SDN 性能优化技术研究综述

孙 涛[1,2] 张俊星[1]

(内蒙古大学计算机学院 呼和浩特 010021)[1] (内蒙古科技大学信息工程学院 内蒙古 包头 014000)[2]

摘 要 软件定义网络(Software-Defined Network,SDN)是一种新兴的网络架构,完全解耦了数据平面与控制平面。控制平面集中制定并下发全网决策,数据平面单纯负责数据转发。通过控制平面的开放接口,SDN 实现了网络的可编程性。在未来 SDN 大面积部署应用的过程中,各个平面的性能优化技术将面临诸多挑战。首先,分析了 SDN 架构中控制平面和数据平面的性能优化技术的发展现状。其次,总结了各平面性能优化过程中所面临的问题。最后,展望了 SDN 性能优化方面的未来研究趋势。

关键词 软件定义网络,控制平面,数据平面,性能优化技术

中图法分类号 TP393 **文献标识码** A

Review of SDN Performance Optimization Technology

SUN Tao[1,2] ZHANG Jun-xing[1]

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

(Institute of Information Engineering,Inner Mongolia University of Science and Technology,Baotou,Inner Mongolia 014000,China)[2]

Abstract Software-Defined Network (SDN) is an emerging network architecture. It decouples data plane and completely of control plane. Control plane focuses on making and issuing whole network decision. The data plane is solely responsible for data forwarding. Through the open interface of control plane,SDN achieves network programmability. In the future,when SDN is widely deployed in a wide area,every plane's performance optimization technologies will face many challenges. Firstly,the status quo of the performance optimization of control plane and data plane in SDN architecture was analyzed. Secondly,the problems faced in the process of optimizing the performance of each plane were summarized. Finally,the future research trends of SDN performance optimization was prospected.

Keywords Software define network,Control plane,Data plane,Performance optimization technology

1 引言

目前,互联网的规模和流量已经大大超出了最初仅用于军事的端到端数据传输的设计目的。无论从软件协议的标准化进程来看,还是从封闭的、复杂的和多样化的硬件网络元件来讲,互联网的日益复杂性和难以管控的现状,已经明显成为了制约互联网创新发展的瓶颈问题。

有学者提出将网络的控制与转发解耦,开放网络的可编程能力,从根本上重组传统的网络架构,将是解决上述瓶颈问题的有效措施,其显著优势是最大程度地简化了网络的控制和管理过程,降低了网络运营的维护成本。2005 年,Greenberg 等[1]对互联网的控制和管理结构进行了全新的设计,提出了 4D 框架,该框架将网络控制决策逻辑彻底从管理网络元件交互的协议中分离出来,初步实现了对网络的灵活管控。

2006 年和 2007 年,美国斯坦福大学的研究人员 Casado 等提出了 SANE[2](Secure Architecture for the Networked Enterprise)和 Ethane[3]两种面向企业网的管理架构。SANE 以数控分离为设计原则,将安全性能作为最根本的设计目标。Ethane 项目实现了基于流表的转发策略。在此基础上,美国斯坦福大学 Mckeown 教授于 2008 年首次提出了 OpenFlow 的概念,并在 SIGCOMM 会议上发表文章"OpenFlow:Enabling Innovation in Campus Networks",详细说明了 OpenFlow 协议的工作原理以及应用场景。2009 年,该研究团队进一步提出了 SDN[4]的概念,该概念引起了业界的广泛关注。2011 年初,Google,Facebook,Yahoo 等公司共同成立了开放网络基金会[5](Open Networking Foundation,ONF),并正式提出了软件定义网络 SDN 的概念。

SDN 目前已被部署应用于一些新兴的网络环境,解决了传统网络架构难以克服的一些困难。例如,数据中心是 SDN 最早的应用场景。利用 SDN 的集中管控和网络可编程的特性,大幅度降低了数据中心的运维成本和网络设备的复杂度,使数据中心的网络性能得到了大幅提升。物联网中类型众多的网络设备的接入和网络协议的采用,使物联网的管理和控制也日益复杂。利用 SDN 实现物联网管理的灵活性和可编程性,无疑是目前最优的解决方案。SDN 的虚拟化分片技术既提高了虚拟网络资源的利用效率,又实现了网络分片的隔离,使虚拟网络分片的安全性得到了大幅提升。最新的移动通信网络 5G 也利用 SDN 来降低运营成本,简化网络管理,

本文受国家自然科学基金项目(61261019),内蒙古自治区自然科学基金(优青培育)项目(2011JQ05)资助。

孙 涛(1977—),女,博士生,副教授,主要研究方向为软件定义网络、网络实验床,E-mail:sunta771031@imust.cn;张俊星(1974—),男,博士,教授,主要研究方向为无线网络、网络实验床,E-mail:junxing@imu.edu.cn(通信作者)。

开放无线网络编程接口,实现虚拟网络的分片隔离,引领了无线网络的发展趋势。

SDN 是面向下一代互联网的新生事物,其研究和应用均处于初级阶段。在未来 SDN 大面积部署的过程中,其架构中各个平面的性能优化将是必然要面临的一类重要问题。控制平面是整个 SDN 体系结构的核心平面,其时延、带宽和吞吐量决定着整个 SDN 网络的性能,降低控制平面的时延、提升控制平面的带宽和吞吐量将是优化控制平面性能的核心问题。数据平面负责全网的数据转发,降低设备复杂度、提高数据转发效率是数据平面优化环节的专注焦点。

本文基于 SDN 性能优化技术的研究,试图全面地介绍、总结和分析 SDN 性能优化技术的发展现状和面临的问题。第 2 节介绍 SDN 的体系结构;第 3 节总结了目前控制平面的优化技术,分析了在控制平面性能优化过程中面临的问题和已有的解决方案;第 4 节阐述了现有数据平面的数据转发性能优化技术和面临的问题;最后总结了全文,并探讨了未来 SDN 性能优化研究领域的发展趋势。

2　SDN 的体系结构

网络可编程、控制逻辑集中化和控制转发分离是 SDN 的三大特点。SDN 实现了网络的自动化管理和控制,大幅降低了网络运行和维护的成本。ONF 提出的 SDN 体系架构由上至下分为应用层、控制层和数据层[6],如图 1 所示。其中,应用层包含各种网络业务应用。网络用户无需了解网络底层的技术细节就可以通过编程来实现和部署多样化的网络应用。控制层由负责产生全网控制逻辑的控制器组成。控制器维护全网状态信息,生成全网视图,根据全网视图制定并下发数据转发规则。数据层由若干交换机等网络元件组成,负责数据的处理、转发和状态的搜集。除此之外,SDN 架构还包括应用层与控制层之间的北向接口和控制层与数据层之间的南向接口。北向接口是为开发者提供网络高层逻辑抽象和业务模型的编程接口,实现了顶层应用与网络之间的信息交换。南向接口负责控制逻辑下发和网络状态的采集。

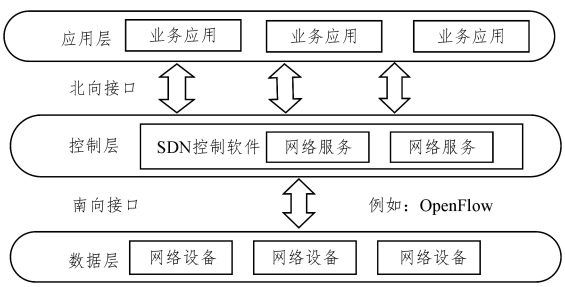

图 1　ONF 提出的 SDN 架构

3　控制平面的性能优化技术

控制平面是整个 SDN 体系结构的控制中心,负责维护全网状态信息,生成全网视图,根据全网视图制定并下发数据转发规则。控制平面的性能对整个 SDN 网络的性能有着决定性的影响。然而,基于数据平面转发设备数量增多的发展趋势以及流控制力度细化和控制平面的管控功能复杂化的现状,控制平面的性能优化已经势在必行。

通常,时延、带宽和吞吐量是评价网络性能的关键指标。对于 SDN 的控制平面而言,时延表示控制平面成功响应一次流请求事件所花费的时间;带宽表示控制平面与数据平面之间的控制通道的容量,直接决定了控制平面可以连接的交换机的数目,影响着整个 SDN 网络的可扩展性。吞吐量是指控制器每秒钟能够处理的流请求数量。下面将从上述 3 个方面对 SDN 控制平面的性能优化技术进行分析和总结。

3.1　单控制器多线程技术

随着 SDN 网络部署规模的扩大,控制平面经历了由单一控制器平面到分布式控制器平面发展的过程。在单一控制器平面的情况下,多线程技术是降低控制平面时延和提高吞吐量的有效方法。

NOX 控制器采用单线程事件循环架构,每秒钟可以处理 30 k 的流初始化事件,同时需要 10 ms 的流安装时间[7]。NOX-MT[8]利用传统的多线程处理技术仅在 NOX 控制器上做了轻微的修改,将 NOX 控制器的性能提升了 30 倍。

Maestro[9]为编程人员保留了简单的单线程编程模式,却能够提供管理并行机制的服务。它将并行机制和传统的吞吐量优化技术并用,充分发挥了高性能服务器的多核并行处理能力,实现了控制器性能的大幅提升。文献[10]中通过实验验证得出结论:多线程控制器在线程支持上基本可以做到吞吐量的线性增长。

Trema[11],Ryu NOS[12]等控制器也都通过多线程技术实现了控制器性能的提升,表 1 给出了简要特征的介绍。

表 1　典型集中式多线程控制器平台的简介

控制器名称	开发语言	开/闭源	开发团队	特点简介
Beacon[13]	Java	开	Stanford	跨平台、模块化,支持基于事件和线程的操作
Floodlight[14]	Java	开	BigSwitch	与 Big network controller 完全兼容,也可应用于 OpenStack
Maestro	Java	开	Rice	平台适应性好,可以有效在多种操作系统和体系结构的机器上运行
Meridian[15]	Java	开	IBM	支持云平台网络应用的服务模块
Mul[16]	C	开	Opendaylight Project	高可靠性,通过模块化应用插件接口实现高性能服务
NOX-MT	C++	开	Nicira	NOX 的升级版本
Ryu NOS	Python	开	NTT	带有一个定义好的 API,可帮助程序员创建网络管理和控制应用
Trema	C,Ruby	开	Stanford	全特性、已使用的 OpenFlow 控制器框架

然而,单一控制器性能的提升已经不足以应对网络规模的扩大,有研究人员已经通过实验证明了随着数据平面交换机数量和终端数量的增加,控制器的吞吐量会呈现下降趋势[17]。同样地,在有限的带宽条件下,控制平面的时延也会随之而大幅下降。另外,在分域的较大规模网络中,单一的集中式控制器与不同域的交换机之间会存在较大的时延,而且随着网络规模的扩大,这种时延也会增加,这些情况都会严重影响控制层面的性能,甚至不可容忍。

同时,基于控制器在整个 SDN 架构中的核心作用,控制器极易成为网络恶意攻击的焦点,即便在有效的安全防范措施下,控制器也会出现不可避免的自然故障问题。因此,单点失效问题对于整个 SDN 网络来说是一个十分严重的安全隐患。

3.2 分布式控制平面部署技术

分布式控制平面部署增加了控制平面中控制器的数量，通过多控制器的协同工作，可以大幅提升控制平面的带宽和吞吐量，同时降低控制平面的时延。目前，SDN 网络架构中分布式控制平面一般可分为平面式和层级式两种格局。平面式格局是指所有控制器都部署在同一层次的不同域中，每一个控制器都拥有全网状态信息，并实现网络状态的同步更新。层级式格局是指将控制器分为局部控制器和全局控制器两类。全局控制器负责全网信息维护，局部控制器仅掌握区域的网络状态，负责所在区域的交换机的信息交互。

HyperFlow[18]是逻辑集中而物理分布式的 SDN 控制平面，是平面式控制平面的代表，它在保留集中式控制平面的同时，实现了控制平面的可扩展性。HyperFlow 是基于 NOX 控制器的应用程序，仅对 NOX 做了较少的改动，通过部署多台控制器来管理 OpenFlow 交换机，控制器之间需要同步全网络视图，每台控制器只需要管理特定域的交换机。HyperFlow 适用于网络状态事件更新频率低于 1000 次/s 的网络。对于网络状态事件更新频率较高的网络，HyperFlow 可能出现性能瓶颈。

Kandoo[19]是典型的层级式控制平面。Kandoo 控制平面共分为两层，位于底层的控制器之间并不互联。每一个本地控制器拥有各自所在区域的网络视图，负责管理分区内的网络设备。顶层是一个逻辑集中的控制器，它将本地控制器上交的分区网络信息进行汇总，生成全网视图。Kandoo 在不修改数据平面的前提条件下实现了良好的可扩展性。实验评估显示，Kandoo 大幅度线性降低了控制平面的带宽消耗。

下面对目前分布式控制平面部署过程中所面临的问题进行总结和归纳。

(1)负载均衡问题

SDN 分布式控制平面部署容易出现局部过载现象，影响了流请求的处理速度，降低了系统的运行效率。控制器负载均衡问题是影响分布式控制平面性能的关键问题之一。

目前，控制平面的负载均衡策略研究主要有以下 3 种方案。

1)借鉴传统的服务器集群负载均衡策略。控制器集群动态负载均衡策略是目前解决多控制器平面负载均衡问题的一种有效方法。文献[20]提出了一种基于 OpenFlow 的虚拟化动态负载均衡策略，控制器通过虚拟机管理机制搜集服务器载荷状态，然后根据载荷均衡调度算法计算服务器总载荷量，交换机将会给载荷最低的服务器发送客户请求，实现服务器处理时间的降低和吞吐量的提高。

2)通过流量重定向方式实现控制平面的负载均衡。BalanceFlow[21]是通过在交换机包含 CONTROLLER X 行为的流表项将流请求定位给载荷量小的控制器，实现流量重定向，减小过载控制器的负载。

3)交换机动态迁移方案。控制器负载均衡问题产生的根本原因是由控制器与交换机之间的静态配置关系导致的。如何根据全网运行的实时状态需要，实现控制器与交换机之间的动态配置，是解决负载均衡的核心问题。ElastiCon[22]是一种弹性控制器分布结构，在这种结构中，控制器池会根据流量情况进行动态的增加或删减，交换机迁移协议实现流量载荷在控制器之间的动态转移。ElastiCon 通过以下两种机制来实现控制器的负载均衡。首先，通过控制器载荷状态的测量和检测算法定期地进行控制器的载荷均衡，实现交换机到控制器之间的映射机制优化；其次，如果目前的载荷量超过控制器的最大处理能力，那么会通过增加控制器数量来扩大资源池，触发交换机迁移事件，实现交换机对新加入控制器的利用。同理，当实际载荷量下降时，会相应地缩减资源池。

以上 3 种方法是目前分布式控制平面负载均衡问题的主要解决办法。但是，控制器集群负载均衡策略通过负载均衡器把任务均衡地分配到控制平面的每一个控制器，但负载均衡器的性能极限对控制平面的扩展会有一定的限制性。流量重定向方法会占用交换机中 TCAM 的资源。相比之下，交换机的动态迁移策略较为合理。

(2)控制器的容量问题和位置部署问题

分布式控制平面的部署还会带来一种开放性的问题，即控制器容量问题和位置部署问题。控制器容量和位置都会给全网的平均时延带来极大的影响[23]。控制器容量是指根据 SDN 应用场景对网络性能的实际需求而计算出的所需控制器数量。控制器容量是部署和管理 SDN 的前提和基础。

目前，控制器网络容量的研究大都以网络测试为基础，比较通用的方法是排队论和网络微积分。排队论的方法主要是得到网络的平均性能，而网络微积分则强调网络的临界性能[24]。控制器部署问题是指对于给定一个 SDN 网络拓扑，控制器应该部署在哪个特定的位置以达到最优的网络性能目标，包括控制平面的时延、带宽、吞吐量以及 SDN 网络的可靠性。

目前，控制器位置部署问题的研究可以分为静态部署方案研究和动态管理方案研究两个方向。静态部署方案是在 SDN 网络拓扑规划时期进行的，根据网络流量和性能需求的预先估算，依据优化算法确定控制器的位置。控制器的位置一旦选定，将固定不变。

文献[25]从控制器位置出发定义了可扩展性控制平面的设计法则。研究者使用 K-critica 算法计算出了控制器数量的最小值和每一个控制器的定位，实现了具有强鲁棒性的控制平面拓扑的构建。文献[26]中，通过简单的 SDN 网络拓扑实例说明了控制器的数量和定位还会影响控制平面的可靠性。

以上这两种静态的 SDN 网络拓扑部署方案无法适应网络状态的实时变化，很难实现控制平面的持续性能优化。

动态管理方案可通过交换机的迁移来调整各域之间的控制器和交换机的映射关系，从而实现对网络的动态管理和控制平面的持续性能优化。文献[27]提出了一种基于 SDN 广域网的多控制器动态部署方案，根据网络的实时状态来确定活跃和休眠的控制器数量和定位。实验评估结果显示，该方法有效缩短了控制平面的响应时间。

由于控制器容量和位置部署问题通常会涉及到控制平面的可扩展性、时延等多个目标的优化，而这些优化目标之间往往又是相互牵制的，随着数据转发平面设备的增多，这些都将成为控制器位置部署问题的研究难点。

(3)东西向接口问题

目前，SDN 网络的部署和应用仍然存在一定的局限性。要实现 SDN 在大规模网络上的实际应用，单一控制器的处理能力已经远远不能满足大规模网络中的海量数据请求，分布式的控制平面将是 SDN 网络发展的必然趋势。SDN 东西向

接口协议的设计是增强控制平面可扩展性的核心技术,目前相关研究尚处于早期探索阶段,研究工作可以从域内多控制器协同工作和域间多控制器通信两个方面进行探讨。

MSDN[28]是一种大型的逻辑意义上的控制器框架,实现了域内的多控制器协同工作。MSDN 由上至下包括负载均衡系统、控制器集群系统和分布式的数据共享系统。当数据流到达该控制器后,首先通过负载均衡系统将其分发至第二层控制器集群系统中的某一个载荷较轻的控制器;然后该控制器再调用第三层的分布式的数据共享系统,查看全网视图,计算数据流的转发路径。MSDN 解决了域内控制平面的多控制器协同工作问题,实验验证控制平面的吞吐量和时延都得到了稳步的提升。

SDNi[29]是 SDN 控制器之间的东西向接口协议,用于实现域之间的流的创建、撤销和更新,以及数据流可达性信息的更新和交换。

"East-West Bridge"[30]是由清华大学毕军教授于 2013 年首次在学术界提出的"东西向网桥"的概念,受到了研究人员的广泛关注。East-West Bridge 的设计中,控制器以全网状态的拓扑形式进行组网,实现了控制器发现、网络视图信息维护和多域网络视图信息交互等功能,并且允许控制器通过标准化插件的形式进行集成,从而实现了异构控制器之间东西向的标准化通信。

东西向协议的发展状况直接关系着 SDN 大规模部署的进度,东西向接口协议一定会成为未来 SDN 发展研究的重点问题。

3.3　控制层面与数据层面的时延缩减技术

目前,有一些研究人员通过减少控制平面与数据平面交互来达到提高控制平面的吞吐量、降低控制平面时延的性能优化效果。DevoFlow[31]利用 OpenFlow 的通配符规则,减少了交换机与控制器的交互次数,同时实现了新型机制,仅对重要的 QoS 数据流进行检测,有效节省了 TCAM 的硬件资源。DevoFlow 将数据流进行了长短流的划分,允许交换机制定本地路由信息用于直接的短流处理,而只有长流才交付控制器进行处理。DevoFlow 通过减少交换机与控制器的交互次数,实现了控制器负载的大幅降低,有效提升了控制器的可用性,但该方法在一定程度上违背了 SDN 数控分离的设计初衷。

DIFANE[32]提出了"权威交换机"的概念,将整个网络划分成几个区域,每个权威交换机管理一个区域范围的交换机。在权威交换机管理区域内的非权威交换机并不直接与控制器交互,其与控制器交互的工作由权威交换机来间接完成。控制器会事先将权威流规则下发到权威交换机,在每条流的第一个分组到达非权威交换机时,非权威交换机直接将分组转发到所在区域的权威交换机。权威交换机查找其流表缓存的流规则,若有匹配的流规则,权威交换机则直接将分组转发出去而不发回原交换机,并且将流规则下发到入口的非权威交换机。如此,DIFANE 保证了尽量把网络的流量保存在数据平面,从而提高分组流处理效率并降低控制器的负载。

基于控制层面与数据层面的时延,一致性问题不可避免,也不容忽视。SDN 网络的一致性问题可以分为单一控制器控制逻辑一致性问题和分布式控制器的状态一致性问题两方面。

单一控制器控制逻辑一致性是指控制器与交换机之间因存在传输时延而导致的控制器下发规则的先后顺序问题,这类问题可能会导致网络丢包、服务中断等现象[33]。DIFANE 和 DEVOFLOW[34]通过在 SDN 数据平面增加控制方法来减少控制逻辑一致性问题的产生,然而这样的做法也违背了 SDN 数控分离的设计宗旨。

Onix[35]利用带复制的事务性数据库模式和 DHT 模式来解决多控制器之间交换全网信息的一致性问题。这两种模式分别适用于网络事件更新缓慢、对稳定性和一致性要求高的网络,以及更新频繁、对网络可用性要求较高的网络。但这项研究并未解决如何分区域部署多控制器的问题。

在分布式控制器控制逻辑一致性问题研究方面,多域控制器之间的网络状态信息共享问题将会是一个研究的热点方向。频繁的全网信息同步更新会导致控制器负载的增加。而由于更新速度缓慢引发的控制器状态不一致又会严重影响控制器的性能。如何寻找一个最优的网络状态同步更新频率,或是根据网络状态变化的实际情况实现动态更新,都将给问题的解决带来新的思路。

4　数据平面的数据转发性能优化技术

数据平面是 SDN 三层网络结构的最底层。OpenFlow 交换机作为 SDN 数据平面的转发设备,依据控制平面下发的规则,采用流表结构进行数据包的转发。降低设备复杂度,提高数据转发效率,是数据平面交换设备的专注焦点。下面将分别从流表结构的优化设计、流表空间的高效利用和快速查表与转发 3 个方面阐述数据平面的优化技术。

4.1　流表结构的设计优化

在 SDN 网络中,网络设备的控制功能与转发功能分离,控制功能全部集中在控制器上,数据转发则由 OpenFlow 交换机来完成。而 OpenFlow 交换机进行数据转发的依据就存在于 TCAM 的流表。

OpenFlow 协议是目前业界所公认的 SDN 网络南向接口的事实标准协议,是 SDN 网络技术的理论基石。南向接口是 SDN 体系结构中至关重要的核心元素,是控制平面与数据平面实现解耦的关键所在。控制器通过南向接口协议实现链路发现、拓扑管理、策略制定、表项下发等控制技术,对网络中所有的交换机实行集中化的统一管理。

2009 年 12 月发布的 OpenFlow v1.0[36]是标准化组织 ONF 制定的第一个商业化版本的 OpenFlow 规范,得到了业界的广泛关注和使用[38-39]。OpenFlow v1.0 的架构原理如图 2 所示。可以看出,OpenFlow v1.0 架构中的核心元素是流表、安全通道和 OpenFlow 协议。

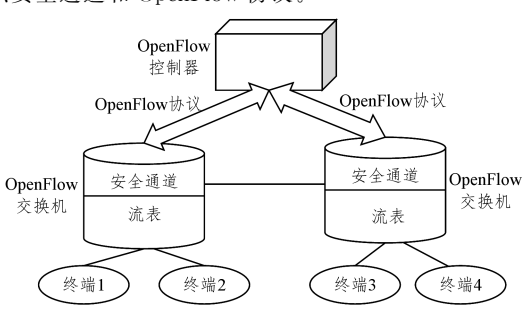

图 2　OpenFlow v1.0 架构原理

自 Open Flow 协议 1.0 发布以来,OpenFlow 协议已经经历了 v1.1,v1.2,v1.3 以及 v1.4 和 v1.5 版本的演进过程。

OpenFlow 流表也随着版本的演进在不断地变化，其匹配字段也从版本 v1.0 的 12 个字段扩展到了 v1.5 版本的 40 多个。然而，网络需求日益多样化，网络控制粒度也在不断地细化，这导致了单一流表将存储越来越多的策略。臃肿的流表对于控制平面和硬件性能都是一种挑战。在协议标准版本 v1.1 发布之后，OpenFlow 交换机实现了“流水线＋组表”的协议架构。“流水线”是将单一的庞大的流表分解成为多张串联起来的流表，从而形成了“流水线”，数据分组根据匹配情况在多张流表之间进行跳转。若某流表项匹配成功，则将执行该流表项中的指令集。用流水线匹配数据分组，减少了总的流表数目，节省了硬件资源，提高了匹配效率。同时，由于控制平面可以分步部署对数据平面的策略，实现了处理逻辑的清晰化和 SDN 的可操作性。

OpenFlow v1.1 还设计了“组表”。OpenFlow 交换机中只存在一个组表，其中包含多个组表项，每个组表项包含多个动作桶。组表解决了当不同数据流对应相同指令集而导致的为每一个数据流添加相同动作而带来的资源浪费问题，提高了数据平面的转发效率。

另外，从 OpenFlow v1.3.0 开始，用户可以按需自定义匹配字段，增强了协议的可扩展性。OpenFlow v1.3.0 中还提出了“漏表项”的概念，附在每张流表尾部用于处理该表中匹配失败的数据分组。“漏表项”的提出，减少了数据平面与控制平面的交互，同样提高了数据平面的转发效率。表 2 将目前已经推出的各个版本进行了对比和总结。

表 2 OpenFlow 规范的比较分析

版本号	发布时间	流表结构	流表	组表	IPv6 支持	匹配域数量	分布式控制器并发通信
v1.0	2009.11	包头域、计数器、动作	单流表	无	不支持	12	不支持
v1.1[40]	2011.02	匹配域、计数器、指令	多流表	有	不支持	15	不支持
v1.2[41]	2011.11	匹配域、计数器、指令	多流表	有	支持	36	支持
v1.3[42]	2012.04	匹配域、优先级、计数器、指令、超时定时器、cookie	多流表	有	支持	39	支持
v1.4[43]	2013.10	匹配域、优先级、计数器、指令、超时定时器、cookie	多流表	有	支持	40	支持
V1.5[44]	2014.12	匹配域、优先级、计数器、指令、生存期限、cookie、标记	多流表	有	支持	＞40	支持

数据平面从单一流表到多级流表的技术改进，有效地提升了流表资源的利用率，增加了流表处理的灵活性，但同时也增加了流表匹配时延与数据流的维护和管控的难度。

随着 SDN 的发展，OpenFlow 在不断演进，其面临的挑战也日益凸显出来。流表规模的合理化、流表结构的优化、流表过载、快速查表与高速转发等都将成为 OpenFlow 未来发展所要解决的问题。

4.2 流表空间的高效利用

基于数据平面的快速、高效的性能需求，在实际的工业设计中，交换机中流表的存储通常是用 TCAM 实现的。目前，市场上常见的硬件 OpenFlow 交换机中的 TCAM 最多能容纳 8000 条表项[45]。TCAM 存在价格昂贵、功耗大并且存储空间有限的问题，这个问题极大地限制了交换设备中流表的规模，也限制了 SDN 的规模。

缓存和压缩是优化管理流表空间的两种常用方法。CacheFlow[46]利用缓存技术将高速的硬件交换机和具有较大流表存储空间的软件交换机相结合，实现了具有任意大流表空间的简单而快速的虚拟交换机。该缓存架构将少部分的处理大量分组的流表项保存到 OpenFlow 交换机中，而将大部分并不常用的流表项放在缓存之中，从而解决了流表规模过大的问题，降低了交换机与控制器之间的流量。

文献[47]分析了当前流表存储结构存在的问题，提出了一种以流表项合并为基础的流表存储优化方案。首先设计一套基于待增流表项和已有流表项值的关联度算法来计算二者的关联度，将待增表项与关联度最高的现存表项合并后再写入流表，利用有限的流表物理空间管理更多的数据流，实现了节省流表空间的目标。

文献[48]将流从法则中分离出来，将流依据包域进行分类，共享代表法则。实验证明，该策略比已有的任何分类器代表法都更能够有效节省存储空间，节省的空间量多达几个数量级，同时降低了控制平面的存储和带宽的需求。

文献[49]采用流表共享策略 FTS 解决交换机流表溢出问题。FTS 将面临 Table-miss 事件的数据包从重载荷交换机分配至轻载荷交换机，避免在热点交换机上频繁触发 Packet-in 消息，既缓解了流表项溢出问题，同时也降低了控制器与交换机的交互频率，提高了数据平面的转发效率。

AdaFlow[50]通过分析预测流的到达情况和流表的资源使用情况，相应地调整新写入流表项的默认滞留时间配置，达到控制活动流表项数量、避免表项溢出、确保数据平面可用性的目标。

面对流表膨胀、流表存储空间资源紧张的严峻挑战，提高有限的流表存储空间的利用率，实现对流表空间的动态管理，解决流表项压缩与合并、流表项溢出、流表项生存时间最优化等问题，已经成为数据平面性能优化的又一大热点方向。

4.3 快速查表与快速转发

CatchFlow[51]系统将硬件交换机和软件处理相结合，高速缓存了存在于 TCAM 的普通规则，同时依靠软件处理小数量的“缓存缺失”流量。基于对具有重叠模式的规则之间的相关性的考虑，CatchFlow 系统并没有盲目地使用现存的缓存替代技术，采用较长的相关链的拼接，存储相关的规则小组，同时保留了规则的原有语义。规则拼接的方法有效利用了 TCAM 空间，实现了流的高速转发和流表规则的快速更新。

ESWITCH[52]实现了一个理想的交换机体系结构，使用动态模板代码生成器对 OpenFlow 管道消息进行快速编译并生成高效的机器代码，通过对快速路径的计算提升数据包的转发速度，降低延迟，增强控制平面的可扩展性。

上述数据转发性能优化方案中，对流到达特征的统计分析，要么增加了控制器算法的复杂度，要么需要在交换机中附加了控制功能，这样又违背了 SDN 数控分离的设计初衷；流资源的共享复用增大了控制器负荷和流表动态维护的难度；在大规模网络中，多流表架构的实现还必须依据应用场景的实际状态选择合适技术的实现方案。总言之，以上各种技术均在提高数据平面数据转发能力方面做出了一些有益的尝试。

5　SDN 在新网络环境中的应用

5.1　SDN 在云数据中心的应用

云数据中心是 SDN 最早的商业化应用场景。目前已经有很多学者利用软件定义网络的可编程性和集中管控的特点来实现数据中心的网络性能提升。

文献[53]提出由于交换机被静态分配给控制器,而动态流量会导致控制器之间的负载不平衡,结果一些控制器没有被充分利用,而过载控制器又可能会响应时间过长。该文献实现了动态的控制器分配,以便实现最小化控制器的平均响应时间。文献[54]中介绍了一个基于 SDN 的系统 Avalanche,并将其用在数据中心的商业交换机中以实现多播。作为Avalanche 的一部分,作者开发了一种新的多播路由算法 AvRA,试图为任何给定的多播组创建最小化路由树,从而实现高效率的带宽利用。

5.2　SDN 在网络虚拟化中的应用

虚拟化网络的目的是多个虚拟化网络服务共同分享公共的物理网络资源,同时各个虚拟化网络服务之间是相互隔离而互不影响的。虚拟化网络环境通过高效灵活的资源管理系统和有效的资源调度算法,既可以实现网络资源的按需分配,又可以实现有限的物理网络资源的高效利用。SDN 控制平面的全网视图抽象,成为了网络虚拟化实现的天然优势。

FlowVisor[55]是 2010 年由斯坦福大学开发的第一款基于透明代理模式的 SDN 网络虚拟化平台,已经在 GENI OpenFlow 实验平台中实现部署。FlowVisor 通过在网络中切分出并行、独立的资源切片,不同用户或应用可以使用自己的 Controllers 来定义不同的网络拓扑,同时 FlowVisor 又可以保证这些 Controllers 之间能够互相隔离而互不影响,从而可以满足各种网络实验创新的需求。OpenVirtex[56]是由斯坦福大学和加州大学伯克利分校共同创建的开放网络实验室(Open Networking Lab,On. Lab)团队基于 FlowVisor 开发的一款 SDN 网络虚拟化平台,允许实验者自定义拓扑和网络编址方案,允许实验者直观地进行虚拟节点的管理,同时可以借助后端数据库对网络状态进行实时记录,实现对实验切片执行网络快照的能力。

5.3　SDN 在无线网中的应用

无线通讯的爆炸式增长,使降低运营成本、简化网络管理和开放无线网络编程接口等属性成为无线网络的发展趋势。软件定义无线网络(SDWN)是将软件定义的思想应用于无线网络,这成为了 5G 移动通信网络的重要发展方向之一。

文献[57]首次提出基于 SDN 的蜂窝网络 CellSDN,其中基于属性的策略由 LTE 网络中的个体用户制定,并通过网络获得细粒度控制。在 CellSDN 中,每个交换机上的本地代理执行深度包检测,并减少控制器上的过多负载。文献[58]提出了 SoftAir,将 SDN 主体集成在 5G 网络中,利用虚拟化实现弹性网络。SoftAir 通过虚拟化提供移动感知负载平衡和资源高效分配。聚合控制由 NFV 提供,创建具有独立协议和资源分配算法的多个虚拟网络。SD-RAN 和 SD 核心网络节点通过 OpenFlow 和公共无线电接口(CPRI)启用和监控。所有管理策略都在中央控制平面上进行定义,从而实现云计算,并提供端到端的 QoS 保证。

5.4　SDN 在物联网中的应用

物联网和 SDN 都是新兴的网络技术。随着物联网中可以连接的网络实体数量的日益增多,众多的网络协议和网络框架使物联网的管理和控制也日益复杂。利用 SDN 实现物联网管理的灵活性和可编程性,已经成为研究的热点。

文献[59]中提出了基于 SDN 的无线传感器网络的新型控制框架 SDN-WISE,该控制框架不仅为网络管理者提供了 API 实现传感器节点的可编程的灵活管控,还降低了传感器节点与 SDN 控制平面的信息交互量。文献[60]实现了软件定义的无线传感器网络框架,该方法的架构组件包括基站 BS 和几个传感器节点。SDN 控制器在 BS 上进行路由选择以代替哑传感器节点。传感器节点包含 SDN 概念中的流表,由 SDN 控制器实现流表填充。

SDN 在新网络环境中的应用还处在不断讨论和探索的阶段。随着应用领域的不断扩展,后续还可能不断地暴露出新问题,其技术方案和标准体系都将成为有待研究和提升的新课题。

结束语　SDN 是当前互联网研究领域的热点问题,被业界认为是未来网络发展的方向,对于下一代互联网的建设有着重要的经济和社会价值。本文在综述 SDN 性能优化技术的同时,分析了未来该领域尚且需要进一步解决的问题和研究的方向。期望通过本文对已有研究结果的综述、对研究目标和方法的探讨分析,以及对研究思路的总结,为相关领域的研究人员提供参考和帮助。

随着 SDN 网络向企业网及运营网的逐渐引入,新问题将不断暴露,其应用场景、技术方案和标准体系都有待研究,SDN 性能提升技术也仍旧面临着一些有待解决的问题。

(1)SDN 网络安全问题

首先,控制平面作为 SDN 全网的指挥中心,其安全性至关重要。随着 SDN 网络规模的增大,分布式控制平面需要在地理位置分散、网络环境异构的条件下实现有效的分域管理,保障实现域内网络状态信息备份和恢复功能。其次,数据平面交换机作为数据转发节点,需要维护大量的转发规则信息,需要兼顾存储效率和转发速度。同时,在控制器故障或突发大数据流以及恶意外来攻击的情况下,必须有可靠的机制来应对,避免对网络性能造成严重影响。最后,应用平面必须保障每一个应用程序的正确性和安全性,以免使整个 SDN 网络的安全性受到威胁。为了实现安全的应用平台,控制平面需要提供有效应用隔离和权限管理机制,从而避免不同应用之间发生冲突,同时还要限制应用程序对底层资源的管理权限。

(2)SDN 网络管控软件复杂度与运算效率均衡问题

SDN 网络的灵活管控特性意味着软件程序的高复杂度。SDN 的分布式控制平面需要同时维护多张逻辑网络视图,通过网络虚拟化技术保证各应用程序共享资源的同时不影响彼此的功能。为了协调各程序的运行,提高资源利用效率,各种算法的复杂度和运算量往往会呈指数级上升。如何在软件复杂度和运算效率之间取得平衡,是 SDN 将要面临的一大挑战。

(3)SDN 网络资源利用率问题

SDN 可以大幅度降低网络的运营成本,这是被许多网络运营商所看好的。SDN 根据应用程序和数据需求来配置网络架构,允许企业在尽可能少的物理硬件上支持这些功能;再加上动态网络配置的运营优势及其对生产力的影响,似乎 SDN 确实能够在很大程度上节约网络硬件成本。但在当前的网络基础设施上到底有多少闲置容量,SDN 可以挖掘多少

在静态环境中被忽略的隐藏资源，或者保持储备以满足高峰需求，这些问题都值得各界研究人员的进一步深入研究和实践探索。

参 考 文 献

[1] GREENBERG，ALBERT，HJALMTYSSON，et al. A clean slate 4D approach to network control and management[J]. ACM SIGCOMM Computer Communication Review，2005，35(5)：41-54.

[2] CASADO M，GARFINKEL T，AKELLA A，et al. SANE：A Protection Architecture for Enterprise Networks[C]// Usenix Security. 2006.

[3] CASADO M，FREEDMAN M J，PETTIT J，et al. Ethane：taking control of the enterprise[C]// Conference on Applications，Technologies，Architectures，and Protocols for Computer Communications. ACM，2007：1-12.

[4] Open Networking Foundation. Software-Defined Networking：The New Norm for Networks[EB/OL]. ONF WhitePaper.

[5] Open Networking Foundation(ONF)[EB/OL]. https://www.opennetworking.org/about/onf-overview.

[6] OpenFlow Switch Specification，version 1.0.0[EB/OL]. http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf.

[7] TAVAKOLI A，CASADO M，KOPONEN T，et al. Applying NOX to the Datacenter[C]// HotNets. 2009.

[8] SGAMBELLURI A，PAOLUCCI F，GIORGETTI A，et al. SDN and PCE implementations for segment routing[C]// European Conference on Networks and Optical Communications. IEEE，2015：1-4.

[9] NG E，CAI Z，COX A L. Maestro：A system for scalable openflow control：TSEN Maestro-Techn. Rep，TR10-08[J]. Rice University，Houston，TX，USA，2010.

[10] TOOTOONCHIAN A，GORBUNOV S，GANJALI Y，et al. On controller performance in software-defined networks[C]// Usenix Conference on Hot Topics in Management of Internet，Cloud，and Enterprise Networks and Services. 2012：10-10.

[11] TAKAMIYA Y，KARANATSIOS N. Trema OpenFlow controller framework[EB/OL]. https://github.com/trema/trema

[12] Nippon Telegraph and Telephone Corporation ，RYU network operatingsystem[EB/OL]. http://osrg.github.com/ryu.

[13] ERICKSON D. The beacon openflow controller[C]// ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM，2013：13-18.

[14] Floodlight Is An Open SDN Controller[EB/OL]. http://floodlight.openflowhub.org.

[15] BANIKAZEMI M，OLSHEFSKI D，SHAIKH A，et al. Meridian：an SDN platform for cloud network services[J]. IEEE Communications Magazine，2013，51(2)：120-127.

[16] SAIKIA D，MALIK N. An Introduction to OpenMUL SDN Suite[EB/OL]. http://www.openmul.org/uploads/1/3/2/6/13260234/openmul-sdn-platform.pdf.

[17] SHALIMOV A，ZUIKOV D，ZIMARINA D，et al. Advanced study of SDN/OpenFlow controllers[C]// Central & Eastern European Software Engineering Conference in Russia. 2013：1-6.

[18] TOOTOONCHIAN A，GANJALI Y. HyperFlow：a distributed control plane for OpenFlow[C]// Internet Network Management Conference on Research on Enterprise Networking. USENIX Association，2011：3.

[19] 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. ACM，2012：19-24.

[20] CHEN W，LI H，MA Q，et al. Design and implementation of server cluster dynamic load balancing in virtualization environment based on OpenFlow[C]// International Conference on Future Internet Technologies. ACM，2014：1-6.

[21] HU Y，WANG W，GONG X，et al. BalanceFlow：Controller load balancing for OpenFlow networks[C]// IEEE，International Conference on Cloud Computing and Intelligent Systems. IEEE，2013：780-785.

[22] DIXIT A，HAO F，MUKHERJEE S，et al. Towards an elastic distributed SDN controller[J]. ACM SIGCOMM Computer Communication Review，2013，43(4)：7-12.

[23] HELLER B，SHERWOOD R，MCKEOWN N. The controller placement problem[C]// Proceedings of the first workshop on Hot topics in software defined networks. ACM，2012：7-12.

[24] 姚龙. 软件定义网络控制器容量及部署问题研究[D]. 合肥：中国科学技术大学，2015.

[25] JIMENEZ Y，CERVELLO-PASTOR C，GARCIA A J. On the controller placement for designing adistributed SDN controllayer[C]// Proceedings of Networking Conference，IFIP Trondheim. Norway，2014.

[26] HU Y，WANG W，GONG X，et al. On reliability-optimized controller placement for software-defined networks[J]. Commun，2014，11(2)：38-54.

[27] BARI M F，ROY A R，CHOWDHURY S R，et al. Dynamic controller provisioning in software defined networks[C]// Proceedings of Networkand Service Management (CNSM). Zurich，Switzerland，2013.

[28] 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)：1969-1974.

[29] MOLNÁR L，PONGRÁCZ G，ENYEDI G，et al. Dataplane Specialization for High-performance OpenFlow Software Switching[C]// Proceedings of the 2016 Conference on ACM SIGCOMM. ACM，2016：539-552.

[30] LIN P，BI J，WANG Y. East-West Bridge for SDN Network Peering[M]// Frontiers in Internet Technologies. Springer Berlin Heidelberg，2013：170-181.

[31] MOGUL J C，TOURRILHES J，YALAGANDULA P，et al. Devoflow：Cost-effective flow management for high performance enterprise networks[C]// Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM，2010：1.

[32] YU M，REXFORD J，FREEDMAN M J，et al. Scalable flow-based networking with DIFANE[J]. ACM SIGCOMM Computer Communication Review，2010，40(4)：351-362.

[33] 柳林，周建涛. 软件定义网络控制平面的研究综述[J]. 计算机科学，2017，44(2)：75-81.

[34] 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.

[35] KOPONEN T，CASADO M，GUDE N，et al. Onix：A Distributed Control Platform for Largescale Production Networks[C]// Operating Systems Design and Implementation(OSDI). Vancou-

ver,Canada,2010:1-6.
[36] OpenFlow Switch Specification, version 1. 0. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 0. 0. pdf.
[37] SUZUKI K, SONODA K, TOMIZAWA N, et al. A Survey on OpenFlow Technologies[J]. IEICE Transactions on Communications, 2014, E97. B(2):375-386.
[38] LARA A, KOLASANI A, RAMAMURTHY B. Network innovation using openflow: A survey[J]. IEEE Communications Surveys & Tutorials, 2014, 16(1):493-512.
[39] NUNES B A A, MENDONCA M, NGUYEN X N, et al. A survey of software-defined networking: Past, present, and future of programmable networks[J]. IEEE Communications Surveys & Tutorials, 2014, 16(3):1617-1634.
[40] OpenFlow Switch Specification, version 1. 1. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 1. 0. pdf.
[41] OpenFlow Switch Specification, version 1. 2. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 2. 0. pdf.
[42] OpenFlow Switch Specification, version 1. 3. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 3. 0. pdf.
[43] OpenFlow Switch Specification, version 1. 4. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 4. 0. pdf.
[44] OpenFlow Switch Specification, version 1. 5. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 5. 0. pdf.
[45] KREUTZ D, RAMOS F M V, VERISSIMO P E, et al. Software-defined networking: A comprehensive survey[J]. Proceedings of the IEEE, 2015, 103(1):14-76.
[46] KATTA N, ALIPOURFARD O, REXFORD J, et al. Infinite cacheflow in software-defined networks[C]// Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, 2014:175-180.
[47] 李向文,吉萌,曹敏,等. 基于资源复用的 Openflow 流表存储优化方案[J]. 光通信研究,2014(2):8-11.
[48] KOGAN K, NIKOLENKO S, EUGSTER P, et al. Strategies for mitigating TCAM space bottlenecks[C]// 2014 IEEE 22nd Annual Symposium on High-Performance Interconnects (HOTI). IEEE, 2014:25-32.
[49] QIAO S, HU C, GUAN X, et al. Taming the Flow Table Overflow in OpenFlow Switch[C]// Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference. ACM, 2016:591-592.
[50] ZHOU B, GAO W, WU C, et al. AdaFlow: Adaptive Control to Improve Availability of OpenFlow Forwarding for Burst Quantity of Flows[C]// International Conference on Testbeds and Research Infrastructures. Springer International Publishing, 2014:406-415.
[51] KATTA N, ALIPOURFARD O, REXFORD J, et al. Rule-Caching algorithms for Software-Defined Networks[R]. Technical report, 2014.
[52] MOLNÁR L, PONGRÁCZ G, ENYEDI G, et al. Dataplane Specialization for High-performance OpenFlow Software Switching [C]// Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference. ACM, 2016:539-552.
[53] WANG T, LIU F, GUO J, et al. Dynamic sdn controller assignment in data center networks: Stable matching with transfers [C]// The 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016. IEEE, 2016:1-9.
[54] IYER A, KUMAR P, MANN V. Avalanche: Data center multicast using software defined networking[C]// 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS). IEEE, 2014:1-8.
[55] SHERWOOD R, GIBB G, YAP K K, et al. Flowvisor: a network virtualization layer [R]. OpenFlow Switch Consortium, 2009:1-13.
[56] AL-SHABIBI A, DE LEENHEER M, GEROLA M, et al. OpenVirteX: make your virtual SDNs programmable [C]// Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, 2014:25-30.
[57] LI L E, MAO Z M, REXFORD J. Toward software-defined cellular networks[C]// 2012 European Workshop onSoftware Defined Networking (EWSDN). 2012:7-12.
[58] AKYILDIZ I, WANG P, LIN S. SoftAir: A software defined networking architecture for 5G wireless systems[J]. Computer Networks, 2015(85):1-18.
[59] GALLUCCIO L, MILARDO S, MORABITO G, et al. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks[C]// IEEE Conference on Computer Communications (INFOCOM). 2015:513-521.
[60] GANTE D, ASLAN M, MATRAWY A. Smart wireless sensor network management based on software-defined networking[C]// 27th Biennial Symposium on Communications (QBSC). 2014:71-75.

(上接第 83 页)

[45] JAFARI S M S, ZIAADDINI F. Optimization of software cost estimation using harmony search algorithm[C]// Swarm Intelligence and Evolutionary Computation. IEEE, 2016:131-135.
[46] MITTAS N, ANGELIS L. LSEbA: least squares regression and estimation by analogy in a semi-parametric model for software cost estimation [J]. Empirical Software Engineering, 2010, 15(5):523-555.
[47] 李效云,杨达,杨叶. 一种改进的类推估算方法及案例研究[J]. 计算机应用与软件,2011,28(7):5-9.
[48] IDRI A, AMAZAL F A, ABRAN A. Analogy-based software development effort estimation: A systematic mapping and review [J]. Information & Software Technology, 2014, 58:206-230.
[49] BARDSIRI V K, JAWAWI D N, HASHIM S Z, et al. A PSO-based model to increase the accuracy of software development effort estimation[J]. Software Quality Journal, 2013, 21(3):501-526.
[50] IDRI A, HOSNI M, ABRAN A. Improved estimation of software development effort using Classical and Fuzzy Analogy ensembles[J]. Applied Soft Computing, 2016, 49:990-1019.
[51] HUANG Z. Clustering Large Data Sets With Mixed Numeric and Categorical Values[EB/OL]. http://www. doc88. com/P-7728902324900. html.
[52] BISHNU P S, BHATTACHERJEE V. Software cost estimation based on modified K-Modes clustering Algorithm[J]. Natural Computing, 2016(3):1-8.
[53] KASHYAP D, MISRA A K. Software Cost Estimation Using Similarity Difference Between Software Attributes[J]. Advances in Intelligent Systems & Computing, 2013, 236:1-6.