软件定义网络控制平面的研究综述

柳 林[1,2] 周建涛[1]

(内蒙古大学计算机学院 呼和浩特 010021)[1] (内蒙古师范大学网络信息中心 呼和浩特 010022)[2]

摘 要 软件定义网络(Software-defined network,SDN)作为一种新兴的网络范式,通过解耦控制平面与数据转发平面,集中控制并且聚集全网视图,在控制平面与数据平面建立开放接口,启用外部应用使得网络具有可编程性,从而弥补当前网络架构所存在的不足与限制。其中,控制器作为SDN中重要的组成部分,成为了研究的热点。针对软件定义网络控制平面控制器的研究,首先总结了当前SDN控制平面控制器技术发展的现状并对其进行归类;其次着重分析了当前控制器存在的一致性、可扩展性、负载均衡等一系列问题;最后探讨了软件定义网络技术未来的研究发展方向及趋势。

关键词 软件定义网络,多控制器,一致性,扩展性,负载均衡

中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.02.009

Review for Research of Control Plane in Software-defined Network

LIU Lin[1,2] ZHOU Jian-tao[1]

(College of Computer Science, Inner Mogolia University, Hohhot 010021, China)[1]

(Network Information Center, Inner Mogolia Normal University, Hohhot 010022, China)[2]

Abstract Software-defined network(SDN) has emerged as a new networking paradigm by decoupling the control plane and data forwarding plane, and centralized controller and the global view of the network, established a open interface between the control plane and the data plane, and enabled programmability of the network by external applications. It maks up for the lacks and limitations of the current networking infrastructure. Thereinto, controller which is an important component in SDN has become a hot spot. This paper focused on the researches of Software-defined network control plane controller. Firstly, we summarized the current status of SDN control plane controller technology development and classification, and then we seriously analyzed the current existing consistency, scalability, load balancing, and other a series of issues in Software-defined network. Finally, we discussed the SDN technology research and development direction of the future and trend.

Keywords Software-defined network, Multi-controller, Consistency, Scalability, Load balancing

1 引言

如今的网络变得越来越复杂,现存的网络协议使得网络的运营及维护管理变得极为困难,而且许久未变的网络架构也制约着网络的持续性发展。为了解决这些问题,斯坦福大学的Nick McKeown教授于2008年4月首先提出了新型的基于软件的网络架构——软件定义网络,之后众多的设备厂商、计算机组织及运营商纷纷加入到了软件定义网络标准的制定中。2011年开放网络基金会(Open Networking Foundation,ONF)[1]成立,它是专门负责制定SDN接口标准的一个标准化组织。该组织制定的OpenFlow[2]协议已成为SDN南向接口的主流协议。除了ONF从用户角度出发定义SDN架构外,还有欧洲电信标准化协会[3](European Telecommunications Standards Institute,ETSI)从网络运营商的角度提出了网络功能虚拟化(Network Functions Virtualization,NFV)[4]架构,以及思科、IBM、微软公司联合推出的OpenDaylight[5]的开源SDN项目。

SDN最大的特点在于具有松耦合的控制平面和数据平面,支持集中化的网络控制,实现底层网络设备对上层应用的透明性。它具备灵活的软件编程能力,使得网络的自动化管理和控制能力获得了提升。其中,控制平面是整个网络架构的核心,扮演着至关重要的角色,它通过集中式控制使得复杂的网络管理变得更加容易。但是随着网络规模的不断扩大,集中控制可能出现的弊端越来越突出。例如集中控制的单点失效问题[6]、可扩展性问题[7-8]、一致性问题[9]、控制器负载均衡问题[10-11]等。所谓单点失效问题是指SDN控制平面由单一物理控制器组成,当单控制器出现宕机或其他问题时,整个网络系统势必瘫痪;扩展性问题就如何实现控制平面由多个

到稿日期:2015-12-01 返修日期:2016-03-02 本文受国家自然科学基金(61262082),内蒙古自治区云计算与软件工程科技创新团队资助。

柳 林(1980—),男,博士生,主要研究方向为软件定义网络、形式化描述技术方法,E-mail:liulin@imnu.edu.cn;周建涛(1974—),女,博士,教授,主要研究方向为云计算与软件工程,E-mail:cszjtao@imu.edu.cn(通信作者)。

控制器组成进行讨论;一致性问题则是研究多控制器之间如何协调使之具有数据或行为的一致性;而负载均衡问题是指基础设施层的交换机提交的处理任务如何合理地分配到不同的控制器中。为了解决 SDN 控制平面存在的这些问题,研究者已经做了广泛而深入的研究,并提出了很多方法及实现来解决 SDN 集中控制所遇到的问题。

本文是基于 SDN 控制平面的研究,试图全面介绍、总结和分析当前控制器在设计实现中所出现的问题。第 2 节介绍 SDN 体系结构;第 3 节对当前 SDN 控制器分类,并阐述研究现状及意义;第 4 节总结分析当前控制器出现的问题及已有的解决方案;最后总结全文并探讨未来的研究和发展趋势。

2 SDN 体系结构

SDN 的体系结构[12]最早由 ONF 提出。SDN 体系结构允许对处理行为和转发建模,支持各种介质和不同的连通方式,处理包括计算、存储和网络功能。网络功能和服务可以覆盖整个 OSI 七层模型,并且在物理资源和虚拟资源上都可以实现。ONF 将 SDN 划分为 3 层[13](见图 1),分别是基础设施平面、控制平面以及应用平面。

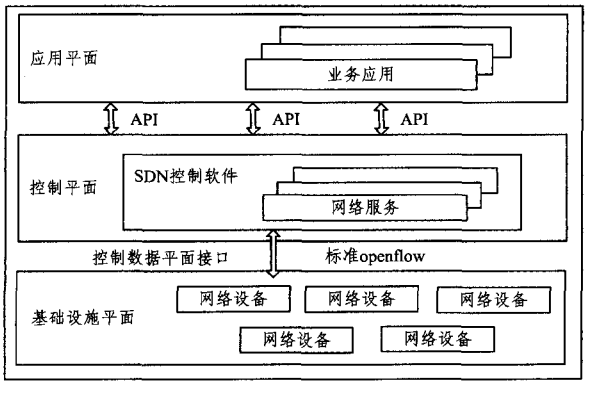

图 1 SDN 架构图

基础设施平面:也可以称为基础设施层或数据平面,由网络设备组成。这些网络设备可以是纯 SDN 交换设备或是兼有传统网络设备的数据转发能力及 SDN 转发功能的设备,这些网络设备基于控制器下发的策略建立的流表进行数据的处理、数据包的转发以及网络状态的收集。因此相对于传统网络设备来说,SDN 网络设备最大化了转发数据的性能,它只是接收控制器的指令来完成数据的转发任务。

控制平面:通常也可称为控制平层,由 SDN 控制系统实现,是整个架构的核心。控制平面向上为应用层提供编程接口,向下则管理和抽象基础设施平面中的网络设备。通过集中维护网络的全局视图,使得网络管理员能够使用 SDN 程序动态地、自动化地对网络资源进行灵活的配置、管理、优化和安全保证。

应用平面:也称为应用层,由应用业务程序组成。这些应用程序根据控制层提供的网络抽象信息执行策略,将策略的结果经控制器下发到基础设施层的网络设备中。应用层还能实现常见的网络服务,如路由、组播、安全、访问控制等,也可以通过定制来满足业务目标。

SDN 这种新型的网络架构将基础设施平面和控制平面相互分离,基础设施平面负责数据的转发,控制平面负责数据转发策略的制定与下发[14]。除了上述 3 层之外,控制平面与基础设施平面之间的接口称为南向接口,OpenFlow 协议就是一种开放的南向接口标准协议。应用平面与控制平面之间的接口称为北向接口,目前北向接口没有一个公认的开发协议,较多控制器使用 RESTful API 架构。而在控制平面内部,SDN 控制器之间的接口称为东西向接口,东西向接口是目前研究的热点问题。

在 SDN 网络中,控制平面性能对整个 SDN 网络的性能起着决定性的影响。到目前为止,不同的供应商、大学和研究机构使用不同的语言和不同的多线程技术,创建了超过 30 种 SDN 控制器。本文着重对常见的 SDN 控制器进行分析与研究。

3 SDN 控制器的分类

3.1 SDN 控制器的研究现状

现今使用的传统网络,控制层集成在网络设备中,并只提供端到端的连接。这种网络配置复杂,管理难度大,网络架构急于被改变,SDN 网络的出现使得连接方式变得更加方便快捷,它提出了可编程网络的概念,使得网络更易于管理和维护。ONF 提出的 OpenFlow 协议初步实现了 SDN 的控制平面与数据平面去耦的设计思想,推动了 SDN 技术的快速发展,简化了网络的管理,是当前 SDN 最成熟也是应用最广的实现方式。

但是,伴随着 SDN 应用范围的不断扩大,集中控制所承受的压力越来越大。据统计[10],一个 1500 台服务器集群每秒产生 100k 个请求,100 台交换机数据中心每秒产生 10000k 个请求。所以针对 SDN 的研究主要是基于控制器的研究,控制器的好与坏决定着 SDN 技术未来的发展。本节对当前的 SDN 控制器进行分类总结,具体分为 3 类(见图 2):单控制器控制平面、多控制器控制平面、混合型控制平面。

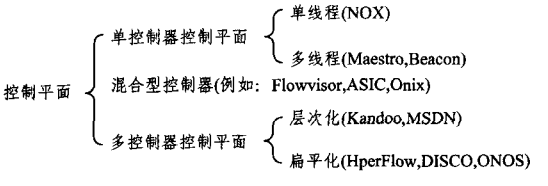

图 2 SDN 控制平面控制器分类图

3.2 单控制器控制平面

单控制器控制平面是在 SDN 技术建立的初期广泛使用的一种控制模式,是由一个物理控制器或一个物理控制器集群所组成的一个控制平面。对于单控制器所存在的单点失效以及处理器计算瓶颈等问题,已有的研究多是利用多线程的方法解决计算瓶颈问题。所以,单控制器可以分为单线程和多线程两种类型。

单线程:NOX[15]是较早提出并得到广泛应用的单线程单控制器,它提出了网络操作系统的概念,类似于读或写各种计算机操作系统所提供的资源,网络操作系统提供了控制和管理网络的能力。同时,NOX 使用 C++和 Python 两种程序

设计语言,实现了对 OpenFlow 网络的集中管理。NOX 使用了模块化的架构,应用层的每一个功能都是 NOX 的一个模块,模块之间可以通过底层消息组件通信。但由于部署的规模不断扩大,单控制器的处理能力不能满足全局网络的需求,在单控制的基础上,研究者提出了利用多线程技术处理不断增加的网络请求。

多线程:这些多控制器的典型代表是 Maestro[16] 和 Beacon[17]。Maestro 是用 Java 语言实现的一款基于 LGPL V2.1 开源协议标准的多线程控制器,主要应用于科研领域,具有很好的平台适应性,可以有效地在多种操作系统和体系结构上运行。除在为开发人员保留了简单的程序开发模型外,还利用了并行技术而且附加了优化吞吐率技术。而 Beacon 是一个快速的、跨平台的、模块化的、基于 Java 技术的 OpenFlow 控制器,它支持基于事件和线程的操作,也是多线程的控制器,具有稳定、跨平台、开源、动态、开发快速等特点,同时具有可定制、易于扩展的界面框架。Maestro 和 Beacon 采用多线程技术提高了控制器的性能,然而,在大规模数据中心网络中,一个单物理控制器不足以支撑整个网络的正常运行[18]。文献[18]通过 8 个实验测试了相应的单控制器的延迟、吞吐率等性能指标,并分析了控制器由于应用场景与实现语言的差异,各有优缺点;同时简单介绍了常见的多控制器,但没有对控制平面常见的问题进行进一步研究与分析。

3.3　多控制器控制平面

目前 SDN 网络的实现大多是在数据中心和校园网络这样的企业级网络,对于更大的广域网络和控制器与交换机之间过于频繁的访问与策略下发这样的应用场景,SDN 单控制器的开销过高,使之成为整个网络的瓶颈[19-20]。同时,单控制器存在单点失效的可能性,一旦控制器在性能或者安全性上不能得到有效保证,随之而来的就是整个 SDN 网络的服务能力下降甚至全网瘫痪。再者,单一控制器也无法满足跨域的部署要求,因此,需要多控制器部署解决以上单控制器所存在的各种问题。

多控制器控制平面是由物理上分布而逻辑上集中多个控制器的控制平面。它解决了单控制器单点失效的问题,同时在应对大规模网络的情况下,多控制器控制平面对于单控制器多线程技术体现出巨大的优势。多控制器控制平面按照结构又分为两类,即层次化的组织模式和扁平化的组织模式。

层次化的组织模式是按功能把控制器分成不同的层,底层控制器向下与交换机连接,向上与中央控制器连接,底层控制器之间没有连接。底层控制器处理本区域内交换机的请求事件;而中央控制器不直接接受交换机的请求事件,它负责底层控制器之间的通信。

层次化的组织模式的典型代表是 Kandoo[21],它是一个具有上下两层结构的控制器,底层是一组控制器,这些控制器之间没有连接,而且不知道彼此的网络现状;顶层是一个逻辑中央控制器,维持整个网络的状态。底层控制器仅仅维护附近路径的局部控制应用。本地的控制节点上,需要全局视图时,重定向到上层控制节点,上层控制节点负责全局视图的维护和网络应用程序的交互。这样减少了新流的请求数量,从而减轻了全局视图节点的负担,优化了性能,提高了网络的可扩展性。

扁平化组织模式按区域对整个网络进行划分,每个控制器负责本区域内交换机请求事件的处理,但每个控制器要求掌握全网视图信息。

扁平化组织模式的代表是 HyperFlow[22] 和 ONOS[23],HyperFlow 提出了一个基于 OpenFlow 的分布式控制平面。在其中,一个域同时运行着多个控制器,每一个控制器处理本区域内的 OpenFlow 交换机。为了使这些控制器获得全网信息,HyperFlow 采用一个分布式文件系统 WheelFS[24],这个文件系统是被设计在广域网环境下使用的。每一个 HyperFlow 控制器有权利在一定的区域内处理网络事件。一旦其它控制器学习到事件信息,它们将重新执行事件去完成全局视图的同步。开放网络操作系统(Open Network Operating System,ONOS)是由非营利性组织 ON. LAB 提出的开源的、分布式的 SDN 网络操作系统,它的设计主要面向服务提供商和企业骨干网,从而满足其对可靠性、灵活性的需求。ONOS 运行包含多个 ONOS 实例,每个实例都可以感知网络的一部分状态,多个实例之间需要共享网络信息,达到每个实例都能了解全局的网络视图。ONOS 采用了分布式哈希表 DHT 数据库的设计,并使用 Cassandra[25] 来保障分布式与可持续性。Cassandra 具有一致性存储的特点,能保障网络视图的最终一致性。

3.4　混合型控制平面

混合型控制平面是一种单控制器或多控制器都能部署实现的控制平面,Onix[26] 和 Flowvisor[27] 是其中典型的代表。

Onix 是另一个应用得比较广泛的分布式控制平台。它设计了能扩展控制应用的平台,最重要的贡献是从应用中抽象了分布式架构下的网络状态和任务,并提供了网络状态的逻辑视图即网络信息库(Network Information Base,NIB)。Onix 提供了一个通用的 API 控制应用,同时允许它们自己在一致性、持久性、扩展性之间做出权衡。

与上面所介绍的所有控制器不太一样,Flowvisor 采取的是一种新的虚拟交换方式,这种方式在同样的硬件转发平面能共享多个逻辑的网络,这些网络各自拥有独立的转发逻辑,对不同的管理对象进行切片,可以使用这个交换级别的虚拟化建立研究平台,并且允许多个并行的网络实验。

除了以上介绍的两个混合型控制平面外,不得不提的是 OpenDaylight 项目。OpenDaylight 是一个由 Linux Foundation 和多家网络巨头如 Cisco,Juniper 等一起创立的开源项目,其赞助商、发起者多为设备厂商而非运营商,其目的在于推出一个通用的 SDN 控制平台。OpenDaylight 不仅仅是一个 SDN 控制器,更是一个庞大的开源项目,其中包含许多子项目,Controller 只是其中的一个子项目。OpenDaylight 支持多种南向协议,包括 OpenFlow(支持 1.0 和 1.3 版本),Netconf 和 OVSDB 等,是一个广义的 SDN 控制平台,而不是 OpenFlow 系的狭义 SDN 控制器。OpenDaylight 的控制器由软件来实现,安放在 JVM 里,目前已经有 4 个版本,分别使用氢、氦、锂、铍 4 个化学元素符号命名。

另外，我国对 SDN 控制器的研究也取得了一定的成果。其中以清华大学的研究成果 ASIC[28] 最为突出，ASCI 也是一种混合型控制器，它是 2012 年由清华大学在 CFI 上提出的。它是一种解决控制平面扩展性的方法，主要解决初始状态流表生成时，大量的流请求信息流向控制器而造成控制器负载过大的问题。ASIC 实现的技术是负载均衡、并行处理、数据共享和集群等。ASIC 具体包括 3 个层次：负载均衡层、控制器集群、分布式数据存储。当一个数据包到来时，首先将数据包通过负载均衡选择到一个控制器来进行处理，控制器根据处理数据包将流表信息直接下发到相应的交换机，完成请求过程。

以上只是对当前 SDN 常见的控制器中的几个典型的有代表性的控制器进行分类说明，表 1 更加全面地对 SDN 常用控制器进行了汇总。

表 1 常见 SDN 控制器及其技术特征

控制器	控制器种类	单/多线程	开发语言	开/闭源	开发团队	其它
NOX	单	单	Python/C++	开	Nicira	最早实现的控制器，支持 OpenFlow 协议，有利于在 Linux 上进行快速的 C 控制器开发
NOX-MT	单	多	C++	开	Nicira	NOX 的升级版本，支持多线程
Beacon	单	多	Java	开	Stanford	跨平台，模块化，支持 OpenFlow 协议，支持基于事件和线程的操作
Floodlight	单	多	Java	开	BigSwitch	使用 OpenFlow 协议和 Apache 许可证的控制器
Maestro	单	多	Java	开	Rice	跨平台，可以加入新的应用程序进行扩展
OpenDayLight	单	多	Java	开	Cisco 等	遵循 OFS 提出的标准 SDN 架构，使用 Apache 许可证
MSDN	多	—	—	—	清华大学	一个大型的逻辑意义上的控制器，整个架构由负载均衡、控制器集群、分布式的共享系统组成
DISCO	多	—	—	—	Thales	开放的可扩展的分布式网络控制平台，处理分布式的、异构性的网络和广域网
ASIC	混合	—	—	—	清华大学	控制器隐藏在负载均衡器中，可以使用 NOX 或 Beacon。
Kandoo	多	单	C/C++/Java/Python	—	Toronto	具有高可配置和可扩展性，使用一个简单有效的方式去创建一个分布式控制平面
Ryu	单	—	Python	开	NTT	带有一个定义好的 API，旨在帮助程序员创建新的网络管理和控制应用
SNAC	单	单	Python/C++	闭	Nicira	集成可扩展的策略定义语言，通过策略管理器来管理网络
Onix	混合	单	Python/C++	闭	Nicira	利用分布式系统实现中央控制层，具有完善的较大规模真实网络的部署方案
ONOS	多	—	Java	开	ON. Lab	SDN 控制具有电信级特性的代表，保证网络敏捷性
HyperFlow	多	单	C++	—	Toronto	逻辑集中控制，物理上分布的控制平面，支持 OpenFlow 协议
FlowVisor	混合	单	—	开	ONF	一个网络虚拟化平台，可将物理网络划分成多个逻辑网络，部署在标准 OpenFlow 控制器与交换机之间

4 当前控制平面的研究问题及进展

伴随着 SDN 控制平面研究的逐步深入，控制平面的问题日渐凸显。本节分别就已有研究的一致性问题、扩展性问题及负载均衡的问题进行归纳总结，并提出整理这些问题研究中尚未解决的部分。

4.1 一致性问题

一致性问题是 SDN 的基本问题，很多问题都可以归结到该问题中。把 SDN 一致性问题分为以下 3 类进行讨论。

(1)SDN 控制逻辑的一致性问题。这方面的研究主要是在控制器与交换机之间由于传输而造成时间延误或是控制器规则下发的先后顺序所造成的一致性问题，这有可能使得网络出现包丢失、服务中断等现象。

DIFANE[29] 和 DevoFlow[30] 通过在 SDN 交换机端增加控制方法来减少上述问题的产生，这样基础层网络设备除转发功能外，加入控制功能的方法与 SDN 最初的设计宗旨相违背。文献[31]聚焦物理网络和 NIB(Network Information Base)在不同等级实施一致性转发状态。目前此类问题的研究仅仅限于单控制器场景，而没有考虑在分布式控制器场景下的应用。

(2)SDN 分布式控制器之间的一致性问题。分布式控制器之间的一致性问题主要研究的是控制器的状态一致性，即控制平面下，控制器的控制数据是否一致。

Onix[26] 提出了多控制器之间应该怎样交换全网信息来解决一致性问题，利用带复制的事务性数据库模式和 DHT(分布式哈希表)模式。前者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。后者则通过 API 等 DHT 机理来实现，构建快速响应的分布式场景，适用于更新频繁、对网络可用性要求较高的网络。但研究并未解决如何分区域部署多控制器的问题，也就是区域之间的控制如何通信而获得全网视图。

(3)并发策略导致一致性问题。并发策略同样会导致一致性问题，可由控制层将策略形成规则，并按两阶段提交方式解决。为了避免基础设施层过多参与，控制层可直接通过并发策略组合的方式来解决，并可利用细粒度锁(fine-grained locking)确保组合策略无冲突发生。

HTF[32] 采用了层次策略方案，它将并发策略分解并组织成树的形式，树的每个节点都可独立形成转发规则。HFT 首先对每个节点进行自定义冲突处理操作，这样整个冲突处理过程就转化成利用自定义冲突处理规则逆向搜索树的过程，从而解决了并发策略一致性问题。但是，研究没有解决当网络拓扑发生变化时策略配置也自动随之更改的问题。

4.2 可扩展性问题

目前，出现了很多解决 SDN 控制平面可扩展性[33-34] 问题

的方案,具体可以划分为分区域控制和集群方式这两种类别。

(1)分区域控制

分区域控制的主要思想就是将现有的网络划分为不同的区域,每个区域内部署一个或多个控制器,而这些控制器通过保证网络状态的一致性来实现对网络的统一管理。目前的研究主要集中于区域控制器之间的消息传递和数据的维护。部分控制器采取了分区域控制的方式,从而避免控制器所接受的交换机提供的请求数量过大。以下 3 种控制器采取了分区域控制的方法构建网络。

HyperFlow 是一种基于 NOX 的分布式控制方案,它将网络划分成多个逻辑区域,每个区域由一个或多个控制器来管理 OpenFlow 交换机,交换机和控制器之间的连接采用就近策略,控制器组件主要负责控制器状态修改时间的捕获及序列化、回放、事件的发送、向交换机发送命令等。事件发送系统持有全网视图,以发布-订阅模式来传输控制器节点之间的网络事件从而完成信息同步。

Onix 的扩展性通过 3 种策略实现。1)分区:控制逻辑通过配置 Onix,确保单个控制器实例在内存中仅有一个 NIB 子集用于数据更新;2)聚集:一个 Onix 实例与其他实例通信,是将一个实例本身的 NIB 数据信息进行整合,这样可以避免底层有过多的特性暴露于上层;3)稳定:控制逻辑负责管理网络状态信息的一致性,从而保证整个网络的稳定。

Kandoo 是一种基于分层思想的多控制器方案。控制器分为根控制器和本地控制器。底层的每个本地控制器负责对网络分区内的设备进行管理,并将分区网络拓扑等信息上交到根控制器,根控制器将分区信息进行聚合,从而实现网络视图的统一。

(2)集群方式

集群方式是指在没有对整个网络进行分区域的情况下提高控制器的控制能力。这种方式利用集群实现可扩展性是在物理机上分布而逻辑上集中的设备上进行的。

ASIC 采用集群方式进行扩展,控制器之间采用 JGroups 进行通信,集群控制器中的节点管理使用 IGMP,而数据缓存使用的是 Hazelcast。该架构中,集群节点都有相同的算法用于生产和更新网络。主要控制器节点负责全局网络视图中的控制器至交换设备的映射、维护和更新,主控制器的选取则通过 JGroups 提供的分布式原语实现,其他节点都需要对主控制器节点进行监听。一旦主节点发生异常,则会选取其他节点替换它。但是,这种方式只适用于一致性要求不高的 SDN 网络。在强一致性要求的网络中,目前还没有更好的方法。

4.3　负载均衡问题

负载均衡是一种服务器或网络设备普遍采用的技术。负载均衡将特定的网络服务和网络流量分担给多个服务器或网络设备,从而提高业务处理能力,保证业务的高可用性。软件定义网络中,负载均衡主要的应用场景是服务器负载均衡和链路负载均衡。

场景一:SDN 服务器负载均衡

负载均衡技术运用到新的网络架构中还面临一些新的问题,如负载均衡模块的设计、服务器运行状态的监控、如何保证负载均衡的灵活性。当 OpenFlow 交换机过多的新流请求集中于某一个控制节点时,如何分配这些请求?当前普遍采用的是实时最小控制器负载选择算法(Real-time Least loaded Server selection,RLS),但该算法只适用于小规模的、相同域的多控制器网络,而不适用大型的多域多控制器网络中。文献[35]采用基于方差的负载同步(Load Variance-based Synchronization,LVS)调度方法,提高了在 SDN 网络中多域多控制器负载均衡的性能。LVS 解决了转发回路和降低控制器同步花费的问题。当一台服务器或域的负载超出阈值时,LVS 表现出极好的控制器状态同步性。

场景二:SDN 链路负载均衡

当前 SDN 网络的链路策略大多采取的是最短路径算法,如文献[36]利用控制器计算链路的带宽和时延。确定最优路径的策略为,当前路径可用带宽最大且时延小于最大时延。文献[37]提出了特定流量的路由算法,该算法将时延和丢包最小的路径作为最优路径。这些路由算法虽然在性能的提升上取得了一定成果,但大都忽略了链路实时状态,仍然会使链路利用率不高或用户体验下降。文献[38]提出了 SDN 架构下一个基于链路的实时状态的负载均衡策略,其通过获取链路的负载情况,实时分析并为不同的业务选择当前负载最优路径,从而有效避免负载不均衡,提高网络资源的利用率。

结束语　SDN 目前已经得到广泛关注,不仅仅在学术上,各大设备厂商、网络运营商也已经开始大规模地投入研发。因此,SDN 技术的发展在带来更多的机遇的同时也带来了新的挑战。本文详细介绍 SDN 控制平面,并对其进行分类,同时总结现有关于控制平面所表现的突出问题并对其进行研究与分析,可以看到,SDN 网络虽然已经得到深入发展,但还面临着许多问题有待进一步研究。例如:

(1)SDN 网络能耗问题

目前,数据中心能耗问题日趋严峻,节能是当务之急。网络设备在整个数据中心能耗中占据着较大的比例,如何降低计算机网络设备的能耗已成为研究的热点。利用 SDN 技术整合数据中心网络设备,从而降低数据中心网络设备能耗,是未来研究的热点和有待解决的问题。

(2)SDN 在广域网络的应用问题

目前的 SDN 应用场景大都存在于高校、企业或者数据中心内部,在大规模的广域网络中的应用还有待开发。将 SDN 部署在广域网级别的规模中,需要考虑控制平面的部署问题、控制器瓶颈问题,以及网络的可靠性、节点的失效、全网视图的共享问题等。因此,SDN 在广域网的部署是有待解决的问题之一。

(3)SDN 控制平面东西协议

同一控制域和不同控制域间的通信协议目前还没有普遍认同的协议,控制器之间如何通信即东西向接口如何定义也是未来有待解决的问题。

参 考 文 献

[1] Open networking foundation(ONF)[OL]. https://www.opennetworking.org.

[2] OpenNetworking Foundation. OpenFlow SwitchSpecification, Version 1.3.4[S]. ONF,2012.

[3] European Telecommunications Standards Institute-(ETSI)[OL]. http://www.etsi.org.

[4] CHIOSI M, CLARKE D, WILLIS P, et al. Network functions virtua-lisation: An introduction, benefits, enablers, challenges and call for action[C]// SDN and OpenFlow World Congress. Darmstadt, Germany, 2012: 22-24.

[5] OpenDayLight(ODL)[OL]. [2016-03-10]. https://www.opendaylight.org.

[6] ZHANG C K, CUI Y, TANG H Y, et al. State-of-the-art survey on software-defined networking (SDN)[J]. Journal of Software, 2015, 26(1): 62-81. (in Chinese)
张朝昆,崔勇,唐翯祎,等. 软件定义网络(SDN)研究进展[J]. 软件学报, 2015, 26(1): 62-81.

[7] LI C, DUAN X D, CHEN W, et al. Thoughts and Practices About SDN and NFV[J]. Telecommunications Science, 2014, 30(8): 23-27. (in Chinese)
李晨,段晓东,陈炜,等. SDN 和 NFV 的思考与实践[J]. 电信科学, 2014, 30(8): 23-27.

[8] MCCAULEY J, PANDA A, CASADO M, et al. Extending SDN to large-scale networks[C]// Open Networking Summit. 2013: 1-2.

[9] ZUO Q Y, CHEN M, ZHAO G S, et al. Research on OpenFlow-Based SDN Technologies [J]. Journal of Software, 2013, 24(5): 1078-1097. (in Chinese)
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.

[10] LIN P P, BI J, HU H Y, et al. MSDN: a mechanism for scalable intra-domain control plane in SDN[J]. Journal of Chinese Computer Systems, 2013, 34(9): 17-20. (in Chinese)
林萍萍,毕军,胡虹雨,等. 一种面向 SDN 域内控制平面可扩展性的机制[J]. 小型微型计算机系统, 2013, 34(9): 17-20.

[11] WEI K. The Load Balancing Research of SDN Based on Ant Colony Algorithm[D]. Jilin: Jilin University, 2015. (in Chinese)
魏凯. 基于蚁群算法 SDN 负载均衡的研究[D]. 吉林: 吉林大学, 2015.

[12] SDN Architecture [OL]. https://www.opennetworking.org/images/stories/downloads/ssd-resources/technicalreports/TR_SDN_ARCH_1.0_06062006.pdf.

[13] FOUNDATION O N. Software-Defined Networking: The New Norm for Networks[OL]. http://bigswitch.com/sites/default/files/sdn-resources/onf-whitepaper.pdf.

[14] 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.

[15] GUDE N, KOPONEN T, PETTIT J, et al. NOX: towards an operating system for networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(3): 105-110.

[16] CAI Z. Design and implementation of the maestro network control platform[M]. ProQuest, 2009.

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

[18] JIANG G L, FU B Z, CHEN M Y, et al. Survey and Quantitative Analysis of SDN Controllers[J]. Journal of Frontiers of Computer Science & Technology, 2014, 8(6): 653-664. (in Chinese)
江国龙,付斌章,陈明宇,等. SDN 控制器的调研和量化分析[J]. 计算机科学与探索, 2014, 8(6): 653-664.

[19] REITBLATT M, FOSTER N, REXFORD J, et al. Abstractions for network update[C]// Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. Helsinki, Finland, ACM, 2012: 323-334.

[20] CANINI M, KUZNETSOV P, LEVIN D, et al. Software transactional networking: Concurrent and consistent policy composition[C]// Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. Hong Kong, China, ACM, 2013: 1-6.

[21] HASSAS YEGANEH S, 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. Helsinki, Finland, ACM, 2012: 19-24.

[22] TOOTOONCHIAN A, GANJALI Y. HyperFlow: A distributed control plane for OpenFlow[C]// Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking. San Jose, USA, 2010: 3.

[23] The Open Network Operating System(ONOS)[EB/OL]. http://onosproject.org.

[24] STRIBLING J, SOVRAN Y, ZHANG I, et al. Flexible, Wide-Area Storage for Distributed Systems with WheelFS[C]// Networked Systems Design and Implementation (NSDI). Boston, USA, 2009: 43-58.

[25] Cassandra [EB/OL]. [2016-01-17]. http://cassandra.apache.org.

[26] KOPONEN T, CASADO M, GUDE N, et al. Onix: A Distributed Control Platform for Large-scale Production Networks[C]// Operating Systems Design and Implementation (OSDI). Vancouver, Canada, 2010: 1-6.

[27] SHERWOOD R, GIBB G, YAP K K, et al. Can the production network be the testbed? [C]// Operating Systems Design and Implementation (OSDI). Vancouver, Canada, 2010: 1-6.

[28] LIN P, BI J, HU H. Asic: an architecture for scalable intra-domain control in openflow[C]// Proceedings of the 7th International Conference on Future Internet Technologies. Seoul, Republic of Korea, ACM, 2012: 21-26.

[29] 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.

[30] CURTIS A R, MOGUL J C, TOURRILHES J, et al. DevoFlow: scaling flow management for high-performance networks[J]. ACMSIGCOMM Computer Communication Review, Toronto, Canada, ACM, 2011, 41(4): 254-265.

[31] REITBLATT M, FOSTER N, REXFORD J, et al. Consistent updates for software-defined networks: Change you can believe in! [C]// Proceedings of the 10th ACM Workshop on Hot Topics in Networks. ACM, 2011.

[32] FERGUSON A D, GUHA A, LIANG C, et al. Hierarchical policies for software defined networks[C]// Proceedings of the First Workshop on Hot Topics in Software Defined Networks. Helsinki, Finland, ACM, 2012: 37-42.

[33] KYUNG Y, HONG K, NGUYEN T M, et al. A load distribution scheme over multiple controllers for scalable SDN[C]// IEEE Computer Society. Taichung, Taiwan, 2015.

[34] JIMENEZ Y,CERVELLO-PASTOR C,Garcia A J. On the controller placement for designing a distributed SDN control layer [C]// Networking Conference,2014 IFIP. Trondheim,Norway, IEEE,2014:1-9.

[35] GUO Z,SU M,XU Y,et al. Improving the performance of load balancing in software-defined networks through load variance-based synchronization[J]. Computer Networks,2014,68:95-109.

[36] CUI H,ZHU Y,YAO Y,et al. Design of intelligent capabilities in SDN[C]// 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE). Shenzhen, Guangdong,China,IEEE,2014:1-5.

[37] CIVANLAR S,PARLAKISIK M,TEKALP A M,et al. A qos-enabled openflow environment for scalable video streaming[C]// GLOBECOM Workshops (GC Wkshps), 2010 IEEE. Miami, Florida,USA,IEEE,2010:351-356.

[38] CAO B,LIU S,SUN Q. Dynamically adaptive load balancing strategy under the software defined network structure[J]. Journal of Chongqing University of Posts & Telecommunications, 2015,27(4):460-465. (in Chinese)
曹侯,刘趣,孙奇. 软件定义网络架构下的动态自适应负载均衡策略研究[J]. 重庆邮电大学学报(自然科学版),2015,27(4):460-465.

(上接第 74 页)

5.2 性能测试

为了测试本文优化方法对自动向量化程序在性能提升方面的作用,选择 SPEC2006 测试集中的 3 个程序作为测试用例,包括 433. milc,456. hmmer 和 462. libquantum。针对这 3 个测试用例分别生成多种优化版本(各版本对应的优化组合为:A、"串行版本",B、"激进循环分布",C、"激进循环分布+向量化",D、"循环聚合优化+向量化"),并测试各版本相对于串行源程序的加速比,加速比越高说明性能提升越明显,测试结果如图 6 所示。结果表明:

1)仅仅对程序进行激进式循环分布(B 类优化)将降低程序的性能,说明激进式循环分布确实引入了过多开销。

2)如果对程序施加 C 类优化,程序的性能较 A 类优化有明显提升,说明向量化提高了程序的性能,但性能仍存在提升空间。

3)对程序施加 D 类优化,即本文提出的优化方法,其效果是上述优化结果中最好的,456. hmmer 的加速比超过了 1. 5。

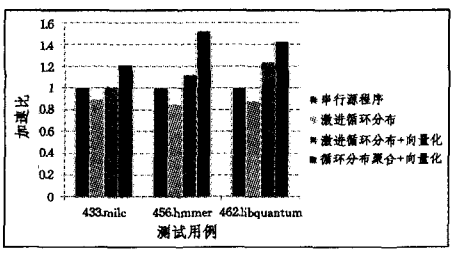

图 6　测试用例多种优化版本的加速比对比

结束语　循环分布是实现部分向量化的有效手段,但目前主流编译器中循环分布过于简单激进,尤其针对 SIMD 短向量部件,难以得到高效的向量化代码。本文提出了面向部分向量化的循环分布及聚合方法,其目的是在保证向量并行性的条件下,通过循环聚合来减小循环开销并提高向量代码的执行效率。实验结果表明,此方法是有效的。然而,本方法目前并不完善,循环聚合显然为数据重用带来了更大的优化空间,怎样在减小循环开销的同时,最大限度地提高寄存器和 cache 的重用也是值得进一步研究的问题。因此下一步工作中将针对此问题展开研究,在循环聚合的基础上实施更有效的数据重用优化。

参 考 文 献

[1] KENNEDY K,MCKINLEY K S. Loop distribution with arbitrary control flow[C]// Proceedings of Supercomputing'90. IEEE,1990:407-416.

[2] MCKINLEY ,KEN K,KATHRYN S. Maximizing Loop Parallelism and Improving Data Locality via Loop Fusion and Distribution[M]// Languages and Compilers for Parallel Computing. Springer Berlin Heidelberg,1997:301-320.

[3] LARSEN S,AMARASINGHE S. Exploiting superword level parallelism with multimedia instruction sets[C]// Proceedings of the SIGPLAN'00 Conference on Programming Language Design and Implementation,2000:145-156.

[4] PARK Y,SEO S,PRAK H,et al. Simd defragmenter:efficient ilp realization on data-parallel architectures[J]. ACM SIGARCH Computer Architecture News. ACM,2012,40(1):363-374.

[5] BARIK R,ZHAO J,SARKAR V. Efficient selection of vector instructions using dynamic programming[C]// 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE,2010:201-212.

[6] KIM S,HAN H. Efficient SIMD code generation for irregular kernels[J]. ACM Sigplan Notices,2012,47(8):55-64.

[7] LIU J,ZHANG Y,JANG O,et al. A compiler framework for extracting superword level parallelism[J]. ACM Sigplan Notices,2012,47(6):347-357.

[8] RAMANARAYANAN R,GUPTA M,CHAKRABORTY S S, et al. Harnessing partial vectorization in Open64 compiler[C]// 2014 IEEE International Advance Computing Conference (IACC). IEEE,2014:813-824.

[9] ALLEN R,KENNEDY K. Optimizing compilers for modern architectures:a dependence-based approach[M]. San Francisco: Morgan Kaufmann,2002.

[10] GCC Team. Gcc,the gnu compiler collection[OL]. http://gcc.gnu.org.

[11] Intel Corporation. Intel® C and C++ Compilers[OL]. https://software.intel.com/en-us/intel-compilers.

[12] Open64. Overview of the open 6 4 Compiler Infrastructure [EB/OL]. http://open64.sourceforge.net.

[13] CHEN K H,Shen B Y,Yang W. An automatic superword vectorization in LLVM[C]// 16th Workshop on Compiler Techniques for High-Performance and Embedded Computing. 2010: 19-27.