke, Gerald S, et al. A three-dimensional numerical solution for the shape of a rotationally distorted polytrope of index unity[J]. The Astrophysical Journal, 2013, 763(2): 116-126

[8] Kong Da-li. Analytical and Numerical Studies of Several Fluid Mechanical Problems[D]. University of Exeter, 2012

[9] 王恩东，张清，沈铂，等. MIC 高性能计算编程指南[M]. 北京：中国水利水电出版社，2012
Wang En-dong, Zhang Qing, Shen Bo, et al. High-Performance Computing on the Intel Xeon Phi-How to Fully Exploit MIC Architectures[M]. Beijing: China Water and Power Press, 2012

[10] Kong Da-li, Zhang Ke-ke, Gerald S. Shapes of two-layer models of rotating planets[J]. Journal of Geophysical Research, 2010, 115(E12003): 1-11