

云原生

Cloud Native

2023.07
第7期



供创原会会员
内部交流使用

创原会 CLOUD NATIVE
ELITE CLUB

本期合作单位

中国信通院 CNCF(云原生计算基金会)
华为云 人民邮电出版社 哔哩哔哩技术团队

主 编

创原会理事会

编委会

蔡适择 曹建农 曹 杰 曾振波 程 俊
慈轶恒 丁维龙 杜頔康 杜 岚 冯 嘉
黄 俊 李炳国 李祥乾 李 洋 林汉生
刘会福 刘克兴 刘如明 鲁 阳 马景贺
沈 剑 王雷博 王尚广 王泽锋 徐中虎
许锋涛 于 淇 袁曙光 张陈毅 张嘉伟
张竞宇 张 荣 张水涛 张宇彤 赵 洲
赵卓峰 周丹颖 周 峰 左 玥

本期专栏作者

程俊 美的楼宇科技研究院研发中心负责人
黄俊 招商证券云原生转型项目调研负责人
沈剑 广联数科 CTO
赵洲 浙江大学教授

编 辑

毛 杰 谢 黎 李 宸 林奇炼

*排名不分先后



创原会官网



创原会小助手

目 录

CONTENTS



行业动态

03-18



技术热点

21-38



行业趋势

41-60



实战分享

63-82



人物专访

85-89





产业动态

标准与开源动态	03-05
云原生发展进入爆发期, 将成为数字世界操作系统	06-08
华为云发布面向消费终端的企业云原生白皮书, 开辟移动时代的云原生路径	09-12
CNCF CTO Chris 答创原会会员问 (节选)	13-16

云原生 Cloud Native

标准与开源动态



中国信通院联合发布《企业数字化转型技术发展趋势研究报告》

报告从企业数字化转型技术的发展趋势、技术应用保障体系、技术转型路径需求、构建转型技术发展体系做了系统性论述，提出未来技术发展的五大趋势，并对每一项技术进行了深入详细的探讨，提出了应用解决方案和实现数字化转型的路径建议，希望为我国产业数字化转型提供有力支撑。报告认为，企业可以通过以下五种路径来高效赋能数字化转型：1) 提升企业基础能力：混合云和数字化PaaS平台将成为企业数字化的基础能力；2) 加速数据资产沉淀：更智能的在线连接和协同加速数据流通，提升整个供应链的数据价值；3) 实现数据价值：业务牵引的精细化数据治理和智能化数据平台建设，助力企业实现数据价值。4) 提升企业决策能力：建设企业智慧大脑，用数据驱动实现企业决策，提升主动服务能力。5) 拓展市场机遇，用无代码和AI中台，增强数据清洗、对齐、分析、决策能力，抓住转瞬即逝的市场机遇。



信通院 × 华为云联合发布《面向消费终端的企业云原生白皮书》

白皮书首次全面介绍了华为终端云服务的云原生改造和实践经验，为更多企业提供面向消费终端的云原生转型路径，助力广大面向消费终端的企业实现高水平数字化转型，促进产业高质量发展。华为终端云云原生工程平台历经六年左右的发展，以效率提升、成本优化、开箱即用、最优体验为关键目标，构建了3个软件工程能力（DevOps、DataOps、MLOps）。通过DevOps端到端工具链为用户提供便捷接入和使用，引入Iac（基础设施即代码）和Bac（构建即代码）的能力实现全流程的自动化，让开发人员更专注业务代码实现。云原生DataOps平台可以实现数据的高效共享和数据作业的持续快速交付，极大提高使用数据的效率并降低使用数据的门槛。云原生MLOps平台通过与现有的DevOps流程和工具相结合，简化端到端机器学习生命周期，提供一个企业内各角色无缝协作的平台，让业务、数据、算法和运维等角色能更高效地协作，提升业务价值产出。FinOps要求企业在结合财务及运营方面不同关注点的基础之上进行能力建设，结合财务人员和运营人员的关注点，实现预算控制、成本分摊、资源优化、架构优化、性能优化等方面需求。





IDC 发布《AI 大模型在工业应用展望, 2023》报告

近日, IDC发布《AI大模型在工业应用展望, 2023》报告, 对AI大模型相关概念进行区分, 总结了AI大模型应用于工业的两条路径, 以及在生产制造、研发设计和经营管理三个领域潜在的应用场景。报告认为, 大模型在工业的应用主要分2个方向, 一是增强场景模型的泛化能力, 提升模型适用性, 二是利用自然语言对话和内容生成能力变更应用交互方式, 生成文档、报表等, 而大模型在工业的应用, 目前在生产制造、研发设计和经营管理领域都形成一些场景, 但都处于起步和探索阶段。大模型未来有望在3个方面对工业进行提升, 包括: 很多企业级应用将增加AI助手, 提升工作效率、企业围绕AI构建知识库驱动持续运营、AI大模型泛化能力提升, AI技术服务商产品适应性更强。同时, 大模型在工业应用也面临4项挑战, 分别是: 数据内容可靠无误对技术的挑战、成本高、与现有企业各方应用的对接和集成、面向具体应用场景的模型开发和积累依然不可避免。



艾瑞网发布《2023 年中国科技与 IT 十大趋势》

报告指出, 2023年中国科技与IT十大趋势包括:

- » “多云混合”向“一云多态”进化: 分布式云加速发展, 云能力从“中心辐射”到“传递下发”;
- » 云基础从软件优先到软硬兼施: 为进一步释放数智平台能力, 云厂商需提升基础硬件资源质量;
- » 更多云产品将以Serverless形式交付: All-on-Serverless理念进一步向AI、中间件等模块渗透;
- » 预训练大模型作为AI基础设施加速应用: 提速AI工业化进程, 为AI开发效率加码杠杆, 倍数释放生产力;
- » AIGC在概念爆发后进入产品化尝试: 提升数字化内容生产质效, 变革人机交互体验;
- » 数据采集由单点走向泛化: 泛化数据采集加速提升数据处理方式的精准性与高效性;
- » 工业数字孪生加强物理与虚拟空间交互: 提升工业数字孪生诊断预测能力, 推动业务全流程闭环优化;
- » 数智反向融合, 形成以智养智的正反馈: AI开始反哺数据治理, 进一步促进人工智能进入小场景;
- » 数据和应用将进一步实现无感知闭环: 应用数据自动治理落入数据层, 数据分析嵌入到事务型应用;
- » “数字碳中和”从概念走向市场: 绿色低碳日益渗透市场环境, 逐步成为数字产业增量新动力。





Istio 项目正式毕业

北京时间7月12日晚, CNCF 宣布Istio项目正式毕业。深耕服务网格, 华为云与Istio社区一起见证了Istio从CNCF的毕业。2022年9月份CNCF 技术监督委员会 (Technical Oversight Committee, TOC) 刚刚接受Istio的孵化申请, 今天在还未满一年之际, CNCF技术监督委员会又通过了Istio的毕业申请, 充分可见CNCF以及业界对服务网格Istio的认可。CNCF积极推动 Istio 与 Envoy 项目紧密协作, 一起构建云原生应用流量管理的基础设施, 此举也将进一步促进 Istio 成为服务网格的事实标准。作为服务网格领域最具影响力的项目, Istio 从诞生之初就获得了技术圈和产业界的极大关注。CNCF 近年来连续的调查报告显示, Istio 一直是生产环境下最受欢迎和使用最多的服务网络 -> 服务网格方案。2023年, 微软宣布停止维护OSM (Open Service Mesh) 项目, 全面拥抱Istio, 并且OSM项目核心的维护者全部投入Istio社区。微软成为继谷歌、IBM、华为、SalesForce、Intel等超大型具有业界影响力的企业之后, 又一家全面接入Istio服务网格生态的顶尖云厂商。此消息对Istio的毕业具有积极的推动作用。



Karmada v1.6 发布! 跨集群弹性伸缩难题迎刃而解

Karmada 是开放的多云多集群容器编排引擎, 旨在帮助用户在多云环境下部署和运维业务应用。凭借兼容 Kubernetes 原生 API 的能力, Karmada 可以平滑迁移单集群工作负载, 并且仍可保持与 Kubernetes 周边生态工具链协同。在多云多集群的场景下, 为了满足跨集群业务的弹性伸缩需求, Karmada 提供了 FederatedHPA, 解决如下场景难题:

- » 无法统筹多个集群的资源, 跨集群弹性伸缩业务;
- » 多个子集群HPA分散管理, 工作负载冗余且复杂度较高, 无法统一高效配置和管理;
- » 如单集群扩容发生故障, 无法在其他集群扩容实例, 存在断服风险;
- » 无法统一限制多个集群资源使用量, 以限制某业务的资源消耗和费用消耗。





云原生发展进入爆发期，
将成为数字世界操作系统



2023年6月6日，中国信息通信研究院云计算与大数据研究所副所长栗蔚在ICT中国·高层论坛-云原生产业发展论坛上发表演讲，系统性地总结了十年间中国云原生发展情况。

过去十年，云原生在经历短暂的萌芽期和发展期后迅速进入爆发期，对传统技术栈产生了颠覆性变化，重塑了企业的技术架构、软件架构、设计理念、组织结构。云原生作为新一代数字基础设施，自身的技术变革已趋于稳定，但基于云原生技术底座的更大一波技术创新、应用创新、模式创新即将到来。



一、过去十年我国云原生发展情况

市场方面

- 我国云原生市场快速发展，市场规模放大明显，市场接纳度不断提升。互联网、软件和信息服务行业引领市场变革，带动云原生技术在传统行业快速应用。金融、电信、能源、制造等行业对云原生已有较高采纳率，并将逐渐加大在云原生领域的投入规模。

技术方面

- 我国云原生核心技术已达较高水平，重点行业应用广泛。云原生理念经多年布道，落地效果良好，以容器、微服务、Serverless为代表的核心技术已达较高水平：容器作为云原生时代软件开发和运维的标准基础设施，在生产环境的采纳率已接近80%；微服务架构作为云原生应用的主要构建方式，在提高开发效率方面作用显著；Serverless作为下一代计算新范式，以FaaS为起点正逐步突破赋能域，未来将有望实现全栈Serverless化。

开源生态方面

- 国内开源技术生态逐步完善，开源贡献率和影响力显著提升。我国云原生开源项目数量已破百，项目类别覆盖云原生全技术领域。目前在主要国际社区项目中，超过20%的开源项目来自中国，国内开发者贡献度跃升至世界第二，国内厂商角色正在从“跟随者”逐步向“引领者”转变。

产品成熟度方面

- 我国云原生产品形态整体趋于成熟，头部行业用户需求从专项能力向综合能力演进。中国信通院已累计对国内90个云服务商和行业用户的191个云原生技术产品进行评估。从产品侧看，容器类产品已高度成熟，中间件管理工具成为热点，云原生安全类产品增长势头强劲。从用户侧看，金融、通信行业用户已进入技术底座建设、业务应用改造和安全防护等体系化能力建设阶段。

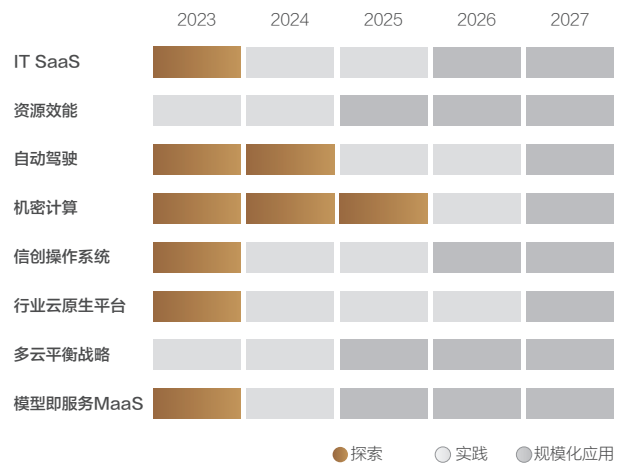
投融资方面

- 据不完全统计，从2018年至今，云原生总投资数达121笔，半数投融资轮次集中在B轮以前。其中云原生安全和云原生数据库为重点聚焦领域，容器、存储、AI、低代码等领域具有较大发展空间。

二、未来中国云原生发展趋势

进入数字化转型时代，算力正像电力一样随取随用，云原生所扮演的角色，就是算力时代的插座。插座提供的标准化能力，使得算力应用可以基于云原生快速部署在各种平台和基础设施上。以云原生为根基，大数据、物联网、生成式AI等创新技术和产品的爆发式增长，将带动宏观经济走向繁荣新周期。云原生将会像蒸汽机之于纺织、电力之于工业一样，成为智能时代的核心动力。

未来五年云原生领域的技术趋势及发展周期预测：





华为云联合中国信通院发布白皮书， 另辟移动时代的云原生蹊径

《面向消费终端的企业云原生实践白皮书》，开辟移动时代的云原生路径

2013年，程序员Matt Stine提出“CloudNative”概念，后来他又将这项技术的特点归纳为模块化、可观察、可部署、可测试、可替换、可处理6项，这就是大名鼎鼎的云原生。

十年过去，云原生的价值在科技界可谓无人不知，谁都能说上来容器化、微服务、敏捷交付等优点。但真正如何做云原生，尤其是消费终端类企业如何全面践行云原生，在今天依旧是问题多于答案。

云原生之路，不能仅在技术层面争论，而应该向实践要回答。



华为终端云, 步入云深处

目前, 大型企业普遍走到了深度用云阶段。随着业务的发展, 企业对于敏捷开发、资源成本管理、云上运维安全等能力的需求会更加迫切。但贸然走向云原生, 又担心会陷入技术路径不清晰、门槛过高、业务发展不可控等问题, 进而产生进退维谷的两难。

华为终端云在发展过程中发现, 企业走向业务全云化, 拥抱云原生, 是一个不可逆的数字化发展趋势。华为终端云的业务范畴, 已经覆盖了全球170+个国家与地区, 而伴随着业务体系的扩展, 以及数字化能力的不断累计, 华为终端云逐渐发现了一系列痛点。比

如说, 业务上线、交付、测试的流程过于复杂, 导致新业务的交付能力与持续性不足, 研发效率变低。再比如, 传统软件开发流程中有大量依靠人工的审核、管理工作, 这些环节增强了业务的不确定性, 同时降低了IT流程间的衔接能力。除此之外, 庞大的业务体系会导致数字化基础设施出现大量重复建设与资源浪费, 长期来看也会拖慢研发与业务进展。这些问题的存在, 阻碍了华为终端业务的发展。

在意识到基础设施与持续交付两方面的产业痛点后, 华为终端云决定基于华为云, 进行全面的云原生改造,

将业务全部搬迁到华为云上, 并且构建面向移动开发的平台化工程。

从2017年开始启动的华为终端云云原生平台工程工作, 到目前已经实现了终端云服务的所有业务上云, 云上主机数量达到数十万个, 云上存储数据已超过EB。在移动互联网时代, 有大量企业都有着复杂而庞大的移动业务, 并且需要全球化进行业务布局, 对于这些企业来说, 云原生平台构建需求非常迫切。华为终端云的成功实践, 就像是一头大象率先过桥, 那么其他动物也当然可以。总结终端云的云原生之路, 其经验可以归纳为一纵一横两个方面。

All in 华为云纵向发展的三个阶段

很多企业在推动云原生时, 最大的困扰在于知道最终想要实现的云原生目标, 但却不知道如何梳理出路径, 规划出节奏。

《白皮书》中完整展现了华为终端云如何实现All in华为云, 拥抱云原生。从方法上来看, 终端云搬迁上云经过了四步方略: 评估、方案、数据搬迁、业务上云。而从发展节奏上看, 整个终端云云原生工程平台, 历经六年左右的发展时间, 最终构建出以效率提升、成本优化、开箱即用、最优体验为发展目标的平台, 整个过程走向可以分为三个阶段。

从2017年到2019年, 是终端云推进

上云的第一个阶段。这个阶段以基础设施上云为主, 还比较少涉及业务改造方面的内容。基于华为云业界领先的云底座, 终端云实现了数十万台云主机、EB级对象存储、TB级CDN搬迁上云, 搭建了全球50ms时延覆盖圈。

基础筑牢之后, 2019年开始的第二阶段中, 终端云开启了高阶服务上云, 将数据库和基础软件大量替换成了华为云提供的服务, 搭建了数万亿级的数据库, 容器化POD数超过10万, 实现了纯在线业务CPU利用率超过40%, 这在企业上云领域是非常高的标准。这一阶段, 终端云的上云重点由基础设施提升到了软件与服务, 为更深度拥抱云原生做好了准备。

第三阶段, 终端云将在云原生探索中凝聚一系列能力, 进行平台化整合和经验沉淀, 并且将这些能力反哺到华为云体系中, 进行SaaS化整合, 从而让更多企业用到领先的移动技术。比如说, 终端云的支付、搜索、浏览器、地图等能力都在这一阶段完成了开放, 这也就是我们经常讨论的华为云与华为终端云的“云云协同”。

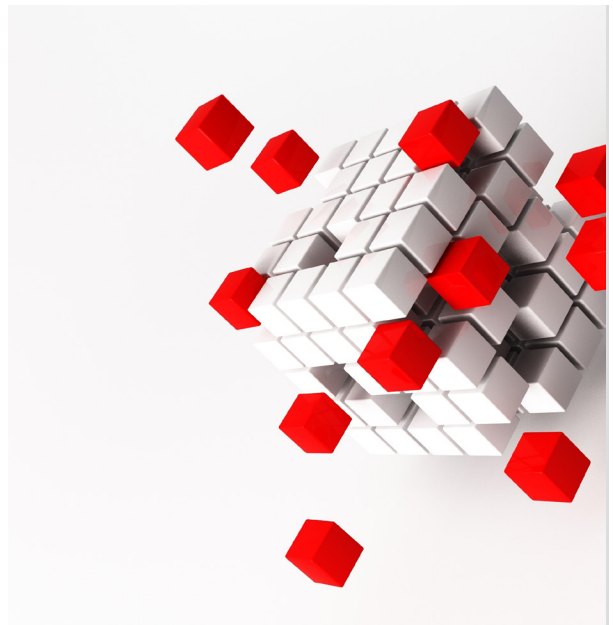
从基础设施上云, 到走向云云协同, 华为终端云的云原生是层层推进, 逐步深化的, 整个纵向路径非常清晰, 节奏有条不紊。在企业走向云原生时, 对阶段性目标与实施节奏的把握是至关重要的, 终端云的实践, 恰好可以作为参考。

沉淀体系化能力 横向筑造三大平台

如果说，阶段性推进是走向云原生的纵向方案；那么沉淀能力，打造平台化工程，则是云原生发展的横向关键。

在云原生进程中，企业需要获取各种各样的能力，并且留存大量开发经验。如果这些宝贵财富用过即弃，那么云原生就会变成同时放水与灌水的水池，浪费大量资源。只有不断进行技术资源的沉淀，并且变成可复用、可标准化的平台，云原生才能焕发出最大价值。

在这个方面，华为终端云也做出了表率。多年的云原生实践，让华为云终端云实现了三大软件工程能力的平台化。这些能力平台不仅可以持续帮助终端云优化开发成本，降低开发难度，带给开发者与终端用户更优体验。还可能通过“云云协同”对外赋能，实现技术能力的社会集约化发展。具体来看，三个软件工程能力平台分别是：



云原生DevOps平台

敏捷开发是云原生的基础，DevOps平台就提供了全自动、一站式、开箱即用的软件工作平台，它沉淀了终端云的优秀开发实践，并按照云原生SaaS平台的方式对外提供服务。通过DevOps端到端工具链，企业可以便捷接入和使用，全面提升软件开发效率，云原生DevOps平台以服务使用者为视角，可全面提升企业应用的开发，运维，运营效率；以软件交付件为核心，实现“业务代码+BaC=软件包，软件包+IaC=可运行环境”。其中，BaC指构建即代码，IaC指基础设施即代码，软件包则应该包括端侧应用，云侧服务以及数据和AI算法等，并且将流程Built-in到软件生产过程，即在BaC和IaC执行过程中可针对流程中需要进行的质量控制项进行控制。

具体而言，云原生DevOps具有四个方面的能力特性构建：

Dev: 构建项目管理服务和祥云Studio，提供个人级、项目级一站式开发平台，通过软件信息树实现全流程可信。

Runtime: 提供大容量、弹性、安全、高可靠的服务运行环境，推动业务按需快速集成。

Ops: 围绕部署的全生命周期管理，通过IaC3.0提升部署效率，服务环境可快速创建、归档和恢复。构建ENS服务提供自动化开通网络能力，提供ServiceInsight汇聚自有服务的所有监控信息。

构建祥云Console，提供一站式的自助申请，祥云平台上的服务上架到祥云Console，对外提供一致的接入和使用体验。

云原生DataOps平台

数据分析能力是现代化软件流程管理的关键，也是效率的来源。DataOps平台是一种面向流程的自动化方法，由分析和数据团队使用，旨在提高数据分析的质量并缩短数据分析的周期。云原生DataOps平台，可以实现数据的高效共享和数据作业的持续快速交付。基于华为云统一元数据、存算分离、多活容灾架构等技术，改造后的数据基础设施，主机数量比IDC减少超过20%，任务减少超过50%，数据共享效率提升80%（原来是集群间数据拷贝，现在是直接访问OBS），人

力维护成本降低50%，3年整体TCO降低20%左右。同时在大数据开发流水线的构建过程中，云原生DataOps创新性的引入了Daac（接入即代码）的概念，实现数据接入自动化。这些技术创新点，极大提高使用数据的效率并降低使用数据的门槛。祥云平台构建的DataOps，可以实现30万+的日均调度，百万级并发调度，秒级调度时延。把数据的共享效率整体提升了80%以上，实现了万人以上的大规模协同作业。

云原生MLOps平台

目前阶段，企业愈发重视AI任务的开发与部署。云原生MLOps平台通过与其他云原生工具结合，实现了简化端到端的AI开发，让业务、数据、算法和运维更加紧密协作，推动模型的快速迭代。终端云MLOps流程，涵盖了计划、数据准备、模型开发、模型训练、模型验证、模型打包、模型部署、模型推理、模型监控等多个方面，从而构建了一站式祥云MLOps平台，提升AI研发与应用效率。

三大软件平台，可以说凝结了华为终端云多年云原生实践的海量技术成果。这些成果的平台化，不仅将继续推动终端云数字化能力持续发展，还将成为云云协同的抓手，降低各行业的云原生实践门槛。

我们提到云原生，通常都在讨论单点技术。但单一技术在云原生进程中是不足的。华为终端云的云原生实践，坚持以整体性、工程化的思维，系统构建能力平台。就像Matt Stine所说，模块化、可替换、可部署，是云原生的核心技术特性。终端云的技术沉淀与平台化打通思路，也值得更多企业在云原生之路上参考。

献给移动时代的云原生路径

华为终端云的云原生改造经验，价值显然不仅局限于终端云的体系内。一方面，其技术成果将通过华为云进行对外开放，成为更多企业可获取、可复用的云原生资源；另一方面，其技术路径本身就可以借鉴，让更多企业可以更快、更好地基于华为云拥抱云原生。从一个很简单的逻辑看，终端云这样体量巨大、业务复杂的体系，都可以通过华为云走向云原生，其他企业自然也可以。华为有一个信条，叫做“自己的降落伞自己先跳”。终端云基于华为云进行的云原生改造，很大程度上又一次展现了这一点。

如今，华为终端云通过深度数字化转型，支撑业务年复合增长率超过30%，依托华为云原生2.0，终端云以应

用为中心构建全面升级的云服务体系，应对了数字化转型中的核心挑战。其带来的价值不仅华为终端云内部感受良多，即使对于开发者与终端用户来说也是一样。过去华为终端云的新业务上线需要人工审核、人工验证，而通过云原生改造，大量服务的上线分发已经实现了全流程自动化，终端用户享受到了更迅捷、便利的新应用获取体验。

面向未来，华为云与华为终端云还会持续推进云云协同，将更多终端云的能力，通过公有云服务与SaaS化对外开放。华为云的用户可以将这些能力按需索取，灵活组合，实现最符合自身需要的云原生能力体系。对与华为终端云来说，未来还将与华为云

进行更紧密的基础设施联合，进一步提升数字化资源的管理能力、运维水平，降低综合开发成本，提升开发经验与技术能力。截至2022年12月31日，华为终端全球活跃用户超过7.3亿，华为终端云服务全球活跃用户超过5.8亿，华为帐号全球活跃用户超过4.2亿，华为全球注册开发者超600万。HUAWEI AppGallery Connect内累计提供的服务达到数百项，全球集成HMS Core的应用超过22.3万个。

这些数字的背后，是云原生之路在支持和赋能，是“云云协同”的战略在推动和牵引。很多企业，经常说在寻找云原生的最优解。答案或许就藏在华为云与华为终端云，这两朵云的合作与协同中。



CNCF CTO

答创原会会员问（节选）

Chris Aniszczyk / CNCF CTO



CNCF 在国内有哪些计划？KubeCon 今年是否在国内举办？



We are going to hold KubeCon CN on Sept 26 - 28th co-located with Open Source Summit China: The theme will be “Cloud: Empowering Innovation Everywhere” and our goal is to establish physical events back in China for the CNCF/LF.

I would like to see more Chinese end user companies raise their opinions, requests and so on to the CNCF project community so we can continue to build open source software that helps with China scale problems. This requires collaboration on all ends though, so we all need to work together!



2023 年云原生技术和产业有哪些发展趋势值得关注？



a. WebAssembly is growing and evolving as a third form of packaging generic compute after VMs and Containers. As WASI v2+ evolves and is standardized to support more “cloud” related workloads we will see more adoption. There are also a lot of new startups in this space.

b. Cloud Development Environments (CDEs) as raising in popularity, from Github Codespaces, to Gitpod, DevPod, DevZero, Replit, Coder, nlocalhost etc are all offering a cloud based managed environment for developers. For businesses, the value will be in what is currently available in VDIs... CDEs represent an evolution of the VDI concept for a modern era and will see uptick in the next couple of years to replace the older VDI technologies

c. eBPF everywhere - this one is simple, modern kernels are in many places now that support eBPF out of the box.

Q 云原生产业发展越来越成熟, 相似的云原生产品又层出不穷, 您认为哪些因素可以让云原生的项目脱颖而出?

A It depends on the product of course, but I believe there are two things that differentiate great products from mediocre ones:

a. developer experience: The focus should be to delight developers. If you make it easy to get started so developers can easily install the tool and be productive quickly, it makes a difference. It is tricky to measure developer experience as there are many ways to do it, there are new frameworks out there like SPACE that can be a useful reference: <https://newsletter.pragmaticengineer.com/p/developer-productivity-a-new-framework>

b. documentation: clear and useful documentation is required to have a good developer experience when developers are still in the learning stages.

Q 如何定义云原生安全? 践行云原生中需要解决哪些安全风险和挑战?

A First I recommend everyone to look at the resources put together by CNCF TAG Security which is some of the best content on this question: <https://github.com/cncf/tag-security/blob/main/security-whitepaper/v2/cloud-native-security-whitepaper.md>

This is hard to summarize in a simple sentence, but I'll say that "cloud native" implies a dynamic and changing environment, there aren't static IPs or machines anymore, it's a ecosystem of services that come and go.

Q 云平台与容器技术的进一步融合是否有一些好的方向? 特别是容器网络与云网络的融合方向、性能提升方向

A I think this is a tough question for CNCF to manage as our general approach is to have the "market" decide the best solutions out there, which would be from end users like yourself. There are use cases that come from telco, industrial, HPC etc that all have their own unique criteria that can improve the core kubernetes project. There are scaling limitations

Q

数字化转型背景下, 云计算和数据底座 (数据湖 / 数据治理) 相互之间的影响、促进、制约关系是什么? 如何解决多种云体系和统一数据底座之间的矛盾, 最佳实践是怎样的?

A

The data gravity problem is easier to tackle than typical digital transformation (e.g., move from onprem classic services to cloud, move from outsourcing dev to bringing in house) as there are less people involved in the transformation. I think if you were going to start your digital transformation journey, work on modernizing how you store foundational data as that affects every part of your stack but generally is easier to manage due to the smaller core team involved in managing that infrastructure.

Q

CNCF 社区针对云原生大数据有哪些短期及长期计划? 开源项目将如何帮助大数据在云原生基础设施更好运行?

A

I think the biggest thing projects can do is decouple consensus mechanisms that were built in house/custom from the storage mechanisms and assume a “kubernetes” or cloud native default environment where there is a shared central orchestrator. If you don't have to worry about service discovery and typical distributed computing problems because you can piggy back on kubernetes, it really simplifies things. There is a new crop of databases, streaming systems that are built for a cloud native era that are shedding the previous dependencies on ZK or manual configuration systems.

Q

Native 编译是 Java 生态迈向真正云原生的重要环节, 海外有什么发展趋势? 有什么比较好的案例, 带来了哪些收益?

A

Java is still used in many traditional enterprises and I don't see it going away anytime soon. Java and the JVM itself is evolving with projects like <https://github.com/quarkusio/quarkus> and <https://www.graalvm.org> - as a reaction to the developments happening in the container and WebAssembly ecosystems. I think the Java ecosystem is mature and the tooling is starting to catch up, which makes the natural transition of moving to say a greenfield implementation.

Q 社区对于多云架构的演进是如何规划的呢?

A I think it depends what you mean by multi cloud? Is it your compute? storage? app? You can technically build a container today that will run in every major world wide cloud. However, if you start using cloud services it leads to an issue of depending on that cloud so you lose your “multi cloud” ability quickly. The problem is that standardization (which enables multi cloud) always lags behind innovation which may be critical for your business which may tie you to a specific cloud.

HOWEVER, I think there is a lot of innovation happening in projects like Crossplane, Karmada, KCP etc that are tackling this problem in different ways. There is more collaboration happening in multi cloud offerings in the last 2 years that I have seen in my whole career so I think the future is promising. It will require input from global cloud providers to make things work.

Q 您对广大云原生开发者有哪些建议可以分享?

A a. don't forsake Linux fundamentals - there are many “cloud native” developers that I teach and have come across that don't know how to write a kernel module or how kernel networking (iptables) works. Linux is only becoming more important as technology like eBPF is becoming critical to many projects (which I can argue is harder to debug than iptables ;p)

b. the best way to learn is to contribute and teach... if you contribute to an open source project you will learn many technologies quickly... this can be reinforced by teaching and also helps everyone!

感谢以下会员积极贡献实践问题

程俊 (美的楼宇科技研究院研发中心负责人)

杜广源 (昆仑数智科技技术总监)

黄俊 (招商证券云原生转型负责人)

江成旺 (美的置业数字化运营中心技术负责人)

刘洋 (骥步科技 CEO)

牛红星 (上海仪电人工智能创新院 CTO)

张水涛 (前海联合交易中心科技创新部副总裁)

张亚祥 (浦发银行云原生开发总监)

赵贵斌 (声网集团环信联席 CEO)

赵旭华 (华人运通云服务和技术安全总监)

周峰 (挚文集团云基础设施部架构师)





技术热点

华为云瑶光面向负载 QoS 感知的资源调度优化探索	19-23
分布式云原生 + GitOps, 轻松交付多集群应用	24-28
多沙箱容器运行时 Kuasar, 启动速度 100% 提升	29-35
云原生数据库八大关键技术解读(一)	36-40
AudioGPT, 全新的音频内容理解与生成系统	41-44

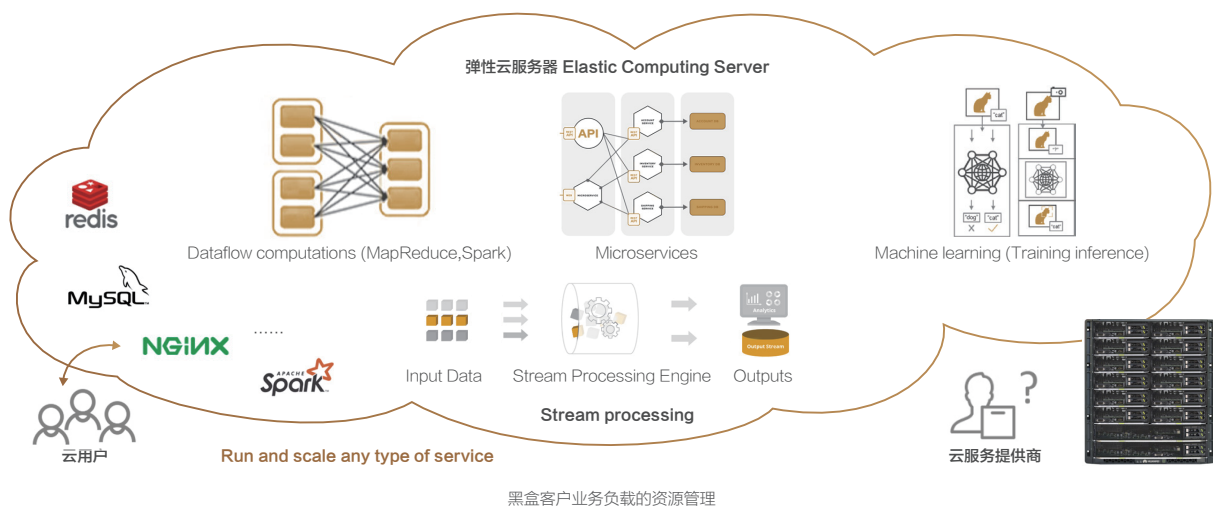
云原生 Cloud Native

华为云瑶光面向负载 QoS 感知的资源调度优化探索

汤闻达 / 华为云技术专家

大规模的云资源运营既要考虑如何保障服务质量 (QoS, Quality of Service) 以提升用户业务上云体验, 又要尽力提高数据中心资源利用率以降低运营成本。事实上, 这两个目标的达成在当前的软硬件技术水平上是一个棘手且复杂的问题。华为云瑶光调度算法团队致力于解决华为云调度场景下面临的多种复杂的优化问题, 为了向客户提供极致、稳定、丝滑的业务上云体验, 全方位持续保障、优化华为云弹性云服务器ECS、容器等云产品的服务质量, 该团队研发了“华为云智能软硬协同算力服务优化与保障平台”——华佗, 本文从典型的云计算资源分配与QoS保障场景介绍华佗平台关于该问题的两个技术探索。

一、未知的用户负载和无所不在的资源争抢是 QoS 保障的难点



公有云计算服务提供商主要向企业、个人提供稳定可靠、安全可信、可持续发展的云服务，这些云服务的技术底座通常由非最终用户所拥有的物理IT 基础设施构建而成。依托虚拟化技术，各云厂商通常采取大颗粒度“切割”物理服务器资源的方式（如CPU核数、内存容量等），向用户提供拥有不同规格资源的计算实例（比如虚拟机，容器）以满足用户业务的差异化资源需求。用户通过结合其业务负载类型和性能要求来选择租用对应的计算实例规格。事实上，用户可以针对其租用的计算实例运行任何需要的业务负载，并且各类负载由于

技术栈和业务场景的不同，它们运行过程中的实际资源需求上呈现出非常迥异的特征。伴随该类典型的资源售卖方式，大量自不同用户的迥异负载共置于同一台物理服务器运行，共享使用大量的资源。作为云服务提供商，我们更愿意看到各类负载能够在资源的使用上“和平共处”，但遗憾的是，争抢时有发生，我们要尽最大努力协调好资源，保障好每一位用户业务的价值。

尽管通过大颗粒资源切割的方式可以给予不同负载一定的性能保障能力，但事实上仍然有一定数量的业务

遭受着不同程度的性能干扰，主要原因是共置于同一台物理服务器的所有计算实例之间有大量“冰山之下”的资源共享，比如LLC（末级缓存），内存带宽等。比如内存密集型负载在共置时若出现LLC的资源争抢，极易导致业务性能颠簸，降低用户的资源使用体验。保障好用户体验的主要挑战在于，基于网络安全与隐私保护原则，云服务提供商不能也不该知晓用户负载的具体用途和敏感资源类别。如何能在面对黑盒用户业务的前提下，既保障用户的体验又能稳步提升资源利用率一直是工业界、学术界研究的热点问题。



未知的用户负载和用户感知

二、QoS 感知的资源调度优化分析

通常而言，尽管基于安全、隐私条款等因素考虑，云服务提供商无法获知用户业务负载的具体细节，但仍可将用户业务作业以计算任务的特点分为“在线作业”和“离线作业”。在线作业通常以服务的形态来处理用户业务请求并执行计算任务，如网页搜索服务、电商交易服务等，其具有较高的实时性和稳定性需求，常常对服务的响应时间极为关注，即呈现出时延敏感（Latency-Sensitive, LS）特征。除此以外，其在资源使用上多呈现随时间动态变化的特点，存在一定的周期

性规律。由于在线作业对性能变化敏感，稍微的性能抖动就会影响大量用户的体验。因此，用户一般在为这类业务购买资源实例时也倾向于考虑资源保障度较高的资源实例，尽管其实际的资源消耗在大量时间内并不高。

相较而言，离线作业通常是计算密集型的批处理作业，典型的如MapReduce、Spark作业、机器学习模型训练作业等。多数离线作业对于性能的稳定性和持续性没有特别苛刻的要求，可以容忍短暂较高的运行延迟，

并可容忍在LS业务出现资源诉求时，提供短时的资源让步，呈现出尽最大努力（Best-Effort, BE）交付的特征。

因此，提升数据中心整体资源利用率的一种理想方法是在理解用户业务的特征后，在保障在线作业性能的前提下，将LS业务的空闲资源临时分配给BE业务（即：将LS与BE混合部署于同一物理服务器，简称“混部（colocation）”），并提供基于业务特征的动态、安全的区分性资源保障能力，即在LS业务需要资源时，能够从BE

业务中抽离资源及时满足LS业务需求。显然，能够鉴别黑盒业务的特征是上述QoS感知资源调度的关键。不仅

如此，基于业务特征的准确识别，还可以结合现有的白盒QoS保障技术实践更多有价值的资源调度优化，下

面简要介绍下本团队近期的技术实践成果。

三、华为云黑盒 QoS 保障技术实践

在探讨QoS保障技术实践之前，先来简单探讨下业务运转的基本特征。计算机业务运转的本质就是对“数据”进行操作，得出对我们有用的“信息”。我们依托“知识”进行程序的开发，以机器能够理解的形式对如何操作“数据”进行逻辑表达，然后通过计算机获得我们需要的“信息”。在当下普遍基于冯·诺依曼架构的计算机中，该过程需要由CPU从存储介质中拉取指令和数据加以操作，并将结果返回存储介质进行存

储，该过程中存在大量的数据搬移操作。具体的搬移路径涉及L1 Cache、L2 Cache、L3 Cache（也叫LLC末级缓存）、内存、硬盘等多个介质，其中关于内存和硬盘的绝大部分交互还涉及OS（操作系统）的统一把控。该路径的任何一个环节出现资源争抢，都将会不同程度地影响业务的性能表现。

为了降低客户在使用华为云ECS、容器等云产品方面所可能面临的QoS劣

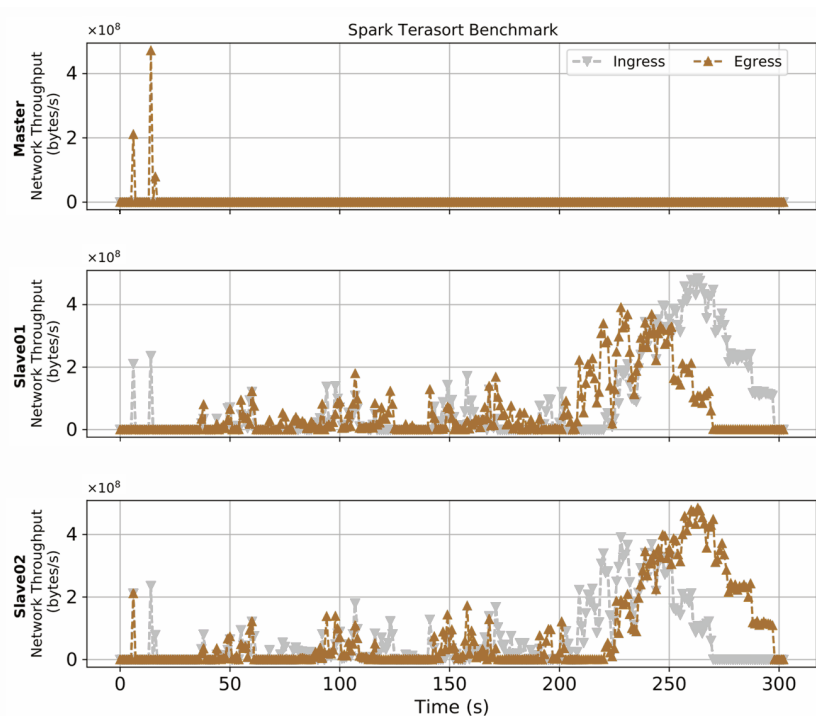
化风险，全链路保障业务的性能表现，我们团队研发了华为云智能软硬协同算力服务优化与保障平台——华佗（Huatu）。Huatu平台致力于提供极致、稳定、丝滑的用户业务上云体验。它能够感知底层资源实时状态，提供业务性能的可观测性与可控性。能够接收上层决策，实现主机级、集群级的微观、宏观资源控制与调度保障，并提供基于QoS感知的资源挖潜能力，提升云资源的复用比，降低运营成本。

下面介绍目前在黑盒QoS保障实践方面的两个技术探索。

区分QoS服务的内存带宽资源调度的内存带宽资源调度

就上述路径中关于存取内存介质的内存带宽资源而言，我们监控发现用户业务对内存带宽资源在使用上存在比较明显的差异性，并且由于该部分资源长期以来一直缺乏合理管控，时延敏感且内存带宽资源敏感的负载极易受共置内存密集型批处理负载的干扰而性能受损，影响业务性能。

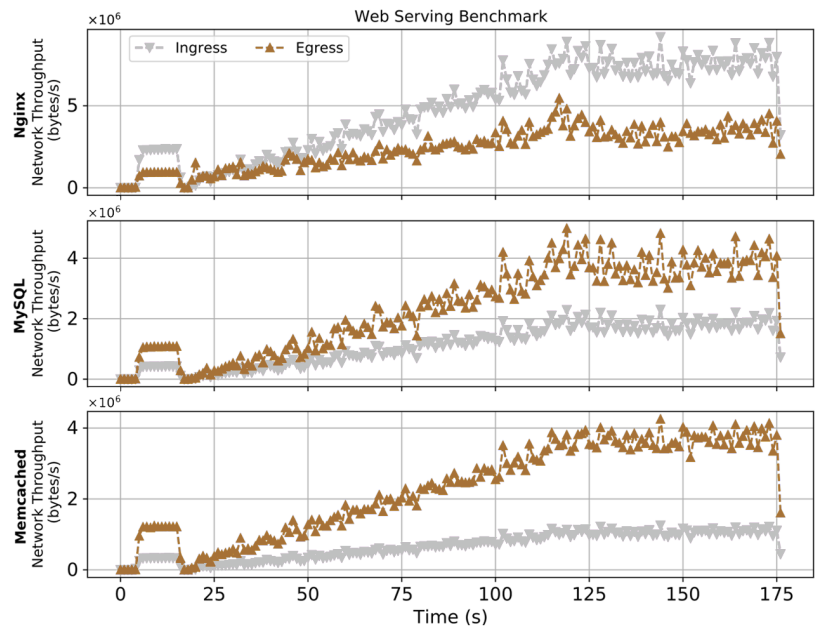
如何知晓用户黑盒负载是否时延敏感？通过大量的白盒实验分析，我们发现时延敏感的业务通常具有类似这样的资源使用模式——用户态的CPU



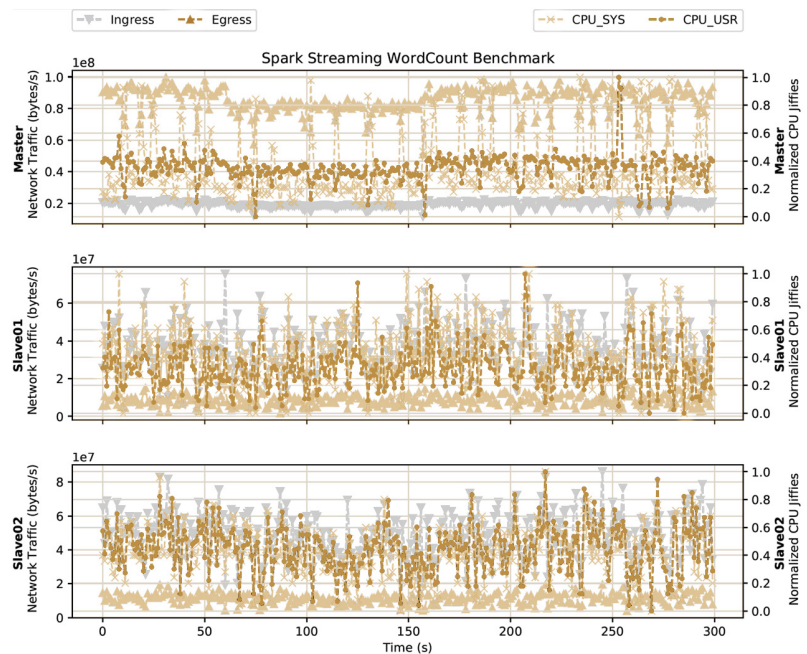
典型 BE 应用在网络接收 (Ingress) 和发送 (Egress) 流量方面的变化情况

资源使用与网络资源的使用呈现出伴随性消耗，不仅如此，时延敏感应用在网络收发数据的模式上比较稳定（比如收发比例、时序变化趋势等），而相比之下，批处理负载在网络资源的使用模式上呈现出较大的跳跃性。据此，我们可基于该模式识别迅速把黑盒业务负载进行有关时延敏感性的动态分类，实际分类效果不错。

在该黑盒负载分类的基础上，为了妥善解决内存带宽资源的争抢，我们研发一种内存带宽资源调度技术 Themis，它可有效削弱共置负载的资源争抢并保障业务QoS。Themis根据负载关于内存带宽&LLC的使用情况，分别为LS业务和BE业务赋予较高和较低的资源优先级，基于一种长-短期资源管理方案对内存带宽资源进行分配、调度。具体来说，长期资源管理周期性地对所有负载的访存资源需求量子进行动态聚类，然后对内存带宽资源、LLC进行分组硬隔离，以达成内存带宽资源长期按需分配，公平竞争，避免饥饿的目标。考虑到长期资源管理造成的资源硬隔离会让部分负载短期内出现资源供需不对等的现象，短期资源管理则提供针对部分分组的空闲资源的加权时分复用，以避免长期资源管理分配的不均，并达成提高吞吐目标。目前关于Themis的详细技术细节已得到业界同行的认可，并正式发表于并行计算领域顶级国际学术会议ICPP 2022。



典型 LS 应用在网络接收（Ingress）和发送（Egress）流量方面的变化情况



典型 LS 应用 HiBench Spark Streaming Wordcount 关于 CPU、网络指标的相关性

负载QoS感知的CPU节能控制技术

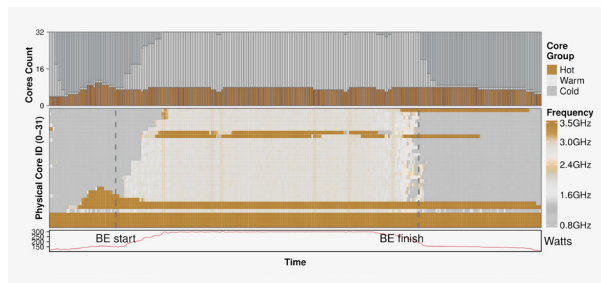
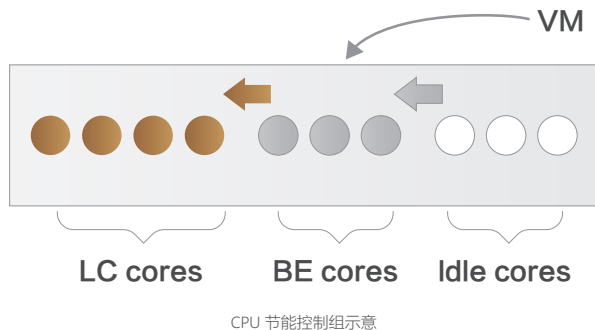
数据中心一般以机柜为最小供电单元，其要求部署在机架中的服务器总功耗不能高于单机柜的承载能力。但随着数据中心业务越来越多，机柜的超电风险持续攀升，并对业务带来了诸多潜在风险。目前华为云已经在机柜热点风险预警、热点

消除、业务调度准入控制等方面做出了大量努力，并已取得显著效果，极大保障了客户上云业务的稳定性。从微观角度看，“雪崩时，没有一片雪花是无辜的”。事实上，服务器的局部功率热点也容易引起TDP（热设计功耗）超限，进而导致用户业务性能受损，这在现网中已是屡见不鲜。

如何在保障业务QoS的前提下，尽可能消除服务器的局部功率热点，进而一定程度上也降低机柜的超电风险？考虑到数据中心有相当一部分能耗来自于服务器CPU，尽管当前CPU已经存在成熟的降频技术能够根据业务的压力自动地调节频率以降低功耗，但这些频率调节技术由于没有充分考虑到其对业务负载性能的潜在影响，极易在启用时导致业务性能产生劣化。

我们发现CPU的调度时延变化可以较好地表征时延敏感性P95尾时延指标，相比CPI、CPU利用率等代理指标，可以较为及时地发现QoS劣化现象。为此，我们提出一种新的面向QoS感知的CPU节能控制技术Demeter。通过将多核CPU在逻辑上划分成热、温、冷三个区域，形成不同的节能控制组，并依据负载压力的波动性设计实现了CPU资源的按需横向扩缩，力求将每一焦耳所能带来的业务价值发挥到淋漓尽致。

实验结果表明，Demeter可在保障原有业务性能几乎不变的情况下，大幅降低整机能耗。相比业界前沿方案，该方案更加适用于公有云环境，并进一步可平均减少10%的整机能耗。为云数据中心的节能减排工作提供强力支撑。目前关于Demeter的详细技术细节已正式发表于云计算领域顶级国际学术会议ACM SoCC 2022。



四、未来展望

从现有的实践经验来看，云场景下良好的画像系统可以帮助提高资源调度的质量，为业务QoS保障提供更坚实的基础。从宏观层面来说，画像系统可以帮助数据中心的提升资产经营能力，通过关注客户的资源购买习惯，其购买习惯直接关系到云产品的规划，资源池的配置，服务器的选型与采购等等；微观层面来说，画像系统可以为客户的资源提供关于安全部署，资源利用率和性能等方面

提供全生命周期的保驾护航。作为云服务提供商，也可借助于该能力进一步挖掘提升资源利用率和保障业务性能方面的平衡。Huatu作为华为云的智能软硬协同算力服务优化与保障平台，后续将结合业务负载在计算、存储、网络等多维资源的QoS诉求，设计新型数据中心智能硬件架构，开展云原生领域的软硬协同资源优化探索，全面持续提升云产品的竞争力，为客户提供极致体验。

未来，华为云将持续优化瑶光面向QoS感知的资源调度能力，在全域调度、容量管理、SLA保障以及更多的场景下持续发挥价值，释放更多技术红利，为各行各业提供极致性能、极致稳定、多样性算力、云边缘协同的云基础设施。同时，华为云将持续发挥全栈技术创新能力，继续探索分布式云形态下全域调度、软硬协同等技术，为企业提供更极致的云上体验。

分布式云原生 + GitOps, 轻松交付多集群应用

多云场景下集群管理和应用交付的挑战

随着业务的全球化发展和应用多元化部署的趋势，越来越多的客户选择通过混合云、多云模式来进行业务部署。选择多云进行部署可以提高部署业务的基础设施稳定性，在单个供应商基础设施出现故障或者访问流量激增时，可以通过配置跨云弹性来提高业务的高可用性，同时，多云还可以避免企业的技术架构被厂商锁定。尽管使用多云的优点很多，但管理多云集群和在多云的场景下发布应用却面临诸多问题和挑战。

1、多集群基础设施的管理及一致性

发布面临的挑战。例如，在多集群场景下的网络策略的配置，TLS证书的发布及更新管理。在现代应用程序的部署步骤中，SSL/TLS证书是很重要的一环。但在部署应用程序时，管理证书的续订通常是事后才想到的。证书的生命周期从90天到13个月不等，为了保持安全访问，这些证书需要在到期前更新/重新颁发。

鉴于大多数 Ops 团队工作繁杂，证书更新有时会被遗漏，这会导致应用间不能正常访问和工作。在多集群场景下，运维团队会每个供应商集群重复上述过程进行证书更新；而通过 GitOps 结合 Cert-manager[1]、Nginx Ingress Controller 可以一致的、统一的管理证书的自动化更新[2]，大大提升 Ops 团队的运维管理效率。

2、由业务场景侧需求和集群基础设施

差异性带来的差异化配置挑战。根据应用程序的业务场景诉求不同，不同集群部署的业务版本，更新频率会存在不同。例如同一家餐厅在不同地域的点餐系统可供给的菜单种类，菜单上新会有差异；或由于跨国公司在不同国家推广策略不同，新的业务软件仅需要部署至部分城市所在集群等。

同时，由于业务部署的基础设施不同，应用程序部署到集群的底层架构、网络连接性、计算存储性能表现可能多种多样。例如同一份应用配置在被差异化渲染后可以被交付和托管在云上的CCE、EKS集群、客户本地数据中心中的集群（存在断连情况）、边缘端无人机的集群上（半连接集群）以及太空中卫星链所组成的集群（短时连接集群）。

因此根据每个集群的性能指标（CPU、Memory）不同，部署业务应用的实例副本数可能会不同；根据每个集群的网络连接情况不同，设置部署业务的版本更新周期，高可用设置（访问某个服务的超时重试次数等）会产生差异；根据每个集群的使用目的不同（早期生命周期阶段的集群通常由开发人员管理，而实际的预发及生产集群的可能由客户的运维团队管理），部署业务的版本和数据库连接池等变量也会存在差异。因此当有M个应用需要交付至N个集群环境中时，差异化配置的复杂度会呈M×N维度爆炸增长。

3、使用 UI 控制台方式交付应用与各厂商控制台风格各异、难以编排大规模微服务交付之间的挑战。

随着微服务规模变大，依赖UI控制台进行应

用交付的方式变得复杂臃肿, 其交付的顺序编排依赖人工, 无法做到自动化; 且无法进行审计和版本控制。

4. 缺乏统一的应用观测视角的挑战。在多云集群场景下, 当前缺乏统一的视图帮助客户查看应用在多集群的

部署情况、应用的健康状态及异常状态定位。

分布式云原生 + GitOps 配置管理简化多云应用交付

为了应对上述多云集群管理和多云应用交付的挑战, 可以基于分布式云原生技术和GitOps 理念开发跨集群配置管理和应用分发的功能, 进而实现屏蔽底层环境差异和不同的管理入口, 将多个集群环境的配置和治理集中于一处, 以一致性的体验完成多集群基础设施的管理以及多云应用的发布及更新。

GitOps 的概念最早由 Weaveworks 公司于2017年提出, 指具备版本控制、拉取和合入请求能力、具备CI/CD流水线发布能力的基础设施即代码 (Infrastructure as Code, IaC), 是一种云原生的持续交付模型。如图1所示, 它的核心是使用 Git 仓库来管理基础设施和应用的配置, 并且以 Git

仓库作为更改基础设施和应用的单一事实来源, 用户从其他地方 (例如集群控制台或者命令行) 修改的配置均会被修正。Git 仓库中的声明式配置描述了目标环境当前所需基础设施的期望状态, 借助 GitOps 能力, 当集群中的实际运行的配置或应用状态与 Git 仓库中定义的期望状态不匹配时, Kubernetes Reconcilers 会根据期望状态来调整当前的状态, 最终使实际状态与期望状态保持一致[3]。

基于上述的思想和技术路线, CNCF 开源社区从17年开始至今, 涌现出很多火热的持续交付项目, 他们以Flux、ArgoCD等CNCF毕业项目为代表, 可以将用户配置在代码仓库中的Kubernetes Manifest (Deployment、Service等

Yaml文件)、Helm Chart、Kustomize、Ksonnet、Jsonnet 定义和组织的应用以自动化的方式部署、将配置变化更改到应用程序的运行环境。

分布式云原生理念最早由华为云提出, 并推出了全球首个分布式云原生产品UCS (Ubiquitous Cloud-Native Service), UCS通过简单易用的 UI 提供对华为云集群、多云集群、本地集群、附着集群和伙伴云集群进行跨命名空间、跨集群的应用分发与配置管理的能力, 并在观测面板中对配置的实时状态的进行收集和展示。用户还可以将它对接到CI流水线后面, 实现多云应用的 CI/CD 流水线的集成和发布。当前UCS提供如下关键能力, 帮助用户实现便捷的多云交付。

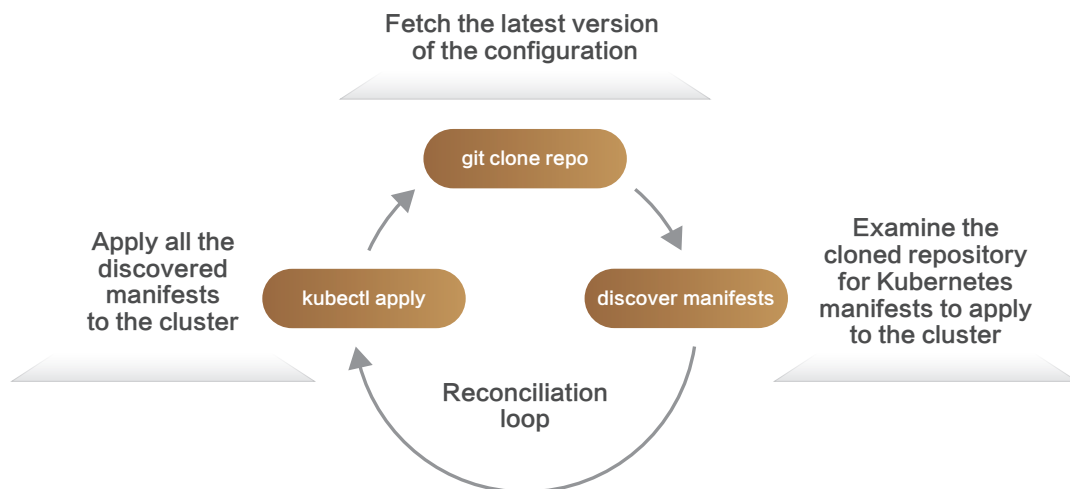


图 1: GitOps Operator 运行方式

2.1 开箱即用的GitOps引擎，兼容主流的开源生态和体验。

UCS 会为每个开启 GitOps 引擎的集群安装一个稳定开源版本的 Flux2 组件，且用户无须运维 GitOps 引擎。每个集群中的 GitOps 引擎会以 Pull 模式、定周期监听和拉取最新的仓库源配置信息并把最新的配置信息及时间同步至集群中。

如图2所示: Source-Controller 主要负责监视 Git 仓库源、Bucket 对象存储桶以及 Helm 仓库的存储配置变化，然后把最新 Commit 记录的制品包拉取至集群本地。而 Kustomize-Controller 和 Helm-Controller 则会负责监听集群本地拉取制品变化情况，其中以 Helm Chart/Helm Release 类型定义的制品会交由 Helm-Controller 进行渲染和同步至集群中；同理，按照 Kustomize 方式进行组织的制品交由 Kustomize-Controller 进行渲染和同步至集群中。

2.2 丰富的多集群差异化配置能力

随着部署应用的规模越来越大，部署集群的底层差异性越来越大，我们发现单一的一份配置对应一个集群的模式会变的越来越繁琐和难以维护，因此面向多个集群的差异化配置策略随之出现。UCS 配置管理功能提供了两种多集群差异化配置的策略: Kustomize 和 Helm Release。

Kustomize 是一个 Kubernetes 应用程序配置管理工具，它提供一种简单灵活的方式来生成 Kubernetes 资源，并可以使得这些资源在不同的环境中用不同的方式进行配置[4]。如图3所示，Kustomize 策略在 Base 目录

下定义所有集群公共部署资源，然后在 Overlay 目录下描述每个集群产生差异化覆盖参数。然后在部署阶段，通过动态渲染参数将最终版本的制品交付至目标集群中。

同理，HelmRelease 也是参考上述思路。将公共定义的资源放置在 templates 目录下，然后结合 valuesFrom/valuesFiles 等方式从 value.yaml 读取每个环境的差异化参数，满足客户差异化的配置诉求。其配置的重点在于做好定义公共部分抽象和少数变量的差异化配置，对

应用本身参数属性和运维参数进行分离，减少重复编辑和维护的成本。同理，按照 Kustomize 方式进行组织的制品交由 Kustomize-Controller 进行渲染和同步至集群中。

2.3 基于Git的可审计、可持续的部署能力

UCS 配置管理将 Git 仓库中最新合入的制品配置信息同步部署至纳管的多个集群中，同时对应用发布行为进行版本化管理和权限控制，提供发布回滚和版本迭代控制，并进行审计跟踪。

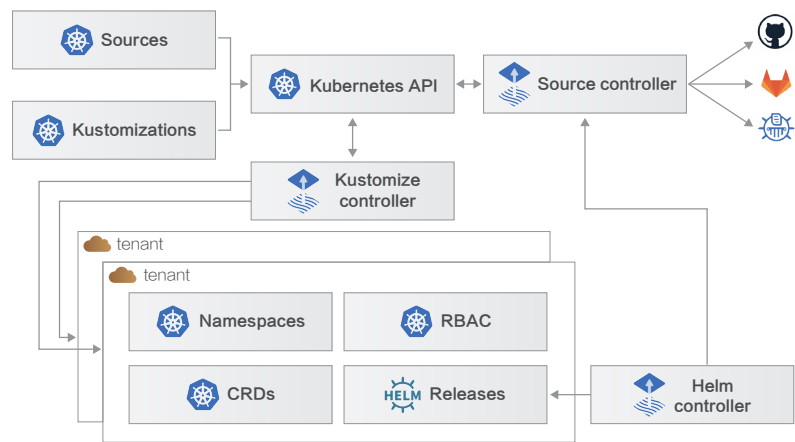


图 2: Flux2 主要组件的运行原理

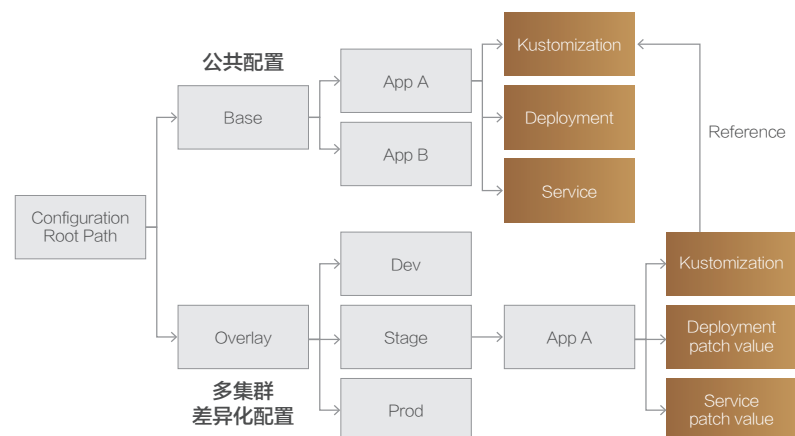


图 3: Kustomize 制品组织目录示意图

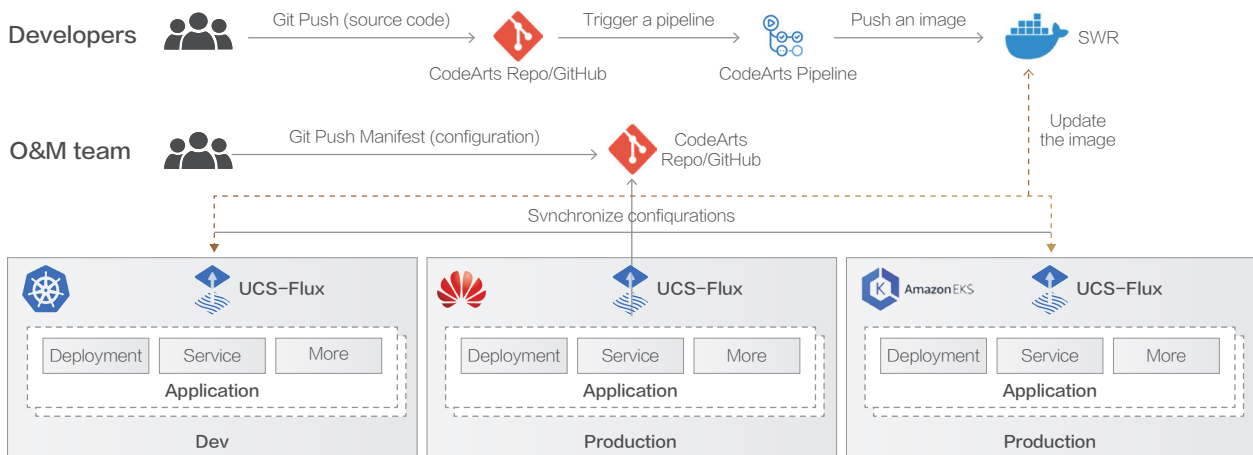


图 4：结合 UCS GitOps 的多云 DevOps 流水线

基于 UCS GitOps+Pipeline 流水线构建多云 DevOps 解决方案

随着 DevOps 价值观和文化的流行，越来越多的公司选择帮助开发团队分担应用程序交付的责任，他们将多云环境下的交付交给专门的运维团队来完成，让开发团队可以更加专注于应用程序的开发和构建本身[5,6]。基于 UCS GitOps+Pipeline 流水线可构建多云 DevOps 的解决方案，实现多云环境下多云应用构建和发布。开发团队和运维团队可以基于 Git 工作流，将现有流程对接到华为云 CodeArts Pipeline 流水线或者企业自建的 CI/CD 流水线之上，从而拥有多云应用的业务开发、集成、测试再到多云应用的部署—全流程 DevOps 体验。具体来讲将分为以下两个阶段：

1、定义和构建多云应用：开发团队进行业务的开发、测试、验证、打包软件和生成镜像。这里可以是采用华为云官方的 CodeArts Pipeline 流水线或者用户自建的 CI 流水线。然后定义每个集群交付资源的原始制品文件。

2、交付多云应用：运维团队首先会根据开发团队提供的原始制品文件对部署在多个集群环境中的差异化内容进行配置。此环节需要做好定义公共部分抽象和少数变量的差异化配置，对应用本身参数属性和运维参数进行分离，减少重复编辑和维护参数的成本。

然后使用 UCS GitOps 统一初始化集群所需的环境和资源，对发布步骤进行编排，通过更新配置仓库来一致的对多个集群进行自动化应用发布；同时运维团队还对应用发布行为进行版本化管理、权限控制和审计，提供发布回滚和版本迭代控制，保证业务应用的成功部署。

下面将以一个详细的例子来解释：华为云某亚太跨国公司客户需要统一管理横跨多国的 Kubernetes 集群和进行业务发布，他们的线上商城业务应用同时运行在香港的华为云 CCE 集群中，新加坡的亚马逊云 EKS 集群中；并且他

们在马来西亚还拥有一部分自建数据中心集群供开发团队进行业务开发和测试验证。由于每个国家消费者的商品喜好差异以及当地的供应链供给不同，商城中发布的商品类别会存在差异。在原有的交付流程中，运维团队会根据每个地域的供应商集群控制面板风格、部署业务版本，业务更新频率等因素，为每个环境单独构建一条流水线独立交付；并在每次发布版本前，运维团队会与开发团队就新版本特点和每条流水线的部署细节进行详细磋商。

而使用 UCS GitOps 可以大大降低交付上述流程的复杂度，如图4的解决方案中所示，客户采用多套环境共享一套 CI 流水线，并将构建的产物统一推送至华为云香港的 SWR 镜像仓库。然后通过差异化配置不同环境的部署参数，将多个环境的发布对接到华为云 CodeArts 配置仓库，实现了从本地集群测试和验证到多个生产集群的发布的无缝切换，也极大的提升了他们多云交付的效率。

总结

综上所述, UCS GitOps 是以 Flux2 为技术内核, 将其与 UCS 的容器舰队/集群模型进行适配的多云交付平台。它通过简单易用的 UI 提供对华为云集群、多云集群、本地集群、附着集群和伙伴云集群进行跨命名空间、跨集群的应用分发与配置管理的能力, 并在观测面板中对配置的实时状态进行收集和展示。它可以帮助您将多个集群环境的配置和治理集中于一处, 以自动化的体验完成多集群基础设施的管理以及多云应用的发布及更新。

在后续的版本迭代中, 下列特性将会逐步支持:

1、支持容器舰队级别的配置分发

通过对舰队内部集群进行标签化管理, 完成舰队视角下应用的一键分发和统一管理。

2、全面对接华为云 CodeArts Pipeline

提供全流程、更好融合体验的多云 DevOps 流水线。

3、在界面中提供对接三方消息系统的应用发布状态感知能力

一方面处理来自外部系统 (GitHub、Bitbucket、Harbor、Jenkins) 的事件, 然后通知 GitOps Toolkit 控制器有关源更改的信息; 另一方面处理由 GitOps Toolkit 控制器发出的事件, 然后根据事件的严重性和涉及的对象将它们转发至外部系统 (Slack、Microsoft Teams、Discord、Rocker)。

参考资料

[1] 在 Kubernetes 环境中自动化证书管理 <https://www.nginx.com/blog/automating-certificate-management-in-a-kubernetes-environment>

[2] 使用 Flux 管理多集群基础设施 <https://github.com/fluxcd/flux2-kustomize-helm-example/blob/main/infrastructure/controllers/cert-manageryaml>

[3] Codefresh Continuous Delivery for Kubernetes

[4] 使用 Kustomize 对 Kubernetes 对象进行声明式管理 <https://kubernetes.io/zh-cn/docs/tasks/manage-kubernetes-objects/kustomization/>

[5] Enterprise CI CD Best Practices

[6] GitOps-2.0 The Future of DevOps Ebook v4

多沙箱容器运行时 Kuasar, 启动速度 100% 提升

在KubeCon + CloudNativeCon Europe 2023 云原生顶级峰会上, 由华为云、中国农业银行、openEuler 社区和 CNCF 项目 WasmEdge 等联合发起的云原生多沙箱容器运行时 Kuasar 正式宣布开源, 受到云原生社区、Rust 社区和云厂商们的广泛关注和热烈讨论。

简单来说, Kuasar 是一个基于 Rust 语言开发、可以同时支持业界多种主流沙箱隔离技术的容器运行时。从架构上看, Kuasar 最大的特色亮点在于: ①基于业界最新的沙箱接口开发, 这种接口不同于当前主流的 Shim v2 接口, 对沙箱的定义和生命周期管理有天然优势; ②采用了 1:N 的容器进程管理模型, 摒弃了传统 Shim 进程 1:1 的做法。这两个有别于传统容器运行时的改动, 为Kuasar带来了100%启动速度提升和99%内存开销优化。

接下来, 就让我们一起走进 Kuasar, 探寻其性能强悍背后的开源故事和架构细节。

容器发展史

早在 2013 年, docker 横空出世, 标志着容器时代到来。最初的容器技术, 其实就是利用 Linux 内核提供的命名空间 Namespace 和控制组 Cgroup 功能, 实现容器进程之间的资源隔离和限制。在容器时代, 容器就是 docker 里唯一的一等公民。不

久, 容器编排领域诸侯争霸, 随着 kubernetes 最终成为主流的容器编排工具, Pod也成为容器编排领域的一等公民, 为了兼容 Pod 这一概念, docker 引入了 pause 容器。但是, pause 容器的引入常常让开发者感到困惑, 因为 Pod 与 pause 容器之间

存在着很多差异。在 kubernetes 中, Pod 是一组容器逻辑和物理资源的载体, 而 pause 容器只是提供了容器间共享的 namespace 而已。此外, 容器运行时里有很多冗余和复杂的判断逻辑用于区分 pause 容器和用户容器, 使得代码阅读和开发变得困难。

沙箱崛起

时间来到2018年，云原生领域飞速发展，许多沙箱(Sandbox)隔离技术都被应用到容器领域，沙箱容器如火如荼。沙箱天生就是个 Pod，它为容器组提供了一个隔离的环境，不同的容器组之间具有更强的安全隔离性。目前容器领域应用较多的沙箱有轻量虚拟机沙箱(MicroVM Sandbox)、用户态内核沙箱(Application Kernel Sandbox)和WebAssembly 沙箱(Wasm Sandbox)。它们都是为满足不同业务需求而孵化的产物，所以它们在不同的维度上有着各自的优势。

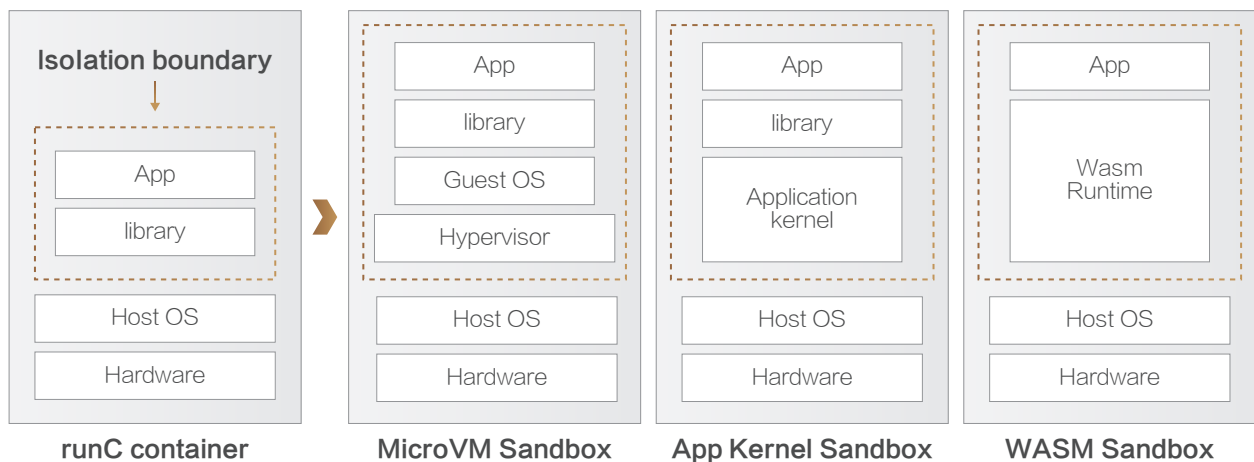
相比于原生的 docker runC 容器，MicroVM Sandbox 适合安全性和兼容性要求高的场景，Application Kernel Sandbox 适合快速资源收缩和安全性高的场景，而 Wasm Sandbox 适合像函数计算这种轻量任务的场景。

目前，主流云厂商都在生产环境中使用了多种沙箱技术，但是没有合适的容器运行时可以同时兼容这些沙箱，每个沙箱借助 Shim v2 均实现了一套管理面的程序，彼此之间不相兼容。因此，云厂商需要维护多个运行时来管理多种沙箱，这无疑增加了运维成本。

Kuasar 诞生

就在刚刚的2023年3月，主流的容器运行时 containerd 在 v1.7.0 版本中(<https://github.com/containerd/containerd/releases/tag/v1.7.0>)引入了 Sandbox API 特性。

Sandbox API 提供了一种管理沙箱容器的新思路，它将容器和沙箱的概念解耦，容器归容器，沙箱归沙箱，Pod 就是沙箱。我们坚信，Sandbox API 的出现使得沙箱成为容器世界新的一等公民，我们需要一个支持多种主流沙箱技术的容器运行时，同时具有可扩展、可维护、可演进机制。因此，Kuasar 应运而生。



Sandbox Classification	Advantage dimension			Vendor		
	efficient	secure	General	AWS	GCP	Azure
runC container	✓		✓	✓	✓	✓
microVM		✓	✓	firecracker	✓	✓
App Kernel	✓	✓			gVisor	
WASM	✓	✓				wasmtime

架构

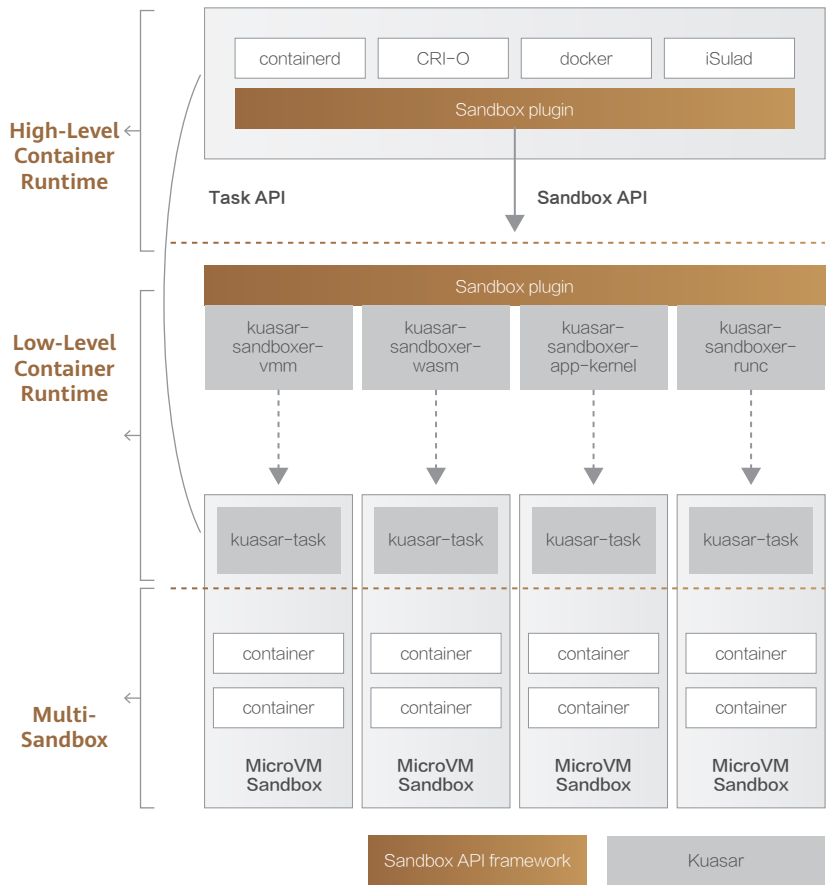
架构总览

Kuasar 是一个多沙箱容器运行时，那么什么是容器运行时？简单说容器运行时是一个负责拉起容器，管理容器运行状态的运行时组件，可以分为高阶容器运行时和低阶容器运行时两类：

高阶容器运行时：负责 CRI 的实现，从高维度管理容器和镜像实例，containerd, CRI-O, docker 还有 iSulad 都是典型的高阶容器运行时。

低阶容器运行时：负责容器和沙箱的具体操作，比如创建删除、资源统计和更新，Kata-containers 和 runC 等都是低阶容器运行时。

从总览图中可以看出，Kuasar 属于低阶容器运行时，和高阶容器运行 containerd 交互，Kuasar 主要由两个模块组成：



Kuasar-Sandboxer: 实现了 Sandbox API，负责管理沙箱生命周期和资源分配。Sandboxer 以插件的形式和上层高阶容器运行时 containerd 交互，关于插件的描述，详见 containerd 社区议题(<https://github.com/containerd/containerd/issues/7739>)。

Kuasar-Task: 实现了 Task API，负责管理容器的生命周期和资源分配。

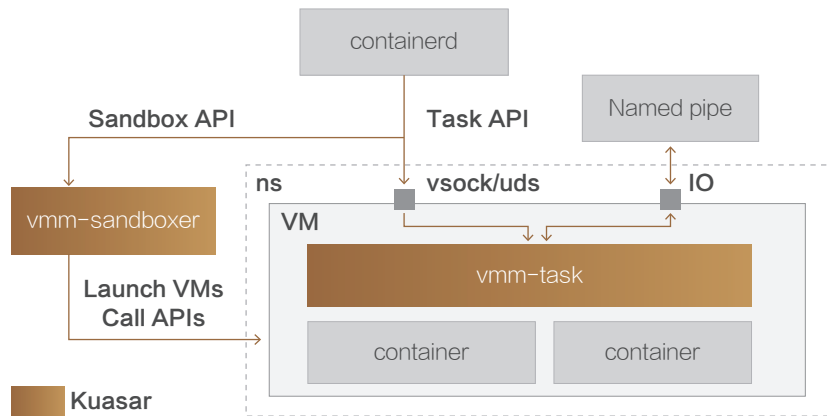
目前，在北向接口层面，Kuasar 正在与 containerd 联合构建最新的沙箱接口标准，sandboxer 插件已经加入 containerd v2.0 的版本路标；此外，openEuler 社区的轻量级容器引擎 iSulad 项目也已经完成与 Kuasar 项目的对接。而在南向沙箱层面，Kuasar 已经支持包括 Cloud Hypervisor (MicroVM 类)、WasmEdge (Wasm 类)、StratoVirt (MicroVM 类)、Quark (App Kernel类) 在内的多类主流安全沙箱。我们已在 Roadmap (<https://kuasar.io/docs/roadmap>)中计划支持更多的沙箱，在未来可以适应更多的云原生场景。

MicroVM Sandboxer

在轻量级虚拟机场景，虚拟机提供了完整的虚拟化层和 Linux 内核，此类的虚拟机包括 Cloud Hypervisor(<https://www.cloudhypervisor.org/>)、StratoVirt(<https://gitee.com/openeuler/stratovirt>)、Firecracker(<https://firecracker-microvm.github>).

io) 和 QEMU(<https://www.qemu.org>)。MicroVM Sandboxer 的 vmm-sandboxer 负责创建虚机和调用 API, vmm-task 作为虚机里的 init 进程负责拉起容器进程, 容器的 IO 流则可通过虚机的 vsock 或 uds 导出。

目前仅支持 Cloud Hypervisor, QEMU 和 StratoVirt。



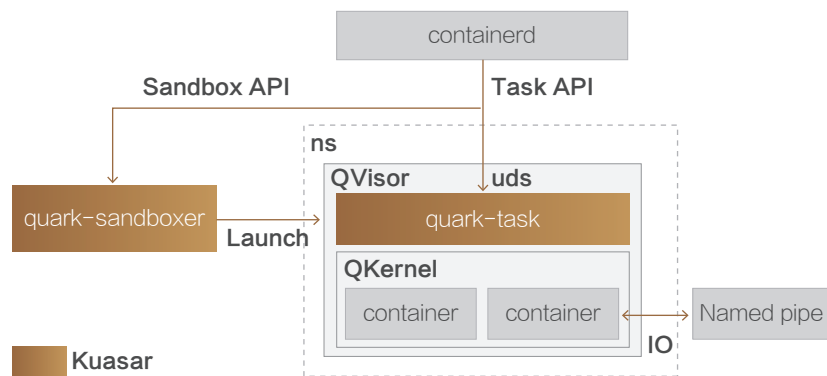
App Kernel Sandboxer

MicroVM 虽然可以大幅降低容器的内存占用, 但是厚重的虚拟化层和标准的 Guest 内核 (如 Linux) 依然给容器带来不可忽略的启动时延和内存占用。而 App Kernel Sandbox 采用了一种更激进思路, 它将 KVM 虚拟化层和 Guest 内核进行深度融合, 采用 9p 或者 io_uring 技术替换标准的设备虚拟化, 实现 IO 通道, 这可以大幅降低虚机启动时间和内存占用。典型代表包括 gVisor(<https://gvisor.dev/>) 和 Quark(<https://github.com/QuarkContainer/Quark>)。Quark 是一款 App Kernel 沙箱, 使用自己的 QVisor hypervisor 和自定义内核 QKernel。QVisor 只负责 KVM 虚拟机

的生命周期管理, 并不模拟任何设备。Qkernel 拦截所有的 syscall, 在需要的情况下, 通过 VM_Exit 或者 eventfd 通知 QVisor 处理。通过将主机进程的内存空间映射到 VM 的物理内存空间中, 实现 QVisor 和 QKernel 的内存共享。App Kernel Sandboxer 的 quark-sandboxer 拉起

Qvisor 和 Qkernel, 每当 containerd 需要在沙箱中启动一个容器时, QVisor 中的 quark-task 将调用 Qkernel 来启动一个新的容器。同一 Pod 中的所有容器都将在同一个进程中运行。

目前仅支持 Quark。



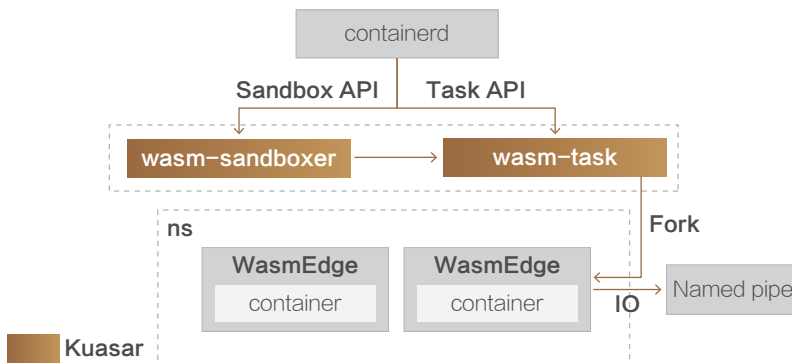
Wasm Sandboxer

如果说 App Kernel 沙箱是在虚拟化和内核层面实现了一套隔离沙箱技术, 那么 WebAssembly 沙箱则是定义了一套新的体系结构, 包含一套指令集和虚拟机。所有程序必须

编译成 WebAssembly 指令集才能在 WebAssembly 虚拟机中运行。另外 Wasm 社区定义了一套 WASI 接口 (对应于 POSIX 接口), 实现对 IO 和网络等资源的访问。由于新的体系结构引入、对部分语言尚未充分

支持, 以及 WASI 接口尚未完善等原因, 目前可能对某些应用程序有限制。但是 WebAssembly 沙箱 (例如 WasmEdge 或 Wasmtime) 可以在保证安全隔离的条件下, 做到极致轻量。当前已经在边缘、函数计算、WEB3

等场景得到应用。wasm-sandboxer 和 wasm-task 在 WebAssembly 沙箱内启动容器。当 containerd 需要在沙箱中启动容器时，wasm-task 将 fork 一个新进程，启动一个新的 WasmEdge runtime，并在其中运行 Wasm 代码。同一 Pod 内的所有容器都将与 wasm-task 进程共享相同的 Namespace 和 Cgroup 资源。

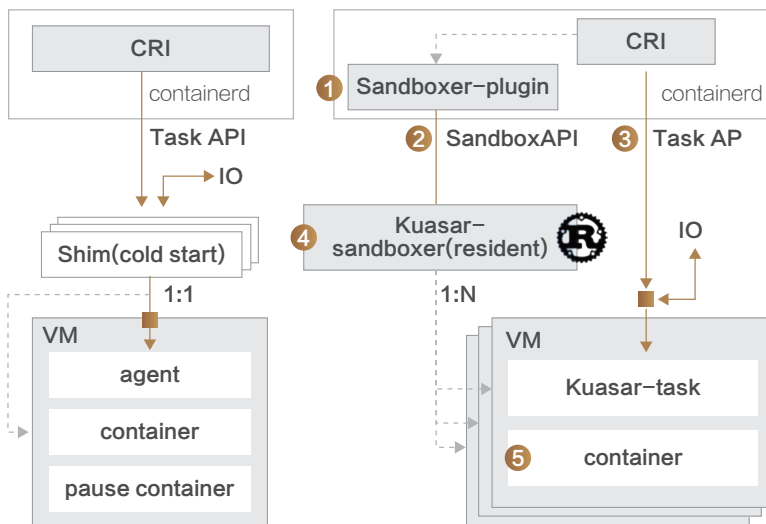


目前仅支持 WasmEdge. 受到某些技术限制(主要是标准输入输出无法重定向), wasm-task 使用 fork 的方式新启动一个 runtime。后续演进可能会选择直接在进程内启动 runtime，实现更快速地启动和更低的内存占用。

模型演进

在目前容器运行时的 Shim v2 模型中, containerd 每创建一个 Pod, 就要创建一个对应的 Shim 进程用于 Pod 的管理, Shim 进程再去创建虚拟机和容器, 在这种场景下, 管理面 Shim 进程和 Pod 的数量关系为 1:1. 但在 Kuasar 中, 只需要运行一个 Kuasar-Sandboxer 进程, containerd 对 Pod 的管理都是调用 Sandboxer 对外暴露的接口, 不再需要为每个 Pod 都拉起一个管理进程, 因此, 管理面 Sandboxer 进程和 Pod 的数量关系为 1:N. 这种模型可以大大减少常驻进程的数量, 整个架构也因此变得更清晰简洁。

Current shim API vs Proposed Sandbox API(Kuasar)



*The comparison in the preceding figure is based on the microVM sandbox scenario

性能表现

说到这, 那么 Kuasar 的性能表现究竟如何呢? 我们选择大家最关心的“端到端容器启动时间”和“管理面组件内存消耗”作为衡量 Kuasar 性能的两个指标, 具体定义如下:

端到端容器启动时间: 在 containerd v1.7.0 的 CRI 实现中, 摒弃了将沙箱伪装成容器的做法, 改用 Sandbox API 新特性来创建沙箱和启动容器, 因此我们需要以 CRI 为入口, 实际测试端到端拉起一个容器进程所需要的时间。

管理面组件内存消耗: Kuasar 作为容器运行时的管理组件, 要额外消耗多少内存, 不包括底层 Sandbox 消耗的内存。

为了对比, 我们选择和开源项目 Kata-containers v2.5.2 进行测试, 两个运行时底层都采用了 Cloud Hypervisor 作为沙箱。为了排除其他因素的影响, Cloud Hypervisor 二进制版本、虚拟机的镜像、虚拟机的内核均保持一致, 容器存储驱动均采用 Overlayfs, 容器镜像相同, 并直接使用本地镜像快照以免访问镜像仓库, 容器网络为 HostNetwork 模式, 避免调用 CNI 插件配置网络。

启动时间

端到端容器启动时间(包含虚机的启动时间)可以通过 `crictl run` 执行的时间衡量,该命令包含对 CRI 接口 `RunPodSandbox`, `CreateContainer` 和 `StartContainer` 的调用。

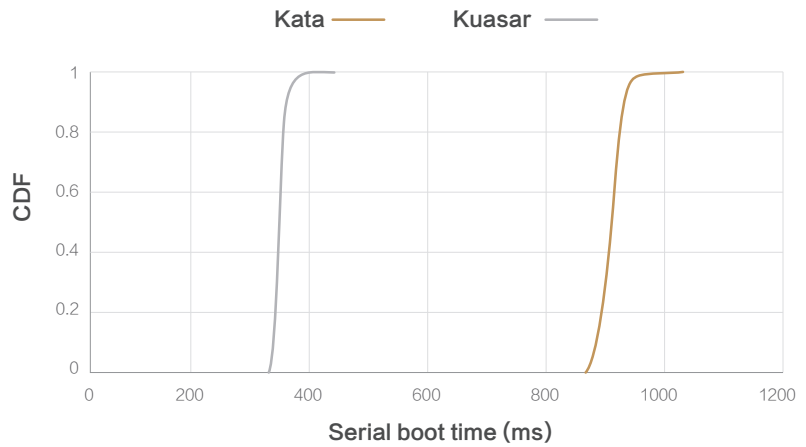
测试分为两组,一组统计单个 Pod 的启动时间,另一组统计并行启动50个 Pod 的时间。每组执行500轮,测得的数据绘制成概率统计分布(CDF)图:

Kata单个 Pod 启动时间大约在850~950 ms,而Kusar 只有 300~360 ms,耗时约为 Kata 的40%。并行启动50个的场景下,Kata 需要1600~1800 ms,而Kusar 只有930~1050 ms,约为 Kata 的60%。通过分析,100%的启动速度提升主要得益于两方面,一方面是 Sandbox API 的实现,使得创建容器不再单独创建 `pause` 容器,节省了准备`pause`容器镜像快照的时间;另一方面得益于1:N 的管理模型,Sandboxer 进程常驻,从而节省了冷启动 Shim 进程的时间,这使得容器的启动速度大大提升。

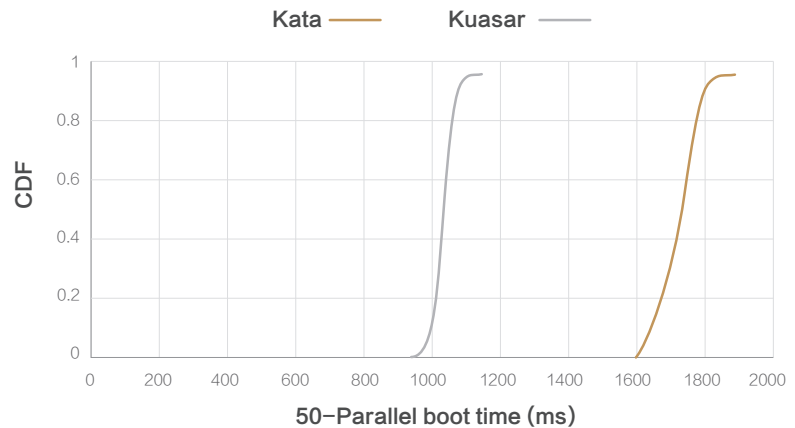
内存消耗

管理面的内存消耗主要是对比 Kusar 的 `Sandboxer` 进程和 Kata 的 `Shim v2` 进程的PSS内存(Proportional Set Size),PSS 是单个进程运行时实际占用的物理内存,包含比例分配共享库占用的内存,可以明确地表示了单个进程在系统总内存中的实际使用量。

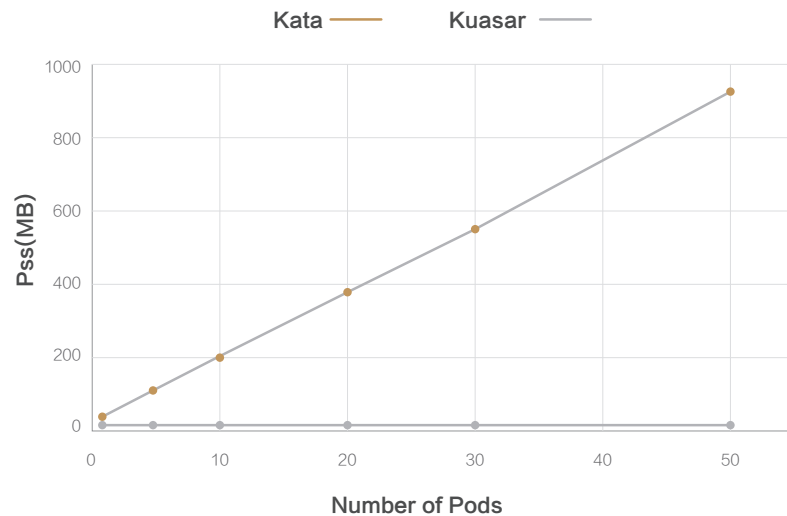
测试共分三轮,每轮分别启动了1、5、10、20、30和50个Pod,查询进程 / `proc` 目录下 `smaps_rollup` 文件中的



单个 Pod 启动时间



并行 50 个 Pod 启动时间



PSS 数值, 对于 Kata 则需要累加所有 Shim 的 PSS 数值。三轮测试得到的数据取平均数后, 绘制成下图:

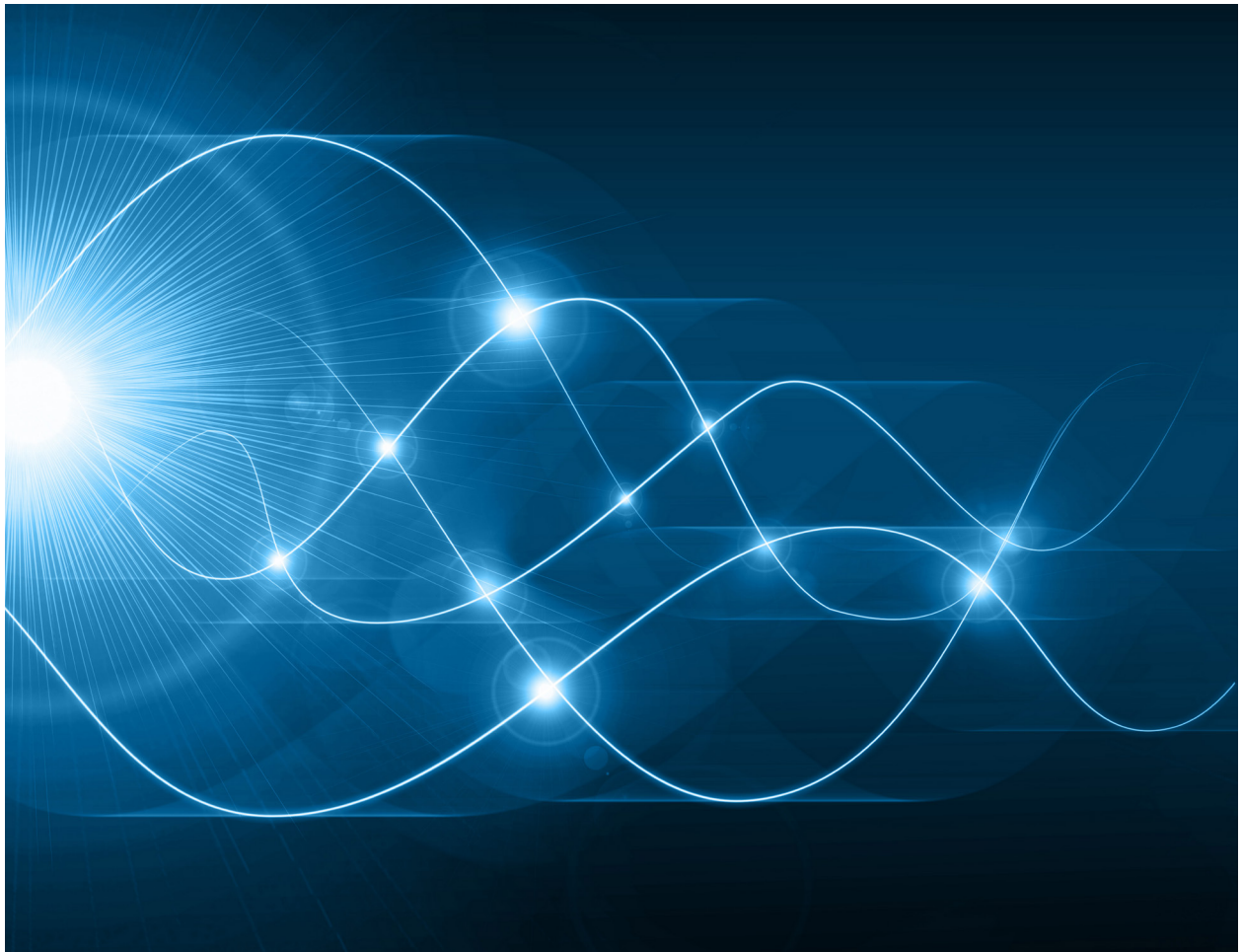
空载的 Sandboxer 进程 PSS 为7MB 左右, 随着 Pod 创建的数量不断增加, Sandboxer进程的内存占用也有小幅上升, 当 Pod 数达到50个时, PSS 只

增加到15 MB, 而 Kata 单个 Shim 进程就占用18 MB内存, 创建50个 Pod 需要消耗将近900 MB 内存。在这一点上, Kuasar 节省近99%的内存, 其背后的原因也可分为两点: 主要是 1:N 的管理模型使得 N 个进程减少为1个进程, 带来的内存收益与 Pod 数成正比; 其次, Kuasar 采用了 Rust

编程语言, 相比于 Kata Shim 进程使用的 Golang 语言, 没有 runtime 运行时, 且内存安全, 因此语言本身也会带来一些内存收益。综上, Kuasar 的性能表现是相当强悍的, 在测试场景下相较于 Kata-containers 具有 100%的容器启动速度提升和99%的管理面内存开销优化。

结 语

面向未来, 作为一个开放和可扩展的多沙箱容器运行时, Kuasar 将发挥沙箱接口的优势, 拥抱业界 DRA(Dynamic Resource Allocation) 和 CDI(Container Device Interface) 等管理接口, 为云原生场景带来更安全、高效、便捷的容器解决方案, 为云原生应用提供更安全的保障。非常期待大家能够体验 Kuasar, 如有任何疑问, 可通过官方GitHub issue 或微信交流群提交问题。



云原生数据库 八大关键技术解读(一)

来源:《云原生数据库白皮书》

数据库上云的技术演进过程中,大体经历了三个阶段:云托管、云服务和云原生阶段。在不同的阶段里所催生出的关键技术也有所不同,云托管阶段基本延续了传统集中式和分布式数据库的技术特点,本质上是将原来服务器上的部署方式照搬到云主机上,并没有针对云环境的定制开发。此阶段主要解决了机房的管理问题。在云服务阶段,通过强大的云管控平台能力,为用户带来了免安装部署和简单且完善的运维能力。此阶段关键技术主要在于强大的云平台工具,且做到了多产品统一运维管理。而数据库产品本身并没有比线下数据库有更多的核心技术演进。

随着云时代的不断发展,华为云基于服务数百万企业客户的经验沉淀,以

及结合自身 30 年的数字化实践总结提出“基础设施现代化、应用架构现代化、开发运维现代化、治理运营现代化”的观点。其中,基础设施现代化的本质是通过将企业应用迁移上云,使用容器、多元算力、Serverless、分布式云等技术,对应用的底层架构进行重构,实现承载应用基础设施资源的高弹性和高可用,最大化利用云平台的技术和优势。应用架构现代化则是指使用微服务、Serverless 等技术,将应用拆分为能独立运行,满足客户需求的独立模块,实现应用的高可用、弹性扩展。应用架构现代化是应用走向敏捷开发的基础。

结合数字化和云原生发展,数据库也开始了从云服务到云原生的蜕

变,例如,云计算带来的资源按需使用、快速弹性扩展、高性能、高可用等,都受限传统的数据库系统架构,而无法在云服务模式下充分对外提供。因此,云原生数据库分别针对容器、微服务、多元算力、Serverless、分布式云高可用、弹性扩展等云原生技术从产品本身做出了关键技术的革新。主要包括:云原生 Serverless、备机资源池、ALT(应用无损透明)、多可用区强一致部署访问、Global Database、NDPQ(NDP+PQ,近数据计算下推+并行查询计算)、日志即数据、云原生 HTAP、全局一致性读、云原生多主等,通过这些改变,摆脱了传统的数据库系统架构的局限性,完全发挥出云计算的优势。下面介绍一些具有代表性的关键技术:

(一) 云原生 Serverless, 让企业摆脱服务器的束缚

云原生数据库存算分离架构具备高性能、高可靠、易扩展的特性,但始终没有摆脱服务器的束缚。数据库所需资源必须首先经过规划、配置,还需对数据库和资源进行监控和管

理。Serverless 引入创新的设计思路,它封装了几乎所有的底层资源和运维工作,使企业更容易使用云基础设施。从这个意义上说,Serverless 是云计算的下一个阶段。Serverless

最重要的三个特征是:隐藏服务器概念、提供极致的弹性伸缩能力和提供按需付费的计费模型,以及兼容处理各种类型的负载。



图 6 传统部署与 Serverless 部署对比

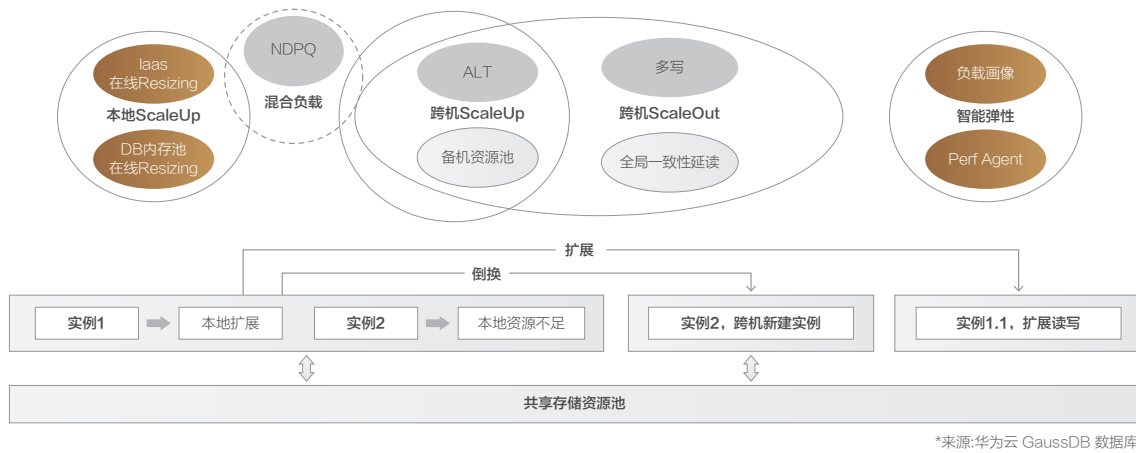


图 7 云原生 Serverless 技术栈依赖

但目前市场上的数据库 Serverless 产品还受到单机垂直伸缩及无状态管理机制的限制，主要适用于单机内部的弹性伸缩，跨机伸缩普遍会引起连接中断使得业务受损，因此主要适用在一些互联网和游戏行业的简单场景中。而云原生数据库 Serverless 将从面向资源转变至面向应用，支持更多的通用型场景，实现事务状态分布式管理。

为了实现云原生数据库 Serverless，云原生数据库产品既需要具备本地的弹性伸缩，也要支持跨机的横向

扩展，规格上限不局限于单台服务器的上限；同时，Serverless 要具备支撑多种混合负载的能力，不拘泥于完全的 OLTP 或 OLAP；为了准确地把用户的资源弹性到所需的容量，还需要精确地预测用户的负载曲线，提前弹性扩容。

为了实现本地的纵向扩展，作为云底座的 IaaS (Infrastructure as a Service) 平台首先需要具备在线调整资源分配的能力，同时运行在 IaaS 平台上的数据库也要同时具备在线伸缩自身内存池的能力，匹配

基础资源的弹性规格。做到 IaaS 分配资源时能用得上，IaaS 回收资源前要能释放出资源。对于跨机的弹性，为了实实例的快速转移，避免临时申请 IaaS 资源消耗时间，管理平台需要事先预测并规划准备好相应规格的 IaaS 实例，用于接管数据库，比如华为云原生数据库的备机资源池技术，就能实现这个目标。除了快速，Serverless 实例迁移还需要应用无感知，那么数据库需要具备事务状态的分布式迁移能力，能将事务从一个实例在线地迁移到另一个实例，比如华为云原生数据库的 ALT

(应用无损透明)技术,就可以实现这样的能力,当数据库在进行主备切换、规格变更或者小版本升级时,

系统可以复制用户的后台会话,在达到安全的事务边界后,确保后端会话操作上下文被完整克隆至目标节

点,从而完成主备切换,保证业务无影响。

(二)多可用区强一致部署,数据访问更加可靠

对于一些可用性要求高的业务,不光需要能应对部分机器故障的问题,还应能应对机房故障、城市灾难等极端问题。为了体现云的跨可用区的高安全、高可用能力,云原生数据库也需要具备跨可用区的部署能力,并且提供跨可用区的读一致性访问,多可用区节点必须读到一致的数据。2个可用区部署下需要保证单可用区故障不影响云数据库的读访问,3个可用区部署下需要保证单可用区故障不影响云数据库的正常读写。单盘访问故障、可用区间网络短时抖动故障基本不影响云数据库的性能。基于存储层底座的跨可用区部署能

力,存储层可以保证数据被一致地分发到不同可用区的节点。存储层应确保全场景负载数据文件绝对可靠,至少三副本存储,并具备多副本强一致访问能力,单副本故障不影响数据可靠性和访问速度,并且用新节点替换故障节点后,故障可以自动恢复并补全新节点的数据。以华为云 GaussDB(for MySQL)云原生数据库为例,存储底座由数百个节点组成,因独立节点故障导致存储层无法写入的概率几乎为零,因为如果所选节点不可用,则可以选择任何其他节点,而这些节点分布在3个可用区中,同时故障概率极低。单个节点故障会影

响延迟,因为失败的写入必须用不同的日志存储节点组重试,但是不影响可用性。对于存储节点中的日志存储节点,一旦不可用,日志存储上的所有存储对象都停止接受新的写入,并变为只读。因此,在短期故障之后不需要恢复。当诊断出长期故障时,故障节点从集群中移除,故障节点上的存储对象副本从可用副本中重新创建到集群中的剩余节点上;对于存储节点中的页面存储节点,每个节点根据其持久化 LSN 决定是否可以为读操作提供服务,实现了云原生数据库跨可用区强一致部署访问能力。

(三)Global Database,数据全球就近访问和实时灾备

随着国内企业规模出海走向世界,原有的本地集中式IT布局难以应对,一方面是用户体验差,跨区域访问时延高,难以满足生产业务甚至是日常办公的需求;另一方面是安全隐患大,某些国家区域的基础设施条件较差,可用性不能保证,如有些国家市电频繁停电、机房高温,可能导致数据丢失。因此,全球分布、统一管理、云原生的IT基础设施将成为核心竞争力。Global Database充分利用云基础设施跨地域分布的能力,通过将多

个数据库实例部署在多个地域,实现全球范围内的低时延同步、本地就近访问、跨地域容灾等能力。确保各地域的应用访问时延小于1秒,当主集群出现地域级别的故障时,1分钟内可进行恢复。Global Database可采用多种策略进行数据全球同步,如基于 REDO 日志、基于逻辑日志或者基于中间件(数据迁移服务)等。其中,基于 REDO 日志的方式由于其较好的 RTO 表现,可做到30秒到1分钟、亚秒级的 RPO(恢复点目标)、

支持多活和 HA(双机),是目前最为常用的技术。基于 REDO 日志的 Global Database,主实例和其他目标实例之间通过 REDO 日志进行全量和增量数据的同步,并在目标实例进行日志回放,数据在目标实例低延迟可读;当主实例故障后,目标实例可进行快速倒换,当主实例恢复后,目标实例可进行回切并补齐数据;通过目标实例发起的写请求可以转发到主实例,并保证多区域的一致性,基于 RPO 目标指定事务的容灾复制策略。

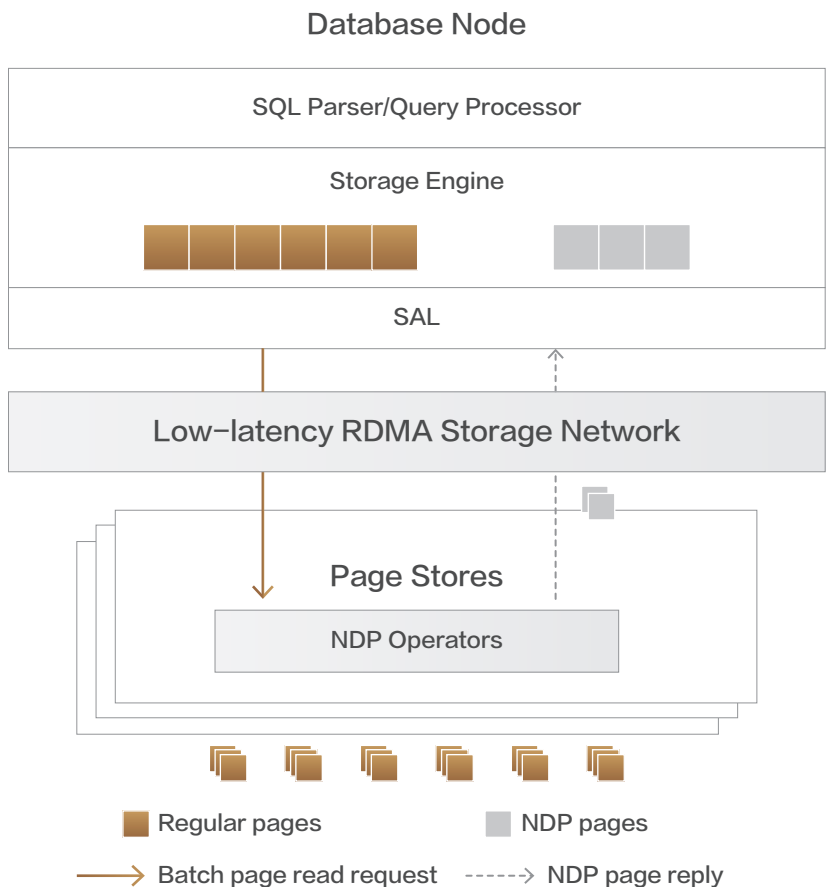
(四) NDPQ 近数据并行查询, 充分利用每一份资源

云原生数据库要想充分利用云的计算资源, 首先应采用“计算存储分离”的架构。在计算存储分离的架构下, 由于计算和存储节点通过网络通信, 网络带宽和延迟往往成为制约。在传统数据库中, SQL 执行引擎从存储中获取数据, 并执行包括投影、谓词计算和聚合在内的步骤。对于经常涉及大型表扫描的分析查询, SQL 执行引擎必须从存储中读取大量数据页。当存储节点与计算分离, 通过网络通讯时, 大表扫描会转化为增加的网络流量。一个典型的例子是对一个非常大的表进行计数查询, 查询对象表的所有页面必须从页面存储池 (Page Store) 发送到要计数的数据库节点, 之后, 数据库节点将丢弃这些页面中的大部分, 因为缓冲区池不能装载这么多数据页, 这是对网络带宽资源的浪费。NDP(Near-Data Processing) 即近数据计算下推的设计思路是避免在分布式系统中移动数据, 并让数据处理在其所存储的地方进行。在云原生数据库中, 存储节点通常由大量性能强大的服务器组成, 这些存储节点上的 CPU 资源利用率较低, 这就为近数据处理 (NDP) 提供了一个绝佳的机会。NDP 功能将选定的 SQL 操作下推到页面存储中, 页面存储过滤掉不必要的数据, 只将匹配的数据子集返回给数据库节点进一步处理。例如, 要处理计数查询, 页数据存储可以计数行, 并将计数而不是实际数据页返回到数据库节点。这样就避免了大量的网络流量, 使用此技术也提升了查询响应时间。GaussDB(for MySQL) 的 NDP 特性架构如下图

所示。数据库节点向页面存储发送 NDP 请求 (请注意, 通常有多个页面存储服务于每个数据库节点), 为了降低 IOPS (每秒 IO 数), 将多个页面分组为一个 NDP 请求 (批量页面读取请求), 页面存储中的 NDP 运算符可以执行上述三种 SQL 操作, 并将较小的数据集返回到数据库节点。数据库节点可以是主节点, 也可以是只读副本节点, 两者都支持 NDP。

例如, 要计算基于某一列的求和值, 传统的方法是通过 IO 读取扫描范

围内的每一个页面到计算节点, 再解析页面内的行, 计算求和值, 这里涉及到大量的 IO, 以及对计算节点计算资源的占用。而基于 NDP 的优化, 页面解析可以在存储节点由专门的 NDP 线程处理, 它会把页面内相关记录的列的求和值直接本地化计算, 最终通过 IO 返回的, 可能只是一行包含本页面最终求和值的结果, 极大地减少 IO 使用量。NDP 得益于华为云的全栈技术能力, 可以在存储层对 SQL 层的下推算子进行处理。



*来源:华为云 GaussDB 数据库

图 8 华为云 GaussDB(for MySQL) 的 NDP 特性架构

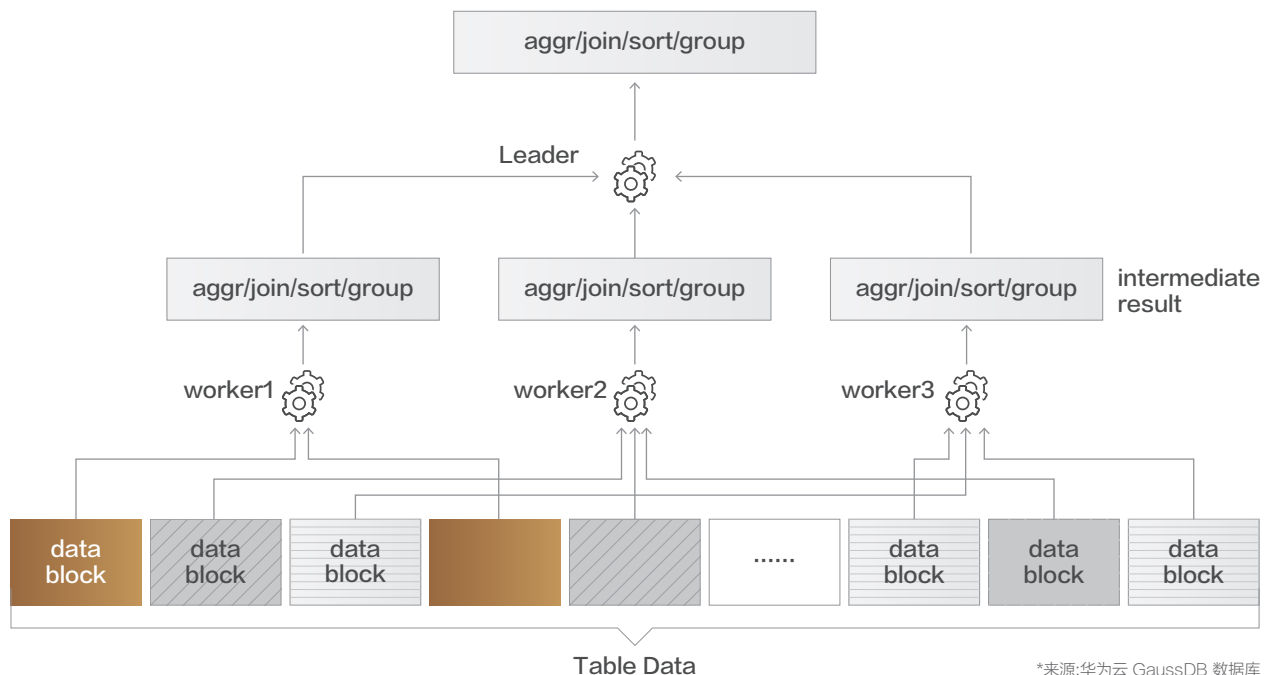
众所周知，软件计算能力的提升一方面依托于 CPU 硬件能力的增强，另一方面也受益于软件设计层面能够充分利用 CPU 的计算资源。当前的处理器普遍采用多核设计，单线程查询的方式至多能用满一个核的 CPU 资源，性能提升程度有限，远远无法满足企业大数据量查询场景下对降低时延的要求。因此，复杂的查询分析型计算过程必须考虑充分利用 CPU 的多核计算资源，让多个核参与到并行计算任务中才能大幅度提升查询计算的处理效率。

下图展示了如何通过多线程并行查询，达到充分利用 CPU 多核资源而提高查询性能的方案。

假设我们要计算某张表的 count(*) 结果，首先把表数据按照一定的粒度，切分成若干个连续的子数据集；之

后启动多个工作线程，每个工作线程负责处理切分过的部分数据，包含若干个子数据集；每个工作线程需要对自己负责的子数据集中的数据计算 count(*)，得到局部的 count(*) 结果；最后各个工作线程把计算得到的 count(*) 的结果汇总给一个协调线程，协调线程把若干个 count(*) 的结果相加，得到整张表的 count(*) 结果。通过将数据切分再转发给多个工作线程，使得多个工作线程可以并发处理数据，以提高查询的效率。这只是 count(*) 的一个例子，实际上很多查询语句的聚合、函数运算和联合等操作都可以进行并行处理，通过对一个查询的每个子步骤都进行并行处理，使得整个查询能最大程度地利用 CPU 的多核优势，更快地完成查询任务。并行查询 (PQ) 是商业关系型数据库系统的事实标准，为分析工作负载提供高性能支持。PQ 通常

采用“leader-worker”设计，要处理的表被划分为非重叠的数据块，并把这些数据块分配给多个 worker 处理。每个 worker 都会产生中间结果，leader 会累计这些结果并做进一步处理，以产生最终结果。PQ 在数据库节点中提供并行性，利用多个 CPU 并发处理查询。华为云 GaussDB(for MySQL)具备 PQ 特性，而且 NDP 和 PQ 可以协同工作，演进到 NDPQ，进一步提高查询性能。NDPQ 在多个层面进行了多重并行，首先在 SQL 节点，由并行查询驱动的并行，多个工作线程并行处理查询，并行下发页面读取请求；其次由于数据页面被打散存储在不同的存储节点，IO 下发后有跨页面存储节点的并行；而当 IO 请求到了单个页面存储节点后，该节点还可以批量并行地读取页面。后两层并行依靠的是在 SQL 节点下发 IO 请求时的逻辑批量读进行优化。



*来源:华为云 GaussDB 数据库

图 9 多线程并行查询提高查询性能方案

AudioGPT, 全新的 音频内容理解与生成系统

赵洲 / 计算机科学与技术学院教授、博士生导师

ChatGPT、GPT-4等大型语言模型(LLM)在语言理解、生成、交互和推理方面表现出的非凡能力,引起了学界和业界的极大关注,也让人们看到了LLM在构建通用人工智能(AGI)系统方面的潜力。

现有的GPT模型具有极高的语言生成能力,是目前最为先进的自然语言处理模型之一,广泛应用于对话、翻译、代码生成等的自然语言处理领域。除了书面语言,用户在自然对话中主要使用口语(Spoken Language),而大语言模型在音频处理领域的福利还差一些:

- » GPT模态限制。用户在自然对话中主要使用口语,对口语理解与合成有极大需求,而单模态GPT不能满足对音频(语音、音乐、背景音、3D说话人)模态的理解、生成需求。
- » 音频数据、模型相对少。基础模型(Foundation Model)少或交互性差。相较于文本模态,用于重新训练语音多模态GPT的数据较少。

» 用户交互性差。用户广泛的使用语音助手如Siri, Alexa基于自然对话高效地完成工作。然而目前GPT之间的交互大多根据键盘输入的文本,交互性差,口语交互更能拉近和用户之间的关系,提升模型易用性。

为应对以上挑战,浙江大学、北京大学、卡内基梅隆大学、中国人民大学的研究人员提出了全新的音频理解与生成系统 AudioGPT。AudioGPT以 ChatGPT 充当负责对话与控制的大脑,语音基础

模型协同完成跨模态转换、以及音频(语音、音乐、背景音、3D 说话人)模态的理解、生成,能够解决 20 + 种多语种、多模态的 AI 音频任务。

论文中有一个多轮对话与上下文理解的例子,可以帮我们更好的理解 AudioGPT是如何与人交互的,示例涵盖文本、图像和语音三种模态的输入, AudioGPT 能够理解不同模态的输入,生成结果也能以文本、图像、音频相结合的方式展现给用户。

AudioGPT: Understanding and Generating Speech, Music, Sound, and Talking Head

Rongjie Huang¹*Mingze Li¹*Dongchao Yang^{2*} Jiatong Sh^{3*} Xuankai Chang³ Zhenhui Ye¹,
Yuning Wu⁴, Zhiging Hong¹, Jiawei Huang¹, Jinglin Liu¹,
Yi Ren¹ Zhou Zhao¹, Shinji Watanabe³

Zhejiang University¹, Peking University²,
Carnegie Mellon University³, Remin University of China⁴

{rongjiehuang, limingze, zhaozhou}@zju.edu.cn, {dongchao98}@stu.pku.edu.cn,
{jiatongs, xuankaic, dongslj}@cs.cmu.edu, {yuningwu}@ruc.edu.cn, {shinjiw}@ieee.org

论文地址: <https://arxiv.org/pdf/2304.12995.pdf>

AudioGPT语音模型在现有模型中具有不同的优势:

1. 相较于文本GPT: ChatGPT、GPT4等语言模型侧重于自然语言处理领域,在多模态理解与合成上福利少了一些,AudioGPT则将重点放在了音频多模态的理解与生成。

2. 相较于Siri等检索模型: 与简单检索并以语音回复检索模型相比,AudioGPT强大的生成式对话,使得用户能够更加实时地获取和处理与内容相关的音频。

3. 相较于视觉任务GPT: 图像不存在像音频一样的语种分类,音频中存在不同语种: 如中文、英文、法语等,AudioGPT还需要进行语种管理。

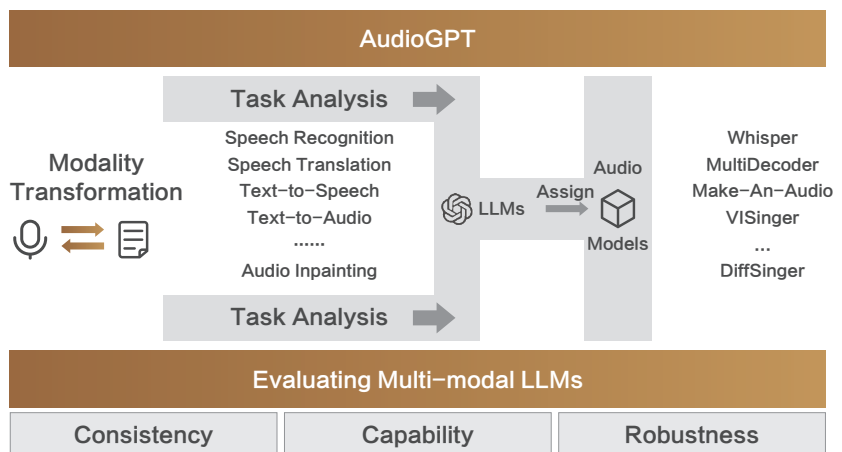
AudioGPT在收到用户请求时使用ChatGPT进行任务分析,根据语音基础模型中可用的功能描述选择模型,用选定的语音基础模型执行用户指令,并根据执行结果汇总响应。借助ChatGPT强大的语言能力和众多的语音基础模型,AudioGPT能够完成几乎所有语音领域的任务。

AudioGPT 为走向语音通用人工智能开辟了一条新的道路。AudioGPT运行过程可以分成4个阶段: 模态转化、任务分析、模型分配和回复生成。



Figure3: Qualitative analysis on multiple rounds of dialogue between humans and AudioGPT.

Figure 1: A high-level overview of AudioGPT. AudioGPT can be divided into four stages with modality transformation, task analysis, model assignment, and response generation. It equips ChatGPT with audio foundation models to handle complex audio tasks, and is connected with modality transformation interface to enable spoken dialogue. We design principles to evaluate multi-modal LLMs in terms of consistency, capability, and robustness.



截至目前，AudioGPT涵盖了语音识别、语音合成、语音翻译、语音增强、语音分离、音频字幕、音频生成、歌声合成等任务。实验结果证明了AudioGPT在处理多模态信息和复杂AI任务方面的强大能力。

目前多模态的LLM层出不穷，Visual ChatGPT, HuggingGPT等模型吸引了越来越多的关注，然而如何测评多模态LLM模型也成了一大难点。为了解决这一难点，研究人员为多模态LLM的性能评估设计了测评准则和测评过程。具体来说，AudioGPT提出从三个方面测评多模态LLM：

- » 一致性（Consistency）：度量AudioGPT是否正确的理解用户意图，并为之分配所需的模型
- » 稳定性（Capability）：度量基础的语音模型在其特定任务上的性能表现
- » 鲁棒性（Robustness）：度量AudioGPT是否能正确的处理一些极端的例子

针对一致性，研究人员设计了一套基于人工测评的流程。如图所示：

Figure 2: A high-level overview of consistency evaluation.

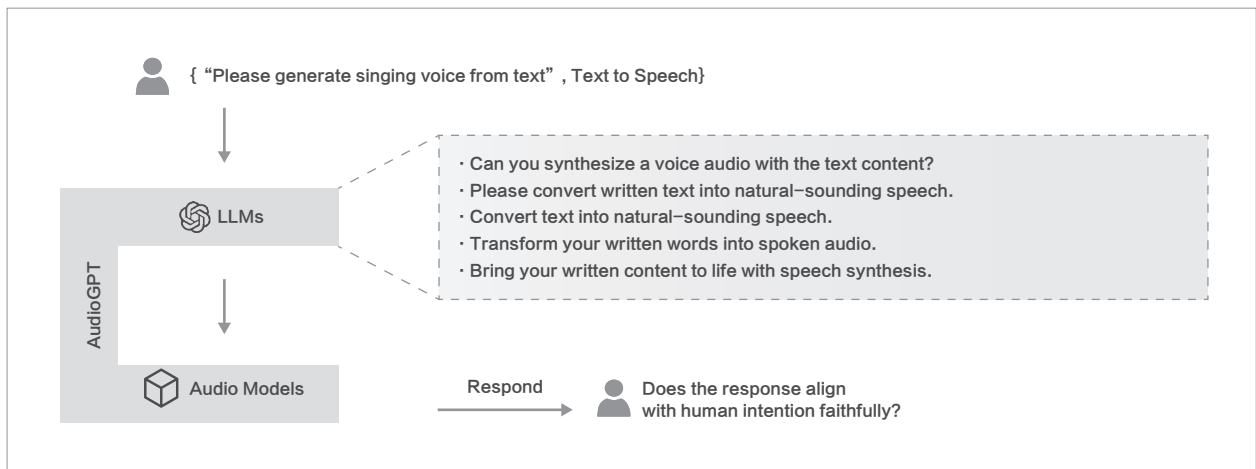


Table 2: Ratings that have been used in the evaluation of query-answer consistency.

Rating	Consistency	Definition
20	Completely inconsistent	Very annoying and objectionable inconsistency.
40	Mostly inconsistent	Annoying but not objectionable inconsistency.
60	somewhat consistent	Perceptible and slightly annoying inconsistency.
80	Mostly consistent	Just perceptible but not annoying inconsistency.
100	Completely consistent	Imperceptible inconsistency

针对稳定性，AudioGPT调研了各语音基础模型在单任务领域的性能表现。

针对鲁棒性, 从四个方面进行评估:

- » 多轮对话的稳定性: 多模态LLM应该具备处理多轮对话的能力并且能处理上下文关系
- » 不支持的任务: 目前多模态LLM并非万能的, 当收到无法解决的请求时, 多模态LLM也应该给予用户反馈
- » 错误处理: 多模态基础模型可能由于不同的原因而失败, 例如不支持的参数或不支持的输入格式。在这种情况下, 多模态 LLM需要向用户提供合理的反馈, 以解释遇到的问题并提出潜在的解决方案
- » 上下文中断: 多模态LLM被期望处理不在逻辑序列中的查询。例如, 用户可能会在查询序列中提交随机查询, 但会继续执行具有更多任务的先前查询

Table 3: Evaluating audio foundation models in AudioGPT.

Task	Audio Model	Dataset	Metrics
Speech Recognition Speech Translation	Whisper MultiDecoder	LibriTTS MUSTC	WER, CER BLEU
Style Transfer Speech Enhancement Speech Separation Mono-to-Binaural Audio Inpainting Sound Extraction	GenerSpeech ConvTasNet TF-GridNet NeuralWarp Make-An-Audio LASSNet	ESD CHME4 WSJ0-2mix BinauralDataset AudioCaption AudioCaption	MCD, FFE, MOS SNR, PESQ, STOI SNR, PESQ, STOI L2 Error, PESO, MRSTFT MOS SNR,PESQ
Sound Detection Target Sound Detection	Pyramid Transformer TSDNet	AudioSet URBAN-SED	mAP F-score
Talking Head Synthesis	GeneFace	LRS3-TEDA	FID, LMD
Text-to-Speech Text-to-Audio Audio-to-Text	Fastspeech2 Make-an-Audio MAAC	LJSpeech AudioCaption Clotho-v2	MCD, FFE, MOS FID, KL,CLAP, MOS CIDEr-D
Image-to-Audio	Make-An-Audio	AudioCaption	MOS
Singing Synthesis	DiffSinger,VISinger	OpencPOP	MCD, FFE, MOS





行业趋势

企业云原生平台核心能力构建方法论(一) 基于 DevOps 的软件工程能力构建 -----	47-50
SaaS 行业展望 -----	51-54
金融数据仓库十大发展趋势-----	55-57
数智融合时代企业业务的提效降本之路 -----	58-60
重识云原生系列四: 硬菜软嚼的云网络(下) -----	61-66

云原生 Cloud Native

企业云原生平台 核心能力构建方法论（一） 基于 DevOps 的软件工程能力构建

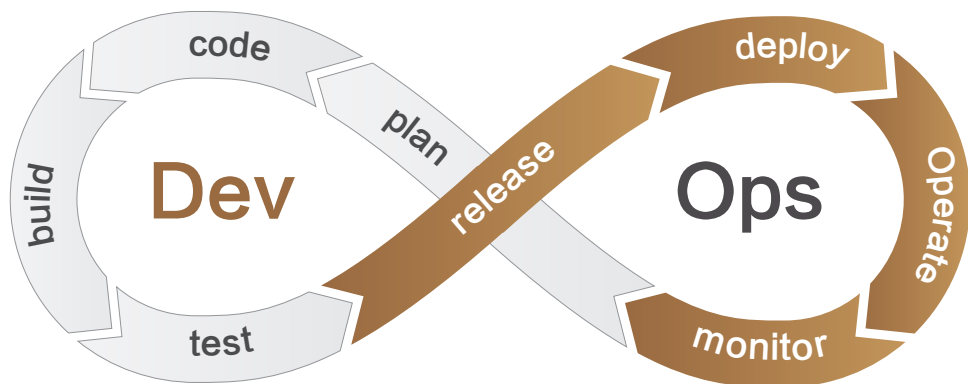
来源:《面向消费终端的企业云原生实践白皮书》

企业软件开发已经进入了新的时代，应用开发、数据开发和模型开发成为了企业软件开发的三个核心领域。DevOps的理念也已广泛被应用于这三个领域，因此构建三大领域端到端的软件工程能力，实现应用、数据、模型的高效交付和运维是核心目标。

同时需要将大容量、高可靠、全球化、安全隐私等关键技术能力下沉在平台内部，以服务化方式向应用提供，通过DevOps端到端工具链为用户提供便捷接入和使用，让开发人员更专注业务代码实现。

基于DevOps 的软件工程 能力构建

云原生DevOps平台的应当以服务使用者为视角，以提升企业应用的开发，运维，运营的全流程运作效率为目标，聚焦Dev-Runtime-Ops进行能力构建。需要以软件交付件为核心，并将流程Built-in到软件生产过程中。



以软件交付件为核心

“业务代码+BaC=软件包，软件包+IaC=可运行环境”。BaC指构建即代码，IaC指基础设施即代码，软件包则应该包括应用、云侧服务以及数据和AI算法等。

**流程Built-in
到软件生产过程**

指BaC和IaC执行过程中可针对流程中需要进行的质量控制项进行控制。比如IaC执行时发现现有网有删除数据库的操作那么主动拦截触发审批，BaC执行时发现此软件版本有使用的开源软件有严重漏洞那么则不允许发布等。

**开发态 (Dev)**

构建项目管理服务和一站式开发平台，通过软件信息树实现全流程可信。解决研发过程中的跨角色、跨组织协作问题，开发人员本地可以完成个人级编码/调测/检视/提交等活动，实现函数/微服务/IaC的一站式操作部署。

**面向云原生应用
开发者提供一站式
DevOps 开发工作台**

覆盖持续开发阶段全链路活动，使能开发者免工具切换完成全栈 workflow，通过个人工作台查看并处理需求、缺陷、检视活动等任务，并能随时监控、管理个人CI&CD workflow

聚焦可信编码

提供IDE工程一键配置、消费软件信息树、部署信息树以及信息仓进行最优软件的推荐、检查，做到过程可信，提供本地代码静态检查以及代码合入能力

**函数开发全
生命周期管理**

提供函数创建、开发、调试、测试、部署、监控和优化的全生命周期一体化作业平台

**微服务开发全
生命周期管理**

提供微服务创建、开发、调试、测试、部署、监控和优化的全生命周期一体化作业平台，支持可视化引用各类中间件、公共组件

IaC 一站式开发部署

提供IaC Model模板化创建、定义自动补全、智能校验、计划预执行能力

**融合 DevOps 持续开发
过程中的工作流和信息流**

结合软件信息树实现编译/构建依赖和漏洞管理

支持本地调测

一键端云联调，基于本地沙箱环境极简化开发调测反馈循环

运行态 (Runtime)

提供大容量、弹性、安全、高可靠的服务运行环境，让业务看不到云，按需快速集成。

可信云服务底座

统一托管业务运行时的IaaS资源（计算、网络、存储等）和数据库，将服务运行时的可信公共能力（可靠可用、服务治理、流量调度、韧性等）下沉到平台，提供包括注册中心、配置中心、灰度发布、限流降级、服务隔离等一系列服务治理能力，实现平台和业务解耦，让业务聚焦代码开发，提升交付和维护效率。

微服务计算平台

微服务改造后，微服务数量急剧发散、调用链路极度增长，在现网出现故障时，定位问题是非常复杂的。因此需要提供一个全托管式微服务运行环境，全面实现容器化和基于流量的弹性伸缩。并提供调用链、日志服务、基础监控等能力，可实现自助诊断能力，而业务只需聚焦自身代码开发。

函数计算平台

函数计算 (Function-as-a-Service 简称 FaaS) 是当今最流行的Serverless计算服务。函数计算将应用程序所需的底层基础设施和管理任务交由平台完成，开发者则可以将更多的精力集中在应用程序的业务逻辑上。通过函数计算平台可提升函数的开发和灰度升级易用性，优化弹性伸缩和调度策略，帮助业务提升研发效率和降低资源成本，助力应用生态全面serverless化。

运维态 (Ops)

随着很多企业逐渐将越来越多的业务系统往云上迁移，企业客户需要将 IT 治理模式延伸或迁移到公有云上。因此需要围绕运维阶段3个核心场景：部署、监控、演练进行自动化、数据化、智慧化运维能力的构建，提升运维效率，为高品质业务提供保障。通过运维管理体系建设，实现流程IT化和可控可回溯的运维安全管控能力，并最终实现无人值守运维。

部署服务

目标围绕部署的全生命周期过程进行管理，通过IaC全自动化部署，实现无人值守变更。依托流程、工具和运营能力，从变更提交到变更结束，打通全流程各个环节，实现场景化的无人值守。

变更管理

完善变更通知，变更检测，变更执行，变更记录各阶段的自动化能力，在变更过程中自动识别高危变更操作，差异化区分不同类别变更操作的处理方式，提升变更流程的自动化率，通过工具和流程保证变更过程和结果的可信。

声明式 IaC

开发人员只需要通过IaC 声明式定义需要资源或状态，后端将复杂的业务逻辑进行封装，简化IaC代码开发。IaC提供标准化模板，并开放参数，定制参数通过规格进行抽象，对类似删除数据库等高危操作进行主动拦截，避免事故。通过支持全球一份IaC代码，让业务维护起来更简单。

服务环境

服务IaC代码执行后生产服务环境，部署服务提供环境的创建，删除，更新，归档，恢复等全生命周期管理，并按照环境聚合环境包含的资源的监控数据，支持对现网存量部署的业务资源按环境维度进行纳管，业务平滑过渡到云原生DevOps平台。

网络自动化

- 抽象“隔离域”概念，对业务屏蔽公有云网络细节，实现网络资源的业务声明式定义和配置自动化。业务通过定义网络隔离域IaC代码实现部署视图IT化，部署服务调用网络服务自动化配置服务需要使用网络信息。关键技术点包括：一、抽象公有云网络资源概念，自动规划和分配网络资源，让业务不感知网络；二、根据服务依赖关系自动生成网络访问策略，实现服务间网络自动按需开通。

监控服务

面向开发和运维SRE，围绕故障生命周期，以自有服务视角汇总IaaS、PaaS、SaaS的监控信息，构建开箱即用的一体化可观测性平台；同时构建运维数仓以及故障RPA处理能力。

故障预防

- 故障预防重点构建健康管家（巡检能力），基于系统化监控及分析能力，面向自有服务和生态开发与运维，提供自定义巡检和编排，可视化度量现网风险，驱动业务改进，提升巡检效率

故障检测

- 围绕服务状态，构建事件中心及全景化事件诊断能力。并给业务提供内网拨测能力，研发发布测试用例能够在拨测平台进行自动化拨测监控，提前发现问题。

故障诊断

- 分为基于专家经验故障诊断工具包括数据库故障诊断、服务调用诊断等诊断工具，同时结合AI算法逐步构建根因分析等智能化诊断能力。

故障通报

- 实现故障通报IT化管理，提升故障通报效率

故障恢复

- 通过知识库、应急预案、故障自愈等能力构建，实现故障自动化恢复。丰富和构建故障恢复原子能力，通过自动化编排故障处理流程，逐步实现故障轻量化、智能化处理。

演练服务

演练服务是基于混沌工程理论主动干预提前识别风险，围绕演练生命周期管理提供平台能力。重点构建以下能力：

IT化管理

- 实现演练计划、演练方案，演练报告的可管理、可重用、可追溯等IT化能力

最小化爆炸半径

- 通过接口级以及会话级故障模式，控制演练最小化爆炸半径，实现常态化的演练

混沌工程

- 建立稳态指标系统，实现全业务的故障场景自动化执行、监控指标采集、测试与度量，快速评估服务可靠性质量属性，识别可靠性风险

集成调用链

- 支持基于规则的链路自动化故障注入能力。



➤ SaaS 行业展望

沈剑 / 广联数科 CTO

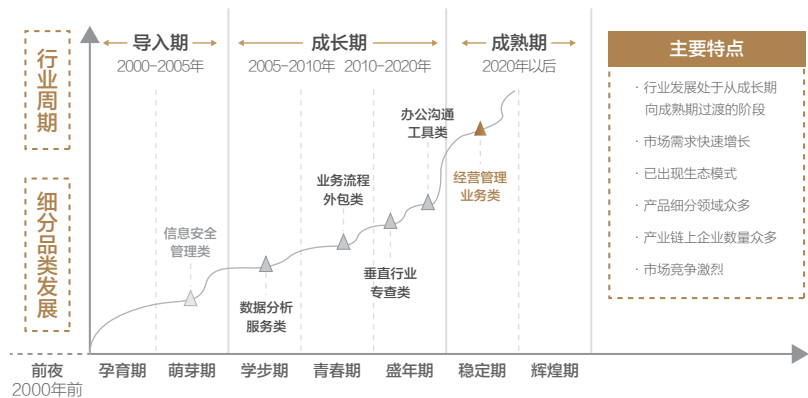
非常荣幸有机会借创原会这个优秀的平台来分享一下我的新书《SaaS创业方法论》的一些重要观点，以及我对SaaS行业2023年的展望。SaaS这个赛道即宽又广，涉及的内容和领域很多，深坑浅坑不断。不过就当前资本市场对SaaS的较低认同度，且中国仍未出现类似美国Salesforce这样的巨头就可以看出SaaS仍有机会！SaaS创业之路仍是一条充满希望的路。

因为当下SaaS和SaaS+X的延伸变化越来越多，所以让我们先回顾一下SaaS是什么？SaaS是Software as a Service的缩写，中文直译就是软件即服务，指通过互联网直接提供软件服务。具体来说，客户可通过互联网向SaaS软件供应商购买和使用软件服务，按订购的软件服务功能的多少、使用软件服务的时长或者使用软件的账户数等向厂商支付费用。这里有几个关

键词“软件、互联网、按需订购和服务”。SaaS领域很宽，分类标准没有完全统一，书中是按提效和增收的思路分为了工具类和商业类。工具类更聚焦在提高管理效率，降低运营成本，而商业类则更着重于为客户创造新的营收项目。当然，我们还可以从横向的业务垂直功能SaaS和纵向的行业垂直解决方案SaaS来进行分类，下图就是来自MobTech机构的SaaS分类图。



接着说说SaaS的发展史，如下图所示，从咨询机构的视角看，国内的SaaS自2000年萌芽开始，目前应该已经逐步迈入成熟期了。但无论是开发SaaS的公司还是使用过SaaS系统的客户可能大多不认同这一观点，相信不少企业都认为SaaS软件服务很多都有些鸡肋的感觉，只要有条件还是自己定制开发一套更好，SaaS的续费率一直提不起来。



回想SaaS软件即服务概念的提出时，从买断软件到租用软件，从此摆脱部署、升级和运维，还能够按需付费，大幅降低使用成本，作为一个软件从业人员当时听到这些真的异常兴奋。然而，虽有SalesForce一骑绝尘的开路，现实却举步维艰。一方面，羸弱的商业信任基础，导致大家不敢把企业数据放到SaaS软件中。就像我们现在服务的4S店，他们的客户电话号码其实在触达用户层面的价值已经很小了（客户更多以微信为沟通工具），却仍然坚持要以独立部署的方式保护起来。另一方面，中小企业人治成分高，管理风格差异很大，导致通用管理模式的提炼难度很大，工具类SaaS的功能边界难于界定，做着做着要么变成定制，要么出现很多冲突性功能，导致软件使用复杂度越来越高，体验越来越差。在这个点上，我非常认同我的新书推荐人前淘宝技术部负责人曾宪杰的观点，他认为大部分行业根据规模可以把企业分成头部、腰部、长尾三类，其中长尾公司会更倾向于选择SaaS服务，但这些公司综合要求高，特别是一致性差，导致SaaS创业死亡率居高不下。

接下来，我分享一些书中的重要观点。

观点 1: SaaS 本质是一种商业模式的变化，它是应对 VUCA 时代的必然趋势。

VUCA是Volatility (易变性)、Uncertainty (不确定性)、Complexity (复杂性)和Ambiguity (模糊性)首字母缩写的组合。对于当下时代的这几个特征大家可谓感同身受。SaaS因其特有的按需订购服务的模式，对客户来说拥有成本更低，免维护，服务使用更灵活，服务迭代更敏捷，更符合这个不确定的、复杂易变时代的特点，是中小企业的数字化的必由之路。

观点 2: 从上云, 到云上服务, 再到云原生, 云计算的发展史见证 SaaS 模式的趋势必然。

其实云计算的发展史见证了SaaS模式的趋势必然。早期, 大家都谈上云的问题, 那时还只是租用云主机, 感觉上是把现实的物理主机搬到云上。接着, 主机的概念开始淡化, 更多的是租用云上的软件服务。以华为云为例, 从租用虚拟机开始, 到租用网站、数据库、数据仓库、AI模型等多种服务。现在开始走向云原生, 不仅仅是业务交付全面上云了, 开发环节也开始全面上云了, 现在大家都开始用全云端开发套件了, SaaS化正变得越来越彻底。

观点 3: 在陪伴式服务中构建差异化竞争力, 中国式 SaaS 不得不做的加法。

大部分中国式SaaS都不得不做“陪伴式服务”这个加法, 而且还得做出差异化的竞争力, 这主要是两方面的原因造成的。一方面, SaaS公司如果没有融资不断撑着, 单靠订阅费前期成本回收周期太长, 现金流会变得极度紧张, 所以大家往往要通过提供差异化服务来寻找新的盈利点。这时常规的陪伴式服务除了软件的基础配置和使用外, 还会包括咨询服务或者是资源上的配套服务。另一方面, 对于很多垂直行业类SaaS来说, 客户更看重使用后是否能够快速看到业绩价值, 如果只是工具效率上的提升、对业绩影响不明显或者业绩提升很慢, 客户都不太愿意续费。营销行业私域运营里的SCRM就是个典型, 目前SCRM软件有好几百个, 卖价已经非常之卷, 但一次营销私域活动要想真的有明显效果, 运营策略、内容文案、软件工具、产品选择缺一不可, 不是单纯软件工具就能够决定效果的。但当下的用户只想为效果付费, 这导致中国式SaaS不得不做更多的加法, 提供除软件工具本身以外的陪跑服务, 帮助用户达成最终目标, SaaS公司客户成功部的建立就是这个逻辑下的结果。

观点 4: SaaS 产品设计特别容易出现的 3 个问题。

除了SaaS企业的商业模式设计以外, SaaS产品设计可能是SaaS企业的最大难点, 通常我们会出现如下这类的问题。

丨 产品设计脱离实际的场景。

产品经理对产品在一二线使用的场景了解不足, 对场景中使用软件的相关环境和制约条件了解不足, 对客户公司的管理体系和执行状况了解不足, 导致产品的功能设计脱离实际的场景。实战中, 很多SaaS初创企业, 不是产品经理下沉到一线的时间不够, 就是一线调研和落地缺少方法论。

丨 缺乏对竞品的系统分析和跟踪

很多创业公司因为创始人的原因, 往往对自己创新的逻辑或者方法过于自信, 以为自己发现了新大陆, 忽视对竞品的系统分析和跟踪, 只是从理论角度或者戴着有色眼镜去给出和竞品的差异化, 而不实际从真实应用场景考察这种差异化在客户层面的存在必要性和竞争力。甚至将竞品之争更多的归咎到客户关系的深浅上, 错失构建核心竞争力的机会。

丨 碎片化时代, 用户愿意投入的学习成本越来越低, 不要盲目做功能的加法

创业公司在找不到突破性创新点的时候, 慌乱之下容易盲目堆砌大量软件功能。或者是从竞品中照搬, 人家有我也要; 或者是领导的思路创新, 提出一个新的解决方案; 或者是在和客户发散性聊天时客户提及的天马行空的想法, 总之做加法, 做尝试是创业SaaS公司很容易涌出的冲动。其实, 碎片化时代, 客户愿意投入的学习成本越来越低, 已经没有用户愿意看用户手册来使用软件了, 你做的越复杂, 功能越多, 越用不起来。你投入越多, 给自己挖坑越深。

观点 5: 高度重视超级用户, 不要害怕为超级用户做些定制。

超级用户对SaaS公司来说意义重大, 但很多创业阶段的SaaS公司都因为担心进入定制化陷阱而忽略了对超级用户的维系。通常情况下, 超级用户往往是有前瞻性、尝试意愿强烈、有较强容错性的企业, 服务好超级用户, 甚至为其进行一些定制是以最小代价, 最快速度对行业实现穿透性了解的重要途径。然而现实中, 很多创业SaaS企业, 在还没有对行业 Know-How有深度认知的情况下, 急于抽象出通用框架或者模式, 急于推广铺开要业绩, 忽略了对超级用户的深耕细作, 导致行业痛点把握不深, 功能设计不接地气。

观点6: TOB企业利用新媒体矩阵宣传难点在内容, 不是陷入俗套没人爱看, 就是内容太过高深曲高和寡。

当下, 很多ToB企业利用新媒体矩阵做营销宣传的时候, 最大的问题往往在内容上。这些内容不是陷入俗套没人爱看, 就是内容显得过于高深而曲高和寡。SaaS公司其实还是需要认真的进行包装, 内容层面要适应当下新媒体发展的趋势。SaaS企业要用好创始人品牌, 用好视频号、抖音、知乎等新媒体, 在内容上多下功夫。

最后, 展望一下2023年中国SaaS领域的发展。

观点 1: 中国的企业级 SaaS 将更聚焦于帮助客户增收。

在当下缓慢复苏的经济环境下, 企业对SaaS的关键需求依次是“增收>降本>合规>提效”。虽然有数字化转型的大旗帜, 在企业压缩预算的背景下, 仍然只有增收最能吸引客户持续的续费购买SaaS服务。可以预见, 在增收需求的驱动下, 一方面陪伴式服务会做的越来越深、越来越细; 另一方面, 更多的SaaS软件商会提出按效果付费, 进一步降低门槛, 突出增收效果。

观点 2: 行业垂直类 SaaS 将因为 AI 大模型技术的突破而被重构。

伴随着AI大模型技术的突破, GPT模型对垂直行业工作效率的大幅提升是一个确定性很高的趋势。未来对于包含了复杂业务过程和流程自动化的SaaS, 如何将AI无缝融合进去会成为SaaS未来的壁垒。

观点 3: 摒弃做大而全的行业 SaaS, 精耕细作单一行业单场景 SaaS 的小而美成为趋势。

国内SaaS之所以被广泛质疑综合壁垒高, 原因在于厂商们过早地将注意力放在全场景, 一体化的产品搭建或产品矩阵上了, 太多人去做大而全的行业SaaS了。而单场景、轻量级SaaS部署成本很低, 客户的决策链条和决策时间也能尽可能缩短, 这对初创的SaaS公司最看重的现金流、人效和续费率来说都至关重要。显然, 小而美将成为趋势。



金融数据仓库十大发展趋势

来源: 金融信息化研究所

通过对百余家金融机构调研提炼出金融业对数据仓库技术的七大关注热度, 如下图所示。在七大关注热点中, 六个都是与“融合”相关。除此之外, 金融机构还关注“普惠”, 比如普惠数据治理、普惠数据仓库周边工具。目前业界有三个比较前沿的方向: 数据网格(Data Mesh)、数据编织(Data Fabric)及现代数据栈(Modern Data Stack)。综合调研和研究, 最终提出金融数据仓库的十大发展趋势。



T+0分析

在一些金融业务场景中，数据分析需求正从离线分析向实时分析转变。离线分析也称“T+1”分析，可以告诉用户过去 1 天或者几小时前的情况，主要支撑实时性不高的报表分析类应用。实时分析也称“T+0”分析，告诉客户目前正在发生什么，典型案例应用场景包括实时风控、实时营销、实时授信、实时运营（实时大屏）、实时运维等。另外，“T+0”分析也给数据仓库带来新的技术挑战：基于事件的处理需要实时高并发入库、入库即分析、流批一体、高性能同时要保证数据一致性等。



湖仓一体

金融业务产生不同类型的源数据，单一的结构化数据处理能力不能满足业务场景的需求。金融机构探索湖仓一体架构，将数据湖和数据仓库有机结合，充分融合数据仓库的高性能与数据湖的低成本，实现冷热数据分级、价值密度分级，解决海量数据中结构化、半结构化及非结构化的数据处理多样性，实现“1+1>2”的效果。湖仓一体关键技术包括：统一访问接口、统一元数据管理、统一存储格式以及计算引擎互通等。



数智融合

金融机构经过电子化、信息化、数字化这几个发展阶段，当前正在向智能化转型。对金融机构而言，数智化转型已经成为关键战略，部分机构已经在智能风控、小微信贷、智能投顾、智能客服等业务场景已有具体应用。但仅依靠 SQL 的数据分析难以满足这些业务需求，需要融合数据平台和 AI 平台。数智融合的关键是能力互补，将数据仓库数据管理能力与 ML 流程生命周期管理结合。目前结合主要包括两种：一是数据仓库将关系型的数据开放给 AI，并作为 AI 流程中数据准备、特征工程等强数据处理负载的分析引擎。二是非结构化数据（图像、视频、语音、文字）处理和模型训练由 AI 平台承载，训练生成的模型可直接部署在数据仓库中，由数据仓库来实现推理，并可以直接与数仓中关系型数据关联分析。



数据共享

数据已成为重要的生产要素，金融数据有序流通与共享开放是释放数据潜能的关键环节。当前各金融机构的数据仓库还处于“PC 时代”，每个数据仓库里的数据都是独立个体，不同机构数据仓库的数据实现安全合规的互联互通还存在困难。数据共享有助于发挥数据的要素价值，如何保证数据共享的安全性、隐私性，以及利益的公平分配等问题至关重要。从技术角度，金融数据仓库需要具备数据共享能力可开放能力，做到细粒度权限控制、加密及隐私计算、安全合规及风险审计。



存算分离

目前数据仓库的架构正向云原生演进，其典型技术特征是存算分离。这种新架构可以给用户带来极致的弹性，同时降低成本和提高资源利用率。部署在私有云上的金融数据仓库，使用存算分离有如下优势：一是有别于传统的节点扩容，用户可以根据业务需要，灵活的单独扩展算力或存储，扩缩容更加便捷；二是多集群共享一份数据，避免数据冗余，不但节省存储空间，同时减少数据搬运带来的网络开销。同一份数据支持多样式分析，保障数据一致性。

从技术角度讲，存算分离面临两大挑战：一是数据仓库的存、算、管是深度绑定的，需要将这三者解耦，在计算层去除计算元数据、存储元数据信息，才能做到无状态。二是存算分离将数据和计算拉远，可能带来性能下降，需要利用缓存、谓词下推及数据预取等技术降低查询延时。



高维分析

金融机构不仅有表格数据，还有图数据、空间数据、时序数据等。如果每种数据都采用独立的分析系统，不仅使用不便，也会导致数据孤岛问题。高维分析尝试构建一套系统，能够实现不同类型数据的融合分析。为了实现这一目标，高维分析需要解决多模存储和多模计算的问题。



HTAP

随着金融创新业务的蓬勃发展，越来越多的数据应用构建在数据仓库之上，有些应用需要对数据进行并发的增删改查。在此情况下，TP 数据库需要考虑如何和金融数据仓库进行更原生的整合，从而更好的支撑这些数据应用的开发和运行。HTAP 的愿景是构建一套系统，既支持 TP 又支持 AP 能力，同时达到降低成本、减少系统运维和 ETL 开销的目的。金融业对 TP 和 AP 的查询性能和资源隔离要求非常高，给 HTAP 技术落地带来了很大的挑战。



数据网格 (Data Mesh)

数据湖和数据仓库将所有数据集中存储，统一由数据中台部门进行治理。但随着金融机构接入的数据源越来越多，数据治理也变得愈加复杂，数据中台部门压力越来越大。为了缓解数据中台部门的压力，数据网格 (Data Mesh) 倡导金融业务部门参与到数据治理中，这样业务部门就可以拥有相关数据的所有权，能够更敏捷、更灵活地开展新业务。当需要整合治理多个业务部门的数据时，可以利用“联邦数据治理”技术和“数据即产品”思维来实现。同时，建议金融机构可以适当培养员工“数据即产品”的思维（即：利用开发产品的思维开发数据），提升员工在做数据治理时的工作效率。



数据编织 (Data Fabric)

数据编织 (Data Fabric) 和数据网格 (Data Mesh) 所要解决的问题是一致的，并与 Data Mesh 类似，Data Fabric 也是让金融业务部门拥有一部分数据的所有权。在整合治理多个业务部门的数据时，Data Fabric 采用逻辑的方式把数据连接起来，统一元数据层，提供全局数据目录供不同部门查找所需数据，利用智能数据技术 (Data Fabric 的核心理念) 节省人力。随着智能技术的发展，越来越多的数据治理任务可以自动化或半自动化，最终实现普惠数据治理的愿景。



现代数据栈 (Modern Data Stack)

金融机构探索利用现代数据栈 (Modern Data Stack) 从重构数据栈的角度来发挥数据仓库的最大价值。在金融机构的数据栈中，数据仓库是一个核心模块，周边还有其它模块负责 ETL、BI、数据质量及数据安全等。通过降低周边工具的使用门槛，推动“人人用数”目标的实现。比如在 Modern ETL 方面，有如下三方面探索：一是 ETL → ELT，ELT 是指先做 EL 把数据接入到数据仓库，再用数据仓库做 T，优势是 EL 可以无代码完成，只需要在数据仓库写 SQL 就可以解决 T 的问题；二是数据分析师 → 分析工程师，赋能数据分析师成为分析工程师，提供简化 DevOps 的工具，让数据分析师不需要依赖数据工程师，可以端到端完成数据分析任务；三是 Reverse ETL，利用无代码工具自动将数据分析结果从数据仓库复制到业务系统 (ERP、CRM 等)，赋能业务发展。

总体来看，可以用融合和普惠来概括金融数据仓库十大发展趋势。其中六大趋势 (T+0、湖仓一体、数智融合、存算分离、高维分析、HTAP) 都与融合有关，四大趋势 (Data Mesh、Data Fabric、现代数据栈、数据共享) 都与普惠有关。此外，通过调研我们发现国内很多金融机构在接下来 1-3 年在融合方面比较有清晰的规划，但是在普惠方面还准备不足，建议金融机构提高关注度，实现数据价值最大化。

数智融合时代企业 业务的提效降本之路

畅聊云原生第八期，由创原会与福佑卡车联合举办，活动以“沉浸其境，共享数智硬核盛宴”为主题，邀请了来自福佑卡车、中国信通院、华为云、新浪、陌陌、北汽福田、G7易流、货拉拉、高等教育出版社、软通动力、环信科技、探真科技、北方工业大学的技术专家及畅销书《人工智能产品经理》作者等20位云原生先锋齐聚一堂，围绕数智融合、云原生数据库、AIGC、代码大模型等技术话题进行了经验分享、问题探讨和未来畅享。

观点一： 数智融合将 赋予公路货运新 动能，为业务提 效降本



陈冠岭

福佑卡车技术合伙人

中国拥有世界上最大的公路运输市场，其中整车运输市场占比约60%，由于运力结构高度分散，市场发展面临着一系列挑战，例如：货主端找车低效、运价不透明、过程不可控、结算不规范，运力端担心货源真假难辨、空驶率高、缺乏标准化、收款缺乏保障，以上种种问题，导致物流运输效率低、成本长期居高。

福佑卡车专注于整车运输的科技物流平台建设，坚持以技术为本，用数智赋能，通过智能定价、智能分单、智能服务帮货主及卡车司机降低信息获取成本、提高车辆运行效率。

■ **智能定价系统：**通过云原生大数据技术破解人工议价，对整车数据进行大量积累、清洗与分析，实时计算整车运价，与市场价吻合度超 95%。

■ **智能分单系统：**基于机器学习和算法，智能分析运单与司机信息，在算法中进行预测，将运单与运力进行最优匹配，最大程度提升车辆运行效率；

■ **智能服务系统：**结合大量真实的运输数据，通过运力风控、智能预警和智能客服，让拥有不同承运经验的运力输出，稳定、统一的标准化服务

福佑卡车技术合伙人陈冠岭博士表示，我们采用云原生技术，对业务进行了一系列升级，福佑卡车实现了用“数据”来驱动一切洞察、决策、行动，将货运准时率提高到95.2%，将车辆空驶率由行业平均值45%大幅降至6.6%、车辆空置率由行业平均值40%降至24.8%，帮助提升司机收入33%。

观点二： 云原生数据库 正走向Serverless、 Regionless、 Modeless



彭立勋

华为云数据库软件
总工程师

从星星之火到燎原之势，在数据库领域，云原生数据库蔚然成风，风起云涌之后的云原生数据库又将怎样演进呢？

华为云数据库软件总工程师彭立勋表示，2025年可能有73%的数据库都是云原生数据库，客户最关注的几个问题，一是资源调度，二是数据访问，三是使用体验，云原生数据库也将随之发生三大转变：以资源为中心到以应用为中心，以地域为中心到以流量为中心，从以负载为中心到以数据为中心，对应到架构的演进，就是Serverless、Regionless和Modeless。

Serverless技术将助力数据库从以资源为中心到以应用为中心的转变。Serverless架构下，用户无需关注数据库运行的背后具体需要多少资源，在流量洪峰时可以极速扩容。华为云云原生数据库可以实现资源的极致弹性，将资源弹性调度速度从分钟级缩短至秒级，不仅业务无感知，而且能够节省大量规划资源、调整资源的运维工作；

Regionless技术可以解决数据库向以流量为中心转变中的挑战。过去，用户需要将数据的处理链路纳入考虑范围，尤其是出海的业务，用户由于数据跨AZ、跨Region而面临时延高、体验差、投入成本大等问题。基于Regionless架构，华为云云原生数据库支持数据全域可用，业务可以在任意地方进行接入和访问，同时带来跨地域的高可用；

Modeless技术将加速数据库从以负载为中心到以数据为中心的转变。Modeless可以提供统一智能的业务入口，自动处理各种类型的数据和工作负载，提升易用性和效率，解决数据库选型难的问题。华为云云原生数据库不仅采用NDPQ技术让数据就近执行，将数据查询性能发挥到极致，还采用HTAP技术，将数据同步时延降低到毫秒级，极大提升数据处理时效性。

我们相信，在以上理念的牵引下，云原生数据库将迎来新一轮变革式的发展与创新，让客户能够像使用水和电一样，更便捷地使用云原生数据库。

观点三： AI眼镜将成为 未来十年颠覆 性的AI产品



张竞宇

《人工智能产品经理》作者

未来人工智能将像水和电一样，在人们生活中无处不在，现在AI技术正处在高速发展时期，未来充满了不确定性，新的技术发展会不断地颠覆现在的产品和行业。畅销书《人工智能产品经理》作者张竞宇认为：“AI眼镜将是未来十年颠覆性的AI产品，是实现增强智能的最佳方案，它的出现却可以拉近人与人之间的距离，提升人与人的沟通效率。”

从数据角度来看，AI眼镜是开启“个人全数据采集”时代的钥匙，它不仅具备个人体征实时诊断的能力，通过对体征数据的实时监测分析，可以为个体提供强大的私人医疗监测服务，同时，随着眼动交互、语音交互、手势交互技术逐步发展成熟，AI眼镜还能生活各方面，比如订餐、出行、网购等方面实现颠覆性的人机交互体验。

从算法角度来看，AI眼镜基于所采集的更全面的个人数据，能训练出最适合自己的、能帮助我们克服自身缺陷的算法。

从算力角度来看，通过将拥有强大算力的微型芯片植入皮肤下，并与AI眼镜互联实现计算和交互，不仅可以采集各种体征信息，还能与人脑进行通讯，真正实现增强智能。

就AI是否会取代人类的问题，张竞宇认为，人工智能的处理逻辑都是基于数学模型的、理性的，而人脑做出的理解、推理和决策在绝大多数不是仅依靠理性推理，还要依靠感性和理性的共同帮助，所以从这个角度讲，人工智能无法替代人类。

观点四： 代码大模型解放 了程序员的双手， 提升企业应用研发 效率，但并不会 完全取代程序员



申博

华为云 PaaS 创新 Lab
技术专家

随着数字化转型日益成为大势所趋，企业数字化转型意味着大量的编码工作，程序员肩负的重担也愈发沉重，代码大模型将AI融入编程软件，通过自然语言描述生成函数级代码，将极大提升代码的生产效率。华为云PaaS创新Lab技术专家申博认为，这仅仅是开始，未来AI在软件开发中的应用不会仅限于代码编写，目前已经开始覆盖代码翻译与迁移、代码总结与解释、代码测试与调试等多个维度，并且将进一步延伸到需求分析、模块设计、安全检测、系统运维等更多环节之中，类似华为云CodeArts Snap的智能研发助手，会有越来越多基于代码大模型的AI工具成为提升软件开发和交付效率的利器。

然而，AI工具仍只能被定位为开发者的助手，只能取代程序员的一部分工作内容，目前的工具擅长重复、通用、明确的任务，而软件开发是充满创新的智力密集型活动，开发者要学会与AI协同，让工具去做工具擅长的部分，而将所节省的精力投入到更高层次、更高价值的创新活动中去，并提升自己的思维层次，增强问题定义和系统设计的能力，例如：如何恰当而精确地描述需求、如何针对业务特点定义接口、如何分解问题并审视AI所生成的结果等。

谈到未来的发展，申博认为，要构建真正实用的智能系统，我们还应当更加关注数理逻辑和知识推理，不但要设计和训练新的模型，还需要研究如何将统计方法与规则算法融合，并在人机交互过程中根据反馈实时在线学习，最终构建持续演进的健康生态。

重识云原生系列四： 硬菜软嚼的云网络（下）

黄俊 / 招商证券云原生转型项目调研负责人

楔子

在上一篇我们讲到伴生于云计算的云网络技术飞速发展，一方面是云网络基础技术发展迅猛，特别是SDN技术的发展与成熟，使得网络架构得到了极大简化，并使传统硬件网络设备开始变得更加开放灵活、软件可定义。同时，SDN技术在云计算领域的深度运用与发展，使得网络虚拟化可以不再基于物理网络设备实现，如此便大大扩展了网络虚拟化的“边界”，也由此孕育出了许多面向云租户的云上网产品与服务，极大地丰富了企业用户上云体验、加速了企业上云进程。

SDN 催化的网络虚拟技术狂飙

接下来，我们分别从网络设备虚拟化、链路虚拟化和虚拟网络三个层面来看看网络虚拟化技术的高速发展。

1.1 网络设备虚拟化

网络设备虚拟化技术是云网络技术发展最为迅猛且影响最大的领域，其主要包括网卡虚拟化（NIC Virtualization）与常规网络设备虚拟化两大方向。

一、网卡虚拟化

网卡虚拟化技术演进路线与我们在

《重识云原生第三篇：软饭硬吃的计算》一文中对于I/O虚拟化的演进路线基本一致，也是经历了纯软虚拟化（即Qemu、VMware Workstation虚拟网卡实现方案）、半虚拟化（Virtio、Vhost-net、Vhost-user系列方案），再到硬件直通方案（SR-IOV），最后到硬件卸载（vPDA、DPU等）。

网卡虚拟化的产品形态即是公有云常见的虚拟网卡，其主要通过软件控制各个虚拟机共享同一块物理网卡实现，虚拟网卡可以有单独的MAC

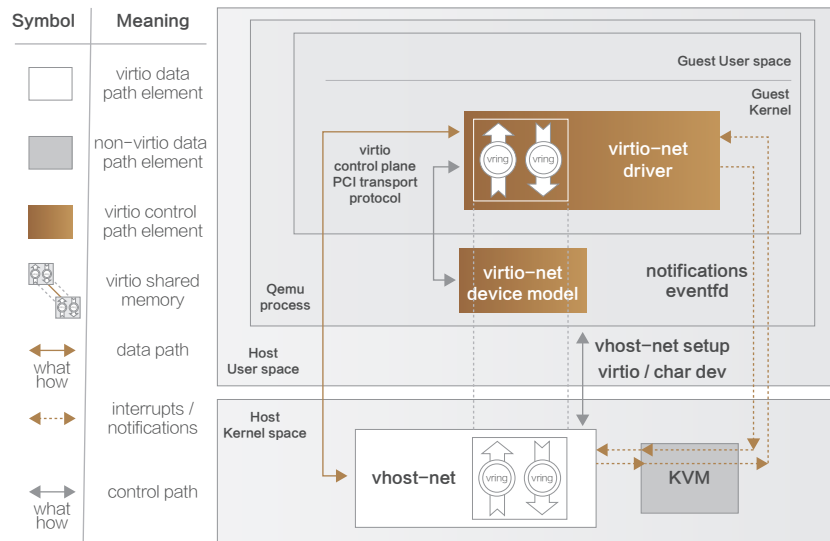
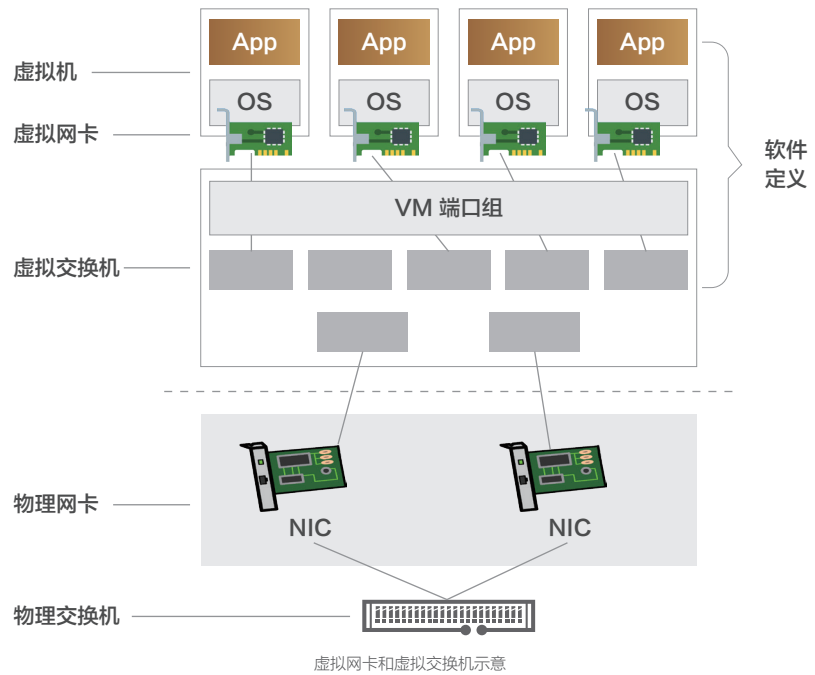
地址、IP 地址，早期结构如下图所示。所有虚拟机的虚拟网卡通过虚拟交换机以及物理网卡连接至物理交换机。虚拟交换机负责将虚拟机上的数据报文从物理网口转发出去。根据需要，虚拟交换机还可以支持安全控制等功能。

虚拟网卡包括e1000、Virtio等实现技术。Virtio是目前最为通用的技术框架，其提供了虚拟机和物理服务器数据交换的通用方案，得到了大多数Hypervisor的支持，成为事实上的标准。

Virtio是一种半虚拟化的解决方案，在半虚拟化方案中，Guest OS知道自己自己是虚拟机，通过前端驱动和后端模拟设备互相配合实现IO虚拟化。这种方式与全虚拟化相比，可以大幅度提高虚拟机的IO性能。

最初的Virtio通信机制中，Guest与用户空间的Hypervisor通信，会有多次数据拷贝和CPU特权级的切换。Guest发包给外部网络，需要切换到内核态的KVM，然后KVM通知用户空间的QEMU处理Guest网络请求。很显然，这种通信方式效率并不高。很快，Virtio技术演进中便出现了内核态卸载的方案Vhost-net。

Vhost-net是Virtio的一种后端实现方案，与之相匹配的是一套新的Vhost协议。Vhost协议允许VMM将Virtio的数据面Offload到另一个组件上，而这个组件正是Vhost-net。Vhost-net在OS内核中实现，其与Guest直接通信，旁路了KVM和QEMU。QEMU和Vhost-net使用ioctl来交换Vhost消息，并且用eventfd来实现前后端的事件通知。当Vhost-net内核驱动加载后，它会暴露一个字符设备标识在/dev/Vhost-net，而QEMU会打开并初始化这个字符设备，并调用ioctl来与Vhost-net进行控制面通信，其内容包含Virtio的特性协商、将虚拟机内存映射传递给Vhost-net等。对比最原始的Virtio网络实现，控制平面在原有执行基础上转变为Vhost协议定义的ioctl操作（对于前端而言仍是通过PCI传输层协议暴露的接口），基于共享内存实现的Vring数据传递转变为Virtio-net与Vhost-net共享；数据平面的另一方转变则为Vhost-net，前后端通知方式也转为基于eventfd来实现。

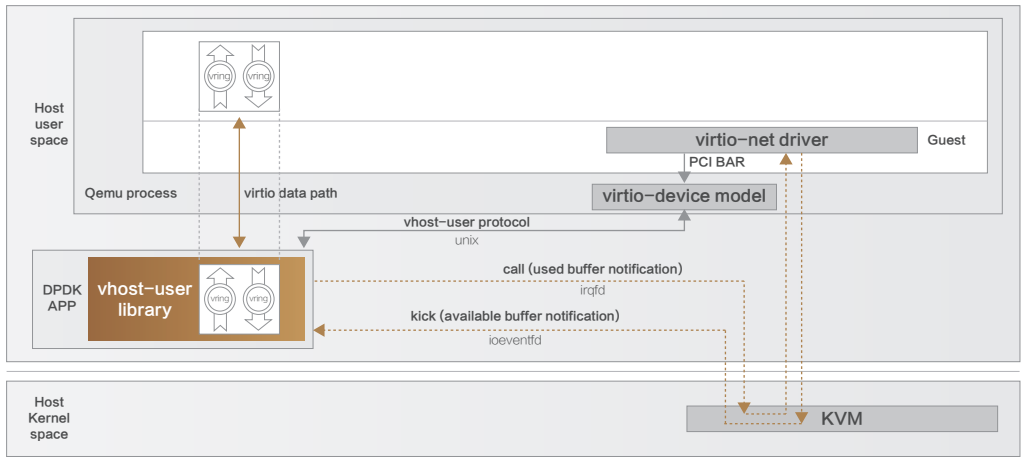


Vhost-net 方案示意图（引自 RedHat 社区博文）

在单次收发的过程中，数据通道减少了两次用户态与内核态的上下文切换过程，实现了数据通道在内核态的卸载，从而较大程度提升了数据传输效率。

但是，对于某些用户态进程间的通信，比如数据面的通信方案（例如Open vSwitch和与之类似的SDN解决方案），Guest仍需要和Host用户态的vSwitch进行数据交换，如果采用Vhost-net方案，Guest和Host之间还是存在多次上下文切换和数据拷贝，为了避免这种情况，业界就想出将Vhost-net从内核态移到用户态的方案，这就是Vhost-user方案的实现思路。

Symbol	Meaning
	virtio data path element
	non-virtio data path element
	virtio control path element
	virtio shared memory
	data path
	interrupts / notifications
	control path



基于 DPDK 的 Vhost-user 方案示意图 (引自 RedHat 社区博文)

基于Vhost协议，DPDK设计了一套新的用户态协议——Vhost-user。该协议将网络包处理（数据平面）卸载到用户态的DPDK应用中，控制平面仍然由QEMU来配置Vhost-user。除此之外，DPDK优化实现了处理器亲和性管理、NUMA调度、使用大页内存代替普通内存、采用无锁技术解决资源竞争问题、轮询模式驱动等技术，由此，大名鼎鼎的DPDK网络卸载方案蔚然成型。

纵如此，技术迭代演进是永无止境的，虽然上述方案对网络性能的提升非常显著，但总体而言，Virtio网络虚拟化方案一直都聚焦在软件层实现，在Host上因软件虚拟化而产生的额外开销始终无法避免。为了进一步提升云端网络服务的性能，RedHat的技术专家们提出可以将Virtio功能Offload到专用硬件，将与业务无关的任务Bypass操作系统和CPU，转而直接交给专用硬件执行，从而进一步提升网络性能，这便是vDPA硬件卸载方案的设计初衷，也是Virtio半硬件虚拟化方案的早期硬件方案实现思路。

该框架由Redhat提出，实现了Virtio数据平面的硬件卸载。控制平面仍然采用原来的控制平面协议，当控制信息被传递到硬件中，硬件完成数据平面的配置之后，数据通信过程由硬件设备SmartNIC（即智能网卡）完成，Guest虚拟机与网卡之间直接通信。中断信息也由网卡直接发送至Guest而不需要Host主机的干预。这种方式在性能方面能够接近SR-IOV硬件直通方案，同时能够继续使用Virtio标准接口，保持云服务的兼容性与灵活性。但是其控制面处理逻辑比较复杂

，通过OVS转发的流量首包依然需要由主机上的OVS转发平面处理，对应数据流的后续报文才能由硬件网卡直接转发，硬件实现难度较大。

不过，随着越来越多的硬件厂商开始原生支持Virtio协议，将网络虚拟化功能Offload到硬件上，把嵌入式CPU集成到SmartNIC（即智能网卡）中，网卡能处理所有网络数据，而嵌入式CPU负责控制路径的初始化并处理异常情况。这种具有硬件Offload能力的SmartNIC就是如今越来越火的DPU。

初代 SmartNIC		DPU
定位	通过从服务器的 CPU 上卸载网络或存储等工作负载，提升云端和私有数据中心中服务器网络 IO 性能	数据中心级数据处理单元，可作为数据中心的极小节点存在
主要特点	释放 CPU 开销；提升网络传输性能；可编程；任务加速；流量管理	包含数据平面和控制平面的双平面卸载与加速；涵盖智能网卡的功能；标准化、高性能、软件可编程的多核 CPU；丰富灵活可编程的加速引擎
社区生态	社区生态繁杂，各家厂商标准不统一，开发难度较高，项目可移植性差	拥有标准社区生态，部分有专用软件开发平台提供上层标准开发接口，如 NVIDIA 的 DOCA SDK，入门和开发难度低
应用场景	对指定业务做加速；存储、安全、数据压缩等	数据中心和云计算；网络安全；高性能计算及 AI；高性能通信；边缘计算；高性能存储；流媒体等
应用价值	处理专用业务；在数据中心的功能较为单一；被动，对其他设备有依赖性；	可单独成为独立的数据中心单元，拥有丰富的功能并可拓展，未来将成为数据中心的标配与 3U 之一（CPU、GPU、DPU）；主动，可成为计算节点、网卡、要素引擎等，可独立存在

表 1：智能网卡和 DPU 简单对比分析

二、网络设备虚拟化

硬件设备虚拟化主要有两个方向：在传统的基于x86 架构机器上安装特定操作系统来实现路由功能。

前者的早期典型产品是Mikrotik公司开发的RouterOS, 其基于Linux 内核开发, 可以安装在标准的x86 架构的机器上, 从而使普通Linux服务器也可以当成路由器使用。此类设备以其低廉的价格和不受硬件平台约束等特性, 占据了相当低端路由器市场。现如今, 国内外主流云厂商对于SDN控制面的实现思路大抵都是如此。

而后者主要是应第一代云计算技术的的市场需求催生, 在计算虚拟化之后, 数据中心通讯规模大幅增长, 传统路由器中的单张路由表已经不能满足需求, 因此催生了虚拟路由转发 (Virtual Routing and Forwarding, VRF) 技术, 即是将路由信息库 (Forwarding Information Base, FIB) 虚拟化成多个路由转发表。此技术诞生背景主要为增加大型通讯设备的端口利用率, 减少设备成本投入, 将一台物理设备虚拟化成多台虚拟设备, 每台虚拟设备仅维护自身的路由转发表即可, 比如思科的N7K 系列交换机可以虚拟化成多台VDC。所有VDC共享物理机箱的计算资源, 但又各自独立工作, 互不影响。此外, 为了便于维护、管理和控制, 将多台物理设备虚拟化成一台虚拟设备的融合

虚拟化技术也有一定市场, 比如H3C公司的IRF技术。

1.2链路虚拟化

链路虚拟化是日常使用最多的网络虚拟化技术之一, 增强了网络的可靠性与便利性。常见的链路虚拟化技术有链路聚合和隧道协议, 此部分内容在《第四篇-硬菜软嚼的网络(上)》篇已有详细阐述, 此处只简单提一下。

链路聚合 (Port Channel) 是最常见的二层虚拟化技术。链路聚合将多个物理端口捆绑在一起, 虚拟成为一个逻辑端口。当交换机检测到其中一个物理端口链路发生故障时, 就停止在此端口上发送报文, 根据负载分担策略在余下的物理链路中选择报文发送的端口。链路聚合可以增加链路带宽、实现链路层的高可用性。

隧道协议 (Tunneling Protocol) 指一种技术/协议的两个或多个子网穿过另一种技术/协议的网络实现互联。隧道协议将其他协议的数据帧或包重新封装然后通过隧道发送。新的帧头提供路由信息, 以便通过网络传递被封装的负载数据。隧道可以将数据流强制送到特定的地址, 并隐藏中间节点的网络地址, 还可根据需要, 提供对数据加密的功能。当前典型的隧道的协议便是VXLAN、GRE和IPsec。当今各大云厂商的云平台的网络实现方案莫不基于此。

1.3虚拟网络

虚拟网络 (Virtual Network) 是由虚拟链路组成的网络。虚拟网络节点之间的连接并不使用物理线缆连接, 而是依靠特定的虚拟化链路相连。典型的虚拟网络包括在数据中心使用较多的虚拟二层延伸网络、虚拟专用网络以及Overlay网络。

虚拟二层延伸网络 (Virtual L2 Extended Network) 其实也可以算是Overlay网络的早期解决方案。为满足虚拟机动态迁移需求, 传统的VPLS (MPLS L2VPN) 技术, 以及新兴的Cisco OTV、H3C EVI 技术, 都借助隧道的方式, 将二层数据报文封装在三层报文中, 跨越中间的三层网络, 来实现两地二层数据的互通。

而虚拟专用网 (VPN, 全称Virtual Private Network), 是一种应用非常久远, 常用于连接中、大型企业或团体与团体间的私人网络的通信方法。虚拟专用网通过公用的网络架构 (比如互联网) 来传送内联网的信息。利用已加密的隧道协议来达到保密、终端认证、信息准确性等安全效果。这种技术可以在不安全的网络上传送可靠的、安全的信息, 总体而言早于云计算技术在企业中落地。

而Overlay网络也已在上篇有详细阐述, 此处就不再复述了。

云网络产品的发展

云网络产品整体而言, 主要经过了三次大的演进, 从最初的经典云网络,

到私有网络 (Virtual Private Cloud), 再到连接范围更大的云联网。云网络

技术的发展也是企业数字化、全球化发展的一个缩影。

2.1 经典(基础)云网络

从2006年AWS推出s3和ec2云服务,开始为公有云用户提供云上服务开始,到2010年,阿里云正式推出Classic经典云网络方案,实现了对云计算用户的网络支持,再后续腾讯云、华为云也提供过类似的基础网络产品。

这些都是公有云早期的云产品,当时用户对云网络的主要需求是提供公网接入能力。经典云网络的产品特点是云上所有用户共享公共网络资源池,所有云服务器的内网IP地址都由云厂商统一分配,无法自定义网段划分、IP地址。

2.2 私有网络VPC

而从2011年左右开始,随着移动互联网的发展,越来越多企业选择上云,企业对云上网络安全隔离能力、互访能力、企业自建数据中心与云上网络互联并构建混合云的能力,以及在云上实现多地域部署后的多地域网络互联能力都提出了很多需求。此时,经典网络方案已经无法满足这些需求,AWS适时推出云上VPC产品服务(即私有专用网络服务,全称Virtual Private Cloud),VPC是企业用户在云上建立的一块逻辑隔离的网络空间——在VPC内,用户可以自由定义网段划分、自由分配网段内IP地

址、自定义路由策略。而后又陆续推出NAT网关、VPN网关、对等连接等云网络产品,极大地丰富了云网络的连接能力,基本实现了私有网络方案最初的设计目标。

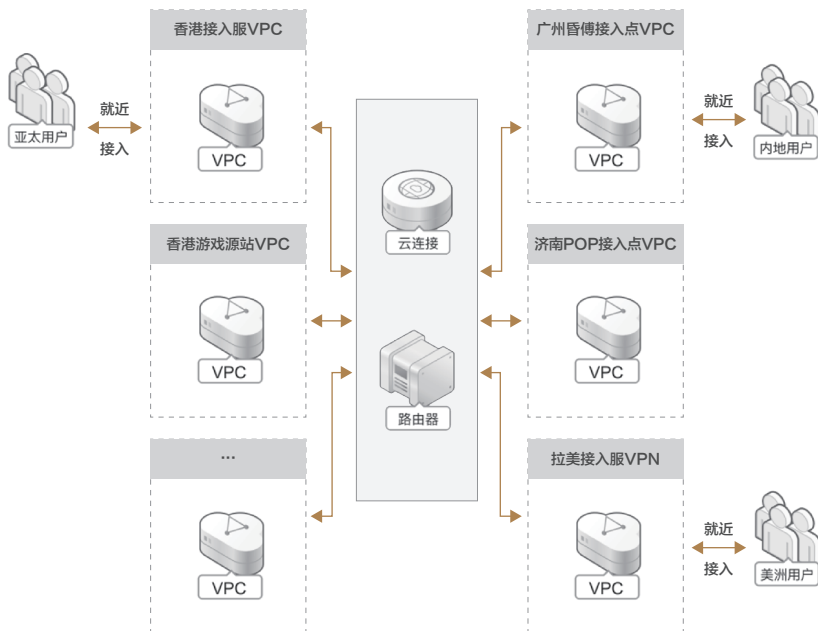
当前,以VPC为核心的云网络产品体系依然在不断发展壮大,又陆续诞生了负载均衡器、弹性网卡、弹性公网IP、云防火墙等配套类衍生网络产品。

2.3 云联网

而近几年,随着经济全球化高速发展,大数据与AI类应用异军突起,需要

云平台网络提供更广泛灵活的接入能力与分发能力,便催生了私网连接、云企业网、云连接器、云联网等各类场景的接入类云网产品,以此来满足企业全球化场景下的各类复杂互联互通需求。

与此同时,传统的CDN分发服务,经过与云联网的深度融合,也焕发了第二春,如今各大云厂商均有推出CDN全球内容分发服务与应用加速服务。同时SD-WAN接入服务的不断完善,也使得企业信息架构一体化的物理边际向着全球大踏步迈进。



华为云联网方案(摘自华为云官网)

云网络技术后续演进展望

云网络从基础核心技术的演进来看,是对网络编址灵活性和网络传输性能提升的持续优化过程:经典云网络和VPC网络在编址方案上,都使用了Overlay技术,在物理网络之上叠加

了一个支持多租户的虚拟网络层,从而有效支持了云上租户间的网络隔离安全性、租户内应用的互联互通灵活性诉求;而在网络性能提升方面,通过重构网络设备形态,在不丧失转

发控制灵活性基础上,来持续持续提升数据转发性能是为当今主流技术演进方向,以虚拟交换机为例,伴随着硬件服务器网络带宽从10G升级到25G,乃至到100G,vSwitch技术经历

了经典网络的Linux内核态数据交换、DPDK用户态数据交换、硬件直通、再到智能网卡硬件卸载(vPDA、DPU等)等演进历程。

而从产品层面看,云网产品经过经典网络,VPC网络,云联网三代产品的发展,云网络产品的能力范围得到了极大延展——从聚焦数据中心内部网络虚拟化,到聚焦数据中心之间的互联互通,再进一步通过sd-wan接入

能力把企业的网络接入能力进行虚拟化从而支持各类场景接入。云网络产品已不仅仅用于连接云内的计算与存储资源,云下接入范围也逐渐扩展到连接企业的总部/分支网络以及云外各类终端,以期最终使企业能在云上构建完整的IT信息服务生态体系。

而在云计算技术发展第三代云原生时代的当下,随着各类虚拟化技术的成熟,各类软件定义硬件的云能力

得以落地运用,伴随着容器技术的大规模商用,终会使业务研发与基础运维一体化协作的技术隔阂会彻底破除,业务应用的全栈可观测性诉求必将使业务服务监控下钻到基础网络层,让网络流量监测带上业务属性,从而使业务全链路监控真正具备生产力价值。而这又都将取决于容器网络与云网络技术的融合深度与应用广度,这一领域也必将成为今后各大云厂商决胜云原生时代的战略制高点!

参考资料

云网络产品和技术浅析

<https://zhuanlan.zhihu.com/p/351449927>

漫谈云计算网络(一):云计算网络技术介绍

<https://blog.csdn.net/qiansg123/article/details/80123088>

漫谈云计算网络(二):云计算网络的应用场景

<https://blog.csdn.net/qiansg123/article/details/80124473>

浅谈云网络的发展历史

<https://baijiahao.baidu.com/s?id=1621622261059186571&wfr=spider&for=pc>

计算机网络的形成与发展过程

https://www.sohu.com/a/540490960_121350545

云计算中的网络基础知识

https://blog.csdn.net/qq_46254436/article/details/104588055

云网络的十五年演进史简述

<https://zhuanlan.zhihu.com/p/307684414>

计算机网络发展历史

<https://zhuanlan.zhihu.com/p/150417958>

挺带劲! 1.2万字总结了网络技术的理论,温故知新

<https://zhuanlan.zhihu.com/p/371546795>

Virtio devices and drivers overview: The headjack and the phone

<https://www.redhat.com/zh/blog/virtio-devices-and-drivers-overview-headjack-and-phone>

A journey to the vhost-users realm

<https://www.redhat.com/zh/blog/journey-vhost-users-realm>





实战分享

B 站: 云原生架构下的 Flink 存算分离改造实践----- 69-77

美的楼宇科技: 基于云原生技术的成本优化实践----- 78-79

去哪儿: 以技术为“魂”，赋能城市旅游 ----- 80-82

云原生 Cloud Native



B 站: 云原生架构下的 Flink 存算分离改造实践

张陈毅、曹杰 / 哔哩哔哩资深开发工程师

1. 背景

在当前整个行业及公司内部降本增效的大背景下，b站内部也在积极推进实时与在线业务资源的整合，往云原生架构迁移，统一资源池与调度，提升资源利用效率。不过面临的现实问题就是，不同业务场景下，资源的规格诉求不尽相同。在线的业务资源池，由于在线业务的属性，一般只具备很强的计算能力而基本不带存储以及io能力。Flink虽然是一个计算引擎，但是由于其stateful的特性，在很多计算场景下，对存储和io其实有比较强的诉求，因此实时的资源池，同时具备很强的存算能力。两种资源池的整合，必然面临兼容性问题，考虑到大数据整体的存算分离发展趋势，我们尝试对Flink进行存算分离的改造，核心工作就是statebackend的远程化。

2. 痛点

Flink里面的StateBackend，是用来存储任务状态的。从用途上划分为OperatorStateBackend和KeyedStateBackend，OperatorStateBackend一般存的是一些与计算逻辑本身无关的数据，如Kafka的offset，体量比较小，不会受数据规模与计算逻辑影响，而KeyedStateBackend相反，存储的是与计算逻辑强绑定的状态数据，如agg/双流Join的中间结果，大小受着数据规模和计算逻辑影响，一般大流量，计算复杂的任务State会非常大。

B站的流计算任务有4000+个，95%的任务均是sql任务，其中50%的任务带状态，且有上百个任务状态大小超过500GB。目前内部的默认StateBackend使用的是RocksDBStateBackend，其在实现上支持了增量快照，可以减少大量重复State数据上传到文件系统，另外历史的Flink集群机器都配置了比较高性能的磁盘，可以支持TaskManager的本地保存大State到RocksDB中，当任务做Checkpoint时，再将增量的State文件上传至文件系统，这一现状及配置能支持所有的流计算任务，无论其KeyedState大小是0或者TB级别。现有的这一环境会有如下两个痛点：

| 磁盘整体利用率低

所有的Flink机器都配置了高性能的大磁盘，能支持超大State任务的健康运行，我们对State做了RetentionTime默认配置，有百分之八十的任务是百GB以内的小状态或者无Keyed State，这些任务TaskManager所运行的机器整体利用率明显偏低，如果所在机器没有大State存在，当前机器的高性能大磁盘存在较为明显的浪费。

I 大任务Rescale慢

超大State在做Rescale时, 任务会先将State数据从文件系统下载到本地并实例化成RocksDB, 然后TaskManager之间根据KeyGroup对数据做重分布操作, 本质是对RocksDB根据KeyGroupId做seek和BatchWrite操作, 当State在TB级别时, State数据的重分布成本是比较大的, 恢复时间需要半个小时左右, 用户体验会比较差, 且易增加引擎侧的值班成本。

3.RemoteStateBackend

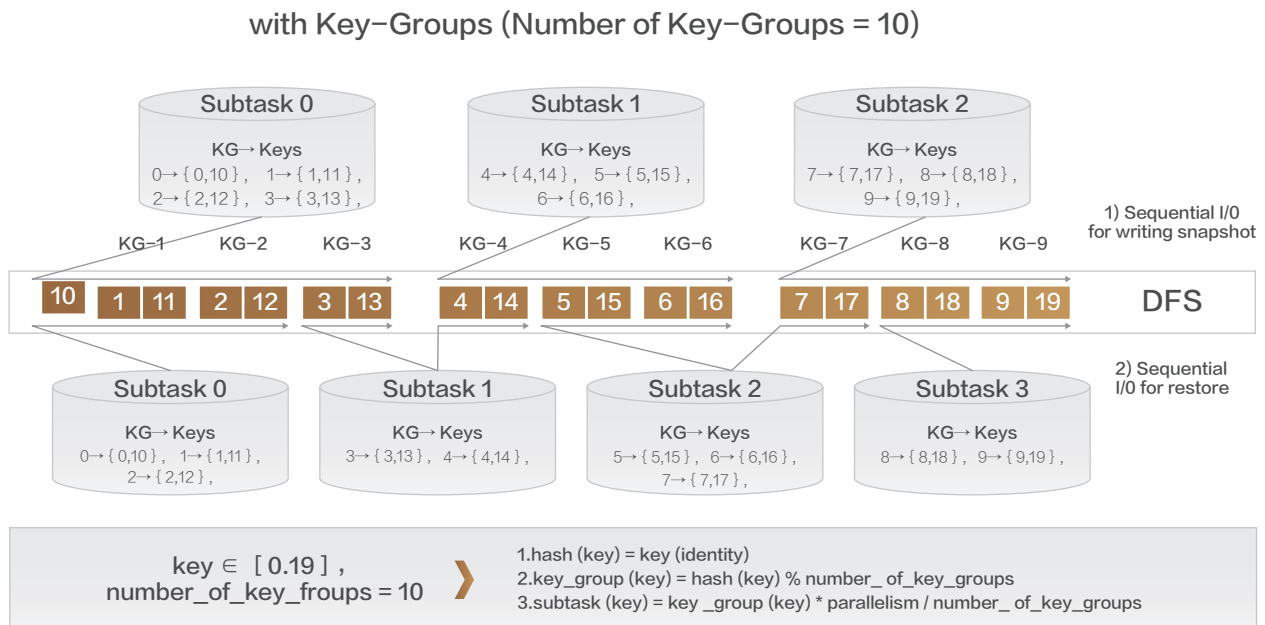
如需解决上面的痛点, 一个是需要将State数据能实时的存储在远程服务中, 减少Flink集群对磁盘的强依赖, 实现存算分离, 这一目的也正和云原生架构演进目标契合; 另一个是State数据可以以KeyGroup为单位来存储, 避免数据的重分布操作。经过与分布式存储团队沟通后, 其自研的Taishan (B站分布式KV存储[1]) 存储基本能满足我们的述求, Taishan存储是基于RocksDB和SparrowDB改造, 采用raft一致性协议保证多副本数据一致性而构建的高可靠、高可用、高性能、高拓展的存储系统。

Taishan存储提供了Java的put/get/del/scan等api, 且具备snapshot的功能, 在api层面和RocksDB的功能基本一致, 且能支持水平扩容和在线升级, 完全能满足Flink的需求, 经过多次沟通和功能支持后具备构建TaishanStateBackend条件。

I 3.1 State切换保障

在KeyedStateBackend中, 每条待计算的数据进入subtask中, 会根据数据的key通过"MathUtils.murmurHash(key's HashCode) % maxParallelism"计算出一个ID作为

KeyGroupId, 这样可以保证相同key的数据会在同一个subtask中计算, 不会导致在计算引擎内出现乱序的现象, maxParallelism则为最大并行度, 其值在任务启动之后就不会发生改变。每条数据根据KeyGroupId会归属到一个Key-Group分片中, Key-Group分片总数为maxParallelism, 每个分片是无法分裂的, 因为maxParallelism的值在重启前后均不允许被修改, 当任务做rescale时, 对State来说是将Key-Group下的数据做移动, 如下图所示。



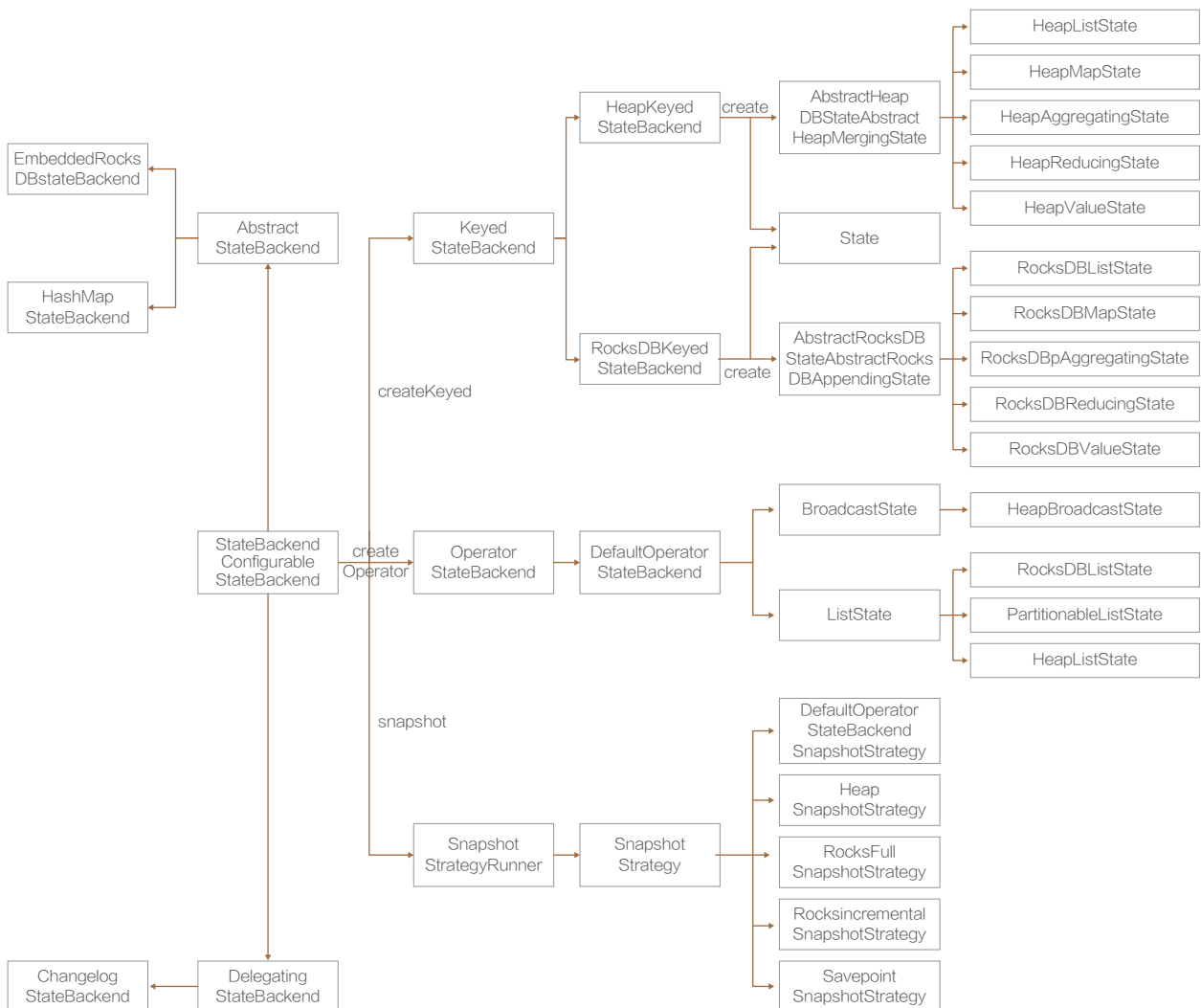
Taishan存储有shard分片概念,正好与Flink的KeyGroup类似, Taishan的shard具备分裂和合并的能力,我们仅需约定Flink的Taishan表关闭此功能即可。Flink还有另一个关键的点是Checkpoint, RocksDBStateBackend的增量Checkpoint是同步过程对RocksDB做snapshot, 异步过程将变动的SST文件上传至文件系统以便任务重启时恢复, Taishan存储提供了可在每个shard上做snapshot的create和restore能力, Flink可以在做Checkpoint时, 在每个subtask内按

照KeyGroupRange的start至end依次做snapshot, snapshotID与Flink的CheckpointID保持一致。

3.2 StateBackend拓扑设计

Keyed StateBackend承担核心的状态数据管理, 管理的State分为Keyed State和Priority Queue State。其中Keyed State有基础的InternalValueState和InternalMapState, 以及复杂的InternalListState、InternalAggregatingState、InternalReducingState和InternalFoldingState, 不同类型的State会应用于

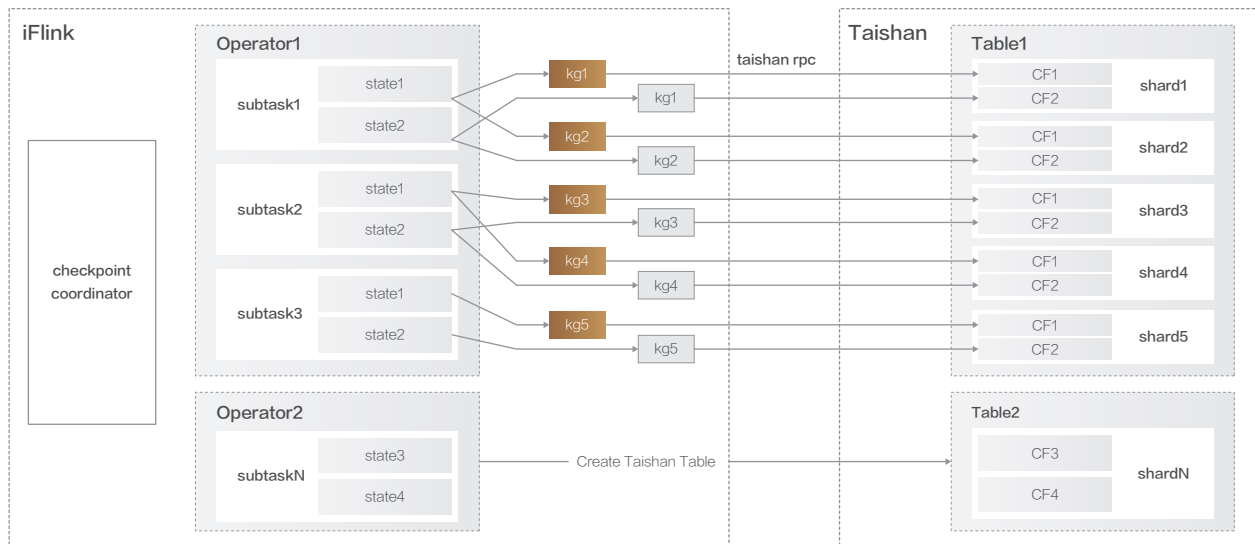
SQL不同场景的算子上; Priority Queue State则需由InternalPriorityQueue实现。Flink流计算的Checkpoint机制是其可靠性的基石, 当一个任务在运行过程中出现故障时, 可以根据Checkpoint的信息恢复到故障之前的某一状态, 然后从该状态恢复任务的运行, 而Checkpoint的执行策略根据SnapshotStrategy的实现决定, Checkpoint的恢复策略根据RestoreOperation的实现决定, 我们需要对如上提到的关键接口做好Remote StateBackend的实现, 整体拓扑结构如下图所示。



3.3 TaishanStateBackend架构设计

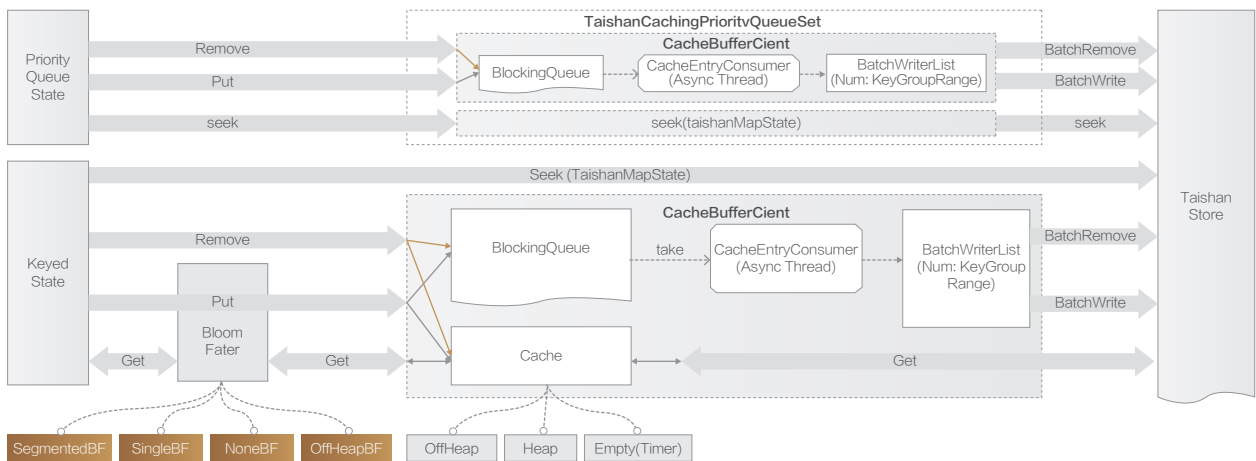
TaishanStateBackend基于如上规则和Taishan存储做了如下约定:

- » KeyGroupId和Taishan Table的shardID做一一对应
- » Taishan表的shard分片数和Flink的maxParallelism保持一致，且始终不可分裂和合并
- » 任务在第一次启动时，每个带有KeyedState的Operator会有且仅创建一张Taishan表
- » 一个Operator下可能存在多个State，发送的KV数据的K会加上ColumnFamily前缀
- » Flink做Checkpoint时，会对Taishan的每个shard做一次snapshot的创建，且每个shard的snapshotID一样
- » Flink做任务恢复时，本质是对Taishan的每个shard根据snapshotID做restore



通过修改成远端存储架构后，主要优势如下：

- Checkpoint 更轻**
 - RocksDBStateBackend的State均是存放在subtask的本地RocksDB中的，仅在做Checkpoint时会根据SnapshotStrategy决定是增量还是全量的方式上传至FileSystem，上传的内容分metadata和KeyedState内容，当修改成TaishanStateBackend之后仅需上传metadata元数据文件至FileSystem即可，单shard的snapshot过程基本上在毫秒内可以完成。
- 存算分离**
 - 改用TaishanStateBackend后，带状态的Operator无需此节点机器拥有高性能磁盘，State数据均存储于远端的Taishan系统，这样使得Flink的container机器减少了对磁盘的强依赖性，从而达到了存算分离的效果。
- 加速任务 rescale**
 - RocksDBStateBackend中当任务做扩缩时，由于subtask下对应的KeyGroupRange的start和end有变化，需要和其他subtask之间对本地的rocksDB实例按照KeyGroupId前缀做State的seek和batchPut操作，从而对State数据按照KeyGroup做重新分布，当State越大时，State数据重分布成本越大，任务rescale对延迟的影响也就越大。而使用TaishanStateBackend后，任务做扩缩操作无需State数据的搬迁操作，因为每个KeyGroup与shard一一对应，仅需修改KeyGroupRange的开始和结束即可。



4. 优化

我们前期在RocksDBStateBackend中已经对State的Read、Write、Delete、ReadNull、Seek等请求做了耗时、请求量和数据包长度的metric统计。通过对线上运行的任务指标做分析可以得知，State的每秒读写请求量最高可达百万级别，每秒十万级别的读写请求量任务也有百来个，另外每个请求数据包的大小在group agg和window agg场景下一般为几十bytes，而在Join场景下单条State的value即使做了zstd压缩后大小依旧会有百KB级别的存在。

在做功能测试时，发现一个任务在同等资源情况下，使用TaishanStateBackend的CPU负载更高，经过诊断发现当State的rpc请求量越大时，网络消耗CPU占比越大。当我们State的数据从本地转换成远端存储后，每次请求均走网络rpc请求，虽然没有RocksDB堆外内存的消耗了，但是对网络的依赖会较大，当rpc请求达到一定阈值时，网络必然会成为当前场景下的瓶颈。为了解决这一问题，我们选择在State与远端KV存储之间添加了一层

缓存层，减少对网络的请求读，另外参考Flink的MiniBatch，希望在缓存层也能做到攒批写的效果。整体结构如上，并在下文中逐步展开说明。

4.1 写优化

4.1.1 攒批优化写

RState的写操作，并未选择从Cache中遍历结果后写出到远端，也并未选择put/remove时直接与远端交互，而是选择将put/remote请求放入当前subtask的BlockingQueue中，使用异步线程去消费put/remote请求，当请求量达到一定阈值（最大攒批量或最大延迟时间）后会flush到远端KV存储，或者是Flink做Checkpoint时主动触发一次flush操作。经过添加指标和压测发现当最大攒批量设置为800时，对远端KV存储的put请求仅会有2-4倍的耗时增加，整体的网络写请求量成百倍的减少，增加了State的写速率。

4.2 读优化

4.2.1 缓存加速读

State的写操作，并未选择从Cache中

遍历结果后写出到远端，也并未选择put/remove时直接与远端交互，而是选择将put/remote请求放入当前subtask的BlockingQueue中，使用异步线程去消费put/remote请求，当请求量达到一定阈值（最大攒批量或最大延迟时间）后会flush到远端KV存储，或者是Flink做Checkpoint时主动触发一次flush操作。经过添加指标和压测发现当最大攒批量设置为800时，对远端KV存储的put请求仅会有2-4倍的耗时增加，整体的网络写请求量成百倍的减少，增加了State的写速率。

4.2.2 ReadNull优化

在实践的过程中，会发现如下两种场景下会存在较大的ReadNull请求，也就是Cache获取结果为null，且从远端获取结果也为null。

a) 当任务的key比较稀疏时，通过指标发现存在大量的ReadNull的请求，一般在去重以及window agg的hop场景下尤为明显。

b) 当任务的key存在天或小时这种周期性变化属性时,会导致大量的ReadNull请求产生,一般在Group Agg场景下尤为明显。

既然是ReadNull请求较多,经过调研后,初步尝试使用Hadoop的Bloom Filter来做ReadNull请求的过滤,当Flink做Checkpoint时,将BloomFilter结果写到远端存储系统,任务restore时亦可恢复。然而无论如何调整BloomFilter的系数和容量大小,随着时间的递增,由于BloomFilter无法做有效的delete,导致BloomFilter的过滤效果越来越低,假阳性越来越高,亦不能解决ReadNull的根本问题。

我们内部的Flink BSQL任务目前对State均有24小时的默认TTL设置,当State的数据超过24小时未访问后,State会失效过期,并随后在RocksDB做compact ion时被移除,这样的配置也基本上能满足大多数用户的实际使用的。

经过Team内部与用户排查故障总结中发现,用户业务行为中存在不少的window 5/10min,以及双流Join 1/2Hour的场景,这样24小时的TTL对这类Case显得是十分的富裕,既然如此,我们可以假设用户多数场景下数据不会存在延

迟,那么任务的Key在5min或者1小时内的实际有效key相对24小时会少很多。这时可以选择在缓存中划出一定比例的内存,用来创建一个独立的KV Cache,其中K用来存储TTL时间内State的key,而V使用Int类型存储当前K失效的绝对时间,单位为秒,当Cache使用驱逐策略删除数据时,可以自定义删除逻辑,删除逻辑为判断当前时间与失效的绝对时间大小对比即可,如果当驱逐数据不满足当前时间小于失效时间时,抛OOM异常,并提示增加相关内存或者调整KV Cache的比例即可。由于堆内存受限于GC的压力,再加上调研使用了堆外内存作为缓存的默认选择,故仅在使用堆外内存时,才允许使用这个独立的KV Cache,我们称之为OffHeap BloomFilter,有关堆内外选择的事项会在下文中做展开说明。OffHeapBloomFilter会将所有有效的Key缓存在内存中,用以充当BloomFilter的效果,从而来过滤大量的ReadNull请求。

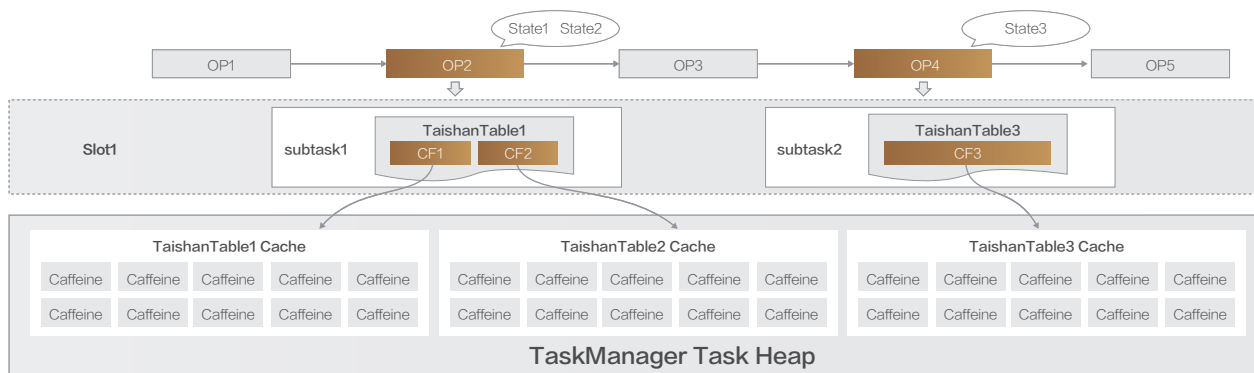
流计算任务做Checkpoint时,不会将OffHeapBloomFilter的数据flush到远端KV系统,当任务重启时,OffHeapBloomFilter会在第一个TTL时间内失效,但是会记录State的key前后数据,当第一个TTL时间之后OffHeapBloomFilter才会开始正常work,这样

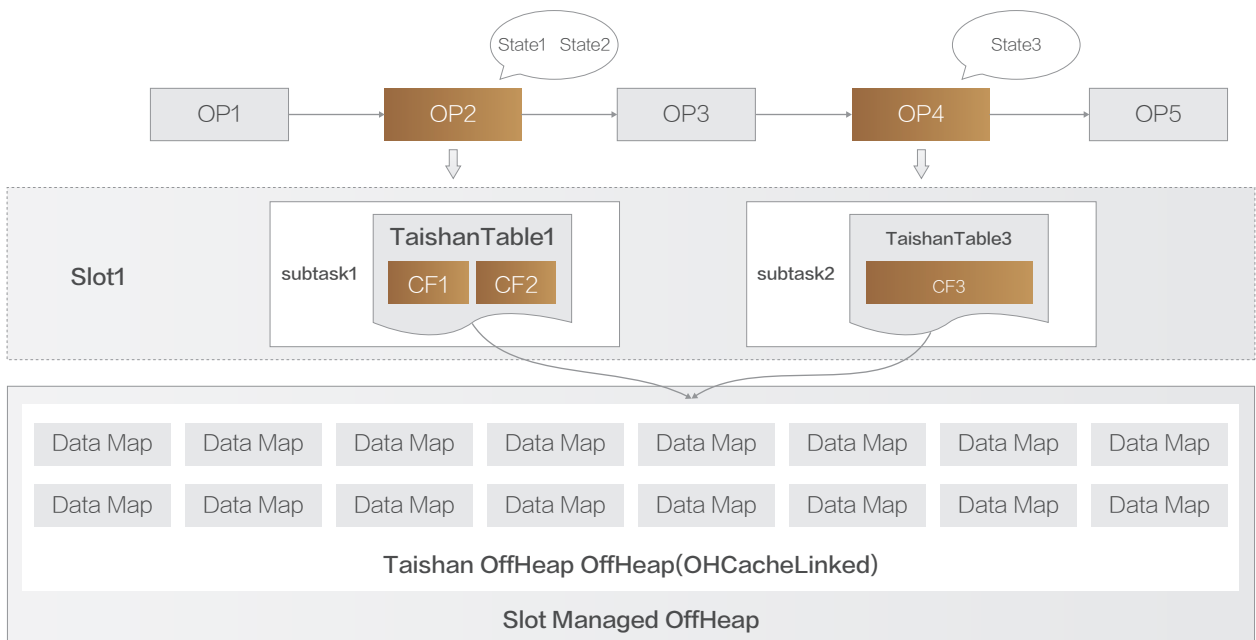
可以保障BloomFilter的准确与有效。

由于Window Agg的Keyed State使用的是Priority Queue State按照window的顺序来做清理的,所以其Keyed State默认是没有TTL的,我们对TumblingWindowAssigner、SlidingWindowAssigner、CumulativeWindowAssigner三种Assigner做了TTL的添加,TTL为Assigner的窗口时间加允许延迟时间,延迟时间一般是用来做任务的重启、失败或堆积的缓冲时间,避免因任务异常时间过长,导致State获取为null,从而引发数据结果不理想问题。同理,我们对Interval Join以及内部自研的两种延迟双流Join的Keyed State也分别提供了相应的支持,TTL设置为左右流的最大时间加允许延迟时间。

4.3 内存模型优化

在缓存读优化中,最初Cache使用的内存是Task Heap的堆内存,与用户代码的JVM堆内存共享,Cache使用的是高性能缓存库Caffeine。当一个Operator存在至少一个State时,创建一个Taishan表,subtask会根据拥有多少个Keyed State而决定创建多少个Caffeine Cache,结构如下。





这样的结构遇见了如下两个问题

1) Caffeine Cache的创建个数等于 $N * M$ ，其中 N 为当前TaskManager的Slot数， M 为Keyed State个数，由于流计算前后可能存在AGG或Filter导致前后Cache处理的State数据量不一样，从而导致对Cache按照时间或数量的驱逐策略难以配置，并且Cache对象之间是不共享的，低频更新的Cache内存无法充分利用。

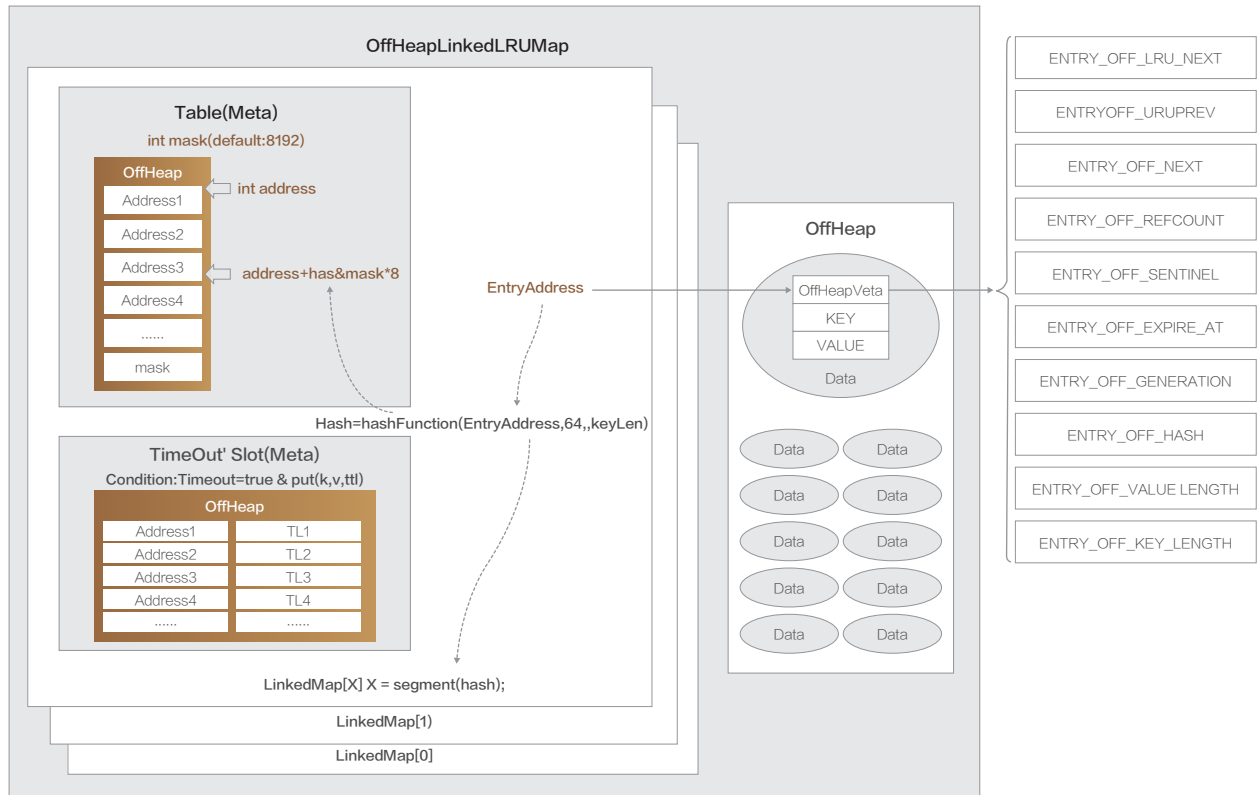
2) 使用中发现经过多次G1参数[2]调整后，依旧存在GC耗时较高，且存在不定时的GC高耗时抖动现象，从而使得任务不那么丝滑，影响用户体验。

经过调研RocksDBStateBackend的堆外内存机制[3]，以及市面上的堆外内存框架后，选择了使用off heap cache（简称OHC）框架[4]来做Cache，并产生了如下的结构。Caffeine Cache被替换成OHC Cache，内存从Task Heap转换成了使用Managed OffHeap，多个subtask共享当前Slot分配的OHC Cache，且分配的Cache对象唯一共享，经过如此替换后可以充分解决上面的两个主要问题，并且执行效率基本符合预期。

由于Taishan存储中早期未使用ColumnFamily的概念，且API层面将KeyGroupId放在了API作为参数，故将原RocksDB的K中KeyGroupId替换成了ColumnFamily，另外由于OHC Cache在堆外共享，可能存在多个State的ColumnFamily和Key均一致，为了避免Operator之间互相影响State数据，故将Taishan State的K添加了Operator前八位作为前缀。

K	结构	说明
RocksDB Key	#KeyGroupId#Key# Namespace#UserKey	RocksDB 原 KV 结构中 K 信息
Taishan Key	#ColumnFamily#Key# Namespace#UserKey	Taishan 的 KV 结构相比于 RocksDB 的 K 替换了 KeyGroupId，取而代之的是 ColumnFamily
Taishan OHKey	#Operator_8#ColumnFamily# Key#Namespace#UserKey	由于内存中多 Keyed State 共享 Cache，故在 Taishan Key 的开头添加了 Operator 前八位

另外针对OHC缓存, 我们内部默认选择OffHeapLinkedLRUMap, 其内存模型结构如下图所示, 并做些许适配:



1) 修改了hashTableSize, 并关闭了Map的自动rehash, 防止因KeyIterator的调用导致数据不准确。

2) 修改了LRU的驱逐逻辑, OHC在CacheSerializer中添加elementCouldRemove方法, OHC驱逐数据前会调用此方法判断数据是否可以移除, Flink端Value Serializer仅需对elementCouldRemove做实现判断当前时间是否大于失效时间即可, 这样的好处一个是可以和异步延迟时间flush做对齐, 保证失效数据均已被写到了远端存储, 另一个是在OffHeapBloom Filter中数据失效的驱逐策略逻辑中有使用。

3) 修改OHC的TimeOut触发逻辑, 虽然能通过指标精准查看Cache堆外内存的实际使用量, 但是对任务的吞吐存在负面影响, 故后续默认关闭了TimeOut功能。

5. 现状与未来

目前B站从2022年11月初开始逐步切换了100+个线上带KeyState的任务, 目前因增加了rpc网路开销导致整体资源使用有微量增加, 但是基本已实现存算分离和加速任务rescale的目的, 整体是符合预期的。

当然还存在一些特殊case, 留存的主要待优化事项如下:

高 QPS 场景

- 部分业务场景中key十分稀疏, 缓存命中率低, 当State数据量足够大时, OffHeapBloomFilter对内存需求过高, 降本增效的大环境下实施起来比较困难且收益比较低, 而关闭OffHeapBloomFilter会导致出现大量的ReadNull现象, 整体表现为单subtask的GRPC请求的QPS过高, 网络压力较大。

大 Key/Value 场景

- 双流Join下存在超大Value以及多字段去重的场景下存在大Key, 在运行一段时间后会存在WriteStall现象, 从而影响任务的健康运行, 这也是下一阶段需要和存储团队做进一步联调优化的地方。

状态的分层存储

- 目前的缓存的实现是使用堆外内存作为存储介质, 上面提到的高QPS和大key/value场景中会因为内存空间限制而导致缓存命中率下降。未来我们计划参考Flink Forward Asia 2022中提到的Tiered State Backend的思路, 将机器上的磁盘和内存都作为缓存加速的资源, 同时保持状态数据完整保存在远程存储上, 形成一套分层状态存储的架构, 这样的架构除了能够解决单一缓存介质的容量限制, 还有能够更加有效地提高混部情况下机器的资源利用率。接下来Flink的实时任务会通过K8S平台与在线业务实现混部和统一管理, 而随着B站在线业务正在积极推进“无盘化”的改造, 本地空闲的SSD磁盘资源能够更好地被Flink任务用作分层状态存储的缓存资源, 进一步提高混部技术下的机器资源利用率。

在如上优化项进行的同时, 会进一步推进TaishanStateBackend的覆盖率, 并选择合适的任务在离在线混部集群中做部署, 最终希望能达到默认开启的效果。

-
- » B站分布式KV存储实践 (https://mp.weixin.qq.com/s?__biz=Mzg3Njc0NTgwMg==&mid=2247484586&idx=1&sn=34be8355205b8c2644553851b122986b&chksm=cf2cc18ff85b4899d04e1bdfb8341f835fcdee7bb32de893fdcaa1b62036c919afc6e3683ece&scene=21#wechat_redirect)
 - » https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/g1_gc_tuning.html
 - » <https://github.com/facebook/rocksdb/wiki/Write-Buffer-Manager>
 - » <https://github.com/snazy/ohc> 作者: 哔哩哔哩技术 <https://www.bilibili.com/read/cv23347156/> 出处: bilibili



美的楼宇科技： 基于云原生技术的成本优化实践

程俊 / 美的楼宇科技研究院研发中心负责人

数据技术已成为企业竞争和创新的重要支撑，作为暖通行业和智慧楼宇行业的领军企业，美的楼宇科技深知数据技术在其发展中的不可或缺性，然而面对着业务的快速扩张、数据极速暴涨的情况，导致资源成本居高不下，业务的访问和计算性能存在严重瓶颈。

成本洞察：确定降本方向

我们主要围绕着业务场景、资源使用情况、成本结构等方面进行了深入分析：

- » **业务场景分析：**根据当前的业务布局，从区域维度（海内外、国内区域分布）、业务分级、业务分类等方面进行分析，寻找可降低资源投入、降低资源配置的或融合使用资源的方向。
- » **资源使用分析：**根据云资源、本地资源的分配情况，持续利用率情况，结合业务场景，分析并列举需要增降配、分离或合并、架构优化的方面，进行统计。
- » **成本结构分析：**根据年度成本结构，按月查看各资源线、业务线的成本结构和成本走向趋势，分析成本构成，挖掘可降本资源、重点降本资源等信息。

经过深入分析，我们发现在数据平台的架构、存储管理和监控、数据备份和恢复策略等方面，具有采用云原生技术进行优化的巨大空间。

成本优化：用运营和技术手段降本

本文重点从存储和计算资源两方向对成本优化方案进行介绍。

存储资源优化

首先，采用数据压缩、去重、快照等技术来优化存储空间的使用，并确保备份和恢复策略的可靠性和效率，避免不必要的存储资源浪费，例如，只备份必要的数据、避免重复备份等，再通过引入存储管理和监控工具，精确监视数据存储的使用情况，及时发现问题并进行解决，并对存储资源进行分类管理，将不同等级、不同访问频率的数据分配到合适的存储介质中，降低

数据存储成本。同时，基于存算分离架构，采用云计算和边缘计算等技术，将数据存储云端或边缘设备上，计算时再将数据拉取到计算资源上进行处理，从而降低数据成本。据我们统计，采用该方案可以节约数据中台的存储成本30%~40%。

另外，采取冷热数据分离策略，将热数据存储在高性能设备上、冷数据存储的低成本介质上，在保障整体数据处理性能的同时，实现成本最小化。我们分析发现，数据中台80%是冷数据，且只有20%的冷数据需要在特定时期内访问，因此采用冷热数据分离能有效实现成本控制，将iBUILDING数据中台50%~90%的数据存储成本降至最低。此外，我们将数据存储在多个分布式源站上，以提高数据的可靠性，同时也为数据的备份提供更全面的保障。采取该方案，可以将数据备份的时间缩短50%以上，在降低备份成本的同时提供更可靠的数据备份与恢复策略，并降低数据中台的运营成本，节省维护和备份开支。

计算资源优化

我们在Hadoop架构基础之上，引入了MPP(Massively Parallel Processing)架构的数据处理引擎，整合成云原生混合架构，混合架构综合了MPP 架构和 Hadoop 架构各自优点：

- » 基于MPP的海量并行处理数据库(Starrocks)，具有更好的实时性能；
- » 基于 MPP- Hadoop 混合框架构建的异构数据融合与集成系统，提升了数据查询和加载效率；
- » StarRocks同时支ES/MySQL/Hive等外表功能，可以实现对Hadoop生态的查询，可以做到存算分离，对于节省成本，打通Hadoop生态很有意义。

相比于之前的架构，新架构使得机器成本降低了50%以上、聚合上卷查询，关联查询方面性能可以提升20-30倍、明细聚合查询性能提升4倍以上。同时，容器技术可以用来保持数据分析应用程序的隔离性，使它们更容易管理，并针对不同业务容器执行不同等级、不同规格的标准化资源配置，来提升计算资源的利用率。

在未来的数字化发展进程中，降本增效是企业必然要面对的事情，楼宇科技iBUILDING数字化平台在不断丰富和完善数字化能力，满足建筑智能化市场的需求，并不断地利用云原生技术进行创新，为市场带来性价比更高、功能更强的产品，为客户创造更多价值。



去哪儿: 以技术为“魂”， 赋能城市旅游

每逢节假日来临之时，面对数以亿计的骤增流量，OTA平台压力倍增。以今年五一为例，文化和旅游部公布的数据显示，国内旅游人数共计2.74亿人次，也意味着平均每5个人中就有一个人在旅行。据去哪儿的数据显示，热门线路火车票平均40s被抢空，售罄速度超过春运。OTA平台去哪儿技术副总裁孙斌表示：“五一放假前的4月27日、28日左右，去哪儿平台同时在线购买火车票人数达到历史新高，热门城市酒店的预订量也达到了疫情前1.9倍，热门城市的机票预订量超过疫情前1.5倍。”

疫情三年，OTA平台几乎处于“放假”状态，当突然面临用户爆发式激增，服务器承载能力是否跟得上？上下数据回传时能否不卡顿、不宕机？前台用户能否像以前一样保持“丝滑”体验？

“在此之前，我们已经预料到，这将是疫情开放以后最‘疯狂’的一次五一，也已做好了充分的压力测试准备。”去哪儿技术副总裁孙斌表示。这轻描淡写背后，是去哪儿和华为云为期五个月的备战。

事实上，这不仅仅是为五一波峰准备，对去哪儿而言，更像是对业务和技术配合度的一次测试，决定了在行业持续复苏背景下，未来是否能以云上的创新竞争力，接住更大的机遇与挑战。



技术为“魂”，三次迭代

“技术力”是去哪儿贯穿始终的核心。这样的思路下，坚持以技术解决产品问题、以技术解决业务增长问题，用技术穿越发展新周期，通过一系列技术升级的关键动作，赋能业务，去哪儿实现了一次又一次跨越式增长。

2005年，一家以搜索为核心，实现线上比价机票和酒店的网站成立了，它的名字叫“去哪儿”，去哪儿创业之初就找到了自己的定位：“低价机票+年轻客群”。

这一年具有跨时代的特殊意义，中国网民数量首次突破一亿，成为仅次于美国的互联网第二大国，也意味着中国互联网迎来了爆发时代。去哪儿上线3个月内，独立用户每两周访问量以超过50%的速度增长，2007年10月访问量突破1200万，2008年突破2400万。

在用户激增的势头之下，去哪儿做出了第一个关键动作——上线了TTS（Total Solution）系统。系统从根源上解决了用户登陆多个机票供应商网站注册的痛点，和机票供应商订票

不出票、临时加价、黑票频出、退改流程繁琐等致命问题。2010年系统一经上线，部分机票供应商单日出票量达到3000-4000张，人工成本却直线下降了2/3。乘势而上，去哪儿决定对机票供应商实施CPC点击收费，当年就实现了盈利。

第二个关键动作，是快速完成PC向移动端迁徙。随着智能手机的发展，互联网圈开启了一场PC端向移动端迁移的用户抢夺战，拼的是速度。2010年10月，移动端的去哪儿旅行APP上线。移动端一路“狂飙”，直接将这个创业8年的公司送进了纳斯达克。具体来看，机票方面，2011-2013年日均出票量从26000涨到74000张，截止到2013年，机票业务占整体业务的70%；酒店方面同步加速发力，据艾瑞咨询数据报道，截至2013年6月，去哪儿一跃成为酒店在线预订领域内的流量第一大入口。在当时，中国有30多万间酒店，其中70%以上是由小商户掌握，对平台方来说挖掘价值潜力巨大，因为酒店业务的利润比机票业务高得多。

捷报接连传来，2013年6月，去哪儿移动端下载量也突破一亿次，成为当时在线旅游行业移动下载量最高的企业，并保持领先地位至今。技术解决增长，在去哪儿面前体现的淋漓尽致。“狂飙”之后，2022年去哪儿决定为自己“瘦身”，提出业务瘦身42%+和效率提升50%的计划，这是它的第三次迭代，即，用技术力降本增效。一方面是因为企业在长期发展过程中，公司组织架构和人员流动非常大，导致目前开发者们维护的系统大多是交接过来的，对于系统全貌掌控力有限。另一方面是，此前系统之间的耦合严重，模块系统彼此关联复杂，系统越来越冗杂，随即导致修改一个小的产品需求、回溯需求以及查找需要修改的功能点的工作时长都达到“天”量级，更别提修改带来的不可预知的影响。这项任务从产品、技术需求和工单问题排查等，都带来较高学习成本和开发成本。于是去哪儿决定在不影响业务开展的前提下，边删、边拆、边改，演化式改造。当他们在专心修炼“内功”时，一次新的大考又来临了。

流量大考，弹性应对

流量大考的信号从“摘星”那一刻开始。

2022年6月29日，工信部信息通信管理局发布了一则消息：为了方便广大用户出行，即日起取消通信行程卡“星号”标记。消息一出当天冲上微博热搜，网友纷纷表示“终于可以回家了”，

“终于可以尽情的出门撒欢了”。

据去哪儿数据显示，消息发布半小时后，平台上国际机票瞬时搜索量翻倍，达到过去两年以来平台国际机票搜索量最高峰。同时，酒店搜索量翻倍，火车票搜索量上涨达到1.5倍。

“瘦身”后的去哪儿能承受突如其来的流量大考吗？“这还只是开始，接下来将是春节、元旦、五一，三节紧密相连，我们也曾担心，服务器是否能承受同一时段涌入的波峰流量。”孙斌表示。

作为去哪儿的技术合作方，华为云的驻场技术团队，则在此时展现了专业而强大的技术服务能力。一方面，驻场团队可以深入了解企业运营逻辑，便于提供更符合客户运维的方案；另一方面，可以及时响应客户的临时需求。

对于去哪儿来说，业务需求很明确：弹性上云、降本增效。针对这些需求，驻场的华为云技术团队持续深入去哪儿业务场景，在不修改基础架构和降低成本的基础上，基于华为云 Serverless 容器 CCI 提供的弹性上云方案，助力去哪儿“0改造”完成 IDC 业务的技术架构升级。

华为云云容器服务 CCI，就好比内置了一个开关，在节假日等高峰时段自动“开闸”扩容，帮助去哪儿增加云上带宽、服务器、计算等资源使用，满足平台海量数据兼容、高速处理。而日常，这一“开关”则自动“关阀”，在保证去哪儿业务有序运营的同时，减少云资源的不必要浪费，满足了去哪儿对“弹性”扩容的诉求，也节省了大量固定成本投入和资源闲置，大

幅降低运营成本。

另外，这一方案具有简单易用、安全快速开放兼容性强、IT 开发投入低等特性，同时还在开源方案基础上形成主备加多活相结合的高可用方案。如果平台服务器等遇到了单点故障，可秒级切换备用方案，确保了去哪儿业务的稳定性和可靠性，并真正做到了“按需弹性、按秒计费”，实现弹性资源 100% 利用率。

在今年不仅如此，华为云的确性运维体系也非常符合客户需要，去哪儿目前业务增长很快，同时也处于上云的关键期，通过引入华为云确定性运维能力以后，在指标体系建设、变更管理优化、故障管理优化、重保、混沌工程等领域展开合作，帮助去哪儿的业务更加稳定，有力地应对业务增长和上云。

五一黄金周到来之前，为应对可能的流量洪峰，双方合作成立了超过 30 人的重大保障项目组，提前 3 个月开始跨区域协调资源，累计为去哪儿高峰提供了充足的 CCI 资源。孙斌记得，在三

月底的时候，去哪儿业务侧觉得资源可能还是不够，于是运维部跟华为云沟通再扩容，华为云快速响应，在四月初完成新一批资源扩容，上浮 20%。

这是中国旅游业经历三年疫情后迎来的首个黄金周，业务侧也极难预估高峰业务量级，能做的就是做好充足的应对准备。

双方提前频繁进行各项子业务的压力测试，不断调试架构和时延等，每日超过五次巡检，五一假日期间持续保障，平均使用核数超过 2.5 万核。事实证明，这次充满不确定性的波峰大考，华为云稳稳地接住了。

从上好云到用好云、管好云，服务更多的企业在数字化的道路上更好做到“精益”与“创新”，不仅是云厂商的产品服务目标，也是千万企业发展的核心诉求。以服务力和创新力为核心，未来华为云还将继续支撑去哪儿云上创新，还将继续做好千行百业的数字化“云底座”和“使能器”。







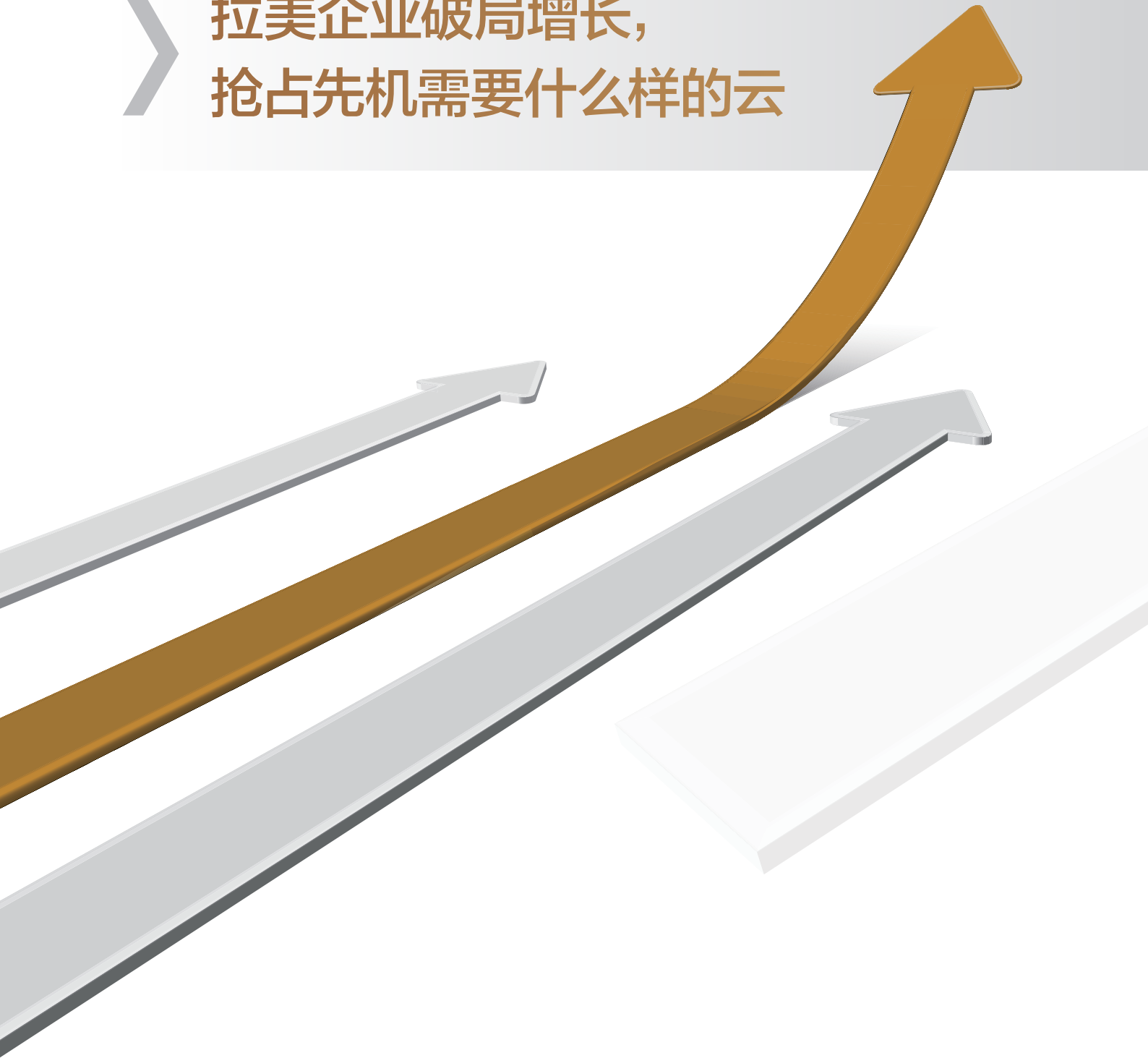
人物专访

拉美企业破局增长, 抢占先机需要什么样的云----- 85-89

云原生 Cloud Native



拉美企业破局增长， 抢占先机需要什么样的云



”

20000公里, 这是地球最远两个点之间的距离, 地理学将地球同一直径的两个端点称为对跖点, 中国与拉美地区就属于这样的对跖区域, 中转24小时起且没有直航, 错位的晨昏, 相反的季节, 中国与拉美地区似乎天然不同频。在很长一段时间里的确如此, 经济活动空间联系存在距离衰减, 较近区位之间的经济联系成本较低, 经济活动自然频繁。然而, 如今的拉美却随处可见中国元素, 电商、手机、短视频、出行平台、Fintech公司、移动支付应用、汽车等行业, 都有不少中国企业的身影。

抛开距离不谈, 拉美和中国有相似的条件, 两者处于相近的发展阶段, 过去十年, 中国对拉美的贸易、投资、金融合作等均增长很快。一方面, 中国品牌或直接或间接地参与到拉美市场的竞争, 另一方面, 拉美地区的本土企业们开始学习, 如何在当地复制中国的成功模式。被视作数字时代商业支撑的云技术, 也随之站在了聚光灯下, 区位和空间位置依然重要, 但云计算正在弥合空间距离与产业生产要素配置的鸿沟。

“



商业向东, 技术向云

尽管距离遥远, 但中国和拉美地区的政府机构、投资人和企业考察团, 还是明显增加了互访频率, 在他们眼中, 拉美地区不只有足球、桑巴和热带雨林, 还有海量的机会和增长。

拉美地区人口基数大, 且年轻人占据了相当比例, 较高的城市化水平、庞大的数字基础设施, 为数字企业打造了良好的营商环境, 拉美地区涌现出了一大批知名的本土互联网公司, 并且来自世界各地的互联网企业也都选择在此处落地生根。

比如拉美最大的在线购物和支付平台MercadoLibre; 提供机票、酒店预订、度假套餐和租车等服务的Despegar.com; 拉美地区最大的外卖平台iFood。以及2023年上半年开放的Tik Tok Shop巴西站; 电商平台SHEIN和Shopee等。

这些企业的兴起, 也改变了拉美当地用户消费方式和生活习惯, 数字经济

俨然成为主流, 拉美地区的商业氛围, 和中国很是接近, 譬如年轻人喜爱的游戏和电商等, 这也为很多企业经验和技术的迁移复制, 提供了得天独厚的优势。

上层应用对底层基础设施的影响显而易见, 不论是海外公司还是本土公司, 都看到了云计算的成本、性能和优势, 拉美企业越来越多地生长在云上, 问题是, 企业应该如何找到适合自己的那朵云?

在华为云云上早餐会上, 来自UOL CTO Alcides Zanarotti Junior, 以及来自Appvantgarde CEO | WebMaps CTO José Chávez就与华为云CTO张宇昕进行了探讨。UOL成立于1996年, 是巴西最大的内容提供商、技术和数字服务公司。2017年, UOL开始尝试公有云服务, 从CDN开始上云, 之后UOL体验到了云服务的便利, 开始迁移邮箱等业务系统。

互联网产品的快速迭代, 需要基础设施尽可能标准化; 云资源快速扩容和及时回收, 确保既能应对大流量, 又不至于耗费高额资源, 此外, 公司业务规模增长, 也带来了运维的复杂性。

“UOL的使命是为客户提供媒体视频, CDN是我们第一个使用公有云的项目, 它能让我们更快地为我们的最终客户带来价值, 可以快速上线新功能, 也可以很快修复各种问题, 从而提高产品性能, 优化产品体验,” UOL的CTO Alcides Zanarotti Junior表示。

UOL作为一家内容提供商, 需要快速发布新闻, 因为有些主题对读者来说很重要。UOL需要在新闻现场, 需要应对突发新闻带来的流量高峰, 需要在在一个面积广阔的国家传播内容, 因此需要云服务有冗余和耐用性, 以及稳定可靠运作的的能力。

作为一家提供汽车定位系统的公司, Appvantgarde CEO | WebMaps CTO的 José Chávez提到, WebMaps有很多服务器设备和数据中心7x24小时运行, 需要做到及时响应, 同时客户流量有波峰和波谷, 这就需要可扩展的能力进行资源分配, 避免造成不需要云资源的情况下产生过多的运维费用, 并且WebMaps业务覆盖面很广, 所以需要云服务商在拉美有足够多的节点。

“我们正在寻找更多可扩展性, 安全合规对我们来说很重要。此外, 高性价比的云服务能将普惠带给我们的合作伙伴。”他说。



做一朵拉美的云



拉美地区的数字化业务增长迅速，新技术新业态层出不穷，企业需要稳定高效、可持续创新的云服务，敏捷构建应用，上云成为共识。数据显示，近年来拉美地区开始广泛使用云计算和大数据等新兴技术，预计未来5年，拉美地区云计算技术使用率将增长25%。咨询公司IDC预计，到2025年，拉丁美洲的云市场规模将达到220亿美元。拉美地区正在成为云计算厂商关注的焦点之一，华为云是更早在拉美地区布局的厂商，张宇昕提到，拉美地区企业业务增长很快，他们首先需要云服务商有足够的资源、能力和技术，满足企业现在及未来的需求。

José Chávez表示，与华为云之后合作，WebMaps可以为客户提供几乎100%可用性的解决方案，并且实现了更快的响应能力，感觉云资源仿佛就在身边；其次，云的安全性对每个公司都非常重要，为此，云服务商也应该具备长期稳定的安全运营经验。“安全，是我认为最重要的事情之一。” José Chávez表示，在过去的几年里，华为云在研究和创新方面投入了大量资金，WebMaps也从其他厂商的合作伙伴，发展为华为云的合作伙伴，华为云

为众多全球客户服务的履历，拥有足够的说服力。第三件事是本地化和全球化，企业需要本地团队的服务，而在出海时，不同地区、不同领域涉及到很多法规或法律以及技术要求，云服务商要帮助企业解决后顾之忧。以当前拉美区域的复杂业务形态为例，每一个企业可能都会用到中心云、边缘云，涵盖原有自建IDC和混合云，华为云可以提供分布式云能力，这种多云状态能够给予客户更多自由的选择，兼顾历史投资和商业创新。

张宇昕表示，华为云除了专注于技术创新之外，还有两部分的差异化能力。一是了解企业的需求：我们有30多年数字化转型的经验。华为本身也是一

个典型的大企业，我们可以将这些经验分享给合作伙伴与客户，帮助客户从传统架构、传统商业模式，转移到新商业模式、新平台、新技术。华为云第二个独特的价值，是本土化和全球化，许多全球合作伙伴和全球客户都想在中国拓展业务，很多中国公司都想走出国门，在全球市场的不同领域开展业务，华为云的销售和支持团队遍布170多个国家，企业想去任何地方，华为云都可以提供属地服务支持。Alcides Zanarotti Junior也表达了对于华为云的认可，UOL的使命是为最终客户提供媒体视频。TikTok不仅在中国，而且在全世界都是一个模范，而华为云也正在与这些大公司合作，这些实践经验都可以很好的共享给我们，这对于我们来说是很好的参考。“上云是一段漫长的旅程，大多数互联网公司从一开始都不是云原生公司，云计算不仅仅是技术的进化，也是商业模式和发展模式的进化。”张宇昕表示，面向未来企业持续的云上需求，华为云坚持“上好云、用好云、管好云”的路径，不仅关注业务拓展、用户体验，更关注云上创新和精益运营，帮助企业成就新云原生企业。



以一切皆服务做锚点

在拉美，为拉美。一切皆服务理念，是华为云的锚点，一切皆服务包含三重含义，基础设施即服务，让业务全球可达；技术即服务，让创新触手可及；经验即服务，让优秀得以复制。

一切皆服务体现在拉美地区，就是不断加码的投入，和兼具拉美地区特点的服务和解决方案，华为云持续提供最新技术，在拉美的商业战略能开放华为30年积累的ICT技术和能力，帮助更多客户和伙伴更简单、快速地完成数字化转型。

据了解，华为云已经为拉美电商、金融、初创、Fintech、媒体、物流、教育、政府、电信、制造等行业的5000多家客户提供了云服务和解决方案。在拉美区域各个国家，与当地的1400多家合作伙伴建立了长期的合作关系，共同推动数字化转型和创新发展。

资源布局直观体现一家国际云厂商的决心，在拉美，华为云当前有三个核心region和最多的AZ可用区。今年下半年，华为云还将在巴西、墨西哥和智利推出新的可用区，华为云在拉美的可用区总数将达到11个。

此外，华为云构建的全球存算一张网KooVerse实现了全球50ms时延覆盖圈，可以为拉美客户提供更加稳定、高效、低时延的云服务体验。目前，华为云是拉美增速最快的云厂商，

如今已跻身拉美地区TOP4。

华为云还将目光聚焦在上层平台能力。从行业维度观测，未来拉美数字经济的增长将以Fintech、教育、电商和媒体与娱乐等行业为主导，华为面向四大类互联网行业客户，发布了相应的行业解决方案，力求打造新体验、新效率、新模式，助力客户云上新增长。

此外，华为云在拉美发布了ESW、CCI、Digital Avatar等一系列云服务，支持拉美数字经济快速增长。同时，华为云通过端到端的效率提升、体验提升以及模式创新，助力拉美客户应对优化内部运营、增强用户粘性、提升收入三大数字经济挑战，实现自身和客户的共同成长。

用云原生的思维践行云原生，华为云联合当地的伙伴成立拉美创原会，与拉美技术精英共同探索新技术分享新实践，创造技术与业务融合的无限可能。

此外，华为云还与AMEXCID、Softex等产业组织达成战略合作，共同发布华为云初创计划，共同培育和扶持139家初创企业；同期华为云在拉美举办开发者大赛，鼓励拉美开发者充分发挥想象力和创新精神，充分发挥云服务商巨头的社会责任和价值。

中国与拉美地区的物理距离没有变化，但是云计算让数字距离无限缩短，正如托马斯·弗里德曼在《世界是平

的：21世纪简史》所写：由于信息和通信技术的发展，“世界正在被抹平”，即资本、技术和信息正在打破一切疆界，整个世界已变成一个地球村。

未来，华为云也会与拉美企业一步步打破数字经济的疆界。

聚八方领航者 论云原生之道

与行业技术精英
共创云原生的无限可能

