第 42 卷 第 6A 期
2015 年 6 月

计 算 机 科 学
Computer Science

Vol. 42 No. 6A
June 2015

软件定义网络性能研究

曾 珊[1,2] 陈 刚[2] 齐法制[2]

(中国科学院大学 北京 100049)[1] (中国科学院高能物理研究所计算中心 北京 100049)[2]

摘 要 服务器虚拟化以及各种云计算应用的出现,改变了传统的网络服务形态。软件定义网络(Software Defined Networking,SDN)技术将网络的控制平面和数据平面分离开来,为未来互联网技术提供了一种新的解决方案。然而,由于数据的爆炸式增长,网络配置要求的频繁变化,需要研究高效的 SDN 实现方案。综述了 SDN 的起源与发展,详细介绍了 ONF 提出的 SDN 典型的三层架构中的基础设施层、控制层、应用层、南向接口和北向接口,然后结合目前业内的研究现状,详尽地分析了 SDN 性能面临的问题和解决思路,最后提出了基于智能流表管理和树状可伸缩的控制平面管理的 SDN 性能优化方案。

关键词 SDN,OpenFlow,控制器,性能

中图法分类号 TP393 **文献标识码** A

Survey on Performance of Software Defined Networking

ZENG Shan[1,2] CHEN Gang[2] QI Fa-zhi[2]

(University of Chinese Academy of Science, Beijing 100049, China)[1]

(Computing Center, Institute of High Energy Physics, Chinese Academy of Science, Beijing 100049, China)[2]

Abstract The emergence of server virtualization and cloud computing applications has changed the traditional form of network services. SDN(Software Defined Networking) decouples the control plane from data plane which provides a new solution for the future internet technologies. But with the explosion of data, the network configuration requirement changes frequently which needs an effective SDN implementation. This paper starts with the origin and development of SDN, and introduces the three layer architecture of SDN proposed by ONF in details which includes the infrastructure layer, the control layer, the application layer, south bound interface and north bound interface. Then combined with the research development of the industry, The problems and the solutions of SDN performance are analyzed in details. In the end, a future SDN performance optimization scheme based on the intelligent flow table management and the scalable control layer management is proposed.

Keywords SDN, OpenFlow, Controller, Performance

1 引言

目前各个数据中心内部都部署了大量的云计算和虚拟化环境,而传统的网络架构无法灵活构建虚拟化环境需要的各种虚拟化网络,且无法进行灵活横向扩展,同时,大规模的虚拟化环境使得网络管理变得更加困难,网络配置更加繁琐,网络链路的调优工作也变得更加耗时耗力。随着数据中心内部流量增大、数据交换频繁,如果不能合理分配传输路径,将会导致数据中心内部数据拥塞。在这种背景下,SDN 的架构被重新提出,并逐步成为各研究机构和企事业单位的研究热点。

SDN 的概念起源于美国 GENI 项目资助的 Clean Slate 课题,在该课题中,以 Nick McKeown 教授为首的研究团队提出了 OpenFlow 的概念并用于校园网络的实验创新,并在 2008 年 SIGCOMM 会议上发表了题为"OpenFlow: Enabling Innovation in Campus Networks"的论文,论文中首次详细地介绍了 OpenFlow,指出 OpenFlow 可以将控制功能从网络设备中分离出来,在网络设备上维护流表(flow table)结构,使数据分组按照流表进行转发,而流表的生成、维护、配置则由中央控制器来管理[1]。

在 OpenFlow 为网络提供可编程特性的基础上,Nick 和他的团队又进一步提出了 SDN 的概念。2009 年 SDN 被 MIT 评为十大前沿技术。2011 年,开放式网络基金会(Open Network Foundation,ONF)成立,其专门致力于创新和发展 SDN,并且对 OpenFlow 进行标准化,先后发布了 SDN 标准架构以及 OpenFlow 协议标准 v1.0—1.4 版本。2012 年开始 SDN 完成了从实验技术向网络部署的重大跨越,在学术界、工业界和产业界都受到了极大的重视,SDN 相关技术研究迅速开展起来,成为近年来的研究热点。2013 年,SIGCOMM

本文受科技部 973 基金项目:物理分析平台建设(2013CB834303),中国科学院战略性先导科技专项(A 类)(XDA10010900),国际自然科学基金项目:基于 IPv6 技术和软件定义网络架构的高能物理数据传输虚拟专用网络和系统平台关键技术研究与应用(11305196)资助。

曾 珊(1987—),女,博士生,助理研究员,主要研究方向为软件定义网络、网络虚拟化、云计算,E-mail:zengshan@ihep.ac.cn;陈 刚(1961—),男,博士,研究员,主要研究方向为网格计算、云计算、网络虚拟化;齐法制(1978—),男,硕士,高级工程师,主要研究方向为软件定义网络、网络虚拟化。

会议收录了多篇相关文章，甚至将 SDN 列为专题来研讨，带动了 SDN 相关研究的蓬勃发展。

2 SDN 架构

本文主要研究 ONF 提出的 SDN 三层架构（该架构是目前使用最广泛的，但并不是 SDN 唯一的架构），如图 1 所示。最顶层为应用层，包括各种不同的网络业务和应用；中间的控制层主要负责处理数据平面资源的编排、维护网络拓扑、转发信息等；最底层的基础设施层负责数据处理、转发和状态收集。其中，以控制层为中心，其与应用层和基础设施层之间的接口分别被定义为北向接口和南向接口，是 SDN 架构中两个重要的组成部分。ONF 在南向接口上定义了开放的 OpenFlow 协议标准，而业界在北向接口上还没有达成标准共识。下面对 SDN 的每一层及其接口进行详细分析。

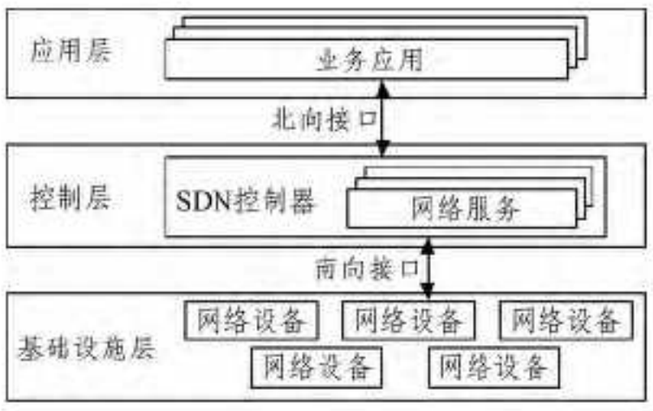

图 1 ONF 提出的 SDN 三层架构

2.1 基础设施层

SDN 架构中的基础设施层负责网络数据的高速转发，根据实际场景的需要，可以采用软件实现的 SDN 交换机，也可以使用硬件实现的 SDN 交换机。其中软件实现的 SDN 交换机通常与虚拟化 Hypervisor 相整合，从而为云计算场景中的多租户灵活组网等业务提供支持，例如 Open vSwitch(OVS)。硬件实现的 SDN 交换机则能够支持基于硬件设备的组网，同时能够满足 SDN 网络和传统网络的混合组网需求，例如 OpenFlow 交换机。下面分别介绍 OVS 和 OpenFlow 交换机。

2.1.1 OVS

OVS 是基于软件实现的多层虚拟交换机，使用开源 Apache2.0 许可协议，由 Nicira Networks 开发，主要实现代码为可移植的 C 代码。它的目的是让大规模网络自动化可以通过编程扩展来支持标准的管理接口和协议（例如 NetFlow、sFlow、SPAN、RSPAN、CLI、LACP、802.1ag）。OVS 支持多种 Linux 虚拟化技术，包括 Xen/XenServer、KVM 和 virtualBox 等。

2.1.2 OpenFlow 交换机

OpenFlow 协议最新发布的版本为 v1.4，随着 OpenFlow 协议版本的演进，OpenFlow 交换机的架构也发生了变化，最新的 OpenFlow 交换机主要由 3 部分组成[3]：一个或多个流表、一个组表(Group Table)和一个安全通道，如图 2 所示。

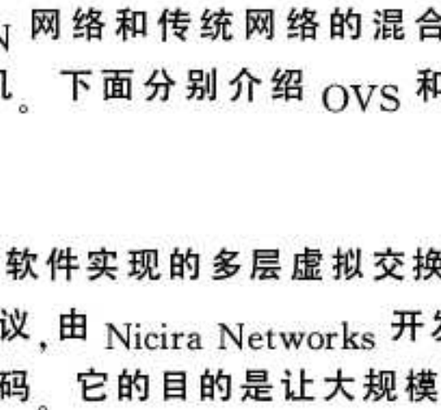

图 2 最新的 OpenFlow 交换机组成

流表和组表执行分组查找和转发，流表结构由匹配域、优先级、计数器、指令、超时定时器和 Cookie 组成，如图 3 所示。

匹配域	优先级	计数器	指令	超时定时器	Cookie

图 3 OpenFlow v1.3 流表结构

组表将多个流编成一个组，然后执行相同的操作集。每条组表记录包括：组标识符、组类型、计数器和动作桶，如图 4 所示。

组标识符	组类型	计数器	动作桶

图 4 OpenFlow 组表结构

安全通道是交换机与控制器进行通讯的接口，在 OpenFlow v1.0 中规定该安全通道需要使用 TLS 安全隧道。而从 v1.1 开始，OpenFlow 不再强制要求使用 TLS 隧道，而是可以使用普通的 TCP 连接。在 OpenFlow 的各个版本中，缺省情况下使用 TCP 6633 端口做为安全通道。

2.2 南向接口

ONF 在 SDN 南向接口上定义了开放的 OpenFlow 协议标准。OpenFlow 协议支持 3 种消息类型：Controller-to-Switch、Asynchronous 和 Symmetric，每一个类型都有多个子类型，控制器和交换机之间通过这 3 类消息进行连接建立、流表下发和信息交换，以实现对网络中所有 OpenFlow 交换机的控制。Controller-to-Switch 信息由控制器发起并且直接用于检测交换机的状态。Asynchronous 信息由交换机发起并通常用于更新控制器的网络事件和改变控制器中交换机的状态信息。Symmetric 消息不必通过请求建立，控制器和交换机都可以主动发起，并需要接收方应答，它们都是双向对称的消息，主要用来建立连接、检测对方是否在线等。控制器和 SDN 交换机之间使用比较频繁的消息类型如图 5 所示：数据分组到达 SDN 交换机时，若在 SDN 交换机的流表没有找到相应的匹配项，则 SDN 交换机会向控制器发送 Packet-in 消息来请求数据分组的转发规则；控制器处理完 Packet-in 消息后，会将数据分组的转发规则通过 Packet-out 消息发送给 SDN 交换机。另外，控制器还会根据上层应用层提供的交换机转发规则修改信息，通过 Modify-state 消息发送给 SDN 交换机。

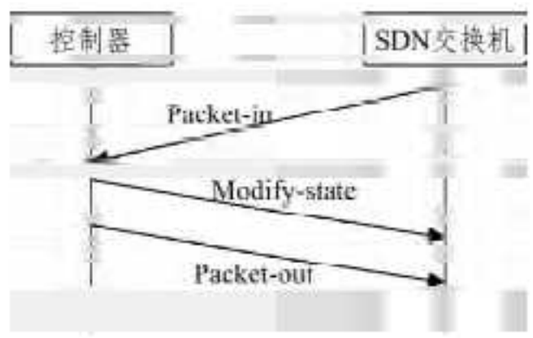

图 5 控制器和 SDN 交换机之间交互示例

2.3 控制层

控制层不仅负责对基础设施层的网络设备的统一管理，还负责向上层应用层的业务应用提供网络功能调用，在 SDN 架构中具有举足轻重的地位。因此，位于控制层的 SDN 控制器(Controller)一直以来都是 SDN 领域关注的焦点。

目前，开源社区提供了很多开源的 SDN 控制器，不同的控制器拥有各自的特点和优势。其中，NOX[3]、Ryu、Floodlight 和 OpenDaylight 等控制器在架构和功能上比较具有典型性，已被业界广泛使用。

2.4 北向接口

SDN 的核心理念在于推动网络业务的创新，而北向接口

是这一理念的最主要推动力。通过北向接口,网络业务的开发者可利用软件编程的方式调用不同的网络资源和服务能力,网络业务编排系统可以获知网络资源的工作状态并对网络资源进行调度,实现资源的统一交付,从而更好地支持云计算等新业务对网络资源的需求。此外,SDN 北向接口还可以提供物理网络视图、虚拟网络叠加视图、指定域抽象视图、基本连接视图以及 QoS 相关连接视图等。但是,目前 SDN 北向接口还没有确定统一的行业标准[4]。

2.5 应用层

应用层位于 SDN 三层架构的最顶层,它包括网络业务相关的管理、安全等基本应用,以及根据用户需求定制有其他指定功能的网络业务。根据具体的业务需求,应用层的内部可以再做更细致的划分,其中下层的应用(例如基本应用)为上层的应用(例如定制业务)提供功能调用接口,使上层应用的开发更加方便。

3 SDN 性能优化调研

SDN 控制平面会不断地收集底层 SDN 交换机的运行状态(例如配置更新等),从而形成整个网络的全局状态;每一个流的第一个数据分组到达 SDN 交换机时,SDN 交换机都需要向控制平面请求数据分组的转发规则,控制平面则将转发规则下发到 SDN 交换机的流表中(流表安装),因此整个 SDN 的性能很大程度上取决于控制平面的性能,而控制平面的性能反过来又受控制平面扩展程度的影响。文献[5]指出目前使用最广泛的控制器之一 NOX 每秒能接收 3 万条流请求,而流表安装的速率只有每秒 275 条[6],流表安装的速率和流请求的频率之间有很大的差距。同时,Benson 等人的研究也表明[7],在一个有 100 台交换机的网络环境中,流速率最坏情况下能达到每秒 1 千万条,NOX 无法满足大数据时代下数据中心的需求。为了提高控制平面的性能,目前的研究主要集中在提高单个控制器的处理能力、降低请求频率以及控制层面的高可扩展性 3 个方面。

3.1 多线程控制器

控制器本质上来讲就是一个软件,目前在提高单个控制器的处理能力上,主要采用并行化和批处理的方式设计多线程控制器。

Maestro[8,9]是采用 Java 开发的多线程控制器,它通过采用并行处理的方法和输入输出批处理的吞吐优化技术大大地提高了控制器的处理能力。NOX-MT 是在 NOX 基础上开发的多线程控制器,文献[5]通过分析和比较 Maestro、NOX、NOX-MT 和 Beacon[10]的性能,发现 NOX-MT 在吞吐和延迟方面更优于另外 3 个控制器。在一个 8 核,2GHz CPU 的机器上,NOX-MT 每秒能接收 160 万条流请求,平均响应时间为 2ms。NOX-MT 通过异步 I/O 的批处理方式来减少 I/O 开销,尽管在性能上有一定的提升,但是还远远达不到实际的要求。其缺陷主要表现在:过多地使用动态内存分配以及基于每条请求的冗余的内存拷贝等。

3.2 降低请求频率

控制平面接收到的请求过多,会使控制平面由于处理不及时而导致延迟过大[11]。因此为了降低控制平面处理的延迟,很多研究都集中在降低请求频率上,主要分为两类方法:一类是通过修改 SDN 交换机来让数据平面处理一部分流请求,而不需要提交给控制平面的控制器;另一类则采用优化 SDN 交换机的组织结构的策略[12]。

3.2.1 DIFANE

DIFANE[13]的基本思想是:首先在交换机中选出权威交换机(authority switch),每个权威交换机管理一定区域内的 OpenFlow 交换机。控制器主动地将分区规则安装到所有的 OpenFlow 交换机上,并根据全局网络信息主动在权威交换机上安装权威规则。当普通交换机收到新的数据分组时,根据其分区规则直接和自己分区内的权威交换机进行通信,由于权威交换机已经提前部署了权威规则,因此可以向普通交换机安装缓存的规则,同时,直接将请求数据转发给目的地而不返回给源交换机。只有权威交换机上也没有找到相应的规则时,数据分组才会提交给控制平面。DIFANE 采用主动和被动两种安装流表的方式将流量保持在数据平面,从而降低交换机向控制平面请求的频率,进而降低控制平面的负载。

3.2.2 DevoFlow

为了满足高性能网络的需求,DevoFlow 采用规则复制和局部操作两种方式来降低数据平面向控制平面的请求频率[14]。规则复制方式是在包含通配符的流表项中的"操作"字段上增加了 CLONE 标志,如果该标志清零,则表示匹配该流表项的报文按正常情况处理;如果该标志被置位,则直接根据匹配报文建立精确匹配的微流,从而细化每一条微流的统计信息。由于带通配符的流表项一般由硬件 TCAM 来实现,而精确匹配的流表项由软件实现,从而也减少了 TCAM 资源的消耗。交换机只需要提前安装带有通配符的流表项,就可以大量减少与控制器之间的报文交互,同时节约硬件资源的开销;局部操作方式包括多路径支持和快速重路由。多路径支持指的是为 OpenFlow 交换机中可复制的通配符流表项提供多个可能的输出端口,DevoFlow 根据概率分布将报文输出至特定端口中,而不是采用传统的等价多路径路由方式(ECMP);快速重路由为 OpenFlow 交换机指定了一到多条备用路径,从而在链路失效时立即转向备用路径,而不是转发给控制平面。另外,针对统计信息收集过程的开销,DevoFlow 采用采样、触发和报告、近似统计的方法来提高统计信息收集效率。采样方法是将统计信息按照一定的样本概率转发给控制器;触发和报告通过设置阈值,在统计信息满足阈值条件时将统计信息发给控制器;近似统计则只将流量最大的 k 条流的统计信息发给控制器,一般情况下,这 k 条流包含了 80%~99%的数据流。

3.2.3 Mahout

A. Curtis 等人在对数据中心网络流量分析的研究中发现,80%的流所传输的数据量均小于 10kB,同时大部分流的到达和离开都非常快,如果每一条流都需要控制器来调度、决策路由,将会带来较大的开销和时延,同时控制平面的可扩展性较差。为了提高数据中心的网络资源利用率、减轻 SDN 控制平面的负载压力,A. Curtis 等人在 2011 年的 INFOCOM 会议上提出了 Mahout 架构[15],其基本思想是:对网络中的流进行检测,识别出少数传输着重要数据量的流,称为大象流。只有大象流的流表项安装才需要控制器处理,其他的老鼠流则由数据平面的交换机自己处理,从而提高控制平面的处理性能。

3.2.4 CMQ

T. Luo 等人提出控制平面和数据平面之间的通信流量主要来源于 Packet-in 和 Packet-out 两种控制消息,通过实验

分析发现，即使在低负载的情况下，Packet-in 消息和 Packet-out 消息之间的往返时间间隔为 3.67～4.06ms 左右，而每隔 0.1～10μs 就有交换机向控制平面提交 Packet-in 消息，请求流表安装的频率要远远高于控制器对 Packet-in 消息处理的频率[16]，为此，T. Luo 等人通过对 OpenFlow 通信协议进行简单的修改，提出了 CMQ（Control-Message Quenching）模式，其基本思想是建立数据分组的〈源地址，目的地址〉对的队列，当交换机收到的数据分组在流表中没有匹配项时，先查看该数据分组的源地址和目的地址是否在该队列中，如果在，则先将该数据分组的请求缓存在交换机中，直到交换机收到了该〈源地址，目的地址〉对的第一个数据分组请求的 Packet-out 消息，再根据新安装的流表信息进行数据转发；如果该数据包的源地址和目的地址不在该队列中，则向控制器发送 Packet-in 消息。通过 CMQ 这种方式，当数据包在流表中没有找到匹配项时，对于每一个〈源地址，目的地址〉对的队列来说，只有收到第一个数据分组的请求，交换机才会向控制器发送一次 Packet-in 消息，从而降低了交换机和控制机之间的交互频率，实验表明，CMQ 模式能够缩短流表安装的时间，提高 SDN 网络的带宽使用率[17]。

3.3 控制平面扩展性

随着网络规模的增大和业务需求的增加，控制平面中单个控制器集中管控带来的单点失效的天然局限性问题，使得研究控制平面的可扩展性解决方案（即多控制器解决方案）变得十分重要。而控制器的数量和它们之间的网络状态（包括拓扑、传输能力、路由限制等）的协同和交互应该如何实现，以保证网络状态的一致性和可扩展性，还需要进行大量深入的研究。目前多控制器的解决方案主要包括 HyperFlow[18]、Onix[19]、Kandoo[20]和 Palette[21]。

3.3.1 HyperFlow

与传统的单控制器网络架构不同，HyperFlow 通过部署多台控制器来分区管理 OpenFlow 交换机，利用分布式文件存储系统 WheelFS，使得每台控制器能够维护全网络视图。如图 6 所示，左边是单控制器的网络部署图，一台控制器管理整个网络中的所有 OpenFlow 交换机，右边是在相同网络拓扑情况下 HyperFlow 部署的多控制器网络部署图，其中每个域中都部署了一个或者多个控制器，每个控制器的管理范围由其颜色进行标识，控制器之间通过消息的发布-订阅模型来互相传输网络事件。

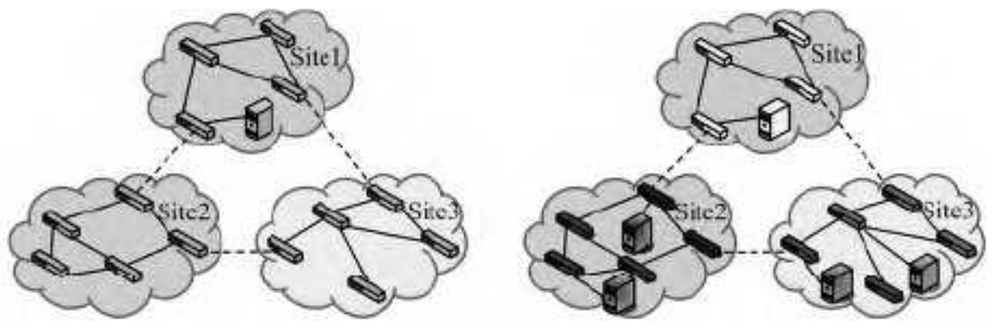

图 6 跨域的单控制器和 HyperFlow 部署的多控制器网络部署对比图

从测试的性能来看，HyperFlow 能够处理的网络事件小于 1000 次/秒，性能还不够高[18]；同时，全网络视图的更新与网络状态信息发布的周期事件和传输时延相关，这是由 WheelFS 的实现机制决定的，因此 HyperFlow 适用于网络状态事件更新不频繁、对网络状态一致性要求不高的网络，而对于网络状态更新频繁或者较大规模网络，HyperFlow 可能会面临性能瓶颈。

3.3.2 Onix

为了解决控制平面的可扩展性、可靠性和通用性等方面的问题，Koponen 等人提出了面向大规模网络的分布式 SDN 部署解决方案 Onix。Onix 网络架构由物理网络基础设施、网络连接基础设施、Onix 和网络控制逻辑 4 部分组成，如图 7 所示。其中 Onix 部分由交换机入口/出口、状态分发入口/出口和网络信息库（Network Information Base，NIB）组成。物理网络基础设施允许 Onix 读写网络状态，网络连接基础设施提供 Onix 与物理网络基础设施之间的通信连接；Onix 采用分布式架构向上层网络控制逻辑提供网络状态的编程接口，网络控制逻辑则通过 Onix 提供的 API 来决策网络行为。

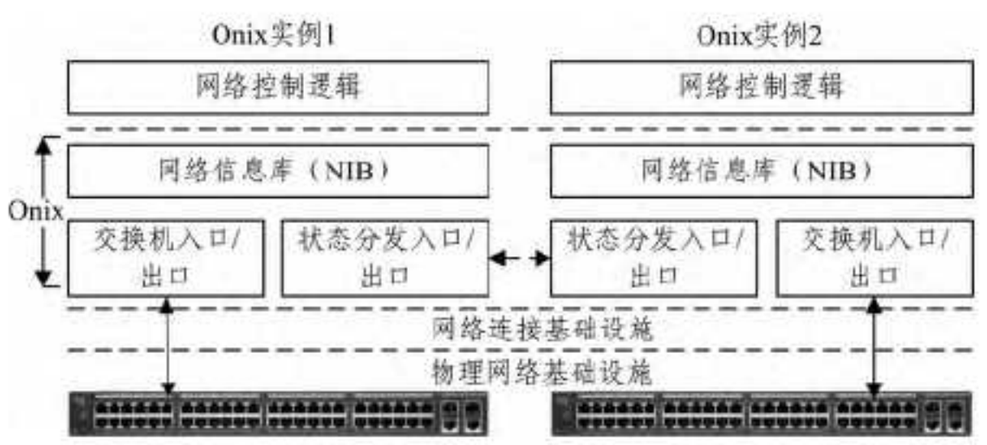

图 7 Onix 网络架构

为了实现控制层面的可扩展性，Onix 采用实例的方式来分区管理不同的域，每个实例中通过 NIB 来维护本分区的路由决策、配置数据平面。实例之间通过图 7 中所示的虚线箭头来协调和共享各自 NIB 中保存的网络状态信息（也就是 NIB 的分发），从而保证整个网络状态信息的一致性。Onix 提供了两种方式来实现 NIB 的分发：带复制的事务性数据库模式和分布式哈希表（Distributed Hash Table，DHT）模式[22]。前者的主要目标是提供一种可靠的分发机制，适用于网络事件更新缓慢、对稳定性和一致性要求较高的网络；后者通过通用的 API、状态触发器和坐标机制等 DHT 实现原理，构建快速响应的分布式拓扑结构，适用于网络事件更新频繁、对网络可用性要求较高的网络。但是在实际部署中，Onix 存在由于采用数据库模式带来的严重性能瓶颈，以及多个 Onix 实例对 DHT 更新可能导致的状态不一致等问题。

3.3.3 Kandoo

Yeganeh 等人在 2012 年的 HotSDN 大会上提出的 Kandoo 框架利用分层的思想实现控制平面的可扩展性，将控制器分为两层，针对不同类型的应用，采用不同层次的控制器进行控制。位于底层的控制器只负责处理它所管理的交换机的流请求，它们之间没有交互，同时底层的控制器可以根据网络流量规模大小进行扩展，顶层的中央控制器（一般只有一台控制器）则负责管理全局的网络拓扑和管理。通过将控制器进行分层管理，大部分的数据请求都在底层的控制器中处理完成，而不需要与顶层的中央控制器进行交互，从而极大地减少了顶层控制器的负载。底层控制器和顶层控制器均可以采取诸如 NOX、Beacon、Floodlight 这样的控制器实现，而为了做到更好的扩展性，顶层控制器也可以采用上述诸如 HyperFlow、Onix 等分布式控制器来实现。

3.3.4 Palette

SDN 的流表一般保存在三态内容寻址存储器中（Ternary Content Addressable Memory，TCAM），而 TCAM 非常耗电而且其容量具有一定的限制，导致 SDN 的流表只能保存有限数量的流表项（一般只有几百条），这无疑给 SDN 的性能带

来了瓶颈。因此，Y. Kanizo 等人在 2013 年的 INFOCOM 会议上提出了 Palette 框架，Palette 结合图论的思想和算法，将原来保存在入口 SDN 交换机上的流表平均分成若干等份的子流表（取决于有多少个入口 SDN 交换机）之后，再将各个子流表分发到网络中的所有 SDN 交换机，这样不仅能够平均整个网络中 SDN 交换机流表的使用大小，同时也能减少流表项的条数。

4 SDN 性能优化拟解决方案

实现高效的 SDN 架构部署需要解决系统可用性、控制平面可扩展性以及控制平面和数据平面之间的延迟问题。为了解决这些问题，本文拟采用两个层次来进行性能优化：1）通过改变交换机流表的设计实现智能流表管理；2）实现树状可伸缩的控制平面管理。

4.1 智能流表管理

通用的 SDN 架构中，控制器只收集 SDN 交换机的网络状态信息，不会收到具体的 SDN 交换机上的转发状态信息，智能流表管理的思路是：SDN 交换机向控制层面报告其数据转发的状态信息，控制层面根据 SDN 交换机反馈的数据转发的状态信息来进行决策，例如：SDN 交换机告知控制器某个〈源 IP 地址，目的 IP 地址〉对的转发时延很长，这说明可能存在路由环路或者交换机本身拥堵，这时控制器就可以通过分析 SDN 交换机反馈的具体数据来进行路径的最优化选择，从而动态地调整整个 SDN 的性能。

4.2 树状可伸缩的控制平面管理

目前存在多种控制平面扩展方案，如 3.3 节介绍的 HyperFlow、Onix 等，但这些方案注重的是对多控制器之间信息的同步，对如何实现多控制器的无冲突高效调度并未涉及，而这是非常重要的，直接决定了控制平面的性能。本文拟采用树形层次结构的方式来对控制器进行层次划分，如图 8 所示。越高层次的控制器的处理能力相对越强，低层的控制器处理能力有限时，就会将请求转发给上一层的控制器，位于根节点的控制器则能处理所有的请求，同时控制器可以根据实际需要很方便地进行扩充或者删减，从而实现了不管是横向还是纵向的可伸缩的控制平面管理，提高了 SDN 性能。

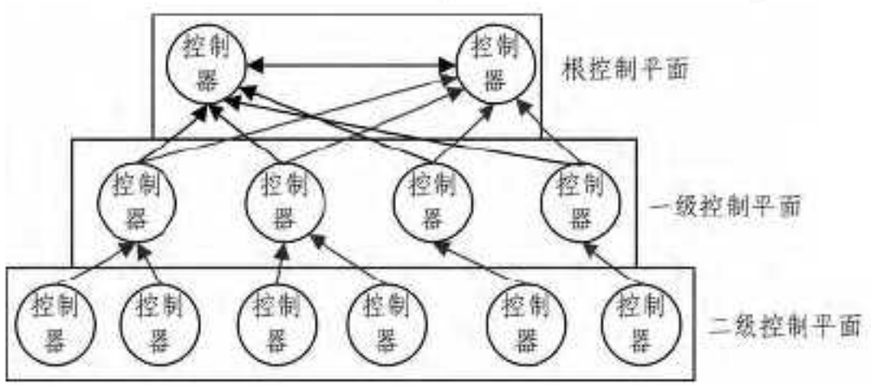

图 8 树状可伸缩的控制平面架构

结束语 SDN 采用控制和转发相分离的架构，使复杂的网络管理变得简单和方便，受到了学术界、工业界、运营商和互联网企业的广泛关注，随着云计算和大数据的兴起，SDN 在数据中心、校园网、企业网以及广域骨干网络中的应用部署也越来越多。在 SDN 快速发展的过程中，作为其主要组成部分的控制器也在不断地发展完善，从最初的单线程 NOX 到各种多线程控制器，以及近年来为了解决单个控制器性能问题的各种方案和实现控制平面高可扩展性的多控制器的解决方案。面对这么多的控制器实现方案，针对特定的应用场景以及特定的需求，哪种控制器实现更符合需求成为了学者以及业界需要考虑的重要问题。同时，结合应用的需要，如何对所选择的控制器实现方式进行性能优化，使其更高效、灵活地服务具体应用需求是未来对 SDN 应用部署需要考虑的问题。

通过本文的工作，能够让研究人员对 SDN 的起源与发展现状以及 SDN 的架构有一定的了解，同时，本文对 SDN 架构中最重要的控制平面部分的广泛调研工作，能够让研究人员对控制平面目前存在的问题、业内已有的解决方案以及未来 SDN 性能优化的研究趋势有一个详细的认识，从而为基于应用的控制器开发以及多控制器方案的设计提供了指导思路。

参 考 文 献

[1] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: Enabling Innovation in Campus Networks[C]// Proceedings of ACM SIGCOMM, 2008. Kyoto: ACM SIGCOMM, 2008: 69-74

[2] Jain S, Kumar A, Mandal S, et al. B4: Experience with a Globally-Deployed Software Defined WAN[C]// Proceedings of ACM SIGCOMM, 2013. HongKong: ACM SIGCOMM, 2013: 78-83

[3] NOX. [OL]. http://www.noxrepo.org/nox/about-nox/

[4] 雷葆华，王峰，王茜，等. SDN 核心技术剖析和实战指南[M]. 北京：电子工业出版社，2013：12-15

[5] Tootoonchian A, Gorbunov S, Ganjali Y, et al. On Controller Performance in Software-Defined Networks[C]// Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. 2012

[6] Curtis A, Mogul J, Tourrilhes J, et al. DevoFlow: Scaling Flow Management for High-Performance Networks[C]// Proceedings of ACM SIGCOMM, 2011. Toronto: ACM SIGCOMM, 2011

[7] Tootoonchian A, Ghobadi M, Ganjali Y, et al. OpenTM: traffic matrix estimator for OpenFlow networks[M]. Passive and Active Measurement Springer, 2010: 201-210

[8] Cai Z. Maestro: Achieving Scalability and Coordination in Centralized Network Control Plane[D]. Texas: Rice University, 2011

[9] Cai Z, Cox A L, Ng T E. Maestro: A System for Scalable OpenFlow Control[R]. Texas: Rice University, Technical Report TR10-08, 2010

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

[11] Jarschel M, Oechsner S, Schlosser D, et al. Modeling and performance evaluation of an OpenFlow architecture[C]// Proceedings of the 23rd International Teletraffic Congress. San Francisco, USA: ITCP, 2011: 1-7

[12] Hu Fei, Hao Qi, Bao Ke. A Survey on Software-Defined Network (SDN) and OpenFlow: From Concept to Implementation [J]. IEEE Communications Surveys & Tutorials, 2014, 16(4): 2181-2206

[13] Yu M, Rexford J, J M, et al. Scalable flow-based networking with DIFANE[C]// Proceedings of the ACM SIGCOMM 2010 Conference. New York, NY, USA: ACM, 2010: 351-362

[14] Curtis A R, Mogul J C, Tourrilhes J, et al. DevoFlow: Scaling Flow Management for High Performance Networks[J]. SIGCOMM Comput. Commun. Rev, 2011, 41(4): 254-265

[15] Curtis A, Kim W, Yalagandula P. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection[C]// Proceeding of INFOCOM 2011 Conference. Shanghai, China: IEEE, 2011: 1629-1637

[16] Luo Tie, Tan Hwee Pink, Quan P C, et al. Enhancing responsiveness and scalability for OpenFlow networks via control-message quenching[C]// 2012 International Conference on CT Convergence(ICTC). 2012:348-353

[17] Xia Weng-feng, Wen Yong-gang, Heng F C, et al. A Survey on Software-defined Networking[J]. IEEE Communications Surveys & Tutorials, 2014, 17(1):27-51

[18] Tootoochian A, Ganjali Y. HyperFlow: A distributed control plane for OpenFlow[C]// Proceeding of the 2010 Internet Network Management Workshop on Research on Enterprise Networking(INM/WREN) USENIX Association, 2010

[19] Koponen T, Casado M, Gude N, et al. Onix: A distributed control platform for large-scale production networks[C]// Proceeding of the 9th USENIX Conf. on Operating Systems Design and Implementation(OSDI). Vancouver: USENIX Association, 2010

[20] Yeganeh S H, Ganjali Y. Kandoo: a framework for efficient and scalable offloading of control applications[C]// Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ser. (HotSDN'12). New York, NY, USA: ACM, 2012: 19-24

[21] Kanizo Y, Hay D, Keslassy I. Palette: Distributing tables in software-defined networks[C]// Proceedings of 2013 INFOCOM. Turin, Italy: IEEE, 2013: 545-549

[22] 左青云, 陈鸣, 赵广松, 等. 基于 OpenFlow 的 SDN 技术研究[J]. 软件学报, 2013, 24(5): 1078-1097

[23] Voellmy A, Wang J. Scalable Software Defined Network Controllers[C]// Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. New York, NY, USA: ACM, 2012: 289-290

(上接第 237 页)

数目的增多,其比重也没有明显增加,这是因为按照空间距离由近及远的顺序查找。

表 1 两种方法计算量对比分析表

节点数量	经典方法主要步骤计算量比重(%)			本文方法主要步骤计算量比重(%)		
	点定位	优化处理	其他	点定位	优化处理	其他
1.0×10^4	20.5	40.3	39.2	6.1	45.3	48.6
1.0×10^6	33.9	32.6	33.5	7.0	43.2	49.8

为了进一步测试本文提出的方法处理大规模数据的性能,对某海域三月份和五月份的海水温度大量数据集分别进行温度场三维可视化仿真测试,并对温度场的三维可视化纵向截面进行切片展示,实验结果如图 2 所示。本文方法单元生成的平均速度可达到 12.9 万单元每秒。图中不同的颜色代表海水不同的温度,从中可以很清楚直观地了解和掌握整个海域内温度随深度变换和季节变化的详细情况。

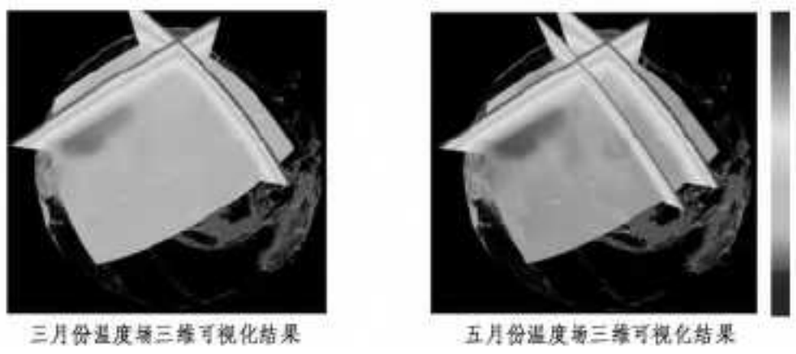

图 2 某海域不同月份温度场三维可视化仿真结果

结束语 本文对海洋环境信息三维可视化快速方法进行了研究,采用模糊 C 均值聚类对原始数据进行预处理得到需要的分类数据集,并提出一种改进的三维 Delaunay 算法快速构建三维曲面,最后采取颜色法对三维曲面进行可视化,并采用该方法对海水温度数据进行了三维可视化,使我们能够清楚直观地了解海水温度的变化情况,该方法同样可以推广到海水盐度场的三维可视化。需要说明的是,本文主要是研究大量数据的快速可视化问题,直接对原始数据提取等温面一定程度上会影响精度,但我们是以对大量数据进行处理为前提,保证了足够的数据密度。下一步将在此基础上进一步研究海水潮流、海流等矢量场的快速三维可视化问题。

参 考 文 献

[1] Liu J, Chen B, Chen Y. Boundary Recovery after 3D Delaunay Tetrahedralization without Adding Extra Nodes[J]. International Journal for Numerical Methods in Engineering, 2007, 72: 744-756

[2] Si H. Constrained Delaunay Tetrahedral Mesh Generation and Refinement [J]. Finite Elements in Analysis and Design, 2010, 46:33-46

[3] 李水乡, 陈斌, 赵亮, 等. 快速 Delaunay 逐点插入网格生成算法[J]. 北京大学学报, 2007, 43(3):302-306

[4] Thompson K E. Fast and Robust Delaunay Tessellation in Periodic Domains[J]. Int J Numer Methods Eng, 2002, 55(11):1345-1366

[5] Bowyer A. Computing Dirichlet tessellations[J]. The Computer Journal, 1981, 24(2):162-166

[6] Watson D F. Computing the N-dimensional Delaunay Tessellation with Application to Voronoi Polytopes[J]. The Computer Journal, 1981, 24(2):167-172

[7] Edelsbrunner H, Mucke E P. Three-dimensional Alpha Shapes [J]. ACM Transaction on Graphics, 1994, 13(1):43-72

[8] Chaine R. A Geometric Convection Approach of 3-D Reconstruction[C]// Eurographics Symposium on Geometry Processing. 2003

[9] 陈定造, 林奕新, 刘东峰. 三维 Delaunay 三角剖分快速点定位算法研究[J]. 计算机工程与科学, 2009, 31(5):79-80

[10] Escobar J M, Montenegro R. Several Aspects of Three-Dimensional Delaunay Triangulation [J]. Advances in Engineering Software, 1996, 27(1/2):27-39

[11] 李强. 三维 Delaunay 剖分在 3D GIS 中的应用[J]. 三晋测绘, 2002, 3(1):14-16

[12] 高新波, 谢维信. 模糊聚类理论发展及应用的研究进展[J]. 科学通报, 1999, 44(21):2241-2251

[13] 汤兵勇, 路林吉. 模糊控制理论与应用技术[M]. 清华大学出版社, 2002.9

[14] Zhu Q, Hu M Y, Zhang Y T, et al. Research and Practice in Three-dimensional City Modeling [J]. Geospatial Information Science, 2009, 12(1):18-24

[15] 詹芳芳, 胡伟, 袁国栋. 二维 LIC 矢量场可视化算法的研究及改进[J]. 计算机科学, 2013, 40(19):257-261

[16] Lorensen W E, Cline H E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm [J]. Computer Graphics, 1987, 21(4):163-169