泛在云场景下 Serverless 计算的机遇与挑战

杨亚南　　张健松　　吴　杰
中国电信云计算研究院

传统云计算到泛在云架构的演变

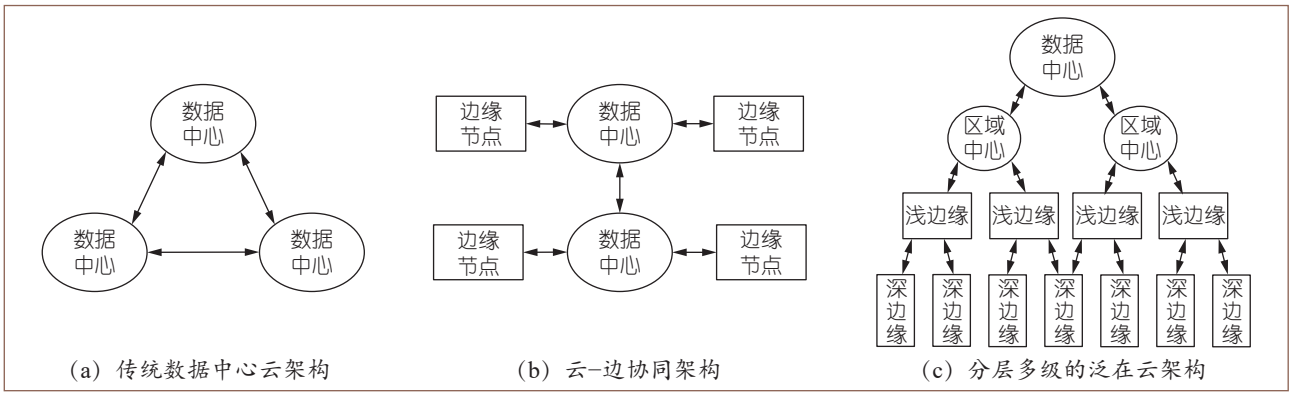
自 2006 年亚马逊网络服务(Amazon Web Services, AWS)推出云计算服务以来,云计算产业经历了近二十年的发展,已成为全球数字经济不可或缺的基础设施。国内云计算产业随后也全面起步,例如 2009 年阿里云计算诞生,2012 年以中国电信为代表的运营商推出云计算服务。传统云计算模式以少数几个超大规模数据中心为主体,各地终端通过互联网访问部署于数据中心内的云计算服务。考虑到大规模供电、散热和占地的需要,云数据中心通常以数十万台节点规模的方式建设在中西部偏远地区。与云计算同时,边缘计算也经历了类似的发展过程,从 1998 年 Akamai 公司成立并推出内容分发网络(content delivery network, CDN)服务,到 2016 年边缘计算产业联盟成立全面推动标准化和产业化,边缘计算进一步向物联网(internet of things, IoT)、智能制造、自动驾驶等领域发展落地。长期以来,云计算与边缘计算发挥着协同互补的作用。随着云计算服务商纷纷推出边缘计算服务,后者也常被看作云计算的一部分。

然而,近年来这种以超大规模数据中心为主体,云-边协同为辅助的传统云计算架构正在悄然经历着基础设施布局层面的深刻变化。首先是 5G 推动下的网络云化。2012 年欧洲电信标准协会成立了网络功能虚拟化(network function virtualization, NFV)行业规范组,标志着网络设施虚拟化和云化升级的开启。基于网络切片、自动化编排调度等 5G 关键能力的需要,5G 核心网和接入网大量采用云化技术来承载网络功能,通过基站虚拟化技术支持灵活部署和动态调整。伴随着 5G 建设的开展,网络基础设施开始升级成为以基站/区、城市、省、全国为覆盖范围的一系列层次化“网络云”。其次是人工智能(artificial intelligence, AI)浪潮下的智算基础设施建设。2014 年的 AlphaGo 和 2022 年的 ChatGPT 掀起了以卷积神经网络和大语言模型为技术代表的两轮 AI 热潮。随之而来的 AI 基础设施建设也在全国各地如火如荼开展,形成了一系列大大小小的“智算云”。

2024 年全国两会政府工作报告提出“适度超前建设数字基础设施,加快形成全国一体化算力体系”[1]。作为国家云建设的主力军,中国电信在国内部署了 800 多个数据中心和 3000 多个边缘节点,总算力规模超过 30 EFLOPS(exa floating point operations per second, EFLOPS),构建形成了“2+4+31+*X*”的层次化分布的广域通用算力节点布局。其中“2”代表位于贵州和内蒙古的超大规模数据中心,“4”代表覆盖 4 个核心经济区域的数据中心,“31”代表 31 个省级数据中心,“*X*”代表覆盖全国 241 个城市的“一城一池”节点。与此同时,以中国电信为代表的运营商积极推动“云网融合”战略,基于国内网络基础设施建设的优势将各地的数据中心连接成一体,形成类似电网和高铁网的全国性云网基础设施。可以看到,在当前算力基础设施建设和互联的背景下,云计算架构正在向层次化布局、云边协同的泛在云(Ubiquitous Cloud)演进,如图 1 所示。

泛在云是中国电信近期提出的新概念,技术层面包含了 2 个方面。一方面是地理上全域覆盖、逻辑上

DOI:10.11991/cccf.202508012
通信作者:吴杰, E-mail: wujie@chinatelecom.cn



（a）传统数据中心云架构　（b）云–边协同架构　（c）分层多级的泛在云架构

图 1　从传统云数据中心架构到泛在云架构

层次化部署的各类大小算力中心，另一方面是全域网络设施与算力设施的深度融合[2]。泛在云代表了运营商进入云计算领域所带来的架构变化，并且这一变化仍处于高速发展的演进过程中。在泛在云的愿景下，算力将变得无处不在，云服务可以随时、随地、以最适配的形式提供给用户，使得身处不同地域、来自不同业务领域的云计算开发者都能享受到灵活便捷的算力供给，帮助其快速构建和部署各式各样的分布式云服务。因此，虽然泛在云在命名上与泛在计算（Ubiquitous Computing）有一定相似度，但本质上是 2 个完全不同的概念。后者由施乐公司帕洛阿尔托研究中心的马克·维瑟（Mark Weiser）于 1988 年提出，强调计算设备融入人们生活环境，通过日常活动中的自然交互持续优化人机体验。

泛在云所代表的数据中心布局的深刻变化给云计算带来了新的机遇与挑战。一方面，云服务部署与用户的物理空间距离更近，使得访问延迟大幅降低，同时能够极大地提高数据隐私保护能力，更好地满足自动驾驶、低空经济等新兴应用在服务稳定性、安全性等方面的需求。另一方面，更复杂的数据中心布局对于应用的性能和成本优化带来新的挑战，计算、网络和存储资源的分配问题，多数据中心间的调度和协同问题，应用服务水平协议（service level agreement, SLA）的保障问题等，使得每个应用单独开发面临巨大的云成本负担。将这些复杂问题从应用层下沉到云服务商侧已成为发展趋势，而基于当前流行的服务器无感知计算（Serverless Computing）解决方案，将服务部署从数据中心云向泛在云扩展，设计承载泛在云高复杂度的创新编程范式和开发框架则是务实有效的路径。

Serverless 计算作为近年来新兴的云计算编程范式，致力于向用户提供编程友好型和免资源运维的云计算服务模式。函数即服务（function as a service, FaaS）是当前 Serverless 计算在通用编程领域的典型代表产品，其提供了一种无状态的函数式编程模型，开发人员仅需要将应用编写为一组可由事件触发的业务函数并上传代码，函数的构建与运行管理（例如环境配置、扩缩容、监控等）则全部由 FaaS 平台负责，同时按照调用次数和实际资源使用量进行计费。凭借上述优势，自 2016 年 AWS 率先推出 Lambda 函数计算服务之后，Serverless 计算的热度不断攀升，目前已在包括 Web 类服务[3]、数据分析查询[4-5]、物联网[6]、机器学习训练和推理[7-8] 等诸多业务场景中得到了广泛应用。如今很多知名云服务商都提供了 FaaS 产品，例如 Google Cloud Function、Microsoft Azure Function、阿里云函数、天翼云函数等，业界已普遍认为 Serverless 计算将成为未来云计算的标准范式。

Serverless 编程模型天然契合泛在云场景下的应用开发部署。然而，现阶段的 FaaS 开发框架在跨域资源整合、算力灵活调度以及用户低时延 SLA 保障方面仍存在不足，需要进一步完善以适配泛在云场景下的业务需求。接下来，本文将分析泛在云场景下的业务特征与 FaaS 编程框架现有能力的匹配程度，并探讨如何基于原生 Serverless 技术构建面向泛在云场景的分布式应用开发和部署解决方案，从而充分发掘泛在云基础设施的应用潜力，为未来的 Serverless 计算发展方向提供建议。

泛在云场景应用特征与需求

以 5G 网络云化、云电脑为代表的运营商主流业务催生了分层多级的泛在云基础设施格局。反过来，泛

在云无处不在的可扩展算力供给能力和“近用户侧”的低时延服务优势也赋予了云应用开发部署更多潜力。结合对当前及未来云应用流行度和运营商自身业务的驱动力评估，本文列举了几项具有代表性的泛在云业务场景。

例如在机器学习方法，由于尺度定律（scaling law）的持续演进，以大语言模型为代表的 AI 技术对算力的需求也与日俱增，目前先进的大模型训练集群规模已达到十万图形处理器（graphics processing unit, GPU）量级。受制于单个数据中心内电力供应、计算能力等因素的约束，利用大量跨地域、多层级的计算节点进行协同训练有望缓解上述问题，目前已有学术界和产业界的研究工作正在探索兼顾成本和性能的跨域 AI 分布式训练方法。

网络云化也是一个典型的泛在云业务场景。多级分层算力架构的优势之一在于可以根据用户位置提供满足不同时延 SLA 需求的服务。例如区域数据中心与用户交互的网络时延通常小于 50 ms，潜边缘交互延迟小于 10 ms，深边缘节点则小于 5 ms。通过将 5G 核心网等运营商业务的控制面功能（例如用户接入、权限校验等）进行泛在云化部署，利用靠近用户的边缘节点处理简单即时业务，利用深边缘节点进行数据存储、分发和任务协同，从而向亿级用户提供低时延、高弹性的网络服务，也能更好地应对公众热点事件等引发的突发流量场景。

此外，以无人机运送、自动驾驶、智慧城市等为代表的低空经济业务也在泛在云场景中具有广阔的应用空间。例如利用边缘节点与智能设备进行实时交互，提供近距数据感知与快速避障能力，而核心算力枢纽存储和分析海量数据，提供最优路线规划能力。同时也能够借助多智能设备间的协作完成任务分发和卸载，以深度利用周边深浅边缘节点的闲置算力。

除此之外，传统的微服务类应用、大数据存储与检索以及云游戏等同样可以利用泛在云业务部署在性能和成本方面的优势。总结来讲，泛在云场景应用具有以下几大特征需求：

强实时性与低延迟需求　跨地域计算节点协同训练大语言模型需要实时响应参数更新，从而加快训练迭代过程；在 5G 核心网、云电脑、云游戏等领域，需要借助靠近用户的边缘节点提供低时延的通信和交互服务（访问时延通常小于 50 ms）；而无人机飞控和自动驾驶则更需要与基站或周边设备进行超低时延通信（通常为毫秒级），从而规划行进路线，规避碰撞风险。

网络化分布式部署需求　运营商类业务例如 5G 网络等需要天然地分散部署在分层多级的基础设施网络之上，从而向不同地域的用户提供服务。大语言模型训练等应用依赖大量计算资源，通常以分布式形式部署到多个算力节点以加速运行。此外，一些敏感数据受到隐私性和安全等限制存储在不同的地域，也需要将服务进行分布式网络化部署以进行本地分析和处理。

边缘算力任务卸载需求　在大模型推理场景中，用户终端内的智能体服务通常依赖边缘算力节点进行任务卸载，例如借助边缘 GPU 设备满足自身迭代所需要的算力需求。运营商领域的网关控制模块也可以从家庭路由器迁移至边缘节点，从而实现统一管理和升级维护。在无人机控制和自动驾驶领域，都需要借助设备和边缘计算节点的交互完成大量计算任务的卸载和协作。

值得注意的是，由于上述业务应用分属不同的领域，在应用开发模式和部署方式存在较大区别，单独开发每个应用势必带来巨大的成本负担。此外，如何在分层多级的泛在云架构优化服务的部署和运行效率（例如资源选型，部署位置等），也要求开发者具备成熟的开发经验，这也无疑提高了泛在云场景中的分布式应用开发门槛和复杂度。

因此，当前阶段迫切需要一套面向泛在云业务场景的编程模型以实现高效的资源管理和服务性能保障能力。该编程模型扮演着“泛在云操作系统”的角色，向上提供友好的用户编程接口，满足泛在云业务的开发和部署需求。向下提供跨域资源管理和算力统一抽象机制，将来自不同地域、不同节点、不同硬件的算力统一纳管，满足业务的多样化算力需求。此外，针对泛在云业务负载高动态变化导致的算力供给复杂性和不确定性，还需要提供高度灵活的资源分配和任务编排调度能力、低时延通信能力以及延迟 SLA 保障能力。

现有解决方案的不足

然而，现有的分布式云编程框架并不能满足上述需求，目前业界在该领域的研究也尚属空白。当前在云计算领域已有大量的分布式应用开发框架，例如面向并行计算编程的消息传递接口（message passing

interface, MPI), 面向图计算的 GraphX 和面向机器学习应用开发的 TensorFlow 等专用领域编程框架, 以及类似 MapReduce、Ray 的通用编程框架, 用于帮助用户构建复杂的分布式计算应用。这些分布式框架通常内置了资源管理模块, 例如 Apache Hadoop 所使用的 YARN; 或者借助一些当前流行的分布式资源管理框架, 例如 Apache Mesos、容器编排框架 Kubernetes 等。尽管现有的分布式编程模型经过不断地迭代优化, 在多语言、跨平台以及多种内置算法库等支持方面已相对完善, 然而距离本文的泛在云编程框架构想还有一定差距, 主要体现在以下 2 个方面:

跨域/业务集群部署协作难 现有的编程模型通常仅限于在单集群内部署运行, 缺乏对跨域资源编排管理和任务调度的支持。尽管国内外已有一些针对跨云算力协作的探索, 例如国内的 CloudExplorer 多云协作平台[9], 加州大学伯克利分校的 SkyPilot 项目[10] 等, 但这些平台目前只初步扮演“算力商店”的角色, 多云之间缺乏跨集群的统一资源表达和数据管理抽象。此外, 跨域间的多层多级复杂网络转发、大流量收发以及网关故障也会增加任务间的通信时延, 降低编排调度的灵活性, 这也使得现有框架无法有效支撑跨域编程。

配置复杂导致学习成本高 泛在云业务场景的受众群体可能来自不同行业, 拥有不同的技术开发能力和编程喜好。现有的编程框架学习成本依然较高, 例如 Map 和 Reduce 接口定义、Spark 框架的数百个调优参数以及 MPI 的并发任务拆分等, 都需要一定的经验和专业能力, 给泛在云架构下分布式应用开发带来了门槛。此外, 一些编程框架还需要开发者显式地管理资源配额, 配置运行环境以及设计任务调度和扩缩容策略等, 这些都会加大应用开发者的负担和运行成本。

面向泛在云的 Serverless 编程模型

Serverless 计算则为泛在云业务场景的应用编程带来了新思路。尽管当前的 FaaS 平台大多以数据中心云架构部署为主, 但已有不少研究者针对边–云场景下的 Serverless 计算适配展开探索, 例如可以对 FaaS 框架进行轻量化改造以运行在资源受限的边缘节点, 或结合软硬件协同设计实现边缘场景下函数任务的灵活分发与卸载。此外, 还可以借鉴“网格计算”思想设计联邦 FaaS 编程模型, 从而支持跨域函数工作流的开发和部署(例如 funcX 项目, 又名 Globus computing[11])。这些研究表明设计一套面向泛在云业务场景的 Serverless 编程框架具备技术可行性。

在泛在云场景中, 底层的基础设施具有高度异构性(例如计算设备类型, 边缘节点容量等)以及跨广域网络分布的特点, 而计算环境由于用户群体的强流动性、业务多样性以及流量的高动态性变得更为复杂。如何打造一个能让用户便捷获取算力的体系, 满足不同业务的多样化算力需求, 使得身处不同地域的用户都能享受到高品质云计算服务是 Serverless 编程模型需要解决的难题。

针对该问题, 图 2 给出了面向泛在云的 Serverless 编程模型设计构想。

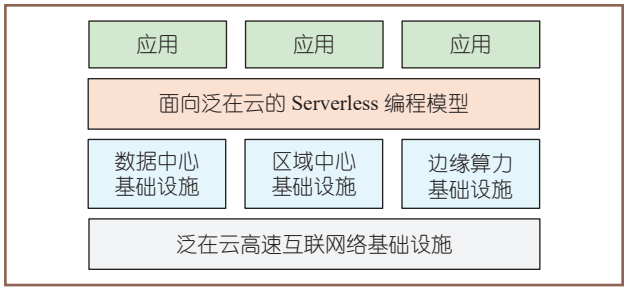


图2 面向泛在云的 Serverless 编程模型

图 2 的编程模型应满足以下目标以克服现有方案的不足:

1)允许用户仅关注业务代码, 借助丰富算子库提供通用编程能力, 支持各种泛在云业务场景;

2)具备高度可扩展性, 能够跨越数据中心以及边缘基础设施实现任务的分发和调度;

3)轻量化及低运行开销, 能够实现快速的任务启动、迁移与释放, 应对瞬息万变的流量和资源变化;

4)结合性能建模、高速通信机制以及智能化服务编排技术保障用户多样化的延迟 SLA 目标。

除此之外, 还要向用户和开发者提供细粒度按需计费、容错性以及可靠性等方面的能力。

需要说明的是, 现阶段 FaaS 编程框架大多基于以 Kubernetes 和容器为主体的云原生软件栈构建, 在运行开销、响应速度、控制能力方面面临一系列的挑战。

表 1 汇总梳理了泛在云场景中分布式应用的运行需求以及当前 Serverless 计算系统的能力，每行代表不同的指标项。本文主要从任务执行时延 SLA 的保障能力，FaaS 函数计算、存储和通信能力以及扩缩容速度方面进行了量化对比，并根据 FaaS 平台的特性能力进行匹配度评估。

表1　泛在云典型业务场景需求与现有 FaaS 平台能力对比

场景及需求指标	AI大模型训练/推理（训练或推理pod）	5G网络云化（控制面网元功能）	无人机、自动驾驶（控制面任务卸载）	现有FaaS平台基础能力	具体量级区间	匹配程度
时延SLA	首推O(100 ms)① 逐字O(10 ~100 ms)①	控制面<10 ms	低空无人机：<20 ms② 自动驾驶：<100 ms②	O(10 ~100 ms) 最大允许运行24 h①	亚毫秒级或毫秒级 数十毫秒级 百毫秒或秒级	× √ √
计算资源	O(1 ~10 cpus)	O(0.1 ~1 cpus)	O(0.1 ~1 cpus)	0.05 ~16 vCPU①	弱算力（低至0.1核） 强算力（百核级）	√ ×
内存资源	O(1 ~100 GB)内存	O(10 ~1 000 MB)	O(10 ~1 000 MB)	128 MB~32 GB①	小内存（低至128 MB） 大内存（百GB级）	√ ×
加速器	以GPU为主	DPU、FPGA等	GPU、NPU、FPGA等	1/16 或整卡GPU①	异构加速器支持	×
数据传输量	单次通信：O(1 ~1 GB)	O(1 ~10 KB)	O(1 KB~1 MB)	同步调用：最大32 MB① 异步调用：最大128 KB①	小通量传输（KB级） 大通量传输（百MB级）	√ ×
通信带宽	O(1 ~10 Gbps)	20 Gbps峰值②	O(10 Mbps~1 Gbps)②	最大5 Gbps①	高速通信能力	×
持久化存储	GB级~TB级	GB级	GB级~TB级	支持对象存储等服务	大/小文件存储能力	√
临时存储	以内存为主	O(10 ~1 000 MB)	O(10 ~1 000 MB)	60 GB临时存储	临时存储能力	√
扩容速度	训练：O(1 ~10 s) 推理：O(100 ~1000 ms)	O(1 ~10 s)①	O(10 ms~10 s)①	单实例并发200 qps[1] 创建实例300/min①	低并发（数十实例/s） 高并发（数百及以上）	√ ×

注：①的数据来源为行业报告、企业白皮书或公开技术文档，②的数据来源为国内外技术规范或标准。O代表数量级范围。√表示FaaS产品能力匹配业务需求，×表示部分满足或不满足需求

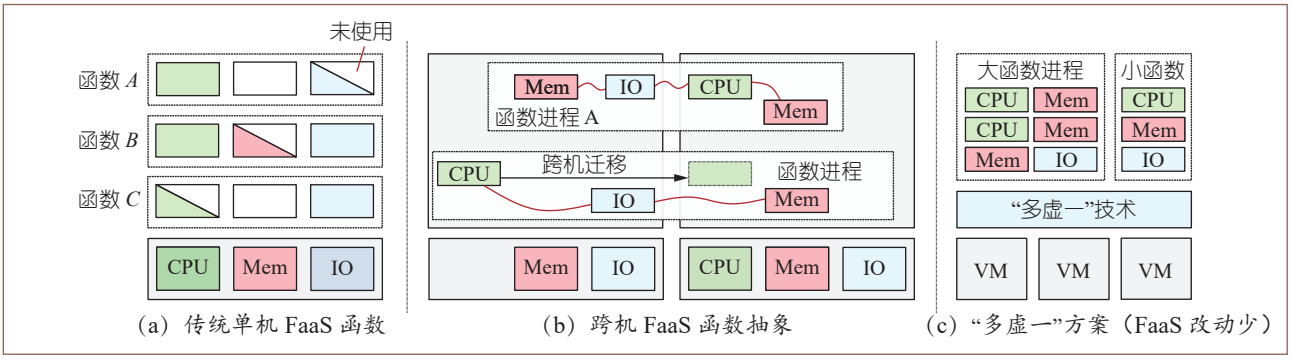
可以看到，当前的 FaaS 编程框架能够满足泛在云业务应用的部分指标需求，但在大算力任务部署需求、超低时延保障能力以及高并发流量支撑方面缺乏适配能力。例如目前小体积的 FaaS 函数尚无法支撑大模型训练任务的拆分封装，FaaS 平台的控制面开销和高转发延迟难以保障亚毫秒级的交互式应用延迟 SLA 目标，FaaS 函数的低并发请求处理能力和冷启动时延也制约着其在高并发场景下的可扩展性。此外，通过对业界现有的 FaaS 平台进行调研分析，可以看到其在多级分层算力基础设施下的跨域调度、多样化应用开发算子库开发等方面仍存在不足。

本文接下来的部分将针对 Serverless 计算目前面临的主要瓶颈，结合业界的最新研究动态来探讨未来的改进空间或可行技术路线。

适配大算力任务需求

由单机函数到跨机函数抽象　泛在云场景下存在大量弱边缘算力节点以及资源碎片，使得 FaaS 函数的计算能力受限。一些具有大算力需求的任务例如大模型训练或推理，任务计算单元通常需要数百颗 CPU 核心和数百 GB 内存（memory, Mem）。因此，如图 3（b）所示，可以借鉴现有跨节点逻辑进程设计理念（例如 ServerlessOS[12] 或 QuickSand[13]）探索跨节点函数抽象机制，将函数运行时所需的 CPU、内存和输入输出（input/output, IO）等资源进行解耦，使得代码执行可以在不同物理节点间灵活地迁移和跳转，一方面能够突破 FaaS 函数算力限制，另一方面也可以充分利用不同物理节点上的闲置资源。相关实验数据表明，借助通信技术的优化，跨机函数迁移执行带来的延迟开销可

[1] 每秒查询请求数量（query per second, qps）



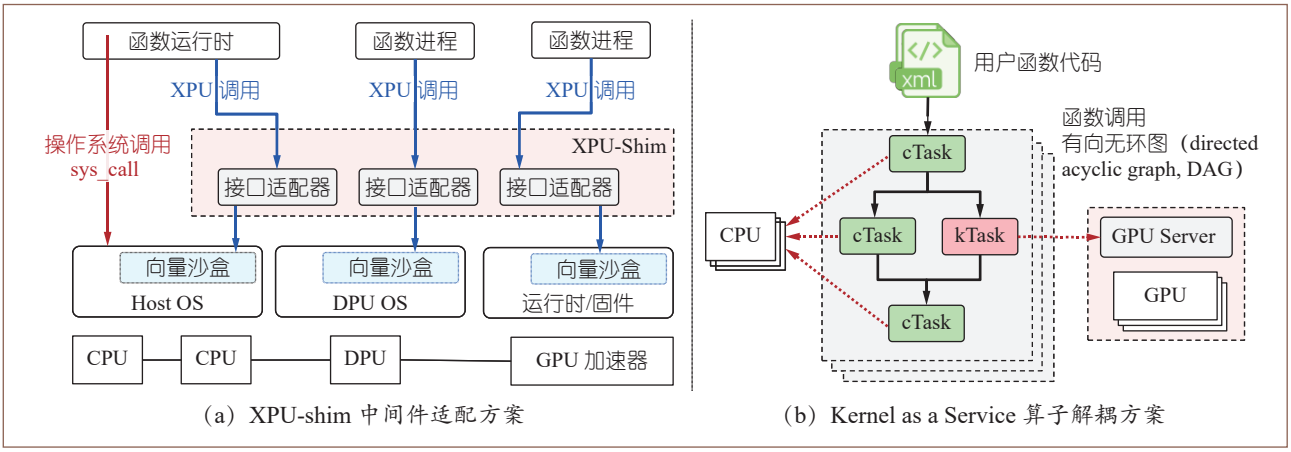
（a）传统单机 FaaS 函数　（b）跨机 FaaS 函数抽象　（c）“多虚一”方案（FaaS 改动少）

图 3　FaaS 跨机大函数实现方案

以降低至亚毫秒级（10~30 μs）[14]，因此利用 FaaS 跨机函数适配大算力任务的运行需求具备可行性。

通常，跨机 FaaS 函数抽象需要借助高速总线或网络加速硬件实现跨机互联和快速任务切换（例如远程直接内存访问技术（remote direct memory access, RDMA），还需要专门设计一套函数协调器实现对位于不同节点上的函数资源的管理和协作，这会增加编程复杂度甚至导致性能瓶颈。另一种可行思路则是结合基础设施及服务（infrastructure as a service, IaaS）层的“多虚一”技术（例如 GiantVM[15]）来创建大算力函数实例，如图 3（c）所示，该架构下 FaaS 函数对底层虚拟化架构无感知，也无需对 FaaS 层软件引入额外修改。实验表明“多虚一”技术具有与传统虚拟机（virtual machine, VM）接近的性能，并能够显著加速 2~3 倍的分布式数据分析任务性能。

从单一算力到异构算力支持　当前 GPU、神经网络处理器（neural processing unit, NPU）、现场可编程门阵列（field-programmable gate array, FPGA）、数据处理器（data processing unit, DPU）等在内的异构计算硬件已经广泛应用于各个云场景。然而，不同设备厂商之间通常具有各自的驱动实现，需要使用特定的软件开发工具包（software development kit, SDK）接口编程，甚至依赖开发者具有特定的领域知识和专家经验。因此，如图 4（a）所示，FaaS 平台可以通过中间层抽象（例如 XPU-Shim[16]）向用户函数提供一套统一的异构算力调用接口，从而屏蔽掉底层异构硬件的实现细节。此外，如图 4（b）所示，也可结合“Kernel as a service”的思想将用户函数进行细分（例如运行在 CPU 侧的 cTask 和 GPU 侧的 kTask[17] 函数），从而与 XPU-Shim 通用加速器适配层（extensible processing unit shim, XPU-shim）搭配使用，以弥补其函数无法同时使用多种类型异构算力的不足。在异构算力虚拟化方面，也可以借鉴 GPU 设备虚拟化的方案，进一步探索 NPU、FPGA 等设备的 FaaS 虚拟化能力。



（a）XPU-shim 中间件适配方案　（b）Kernel as a Service 算子解耦方案

图 4　异构算力统一抽象方案

适配超低时延任务需求

从多层虚拟化到扁平化 FaaS 软件栈 现有的 FaaS 函数多基于微虚拟机(MicroVM)和容器作为载体构建,如图 5(a)所示,无论是函数实例启动还是运行时调用,都需要跨多个虚拟化层执行系统调用、地址转换、内存复制等操作,当前厚重的 FaaS 虚拟化软件栈并不适用于泛在云场景。例如创建一个函数实例通常需要耗时数百毫秒或数秒,而运行一个简单函数可能需要数十到数百 MB 内存的软件栈开销。因此,如图 5(b)所示,可以结合 WebAssembly [18] 和库操作系统 LibOS[19] 技术构建更轻量级、更扁平化的 FaaS 虚拟软件栈(例如 Alloystack[20])。实验表明,通过宿主机 OS 直接管理函数进/线程并结合按需加载等技术,FaaS 函数的冷启动时延可降低至 1 ms,端到端延迟性能超过现有软件栈数十倍。

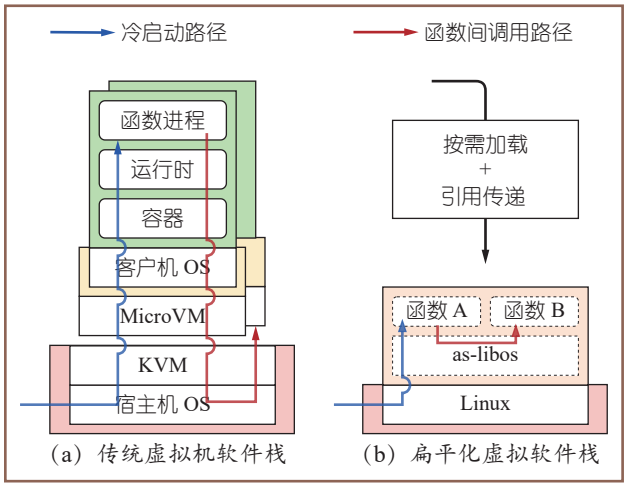


图5 FaaS 函数虚拟化软件栈

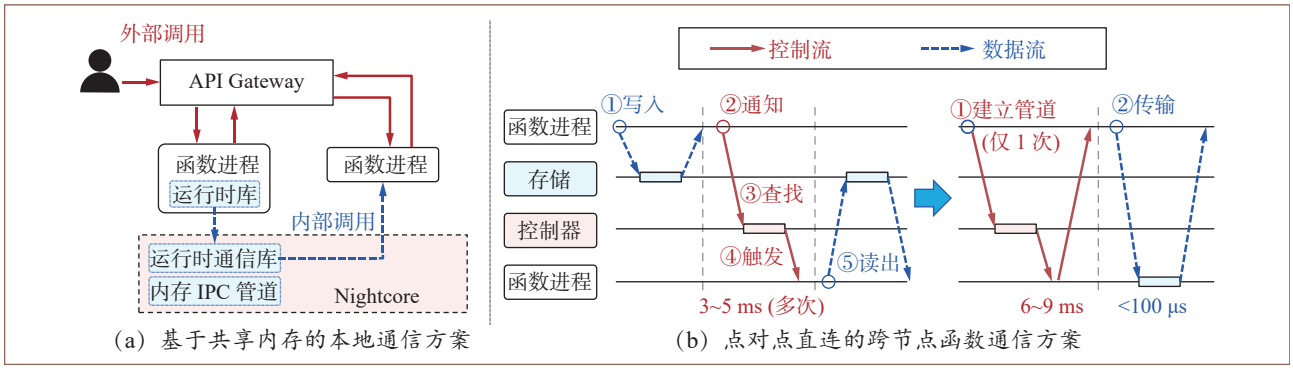
从通用编排框架到定制化原生编排框架 在扁平化 FaaS 虚拟软件栈的基础上,泛在云场景下更大的集群规模对 FaaS 函数的编排调度提出了更高的挑战。华为云的公开 Trace 数据曾显示,其部分数据中心内平均函数实例调度延迟超过 1 s(无法满足一些高并发泛在云业务场景的扩展需求)。其主要原因是 FaaS 平台通常基于 Kubernetes 二次构建,而 Kubernetes 内部的多层复杂抽象、状态同步机制以及频繁的远程过程调用(remote procedure call, RPC)调用容易出现性能劣化,从而拖慢 FaaS 平台扩容速度。因此,可以对现有函数编排框架进行定制化设计(例如 Drigient[21])。相关实验数据表明,通过操作对象和状态精简以及控制/数据面优化设计可大幅降低函数编排层的开销,实现每秒数千实例的并发创建。

从单集群服务管理到跨集群服务网格 现有集群内的通信方案通常以网关、重叠网络、路由、实例端点为体系进行构建,服务的跨集群通信需要进行互联网协议(internet protocol, IP)二次映射、多层网关转发和路由实现,在泛在云大流量业务场景下频繁的重定位和 IP 映射还原等操作可能导致高通信延迟的问题。为了解决该问题,可以通过服务网络(例如 lstio[22])构建跨集群的大二级增强网络,实现跨集群的 FaaS 服务注册与发现能力,降低 FaaS 集群间的通信延迟与运行开销。

适配大通量 IO 任务需求

从控制数据耦合到控制面 by-pass 传输 泛在云场景下一些 IO 型任务通常具有大数据量通信的需求,而现有的 FaaS 平台函数间通信能力较弱。一方面体现在通信控制面的开销,由于应用程序编程接口网关(application programming interface gateway, API Gateway)和相关组件频繁参与请求的流转和函数间调用,从而导致了较大的通信延迟(例如华为云 Trace 显示函数请求在宿主机节点和前端组件间的平均流转时间均超过 100 ms)。在请求流转关键路径上减少或绕过控制面参与(例如 Nightcore[3])或采用本地 IPC 通信可以实现节点内微秒级的函数通信时延(图 6(a))。对于函数间跨节点调用场景,则可以借助 RDMA 或计算快速链接(compute express link, CXL)协议(例如 RMMAP[23], FUYAO[24] 等)实现函数间“点对点”直连通信,从而减少控制面的参与,可将跨节点函数通信延迟降低至约 100 μs(图 6(b))。

从网络通信到内存原生数据面通信 另一方面,现有的 FaaS 平台尚缺乏针对数据面的高效数据传输机制。例如 AWS Lambda 允许函数间通过超文本传输协议(hypertext transfer protocol, HTTP)或 RPC 协议进行直接交互,同时也提供了借助 S3 对象存储服务的间接交互方式用于满足函数间状态保存或持久化的需求,然而外部存储服务通常具有较大的访问延迟和带宽限制(例如读写 1 KB 数据分别耗时 12 ms/25 ms),在一些分布式任务中频繁 IO 会导致大量的通信延迟。通过借助内存型键值存储数据库(读写 1 KB 数据约



(a) 基于共享内存的本地通信方案　(b) 点对点直连的跨节点函数通信方案

图 6　高性能 FaaS 函数间通信方案

200 μs)[25] 或本地共享内存绕过网络协议栈(32 qps 并发下读写 1 KB 数据约 20 μs)[26], 则可以大幅改善本地节点 FaaS 函数间的数据面通信能力。

适配大规模高并发任务需求

从单集群调度到跨域协同调度　泛在云场景中大型分布式任务的部署需求也要求 FaaS 平台具备跨域任务调度能力, 同时能够保障任务端到端性能(例如最小化总体完成时间, 减少平均等待时延等)。可以借鉴现有的联邦式 FaaS 平台的设计思想(例如 UniFaaS[27]), 允许开发者根据任务性能−成本建模来制定任务调度策略, 或结合智能化学习方法、博弈论等研究实现多租户 FaaS 工作流的运行性能优化与延迟控制。

从传统扩容到基于快照的实例创建　泛在云场景中一些边缘业务可能由于跨区域用户或终端设备的高流动性产生大型突发流量, 因此 FaaS 平台应该具备快速 0—1 和 1—*N* 的函数实例创建能力。前文提到的扁平化 FaaS 软件栈和轻虚拟化技术可以进一步结合快照机制(例如 sfork[28]、remotefork[29] 等)加快函数实例的创建, 实验表明, 相比从零冷启动过程, 利用快照创建函数实例可有效降低函数启动时延 1 个数量级。此外, 通过实例预留、负载预测等函数实例缓存策略也可以减少大量冷启动调用的发生, 从而降低 Serverless 函数的扩缩容压力。

系统设计构想与应用场景

基于上述关键技术的设计思路, 图 7 给出了泛在云场景下基于 Serverless 的跨域用户无感知的编程框架体系架构。如图 7 所示, 自底向上为基础设施层、编程框架层以及用户业务层。其中, 扁平化的 FaaS 函数虚拟化软件栈和跨域任务调度及资源编排在中间层扮演着重要角色, 其具备如下能力: 通过①跨机函数抽象与快速启动能力提供资源供给弹性和灵活性; 通过②控制面解耦的内存原生通信系统提供跨集群/节点/实例的高性能通信能力; 通过③高扩展的服务编排及智能化函数调度实现跨域资源的灵活调配和高效管理; 通过④多样化时延服务质量目标(service level objective, SLO)保障能力向泛在云不同业务场景的用户提供高品质云计算服务。

在该体系架构下, 面向泛在云场景设计的 FaaS 编程框架将结合现有的先进技术和设计理念, 将泛在云场景下的分层多级计算基础设施的能力充分调动起来, 支撑小到一个临时用户测试任务, 大到分布式跨域应用的快速、高效及灵活部署, 解决泛在云场景中分布式应用开发及运行时管理的高复杂度问题。

总结与展望

在云计算从传统数据中心云到泛在云演变的大背景下, 本文讨论了如何基于 Serverless 编程范式来应对泛在云场景下分布式应用开发部署以及资源管理面临的高复杂度挑战。结合当前中国电信的“云网融合”一体化智能调度的发展战略, 如何以泛在云网基础设施为底座, 通过云计算体系和系统层面创新实现无处不在、高效智能、安全可靠云计算服务的目标也驱动了本文中面向下一代云计算编程模型和资源管理的思考。

智能泛在云的展望　如今随着 AI 技术的飞速发展, 智能化已经与云计算变得密不可分。一方面, 泛在云架构、编程模型等系统层的演进将使得其能够有效承载包括大模型训练、自动驾驶等在内的智能化应用和新兴应用的部署需求, 充分释放泛在云的计算潜力。另一方面, 诸如用户行为预测、智能调优以及智能

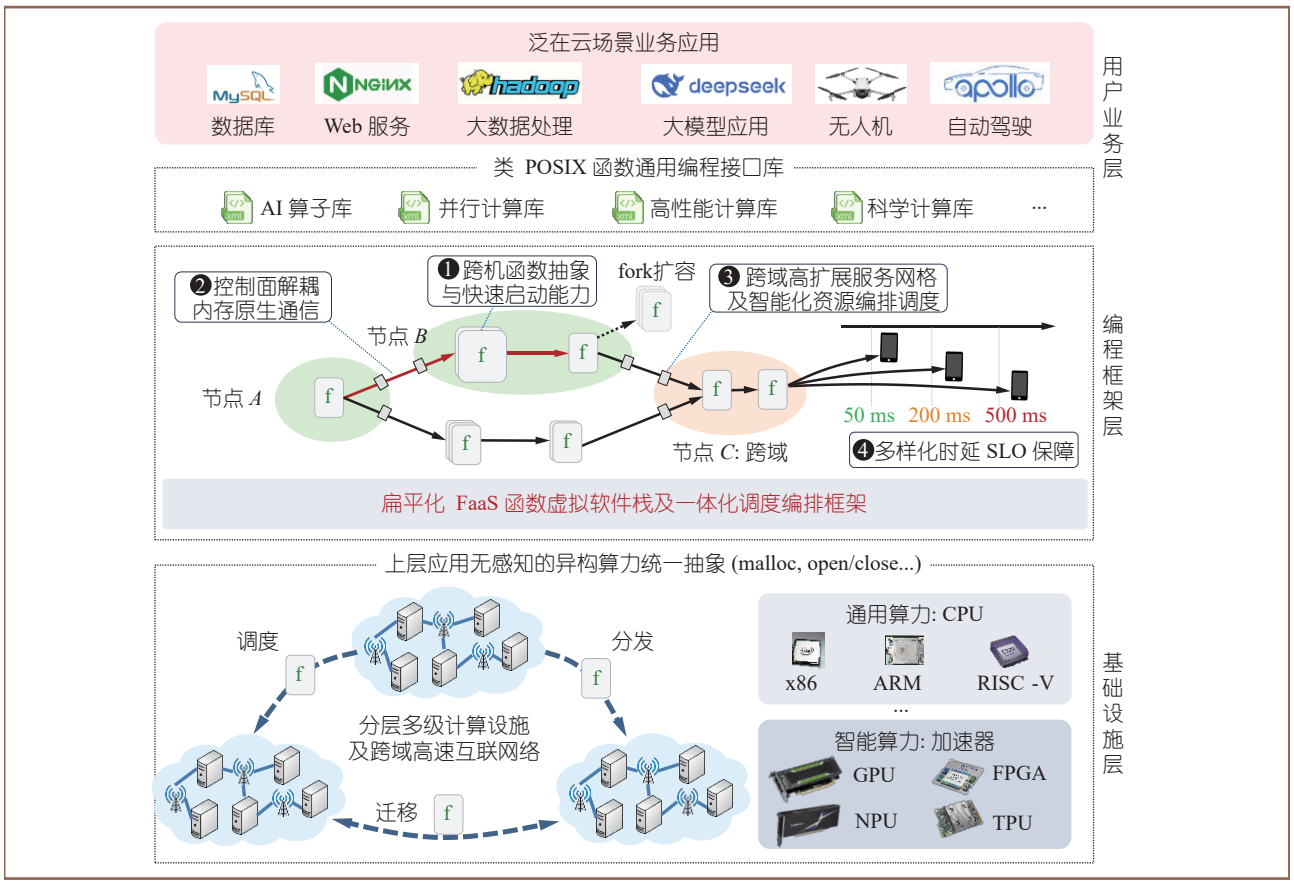


图 7　泛在云场景下的跨域用户无感知的编程模型体系结构

化编排调度等也能够用于泛在云的业务场景，改善泛在云业务应用的运行效率。例如，通过机器学习等方法实现对业务流量的特征预测，优化指导 FaaS 函数的扩缩容策略以降低冷启动[30]。也可以结合深度学习等方法对泛在云场景下的任务工作流进行建模，在满足端到端延迟约束的前提下减少用户的运行成本[31]。此外，还可以结合智能运维等方法优化大规模分布式任务的运行效率[32]。这些面向人工智能的计算机系统研究（System for AI）和 AI 辅助的系统优化技术（AI for System）共同组成了智能泛在云的内涵，并将推动泛在云架构继续朝向智能泛在云的方向演进，也是本文未来的探索方向。■

致谢：本篇文章也要感谢来自天翼云科技有限公司的高键总及中国电信上海市分公司的许浩专家，他们曾为本文的撰写给出了富有价值的指导建议。中国电信云计算研究院的全硕研究员也提供了参考资料。在此一并致谢！

杨亚南

CCF 专业会员。中国电信云计算研究院研究员，ACM SIGOPS 中国分会 (ChinaSys) 优博。主要研究方向为下一代云与 Serverless 计算。

yangyn11@chinatelecom.cn

张健松

CCF 高级会员。中国电信云计算研究院资深主任研究员。主要研究方向为云计算、体系结构与软硬件联合设计。

Zhangjs15@chinatelecom.cn

吴　杰

CCF 会士、2011 年 CCF 海外杰出贡献奖获得者。IEEE Fellow, AAAS Fellow 和欧洲科学院院士。中国电信股份有限公司首席科学家。主要研究方向为网络与分布式计算。

wujie@chinatelecom.cn

参考文献

[1] 中华人民共和国中央人民政府. 政府工作报告: 2024 年 3 月 5 日在第十四届全国人民代表大会第二次会议上 [EB/OL]. (2024−03−05)[2025−06−18]. https://www.gov.cn/yaowen/liebiao/202403/content_6939153.htm.

[2] 刘志军, 周国强, 雷波, 等. 云网融合 [M]. 北京: 人民邮电出版社: 2023.

[3] Zhipeng Jia, Emmett Witchel. Nightcore: Efficient and Scalable Serverless Computing for Latency-Sensitive, Interactive Microservices[C]//*Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. New York: ACM, 2021: 152−166.

[4] ONAS E, Qifan Pu, VENKATARAMAN S, et al. Occupy the Cloud: Distributed Computing for the 99%[C]//*Proceedings of the 2017 Symposium on Cloud Computing*. New York: ACM, 2017: 445−451.

[5] Rohan Basu Roy, Tirthak Patel, Richmond Liew, et al. Propack: Executing Concurrent Serverless Functions Faster and Cheaper[C]//*Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*. New York: ACM, 2023: 211−224.

[6] Bin Wang, Ahmed Ali-Eldin, Prashant Shenoy. LaSS: Running Latency Sensitive Serverless Computations at the Edge[C]//*Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*. New York: ACM, 2021: 239−251.

[7] Yanan Yang, Laiping Zhao, Yiming Li, et al. INFless: A Native Serverless System for Low-latency, High-throughput Inference[C]//*Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. New York: ACM, 2022: 768−781.

[8] Diandian Gu ,Yihao Zhao, Yinmin Zhong, et al. Elasticflow: An Elastic Serverless Training Platform for Distributed Deep Learning[C]//*Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*.New York: ACM, 2023: 266–280.

[9] 飞致云, 天翼云息壤一体化智算服务平 [EB/OL]. (2014−10−1)[2025−8−5]. https://www.fit2cloud.com/index.html.

[10] Zongheng Yang, Zhanghao Wu, Michael Luo, et al. Skypilot: An Intercloud Broker for Sky Computing[C]//*20th USENIX Symposium on Networked Systems Design and Implementation*. Boston: USENIX Association, 2023, 437–455.

[11] Ryan Chard, Yadu Babuji, Zhuozhao Li, et al. Funcx: A Federated Function Serving Fabric for Science[EB/OL]. (2020−05−07)[2025−06−18]. https://arxiv.org/abs/2005.04215v1.

[12] Zaid Al-Ali, Sepideh Goodarzy, Ethan Hunter, et al. Making Serverless Computing More Serverless[C]//*Proceedings of 2018 IEEE 11th International Conference on Cloud Computing* . Piscataway: IEEE, 2018: 456−459.

[13] Zhenyuan Ruan, Shihang Li, Kaiyan Fan, et al. Unleashing True Utility Computing with Quicksand[C]//*Proceedings of the 19th Workshop on Hot Topics in Operating System*. New York: ACM, 2023: 196–205.

[14] Diego Ongaro, Stephen M. Rumble, Ryan Stutsman, et al. Fast Crash Recovery in RAMCloud[C]//*Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. New York: ACM, 2011: 29−41.

[15] Jin Zhang, Zhuocheng Ding, Yubin Chen, et al. GiantVM: A Type-II Hypervisor Implementing Many-to-One Virtualization[C]//*Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. New York: ACM, 2020: 30−44.

[16] Dong Du, Qingyuan Liu, Xueqiang Jiang, et al. Serverless Computing on Heterogeneous Computers[C]//*Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. New York: ACM, 2022: 797−813.

[17] Nathan Pemberton, Anton Zabreyko, Zhoujie Ding, et al. Kernel-as-a-Service: A Serverless Interface to GPUs [EB/OL]. (2022−08−15)[2025−06−18]. https://arxiv.org/abs/2212.08146v1.

[18] Phani Kishore Gadepalli, Sean McBride, Gregor Peach, et al. Sledge: A Serverless-first, Light-Weight Wasm Runtime for the Edge[C]//*Proceedings of the 21st International Middleware Conference*. New York: ACM, 2020: 265−279.

[19] Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, et al. Unikernels: Library Operating Systems for the Cloud[C]//Architectural Support for Programming Languages and Operating Systems, Houston : ACM, 2013: 461–47.

[20] Jianing You, Kang Chen, Laiping Zhao, et al. AlloyStack: A Library Operating System for Serverless Workflow Applications[C]//*Proceedings of the Twentieth European Conference on Computer Systems*. New York: ACM, 2025: 921−937.

[21] Lazar Cvetković, François Costa, Mihajlo Djokic, et al. Dirigent: Lightweight Serverless Orchestration[C]//*Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*. New York: ACM, 2024: 369−384.

[22] Google, IBM, Lyft. Istio - connect, secure, control, and observe services[EB/OL]. (2017−05−24)[2025−06−18]. https://istio.io/.

[23] Fangming Lu, Xingda Wei, Zhuobin Huang, et al. Serialization/Deserialization-Free State Transfer in Serverless Workflows[C]//*Proceedings of the Nineteenth European Conference on Computer Systems*. New York: ACM, 2024: 132−147.

[24] Guowei Liu, Laiping Zhao, Yiming Li, et al. FUYAO: DPU-Enabled Direct Data Transfer for Serverless Computing[C]//*Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. New York: ACM, 2024: 431−447.

[25] Ana Klimovic, Yawen Wang, Patrick Stuedi, et al. Pocket:

Elastic Ephemeral Storage for Serverless Analytics[C]// *13th USENIX Symposium on Operating Systems Design and Implementation,* Carlsbad: USENIX Association, 2018: 427–444.
[26] Peter Pietzuch, Simon Shillaker. Faasm: Lightweight Isolation for Efficient Stateful Serverless Computing[C]// *Proceedings of the 2020 USENIX Annual Technical Conference*. Berkeley: USENIX Association, 2020: 419–433.
[27] Yifei Li, Ryan Chard, Yadu Babuji, et al. UniFaaS: Programming Across Distributed Cyberinfrastructure with Federated Function Serving[C]//2024 IEEE International Parallel and Distributed Processing Symposium. San Francisco: IEEE, 2024: 217−229.
[28] Dong Du, Tianyi Yu, Yubin Xia, et al. Catalyzer: Sub-Millisecond Startup for Serverless Computing with Initialization-Less Booting[C]//*Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. New York: ACM, 2020: 467−481.
[29] Wei Xingda, Lu Fangming, Wang Tianxia, et al. No provisioned concurrency: gast rdma-codesigned remote fork for serverless computing[C]//17th USENIX Symposium on Operating Systems Design and Implementation. Boston: USENIX Association, 2023: 497–517.
[30] R B Roy, T Patel , D Tiwari. Icebreaker: Warming Serverless Functions Better with Heterogeneity[C]// *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. New York: ACM, 2022: 753–767.
[31] Zhuangzhuang Zhou , Yanqi Zhang , Christina Delimitrou. AQUATOPE: Qos-and-Uncertainty-Aware Resource Management for Multi-stage Serverless Workflows[C]// *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. New York: ACM, 2022: 1–14.
[32] Li Yu, Zhiling Lan. A Scalable, Non-Parametric Method for Detecting Performance Anomaly in Large Scale Computing[J]. *IEEE transactions on parallel and distributed systems*, 2016, 27(7): 1902−1914.

Serverless Computing with Ubiquitous Cloud: Opportunities and Challenges

Ya'nan Yang, Jiansong Zhang, Jie Wu
China Telecom Cloud Computing Research Institute

Abstract: The rapidly emerging multi-layer hierarchical cloud and network infrastructures have enabled cloud computing to evolve from a data center-centric model to the ubiquitous cloud. Serverless computing, also known as "Function as a service (FaaS)", inherently aligns with the requirements of ubiquitous cloud applications. However, existing FaaS programming frameworks exhibit limitations in large-scale cloud service deployment, geo-distributed resource management, and function runtime abstraction. For example, the constrained CPU resources of individual functions struggle to meet the demands of complex tasks, and there is a lack of industry practice in cross-region function scheduling with heterogeneous hardware. Moreover, inter-function communication overhead and long cold-start latency can significantly degrade the user experience. This leads to a challenging and urgent problem for cloud providers to exploit the potential of serverless computing in such scenarios. In this paper, we present several system-level analyses and discuss the design of a generic proposed cross-region FaaS framework that aims to facilitate the deployment of distributed cloud services under SLA guarantees while achieving high resource utilization on ubiquitous cloud infrastructures. Finally, we further propose a future development of intelligent ubiquitous cloud ecosystems.

Keywords: Ubiquitous Loud; multi-layer hierarchical cloud infrastructure; serverless computing; distributed system; cloud programming paradigm; region-less scheduling; latency service level agreement guarantee

摘　要: 算力和网络基础设施的异构化、大规模化和跨地域化的快速发展使得云计算从数据中心云向泛在云的方向演变，Serverless 计算作为新兴的函数式编程范式天然契合泛在云业务场景。然而，现阶段的 Serverless 编程框架在大规模服务构建、跨地域资源调度以及函数运行时优化等方面仍存在不足，例如有限的单函数计算能力难以支撑大型复杂任务的计算需求，跨地域的异构资源整合和函数调度方面仍缺乏产业实践，同时函数间的通信开销和长冷启动时延问题影响用户体验，需对当前的 Serverless 编程框架进一步改进以适配泛在云业务场景需求。针对上述问题，本文初步探讨了如何基于 Serverless 计算构建面向泛在云的跨域通用任务编程框架，以实现服务水平协议（service level agreement, SLA）保障下的分布式云服务开发部署和分层多级算力基础设施的高效利用，最后也展望了对未来智能泛在云的发展构想。

关键词: 泛在云；分层多级算力；serverless 计算；分布式系统；云编程范式；跨域调度；时延服务等级协议保障
中图分类号: TP31
中文引用格式: 杨亚南, 张健松, 吴杰. 泛在云场景下 Serverless 计算的机遇与挑战 [J]. 计算, 2025, 1(4): 1−11.
英文引用格式: Ya'nan Yang, Jiansong Zhang, Jie Wu. Serverless Computing with Ubiquitous Cloud: Opportunities and Challenges [J]. *Computing Magazine of the CCF*, 2025, 1(4): 1−11.

（本文责任编委：**翟季冬**）