基于可编程网络的算力调度机制研究

Computing Power Scheduling Mechanism Based on Programmable Network

李铭轩 /LI Mingxuan，曹畅 /CAO Chang，杨建军 /YANG Jianjun
（中国联合网络通信有限公司研究院，中国 北京 100048）
(China United Network Communication Research Institute, Beijing 100048, China)

摘要：结合最新的可编程网络技术，提出了算力资源调度技术，并介绍了技术架构和算力调度机制。在算力资源调度技术架构的基础上，进一步提出了整体平台功能架构和编程架构。基于可编程网络的算力资源调度技术解决了目前算力调度过程中无法实现的网络参数问题，从而能够更好地实现网络和算力的融合。

关键词：可编程网络；云原生；P4；无服务

Abstract: Combined with the latest programmable network technology, the computing power resource scheduling technology is proposed, and the overall technical architecture and computing power scheduling mechanism are introduced. Based on the technical architecture of computing power resource scheduling, the overall platform functional architecture and programming architecture are further proposed. The computing power scheduling mechanism based on the programmable network solves the current bottleneck of network parameters that cannot be achieved in the current computing power scheduling process, which can better achieve the integration of network and computing power.

Keywords: programmable network; cloud native; P4; serverless

DOI：10.12142/ZTETJ.202103005
网络出版地址：https://kns.cnki.net/kcms/detail/34.1228.TN.20210615.1137.002.html

网络出版日期：2021-06-15
收稿日期：2021-05-15

软件定义网络（SDN）通过网络控制逻辑（控制平面）与转发流量（数据平面）的分离，将传统封闭的网络体系解耦为数据平面、控制平面和应用平面，简化策略实施和网络配置[1]。2008年，以斯坦福大学Nick MCKEOWN教授为首的研究团队提出了OpenFlow以及SDN技术。自此，SDN技术获得了业界的高度关注，一系列相关应用被提出，极大地促进了网络创新发展。2014年，在SDN基础上，研究者又提出了可变成数据平台技术，将网络编程能力扩展到数据平面，进一步开放了网络设备的可编程能力[2]。

在从原有的虚拟化技术向云原生技术的演进过程中，传统的算力资源调度技术往往基于网络互通，并通过集群自身的调度策略来实现算力的动态调度和应用分配。在这一过程中，固化的组网方式已经无法满足业务需求，网络编排能力已成为算力调度能力的制约因素。文章中，我们着重研究基于可编程网络的算力调度机制，以期能够将可编程网络的最大优势应用于传统的算力调度机制中。

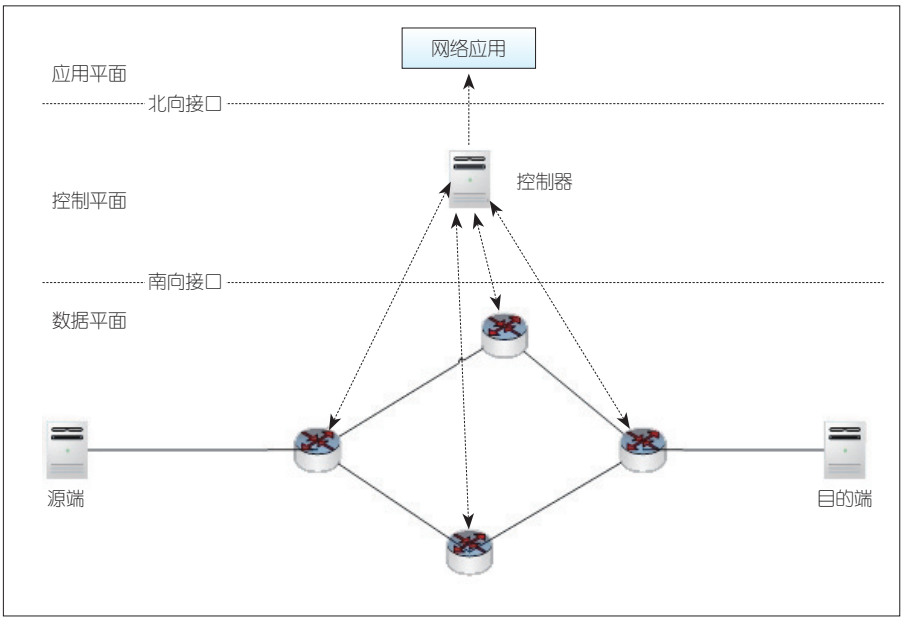
基金项目：国家重点研发计划（2019YFB1802800）

1 可编程网络介绍

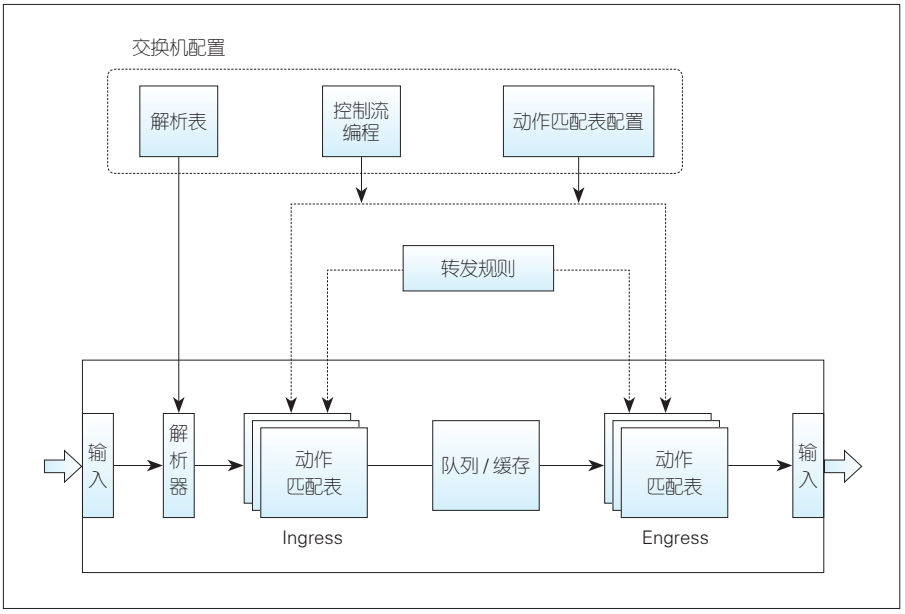
现有的网络技术尤其是SDN技术的发展，使得传统网络转发设备能够从固化在芯片上的转发机制向基于通用芯片承载的转发机制转变，同时也可以为SDN的实现带来可能性[3]。现有的以SDN为代表的可编程网络实现技术，主要是基于可编程的网络协议和转发控制协议：应用平面的网络应用通过控制平面的控制器向底层的数据平面SDN数据转发设备下发路由协

议和转发策略，从而实现网络转发机制[4]。SDN 网络架构如图 1 所示。

基于上述 SDN 网络架构，网络应用通过控制器对 SDN 路由器的转发机制进行控制。数据报文从源端服务器，经 SDN 路由器，发送至目的端服务器。在此过程中，网络应用程序可以通过控制器来选择不同的转发路径，进行数据转发链路的路由。在整个网络架构中，为了实现两层解耦，可以通过南向接口对底层数据平面的转发功能进行封装，实现控制器和路由器之间的对接；再通过北向接口对 SDN 控制能力进行封装，对上层应用提供统一的能力开放。网络应用则通过调用北向接口来实现对控制器的调用和控制[5]。



▲图 1 软件定义网络架构图



▲图 2 可编程网络数据平面抽象转发模型图

1.1 可编程网络技术架构

SDN 的可编程网络技术架构，可以实现 SDN 路由器的流量转发策略控制和软件定义设定。在数据转发平台，现有技术从原有的基于 OpenFlow 协议相关的数据转发，逐渐演进到与协议无关的面向高级编程的数据转发平面。可编程网络技术通过代码级的自定义网络数据平面来实现可编程能力，同时还可以实现数据转发控制和策略[6]，具体实现流程如图 2 所示。

根据交换机的配置，传统的可编程网络数据转发对解析表和控制流进行编程，并且将解析表下发至数据转发平台的解析器中，再通过控制流编程将动作匹配配置下发至数据转发平台的动作匹配表中[7]。

数据流在输入时，首先经过解析器进行报文解析，并通过 Ingress 负载均衡器进行转发设置，以实现动作匹配配置，然后进入转发的队列和存储中等待；数据流从队列中出栈时，先通过对应的 Engress 反向负载均衡器的解析，再进行报文输出，从而实现了完整的数据平面数据转发机制[8]。

1.2 控制面 / 用户面分离实现机制

传统的 SDN 技术通过控制面和数据面的分离，实现网络数据流转发和控制策略制定之间的分离，也为基于可编程网络的算力调度提供了可能。

从上述的技术架构可以看出，现有的 SDN 技术实现了两层解耦：一方面，将控制器集中于上层控制面，实现对 SDN 路由器或数据转发设备的统一管理；另一方面，使数据平面的 SDN 路由器和转发设备脱离了传统模式（固化在设备内），并使得转发功能和转发规则通过南北向接口开放至上层控制器和网络应用。这种控制面 / 用户面分离机制，为网络可编程的进一步实现提供了可能。

2 算力调度机制

随着云计算技术的不断发展，基础设施领域已有越来越多的企业采用云计算作为统一资源的管理方式。随着云资源池规模的不断扩大，算力节点的调度主要采用分布式的方式；而传统的基于云计算或云原生的算力资源调度是以虚拟化技术和进程共享技术来实现的[9]。基于 OpenStack 或 Kubernetes 的算力资源调度将算力节点的空闲度作为算力调度策略的主要评判依据。本文中，我们以 Kubernetes 的资源调度组件 Scheduler 为例，重点阐述云原生的算力调度机制。

Scheduler 是 Kubernetes 的核心组件，负责为用户声明的 Pod 资源选择合适的算力节点，同时保证集群资源的最大化利用，其任务资源调度流程如图 3 所示。

现有的 Kubernetes 资源调度机制根据用户的请求，从资源管理器中获取资源信息，并且根据具体的调度策略将任务调度至具体的算力节点上。

在网络可达的情况下，现有 Kubernetes 算力节点运行状态监测控制主要通过算力节点代理监测的方式来实现。通过采集和上传算力节点上的中央处理器（CPU）、存储、内存等信息，并将这些信息上传至资源管理器，再经资源调度器进行策略调度[10]，从而将任务调度至指定的算力节点上。这种算力资源调度机制虽然能够解决分布式环境中、算力资源非均衡情况下的算力动态调度问题，但必须基于网络可达的情况。该机制并没有考虑到网络质量、算力节点的连接，以及传输过程中的网络情况。随着分布式计算，尤其是大数据等多集群甚至是跨数据中心协同处理的发展，网络的数据传输质量往往会成为影响上层用户体验的关键因素，同时也会成为跨数据中心算力调度和集群高效协同的制约因素。传统的算力调度机制仅实现了计算资源在非均衡状态下的动态调度，却未考虑网络的服务质量（QoS）或体验质量（QoE）。在算网融合应用快速发展的趋势下，传统的算力调度方式已无法满足当前需求。

SDN 技术，尤其是可编程网络技术的发展，促使网络能力进一步开放、可编程化，并使传统算力调度机制能够更好地融合网络因素。本文中，我们通过算力和网络协同的方式来实现算力最优化调度，极大地发挥了网络在数据传输和转发方面的优势。

3 基于可编程网络的算力调度方案

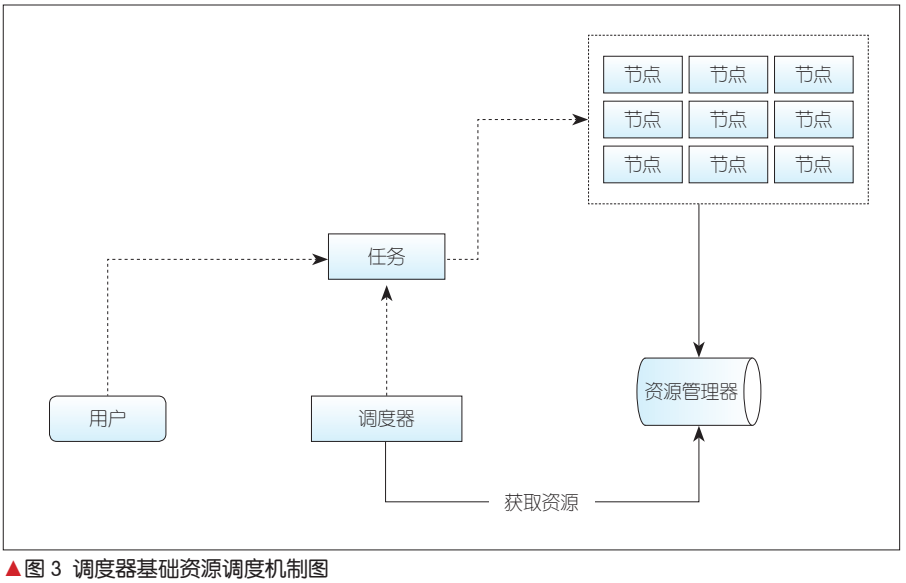
基于网络的可编程、可控制等能力，并结合算力节点空闲度和计算能力等因素，基于可编程网络的算力调度机制能够在网络路由和路径选择方面实现算力调度。本文中，我们研究了通过编排调度方式来实现算力服务的编排和管理，以及可编程网络能力的开放。

3.1 整体网络架构

基于可编程网络的管控分离能力，本文在 Kubernetes 原有调度方式基础上，进一步研究了基于可编程网络的算力调度机制。其中，网络拓扑采用数据控制面和数据转发面分离模式，容器计算集群承载具体的算力分配和容器承载，控制集群和可编程网络的数据控制面对接以实现网络控制。基于可编程网络的技术架构如图 4 所示。

基于可编程网络的算力调度架构，面向分布式集群通过容器控制集群，向数据控制面下达网络控制指令；通过控制器，向数据转发面的转发设备发送数据转发策略和数据路由表等网络转发协议；通过数据转发面接入的计算集群，实现算力节点的调度和容器承载。基于可编程网络架构的数据转发，可以改变原有容器资源仅能在 overlay 的数据中心内调度的情况，实现基于 underlay 的跨数据中心的算力资源调度[11]。

在每一个算力节点上，该架构采用传统的 master/agent 模式来代理、发布算力节点的计算、存储和应用输入输出（IO）等情况，并尝试将这些情况反馈至控制集群中服务器的调度器，从而实现集群内算力节点的统一管理。网络 QoS、QoE 以及转发设备等状态，通过数据转发平台上传至控制平面，



▲图 3 调度器基础资源调度机制图

然后被统一管理。在整个算力资源编排调度的过程中，容器控制集群为用户和开发者提供了统一入口，并通过统一配置脚本、开发变成方式，来实现面向服务的算力和网络的统一编排和调度，从而实现面向服务的基于可编程网络的算力调度。

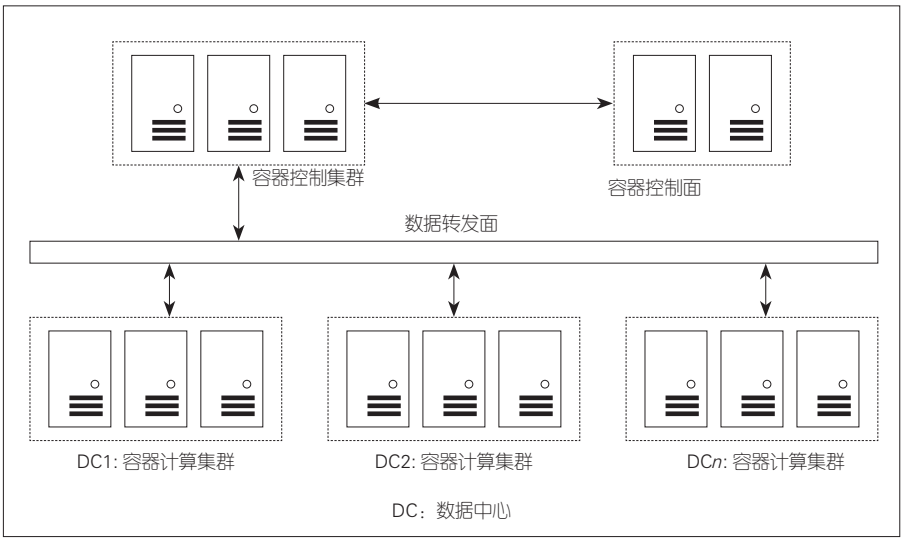
3.2 算力调度机制

在算力调度方面，Kuberenetes 云原生平台提供服务编排调度能力，集成网络编排能力和计算服务编排能力，并通过 Knative 实现统一的应用能力封装和消息队列。整体算力调度层为上层门户提供应用程序编程接口（API）网关，也为上层应用提供统一的 API。这可以开放可编程网络的算力能力，屏蔽底层网络和算力的差异性，并且可以为开发者和用户提供统一门户，进一步降低了可编程网络算力调度的开发门槛。算力调度平台具体的架构如图 5 所示。

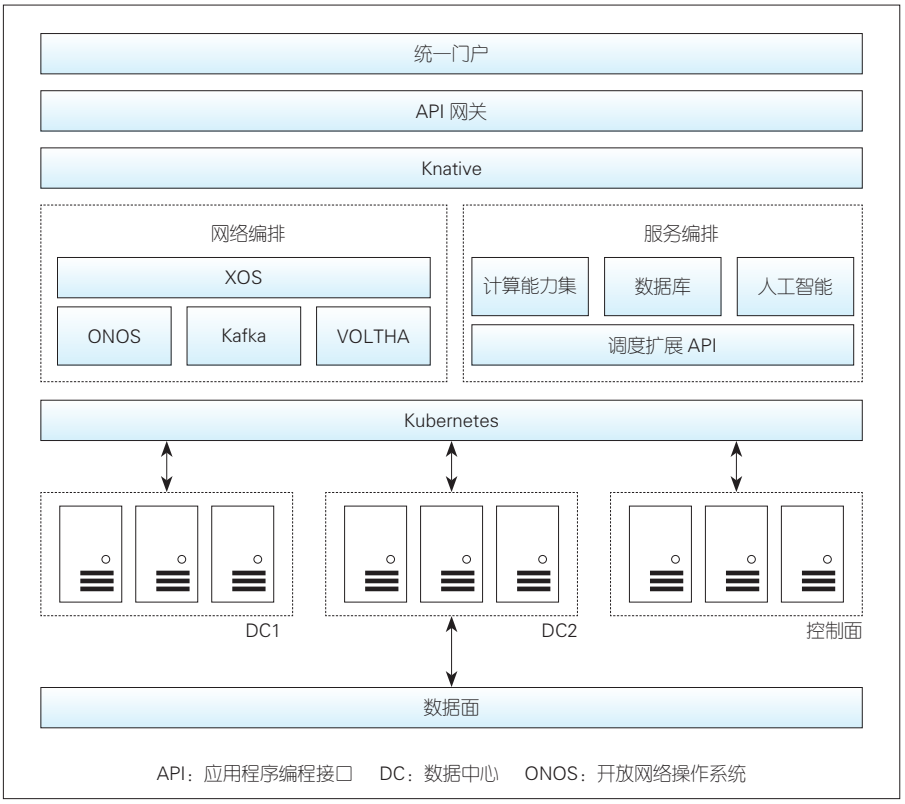
整体算力调度机制由 Kubernetes 实现统一的算力网络资源调度。其中，根据资源服务对象的不同，Kubernetes 调度能力可以分为两个方面：一方面是以基础设施平台即服务（i-PaaS）能力为主，实现对底层基础设施算力资源的调度，借助控制平面的对接来实现对网络数据面的调度和管理，通过对接不同的 Kuberentes 云原生集群实现对底层云原生集群的调度管理；另一方面是以应用层 PaaS（A-PaaS）能力为主，实现对网络编排和计算服务编排的服务能力管理。面向上层的能力调度主要包含网络编排和服务编排两个方面。

（1）网络编排

网络编排主要是指，对底层的网络服务编排能力进行硬件资源的抽象和能力的建模，并通过服务编排来实现网络控制。本文中，我们提出基于



▲图 4 基于可编程网络的技术架构图



▲图 5 算力调度平台架构图

SDN 的宽带接入（SEBA）容器化架构，以实现 SDN 网络访问。SEBA 的核心组件主要包括开放网络操作系统（ONOS）、Kafka、VOLTHA、XOS。

• ONOS：实现 SDN 网络操作系统，对网络服务编排实现统一的资源调度和管理。

• Kafka：实现 REST 的消息队列管理，并通过上层的服务能力对底层硬件的访问请求消息进行统一管理。

• VOLTHA：实现底层网络接入设备和转发设备的硬件资源抽象，从而使用和访问上层的网络功能。

• XOS：实现网络功能虚拟化和服

务化，并可以基于 SDN 控制器的可编程能力实现网络控制和功能软件定义能力。

（2）服务编排

服务编排可以实现对 PaaS 和软件即服务（SaaS）能力的容器化调度。由于云原生具有服务化和微服务化的能力，因此在实现算力调度的过程中，基于不同的应用场景，我们提出了 3 个方面的服务能力。

• 计算能力集：集成目前云原生统一的计算型能力库，包括 Spark、Hadoop、Hive、Flink 等。

• 数据库：采用传统的数据库服务能力，为上层的应用和业务场景提供一键部署式的云原生数据库，包括 Mysql、MangoDB 等。

• 人工智能：包括面向人工智能场景的推理和训练，以及对硬件加速有特定需求的算力调度能力。

这些服务能力统一由 Kubernetes 来实现编排。通过 Kubernetes 的调度扩展接口和平台内部调度器对接，从而能够实现 PaaS 和 SaaS 服务的容器化调度。

通过 Knative 来完成统一服务能力的封装和打包，通过 Knative 的 API 网关提供统一的网络和算力调度接口，并通过统一的门户对外开放，开发者可以根据网络和算力调度能力进行网络编程。这样可以进一步融合底层网络和算力，实现基于可编程网络的算力调度。同时，用户也可以更加关注上层业务逻辑和业务流程。

3.3 可编程网络编排机制

构建在传统 SDN 架构上的可编程网络算力调度机制，不仅实现了网络控制面和用户面的分离，还实现了基于云原生统一 Kubernetes 平台的服务编排调度能力。在可编程网络服务编排能力方面，随着网络转发设备的普及，基于 P4 的网络可编程能力实现了网络的可编程。另外，网络组件本身也可以进行容器化，并可以调度到具备 P4 功能的白盒交换机上。基于容器化的可编程网络编排架构如图 6 所示。

依据上述可编程网络算力编排技术架构，P4 交换机集成了专门针对 P4 的运行时（Runtime）。在通用计算节点上，运行时集成了运行应用程序的镜像。在整个技术架构中，面向上层的网络功能和应用程序提供了统一的容器封装能力，用于打包和封装容器镜像。其中，在网络功能容器化封装的过程中，P4 编译器专门用于服务网络功能程序，即将网络功能程序编译成可在 P4 交换机上运行的可执行程序后，再进行容器化封装。Kubernetes 平台实现了 P4 交换机和通用计算节点的算力资源调度和服务编排。根据网络功能和应用程序的不同，Kubernetes 平台分别将网络功能调度到 P4 交换机上运行，将应用程序调度到通用计算节点上运行 [12]。面向上层开发者则提供统一的开发平台、API、网络可编程能力和应用程序开发能力，从而实现可编程网络和应用程序。这样一来，基于可编程网络的算力调度技术在代码开发阶段就能够进行融合开发，满足目前越来越多的算网融合场景下的应用程序开发需求。

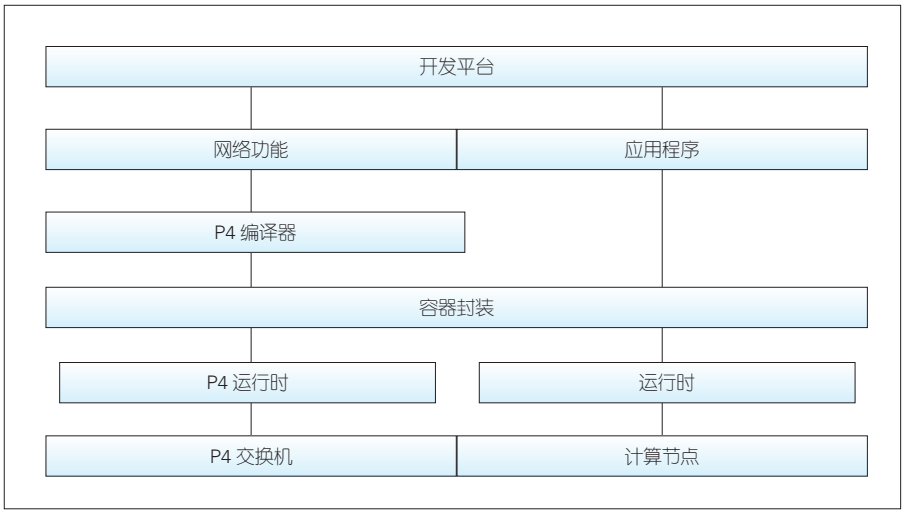
4 结束语

异构算力网络是下一代云网融合 2.0 的发展趋势。基于可编程网络的算力调度机制，能够在网络可编程的基础上，实现传统算力调度无法实现的基于网络的算力调度方式。该机制可以根据网络情况进行算力调度，也可以基于算力调度需求进行网络适配和可编程，从而真正实现云网融合 [13–15]。本文中，我们所提出的基于可编程网络的算力调度技术，是基于云原生技术来实现算力网络的融合调度。传统的云原生调度仅能基于 overlay 网络实现算力调度，而该技术则可以实现基于 underlay 网络的算力调度和服务编排能力。该技术还可以提供业务感知的网络编排能力，从而能为后续的网络感知业务提供新的研究思路和发展方向。

致谢

本研究得到中国联通研究院李建飞高级工程师的帮助，同时也得到了中国联通研究院首席科学家唐雄燕的指导，谨致谢意！

下转第 61 页 ➡



▲图 6 可编程网络算力编排架构图