产业动态 | 技术热点 | 行业趋势 | 实战分享 | 人物专访

云原生

Cloud Native

2022.05
第3期

供创原会会员
内部交流使用

创原会
全球云原生交流平台

目录 CONTENTS

标准与开源动态

中国信通院联合IDC发布《全球云游戏产业深度观察及趋势研判研究报告（2022年）》

报告从全球云游戏产业发展情况入手，深入阐述了云游戏的现状及未来，总结了云游戏产业历经的各个阶段，梳理了全球产业链分布地图。对中国及海外区域在市场规模、用户规模、接入类型、产业生态等不同维度的差异表现进行了重点剖析，并结合国内云游戏行情，详细介绍了中国云游戏的用户行为特点。最后，从内容、场景、入口、分发、终端、网络、算力、成本、政策、生态十个层面对发展趋势进行了深度研判。

中国信通院发布《数据中心白皮书（2022年）》

在数字经济时代，算力正在成为一种新的生产力，广泛融合到社会生产生活的各个方面，为千行百业的数字化转型提供基础动力。数据中心是算力的物理承载，是数字化发展的关键基础设施。《数据中心白皮书（2022年）》基于全球视角和我国现状，梳理了数据中心产业总体及热点情况，重点从市场、技术、政策等维度分析了我国数据中心产业的发展，并研判了我国数据中心发展趋势。以白皮书发布为契机，中国信通院将进一步加强产业研究、推动行业交流，促进我国数据中心产业低碳高质发展。

华为云云原生开发者认证人才计划启动

面向高校学生、个人开发者、企业开发及运维人员，华为云推出了云原生开发者认证（HCCDA - Cloud Native），从开源组件到华为云上服务的介绍，使您掌握云原生的核心理念和架构，具备基本开发实践能力，华为云开发者学堂上线“云原生入门级开发者认证人才计划”，赋能开发者，帮助大家快速获得相关专业认证。

Gartner发布新兴技术研究：预计2026年全球30%的企业机构将拥有元宇宙产品和服务

目前被定位成元宇宙的解决方案虽具有潜在的兼容性，但它们并不符合完整元宇宙的定义。早期解决方案可能具有一个或多个属性（持久性、去中心化、协作性和互操作性），但不具备成熟元宇宙所要求具备的所有属性。早期解决方案包括游戏、虚拟协作、导航应用、社交媒体

和非同质化通证（NFT）。Gartner预测成熟的元宇宙将不依赖于设备、不属于任何一家厂商并且将拥有由数字货币、非同质化通证或类似元素所组成的虚拟经济体系。

KubeCon & CloudNativeCon 2022 Europe在西班牙瓦伦西亚成功召开

KubeCon & Cloud Native Con是全球顶级云原生盛会，也是技术爱好者最向往的云原生交流盛典，大会汇聚了全球最活跃的云原生开源项目、最先进的云原生技术与最前沿的云原生行业实践，不断推动云原生的技术进步与产业成熟。2022年5月16日至5月20日，KubeCon & Cloud Native Con Europe 2022在西班牙瓦伦西亚重新开启，这是自2020年新冠疫情以来首次举办线下会议，沉寂已久的全球云原生企业及技术爱好者的热情被再次点燃。华为云成为入选顶级赞助商名单的唯一中国企业，同时也是大会演讲议题最多的中国企业，再次表明华为云在云原生产业投入的决心，以及CNCF及业界专家对华为云在云原生领域所获成绩及卓越贡献的认可。

随着云原生技术的全球普及，CNCF会员数快速增长

CNCF于3月公布最新会员名单，新增 68 名白银和最终用户成员以及最终用户支持者。这些新成员将与超过 775 家其他成员并肩工作，构建和使用云原生技术，使世界各地的组织能够应对 5G、边缘、大规模等挑战。CNCF 年度调查显示，2021 年是 Kubernetes 跨越鸿沟的一年，参与调查的 96%的组织使用或评估了 Kubernetes。CNCF 项目的采用量也显著增加，其中 containerd 增加了 500%，Fluentd 增加了 53%，Prometheus 增加了 43%。

CNCF 总经理 Priyanka Sharma 说“我们看到新成员的增长速度加快，这可以归因于 Kubernetes 和容器技术的成熟和普及。CNCF 项目在帮助 Allianz Direct、富达投资和沃达丰等最终用户应对紧迫和不断变化的基础设施挑战方面发挥了不可或缺的作用。我们很高兴有这么多新成员加入，继续我们的使命，让云原生计算无处不在。”

Google宣布将Istio捐赠给CNCF

在4月份举办的IstioCon 2022上，谷歌副总裁兼研究员在主题演讲中表示，“Istio商标将被移交给CNCF，并由该基金会如同Kubernetes等其他商标一样负责管理，由此降低广泛的社区风险并建设真正开放的生态系统。如此一来，Istio将成为完整Kubernetes集合堆栈中的组成部

分。” Istio服务网格的支持者及其他IstioCon与会者对此纷纷表示赞赏，认为这将扩大Istio项目在用户及供应商合作伙伴中的采用比例。Linkerd社区负责人们也表示，此次捐赠将为CNCF各服务网格项目间的协作铺平道路。

首个云原生批量计算项目Volcano正式晋级为CNCF孵化项目

4月初，云原生计算基金会（CNCF）宣布，由华为云捐献的业界首个云原生批量计算项目Volcano正式晋级为CNCF孵化项目。目前Volcano在Github上已获得2.2k+star，490+fork。来自世界各地的300+开发者参与代码贡献，超过50家企业与科研机构参与项目合作，合作伙伴包括华为、腾讯、百度、爱奇艺、滴滴出行、京东、建信金融科技等。可以预见，Volcano 将在企业数字化、云原生转型过程中发挥越来越重要的作用，CNCF云原生技术版图对于批量计算的支持也已将更加成熟。

Kubernetes 1.24 正式发布，Dockershim从Kubelet中移除

» Kubernetes 1.24 已正式发布，这也是 2022 年的首个大版本更新。重大更新包括：
» 从 kubelet 中删除 dockershim 组件
» 默认关闭 Beta 状态的 API
» 提供的签名的发布工件 (Signing Release Artifacts)
» OpenAPI v3
» 存储容量和存储卷扩展功能正式 GA
» NonPreemptingPriority 正式进入稳定状态
» 迁移存储插件
» gRPC 探针升级至 Beta 版本
» Kubelet Credential Provider 升级至 Beta 版本
» Contextual Logging 进入 Alpha 阶段
» 避免为服务分配 IP 时发生冲突
» 从 Kubelet 中删除动态 kubelet 配置

KubeEdge发布v1.10.0版本

» 2022年3月KubeEdge发布v1.10.0版本，新版本发布了多个重要特性：
» 基于keadm的安装体验升级
» 基于Kubelet裁剪的下一代Edged预览版
» Edgemark: 支持大规模的kubeEdge集群性能测试
» EdgeMesh代理隧道支持quic协议

Volcano正式晋级为CNCF孵化项目

4月7日，云原生计算基金会（CNCF）宣布，由华为云捐献的业界首个云原生批量计算项目Volcano正式晋级为CNCF孵化项目。这意味着Volcano的技术生态受到业界广泛认可，CNCF云原生技术版图对于批量计算的支持也已趋于成熟。

张宇昕

华为云CTO

华为云一直致力于云原生技术、产业和生态的建设，Volcano融入了华为云在云原生、AI、大数据、HPC等领域中沉积的行业和技术经验，促进云原生技术与企业数据资产高效融合、充分释放数据红利，加速企业数字化、智能化进程。

Chris Aniszczyk

CNCF首席技术官

对于复杂的Kubernetes工作负载，尤其是AI、大数据等领域，批处理可以大大简化部署。以云原生的方式来精简大批量数据的处理是非常新颖和有价值的实践，Volcano使得Kubernetes能够成为世界级的工具，助力科学研究、高性能计算等行业的发展。

Volcano项目于2019年6月开源，2020年4月正式成为CNCF沙箱项目。Volcano自2020年进入CNCF以来，在人工智能、大数据、基因测序等海量数据计算和分析场景得到快速应用，并构建起完善的上下游生态，目前腾讯、爱奇艺、小红书、蘑菇街、唯品会、鹏城实验室、锐天投资等企业均已将Volcano应用于生产环境。

自加入CNCF以来，Volcano社区已吸引2.6万全球开发者、并获得2.3k Star和530+ Fork。Google、Facebook、Amazon、Red Hat、华为、百度、腾讯、建信金融科技等科技巨头纷纷加入Volcano社区贡献，海内外生产落地用户广泛分布于互联网、先进制造、金融、生命科学、科研等行业。Volcano也因其创新的技术理念、活跃的社区生态获得第二届“中国优秀开源项目”和 “2021年OSCAR尖峰开源社区及开源项目奖”，其作业管理能力被写入由中国信息通信研究院牵头制定的《高性能计算(HPC)云平台标准》，成为行业标准。

过去两年，Volcano全球生态发展迅速，一批行业标杆用户不仅积极地推动Volcano落地生产环境，也基于自身实践反哺社区，实现双赢。

张雷
小红书技术部负责人

云原生批量计算项目Volcano应用于小红书大规模机器学习平台、大数据平台等生产系统，支撑着搜索、推荐、广告、内容审核等多项关键业务，Volcano大大简化和加速了大数据以及AI应用在云原生环境的落地进程，小红书业务系统借助Volcano提供的丰富功能和优越性能，实现了资源成本的降低和作业性能的提升，期待Volcano在云原生领域持续深耕，发挥更大价值。

常峰
中科类脑研发总监

Volcano是最早针对批量计算场景开源的云原生项目之一，其动态可配的高级调度策略和优秀的资源管理能力解决了AI场景下作业调度、生命周期管理、异构硬件支持等多个问题。在落地实践的过程中，我们基于Volcano的能力做扩展，有效提升了系统稳定性和资源利用效率。期待加入CNCF后，在社区的加持下Volcano能持续孕育出更多优秀的解决方案和最佳实践。

截止目前，Volcano社区共发布21个版本，最新版本为v1.5.1。2022年，社区将会进一步扩大技术版图，聚焦以下能力建设：

跨云跨集群调度：跨集群一直是分布调度系统解决大规模、灾备等问题的主要解决方案。同时，为了降低厂商绑定的风险，并最大限度兼顾不同云厂商的优势，多云环境下的负载高效分发逐渐成为趋势。Volcano将会通过多个项目构建分层调度体系，基于全局资源视图，提供多样化策略如成本优先、效率优先等，为作业发放提供最佳决策。

» **在离线作业混部：**针对业界普遍存在的数据中心集群资源率低下的问题，Volcano将围绕业务感知、在离线统一调度、资源超卖、资源隔离与抢占、动态调度等能力的构建，在保证业务稳定性的前提下实现降本增效。

» **弹性调度：**针对弹性训练、竞价实例的场景，Volcano将会增强基于min，max的调度能力、作业感知、资源抢占能力，实现资源利用最大化。

» **GPU虚拟化：**推理场景以及GPU开发的场景，GPU使用率普遍偏低，Volcano已实现多容器共享使用GPU，未来将进一步增强算力、显存的隔离能力，保障在提升利用率的同时，降低业务间的干扰。

» **细粒度资源管理：**Volcano目前通过Queue提供资源的高效复用，针对更复杂的场景，Volcano将会通过Hierarchy Queue,、Policy per Queue、Plugin per Queue等机制提供更细粒度的管理和共享。

» **工作流管理：**工作量的编排使用越来越广泛，Volcano将基于子项目JobFlow，构建多场景、轻量化、高性能的编排能力。

» **基于真实负载的动态调度与重调度：**针对当前基于资源申请进行的负载调度、资源碎片化引入的节点使用率不均衡的问题，Volcano将结合监控能力构建基于真实负载的动态调度和重调度。

华为云一直是云原生新技术的探路者、产业新格局的开拓者，2015年华为作为唯一亚洲企业参与云原生计算基金会（CNCF）创建，并一直是CNCF核心项目的主要贡献者，代码贡献稳居亚洲第一。除Volcano项目外，华为云还捐献了首个智能边缘计算项目KubeEdge、首个多云容器编排项目Karmada，完善了CNCF的技术生态；同时，华为云还拥有服务网格顶级开源社区Istio在亚洲的首个指导委员会席位。为进一步推动云原生技术在各行业的落地、构建产业融合新格局，华为云联合中国信通院先后发布了《云原生2.0白皮书》、《数字政府云原生基础设施白皮书》为企业落地云原生提供体系化的理论参考，并与CNCF、中国信通院联合成立了全球云原生交流平台——创原会，为全球企业提供共享、共创、共赢的云原生交流平台，已服务于500+企业的技术管理者。未来，华为云将持续与广大客户一起共建云原生产业生态圈，做深耕数字化的先行者。

云原生安全成熟度模型解读

刘如明
中国信通院云大所云计算部
高级业务主管

杜岚
中国信通院云大所云计算部
工程师

云原生技术已经大范围普及并深入应用到了企业核心系统，但云原生在拥有敏捷、弹性、可迁移等优秀特性的同时，也带来了新的安全风险，镜像漏洞、容器逃逸以及微服务细粒度拆分带来的服务交互安全等问题正威胁着企业的云原生应用，云原生安全建设成为企业云原生平台建设和应用云原生化改造进程中的必备项。中国信息通信研究院（以下简称“中国信通院”）发布的《中国云原生用户调查报告（2021年）》显示，安全性是企业大规模应用云原生技术时的最大顾虑，其中容器安全、API间认证鉴权、微服务入侵检测、代码安全扫描等问题受到重点关注，企业云原生安全建设正在进行中。

在此背景下，中国信通院联合业界20余家单位的近40名专家历时1年完成了国内首个云原生安全成熟度模型标准的编撰工作。目前云原生安全成熟度标准文稿及评估方案已经完成，首批评估工作正在进行中。

一、云原生安全成熟度模型价值

云原生能力成熟度模型（CNMM-TAS）以提升企业研发效能、促进业务创新发展为目标，从技术架构（T）、业务应用（A）、架构安全（S）三个方面助推企业云原生能力建设。云原生安全成熟度模型（CNMM-TAS）融合了零信任、安全左移、持续监测与响应以及可观测四大理念，从基础设施、基础架构、应用服务、研发运营、安全运维等5个层面综合评估企业云原生平台及应用的安全成熟度，帮助企业快速对照、定位安全能力水平，诊断自身问题，根据业务需求结合模型高阶能力定制安全架构演进方向。

云原生安全分级评估方法，助力云原生安全建设

多维度把脉
准确定位企业云原生安全水平

差异化分析
详细诊断企业当前云原生安全能力建设短板

定制化提升
明确输出企业未来能力改进方向和计划

二、云原生安全成熟度评估全貌

评估面向云原生安全防护体系，包括基础设施安全、云原生基础架构安全、云原生应用安全、云原生研发运营安全和云原生安全运维5个能力域、15个能力子项、46个实践项和近400个细分能力要求，具体如图所示：

三、云原生安全成熟度模型解读

（一）基础设施安全域

基础设施安全域是指承载云原生架构的底层基础设施的安全能力，包括计算安全、网络安全和存储安全三大能力子项。

1 计算安全

计算安全是指承载云原生架构的底层算力支撑单元的安全能力，评估包含资源隔离、访问控制、安全加固和攻击防护四个实践项：

» 资源隔离：计算资源的隔离与管理。

» 访问控制：主机资源的访问控制。

» 安全加固：系统的基础安全加固与配置。

» 攻击防护：主机攻击检测与防护。

2 网络安全

网络安全是指承载云原生架构的底层网络通信单元的安全能力，评估包含访问控制、安全通信和网络攻击防护三个实践项：

» 访问控制：基础网络的边界访问控制、网络划分及隔离。

» 安全通信：基础网络通信中的机密性、完整性保护。

» 网络攻击防护：基础网络层面的攻击检测与防御，以抗拒绝服务攻击和网络入侵防护为主。

3 存储安全

存储安全是指承载云原生架构的底层存储单元的安全能力，评估包含数据保护、数据备份恢复和剩余信息保护三个实践项：

» 数据保护：存储数据的访问控制与加密保护。

» 数据备份恢复：充分的数据备份机制与数据恢复能力。

» 剩余信息保护：存储单元被释放或再分配前数据得到完全清除，防止信息泄露。表1

（二）云原生基础架构安全域

云原生基础架构安全域是指云原生PaaS平台的安全能力，包括云原生网络安全、编排及组件安全、镜像安全及容器运行时安全四大能力子项。

云原生安全成熟度模型	能力域	能力子项	实践项				
	基础设施安全域	计算安全	资源隔离	访问控制	安全加固	攻击防护	
		网络安全	访问控制	安全通信	网络攻击防护		
		存储安全	数据保护	数据备份恢复	剩余信息保护		
	云原生基础架构安全域	云原生网络安全	访问控制	安全通信	网络攻击防护		
		编排及组件安全	集群组件安全加固	敏感信息保护	访问控制		
		镜像安全	镜像仓库管理	镜像扫描			
		运行时安全	容器运行时检测	安全策略管理	容器数据信息加密	容器资源隔离限制	
	云原生应用安全域	微服务安全	微服务组件安全	服务安全			
		无服务器安全	无服务器平台安全	无服务器应用安全			
		通用安全	访问控制	安全通信	API安全	攻击防护	
	云原生研发运营安全域	安全需求	安全需求分析				
		开发安全	制品安全	开源安全	安全设计	代码安全	
		测试安全	静态测试	动态测试			
	云原生安全运维	安全管理	资产管理	安全审计	策略管理	身份管理	密码管理
		安全运营	监测预警	响应处置	溯源分析	情报管理	

云原生安全成熟度模型

	计算安全	网络安全	存储安全
L1	计算资源隔离	云基础设施管理流量和业务流量分离	限制平台管理员访问用户业务数据
L2	多租户计算资源管理和隔离 基于用户角色的访问控制 云主机系统软件漏洞扫描 云主机安全配置基线检测 云主机入侵检测	多租户网络隔离 ACL资源访问控制 网络安全组设置 通信传输、边界防护、入侵防范等安全机制	租户间数据隔离 数据存储备份和恢复与完整性校验 虚拟机回收时内存和存储空间完全清除 业务应用数据删除时对应云存储中所有副本删除
L3	漏洞扫描结果分析与修复方案建议 基线检测结果分析与修复方案建议 入侵行为告警和处置建议	网络安全策略设置 VPN加密通讯	身份鉴别信息加密存储 备份策略设置
L4	强身份鉴别措施 威胁评级 自动修复 智能化异常行为检测 入侵行为自动处置 处理情况跟踪	安全策略合规审计、潜在风险检测 云内资源的主动外联网络侧检测与阻断 智能化异常流量检测与防护1	异地备份与恢复 主备备份 云盘加密
L5	云主机自动安全加固和攻击防护	0day、高级可持续威胁攻击防护 攻击防护能力自适应优化	数据操作行为的异常检测 实时动态数据屏蔽

表1 基础设施安全域各等级关键特征说明

1 云原生网络安全

云原生网络环境建议与物理网络或云主机网络进行区分，划分独立的容器网络层，为该层网络提供更为细粒度的访问控制与安全防御措施。评估包含访问控制、安全通信和网络攻击防护三个实践项：

» 访问控制：云原生网络的访问控制与细粒度隔离。

» 安全通信：云原生网络通信中的机密性、完整性保护。

» 网络攻击防护：云原生网络的攻击检测与防御，包括南北向和东西向的网络攻击防护。

2 编排及组件安全

编排及组件安全是指容器编排引擎及其核心组件的安全能力。评估包含集群组件安全加固、敏感信息保护和访问控制三个实践域：

» 集群组件安全加固：集群组件的漏洞修复与安全配置。

» 敏感信息保护：集群组件的敏感信息加密保护。

» 访问控制：集群组件间以及外部对集群的访问控制。

3 镜像安全

镜像安全涵盖镜像自身安全和镜像仓库安全两大方面。评估包含镜像仓库管理和镜像扫描两个实践项：

» 镜像仓库管理：镜像仓库自身的脆弱性加固、攻击防护能力、镜像传输安全与镜像的管理扫描能力。

» 镜像扫描：镜像漏洞、不安全配置以及恶意程序的扫描。

4 运行时安全

运行时安全指对云原生平台上容器运行时的安全防护能力。评估包含容器资源隔离限制、容器数据加密、安全策略管理以及容器运行时检测四个实践项：

» 容器资源隔离限制：容器资源细粒度的隔离及使用权限控制。

» 容器数据加密：容器内关键数据的完整性与机密性保护。

» 安全策略管理：容器\容器内部资源访问以及镜像使用等一系列容器安全策略的配置及管理。

» 容器运行时检测：容器异常行为的检测与审计，包括但不限于容器逃逸、异常连接、敏感操作以及高危系统调用等行为。表2

	云原生网络安全	编排及组件安全	镜像安全	运行时安全
L1	网络架构业务与管理分离	编排组件访问控制	镜像集中管理	容器健康状态监测 资源隔离与使用限制
L2	容器外网访问限制 四层网络流量限制	安全基线扫描 安全漏洞扫描及修复 敏感信息加密 集群组件安全通信	批量化镜像漏洞扫描与修复建议 镜像仓库脆弱性检测与修复 镜像仓库访问控制 镜像传输加密	特权容器与特权行为限制 基于白名单的容器内程序运行控制 容器存储数据全盘加密
L3	七层网络流量限制 容器网络拓扑和流量可视化 攻击阻断	限制外网访问容器编排组件 敏感信息托管保护 集群共享存储内容加密	镜像推拉限制 镜像扫描策略设置 镜像内部配置及恶意程序扫描	容器内恶意文件、异常进程、 高危行为、逃逸攻击检测 容器级别攻击阻断
L4	流量加密 流量审计和流量镜像能力 微隔离的自动生成与告警 自动化攻击阻断	集群编排组件的攻击检测和阻断 安全事件审计	DevOps流程中自动化镜像扫描 分阶段匹配处置手段 镜像风险评估及处置建议除策略 风险镜像跟踪	基于行为分析的容器内异常行为检测 容器内资源级别的攻击阻断 自动化攻击阻断能力 安全容器
L5	全流量威胁检测和智能阻断	集群组件智能安全加固	环境自适应智能化漏洞威胁排序 基于漏洞风险排序结果自动加固	基于可信执行环境的数据保护 实时自动化、多级联动的威胁响应能力

表2 云原生基础架构安全域各等级关键特征说明

（三）云原生应用安全域

云原生应用是指云原生PaaS平台上部署的应用安全防护能力。包括通用安全、微服务安全和无服务器安全三大能力子项。

1 通用安全

通用安全是各种形态的云原生应用的共性安全能力。评估包含访问控制、安全通信、API安全和攻击防护四个实践项：

» 访问控制：细粒度的云原生应用访问控制。

» 安全通信：应用间通信的机密性与完整性保护，以及异常流量监测与阻断。

» API安全：海量API安全管理、检测与防护。

» 攻击防护：包含南北向与东西向的应用攻击防护。

2 微服务安全

微服务安全是微服务自身框架及微服务应用的安全能力。评估包含微服务组件安全和服务安全两个实践项：

» 微服务组件安全：微服务框架中组件的脆弱性扫描与安全加固。

» 服务安全：微服务应用的状态监测、高可用以及攻击检测与响应。

3 无服务器安全

Serverless（无服务器）是一种将基础设施资源抽象成按需使用的服务，用户只需关注应用逻辑，而无需管理复杂的基础设施运维工作的应用模式，当前实现形态以FaaS和BaaS为主。无服务器安全评估包含无服务器平台安全、无服务器应用安全两个实践项：

» 无服务器平台安全：平台账号及资源权限管理、计算单元隔离。

» 无服务器应用安全：资产梳理、攻击检测与响应，特别是资源耗尽型攻击。表3

（四）云原生研发运营安全域

云原生研发运营安全指结合人员管理体系、制度流程，在应用设计研发阶段便引入安全措施，实现安全左移，包括安全需求、开发安全和测试安全三大能力子项。

1 安全需求

安全需求是指开始研发前应具备完整安全策略和开展应用安全需求分析过程。评估包含安全需求分析一个实践项：

» 安全需求分析：梳理安全需求、制定测试方法、对安全需求进行规范化管理。

2 开发安全

开发安全是指应用开发阶段的安全措施。评估包含制品安全、开源管理、代码安全和安全设计四个实践项:

» 制品安全: 应用程序交付物的安全保障措施。

» 开源管理: 对开源软件的安全管理。

» 代码安全: 代码编写规范制订, 代码进行安全检测与修复。

» 安全设计: 标准化的安全设计规范与威胁建模分析

3 测试安全

测试安全是指应用测试阶段的安全测试要求。评估包括静态测试、动态测试两个实践项:

» 静态测试: 对应用的源代码、二进制文件等进行分析。

» 动态测试: 对运行状态的应用进行分析测试。 表4

	通用安全	微服务安全	无服务器安全
L1	用户、服务访问控制	微服务高可用	用户权限控制
L2	基于角色的访问控制 API识别、监测与脆弱性扫描 南北向通信的机密性、完整性保护 南北向Web应用攻击防护	访问控制策略统一管理	函数隔离 函数权限控制
L3	内部应用间细粒度的访问控制 API异常行为检测与响应 东西向通信的机密性、完整性保护 南北向流量分析与异常拦截	服务降级与隔离 微服务组件安全扫描及修复建议	资源监控及告警 滥用攻击检测
L4	应用微隔离策略 东西向应用攻击防护 东西向流量分析与异常拦截 基于行为分析的API异常行为检测与响应	服务拓扑可视化 异常行为检测与响应	资产业务逻辑梳理 异常行为检测与响应
L5	智能化攻击检测与响应	微服务组件的自适应安全加固 智能化攻击检测与响应	智能化攻击检测与响应

表3 云原生应用安全域各等级关键特征说明

	安全需求	开发安全	测试安全
L1	安全基本需求清单, 产品基本安全能力需求	开源软件使用规范 项目级安全编码规范 安全设计规范	手动安全测试
L2	更全面的需求清单 有相应的安全测试用例 针对应用场景特点制定安全需求与用例	制品安全检查 开源组件自动化安全扫描 团队级安全编码规范 安全功能标准化设计	明确的安全测试要求与测试用例 安全测试和合规扫描
L3	安全需求与其他功能性需求同步开展测试 安全需求包括功能性需求和非功能性需求	制品可追溯性 建立标准化、安全的技术栈 在安全设计中进行威胁建模分析	安全测试流程和规范 端到端的测试工具链 流水线自动集成安全测试、生成测试结果
L4	自动化安全需求管理平台	制品自动化检查、清点 插件自动化安全检测能力 支持标准化的安全功能组件	人工渗透测试流程
L5	智能化安全需求管理平台 安全需求自动化验证能力	开源组件修复 编码工具安全问题自动化识别和修复 智能化威胁建模并输出安全设计方案	安全测试智能化并内嵌到开发与交付过程

表4 云原生研发运营安全域各等级关键特征说明

（五）云原生安全运维域

云原生安全运维域是指对云原生平台与应用各层级对象统一的安全管理，以及全局覆盖和多点联动的安全运营能力。包括安全管理和安全运营两大能力子项。

1 安全管理

安全管理是指对基础设施、云原生基础架构以及云原生应用统一的资产\策略\权限管理以及安全审计。评估包含资产管理、安全审计、策略管理、身份管理和密码管理五个实践项：

» 资产管理：云原生资产的识别与管理，各类资源层级及拓扑关系的可视化。

» 安全审计：云原生日志采集与安全审计。

» 策略管理：云原生安全策略统一配置管理。

» 身份管理：云原生平台统一身份和权限管理。

» 密码管理：平台中各类凭据、密钥、证书等关键信息的统一管理。

2 安全运营

安全运营是指覆盖云原生全流程全要素的安全风险监测、统一分析以及联动响应能力。评估包含监测预警、响应处置、溯源分析和情报管理四个实践项：

» 监测预警：云原生平台统一的威胁检测、告警和可视化。

» 响应处置：云原生平台统一联动的威胁响应处置。

» 溯源分析：云原生平台综合的攻击溯源分析。

» 情报管理：获取、处置和分析安全情报信息。表5

	安全管理	安全运营
L1	节点、容器级资产可视化	独立监测 手动处置
L2	云原生多层级资源可视化与查询 云原生平台日志采集和安全审计	统一监测 多设备联动的威胁响应 人工溯源分析 漏洞、IOC情报实时获取
L3	日志分析、安全行为审计 应用开发测试阶段策略统一配置管理 容器安全策略统一配置管理 网络安全策略统一配置管理	监测及处置进展可视化 安全事件情报自动化获取 情报处置
L4	网络拓扑可视化 服务访问关系可视化 资产的关联显示与查询 多账号统一身份管理与访问控制 身份滥用和异常身份操作监控告警 支持第三方密码管理系统	基于威胁情报的实时威胁检测与告警 智能化威胁检测能力 基于预设策略的自动响应能力 攻击路径自动化构建
L5	混合多云资产实时识别、关联关系智能化分析 研发运营全流程自动化安全审计	攻击诱捕 攻击者画像、定位、部分攻击反制 情报自主发现和分析预警

表5 云原生安全运维域各等级关键特征说明

DevSecOps将成为企业构建安全软件体系的利器

马景贺
极狐（Gitlab）DevOps技术布道师
LFAPAC开源布道师

安全是企业的生命线，企业的发展必须建立在安全的前提之下，在数智化飞速发展的现代社会，安全受到再多的重视也不过分。安全之所以如此重要的原因是安全就等于金钱。安全问题一定会直接或者间接为起来带来经济损失，诸如近些年频发的软件勒索，就是因为企业的安全防护体系不够完善，最后被攻击者击破防御网，从而盗取了大量企业核心数据，只有在交付高额（甚至达千万美元）赎金之后才有可能要回数据。更加麻烦的一点是如果盗取的数据与客户相关，那么就面临失去客户的风险。因此，对于安全的重视与投入应该伴随企业发展的全过程。让安全促进企业发展，用发展来加固企业安全。

然而，企业内安全的落地实践却充满挑战，首先就是安全的主题难以界定。在传统的软件研发模式中，开发、运维、安全的界限比较清楚，软件的复杂度相对现在而言低很多，交付周期也较长，在这种模式下靠专门的安全团队、专业的安全人员去保证软件的安全是没有太大问题的。但是，随着软件规模的爆炸、交付的加速、复杂度的提升，传统的安全保障模式受到了挑战，那在这种情况下，到底谁是安全的第一责任人呢？安全？开发？还是运维或者测试人员呢？安全主体的界定模式就会导致推诿与甩锅，出了安全问题很难去快速找到相应的负责人进行修复；

其次就是ROI不清晰，通常研发或者测试的工作，可以基于工作量、产出来衡量，而安全的工作结果却很难直接呈现，投入产出也不好衡量，这就导致企业内安全人员的职业发展要比其它岗位更难，岗位很重要但是不受重视；再加上安全内容很碎片化，如用户密码属于安全，配置管理属于安全，容器镜像也涉及安全，引入的开源软件也涉及安全问题，这些繁而杂的问题导致安全人员的工作很难统一，再加上互联网时代，企业追求业务快速上线、快速抢占市场、快速赢利，常常出现安全让位业务，导致安全工作容易被忽视，员工也就自然不愿意承担安全的工作。长此以往，安全人员比较容易离开公司，安全防护体系受到影响。

再者就是企业在安全方面的短视，安全是一个长期的、持续的工作，需要真正的专业人士，甚至大量资金投入。但是企业往往会因为一个专业安全人员价格过高、一个商业的安全产品价格过高而选择忽略放弃安全，原因很简单：全球这么多软件，我怎么可能是那个被攻击的倒霉蛋。这恰恰表现了企业存在的侥幸心理。侥幸心理是安全的最大障碍。

在企业数字化转型加速的今天，作为企业信息化、数字化核心的软件系统，其安全问题再一次回到了大家必须要讨论实践的面前。安全问题伴随着软件的诞生而诞生，是无法消除的。但是只要通过合理的流程、有效的手段，就能够避免一些安全问题，而这恰恰是 DevSecOps 所要做的事情。

从DevOps到DevSecOps，软件开发经历了什么?

DevOps 的诞生是为了让 Dev 和 Ops 之间的壁垒能够被打破，从而加速软件的交付。但是随着如今企业上云进程的加速、开源软件采用率的提升、软件交付周期的缩短（从以年度、月度为交付周期变为以周甚至天为单位的交付），在软件开发生命周期（从研发到交付上线的整个过程）不会有任何变更的前提下，留给安全的时间越来越短，因此必须要在 DevOps 中有效的融入 Security，以便在保证软件交付速度的前提下，还能不丢失安全性，这也是 DevSecOps 的真正由来。

看似从 DevOps 到 DevSecOps，只是多了一个 Sec，但是一字之变，内容却是千变的。DevSecOps 有三个重要特征：

1 安全左移

其实这个和软件开发生命周期有关，一般软件开发生命周期，包含计划、编码、运维、部署上线等，这是一整个SDL。在 DevSecOps 模式下，安全都需要尽量左移，不要在测试阶段才开始介入，尽可能在plan阶段就介入。那为什么是左移而不是右移呢？其实还是钱的问题，安全和钱挂钩。因为越早发现安全问题，修复问题所花成本越低，有数据显示，如果在研发阶段发现了一个安全问题，修复它的成本是一美分；到测试阶段发现一个安全问题，修复它的成本到了十美分；而到上线的时候，再去修复这个安全漏洞，同样的安全漏洞修复成本是100美分，相当于扩大了100倍。所以左移是为了见可能早的发现问题，从而以更低的成本来修复问题。此外在方法上，通过对比使用DevSecOps前后修复漏洞所需的时间，二次相差50%左右，效率提升明显。

2 持续自动化

研发人员提交代码后，往往需要调用一些安全服务进行扫描、检测，如今可以通过CI/CD把原来需要手动做的安全检测集成到整个研发流水线中，实现持续自动化检测，一旦扫描出漏洞，还可以自动反馈，结合流程工具自动化跟踪，直至问题闭环、版本正式发布，从而减少人工操作，提升整体研发效能。

3 人人为安全

DevSecOps 模式下，任何与软件研发有关的人员都是安全的主体，因为每个人都可能成为安全问题的瓶颈点。研发、测试、运维、安全、甚至产品、销售以及市场都应该为整个软件的安全负责人，所有人一起为安全负责，才有可能构建起整个软件安全，加固企业的安全防护体系。

当然，仅仅通过口号是无法落地 DevSecOps 的，需要制定适宜的流程（大家都任何）、选择合适的工具（非最贵、最好）、多团队协同（大家一起干）来共同实践落地 DevSecOps。

在 DevSecOps 中有一个常见的误区，用安全工具扫描出安全问题就算是实现安全了。其实安全难的一点不在于发现问题，难点在于如何去解决问题，在这一点上，首先要做到多种防护手段（SCA、SAST、DAST、Pen Testing 等）的安全漏洞报告要能够统一展示，而且是透明的，软件研发相关的人员

都可以查看到，这样才能让研发、测试和安全团队能即时了解、有效配合、效能化，更好、更快地修复漏洞；接着还要为这些安全漏洞建立追踪机制，其次要明确追踪、修复责任人、修复方案、修复时间等关键信息。

DevSecOps实践：基于Kubernetes构建纵深防御

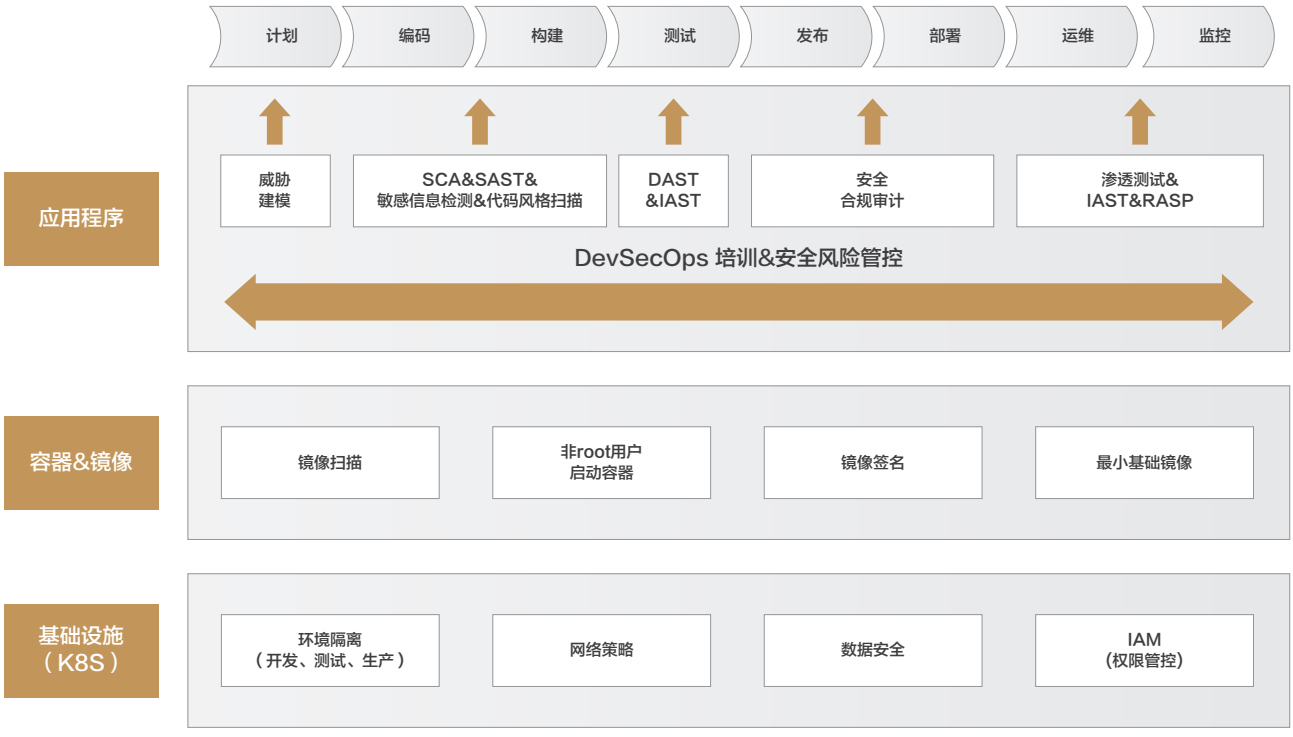
云原生是时下非常火热的技术手段，很多企业在向云原生转型，而云原生本身的复杂性让 DevSecOps 的落地显得有些吃力。对此，需要构建立体的纵深防御体系。如图所示：

这种纵深的安全防御体系有纵深，有横阔。

纵深方面从基础设施做起（主要针对 Kubernetes），需要做到环境的隔离，网络的策略安全、数据的存储安全以及对于基础设施的访问控制等；

再上一层是容器和镜像安全。如果说容器是云原生时代的核心，那么镜像就是容器的灵魂，需要通过“预防为主、防治结合”的手段来实践容器和镜像安全。所谓预防，是指在选择使用基础镜像时选择安全的镜像，在编写 Dockerfile 的时候遵循一定的正模式（非root用户启动容器，选择满足功能的最小的基础镜像等）；所谓治，是指要对容器镜像进行安全扫描，在发现安全问题的时候，及时进行修复。当然，最上层也是大家非常关注的应用程序安全，这是一个复杂的安全防护，需要提供多种安全防护能力，覆盖软件开发的全生命周期，完成软件从静态到动态，从编码到上线全过程的安全防护能力。

构建安全的企业业务体系需要多方联动、多措并举，基于DevSecOps企业可以更高效的协同多方力量共同保证软件安全、软件供应链的安全。



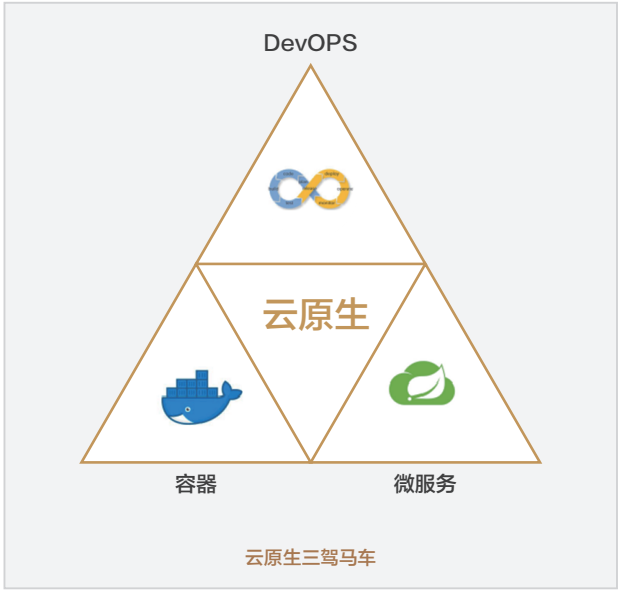
基于Kubernetes构建的安全纵深防御

重识云原生
不谋全局不足以谋一域

黄俊
招商证券核心交易系统容器化改造项目
技术经理，云原生转型项目调研负责人

“三驾马车”奠定云原生体系核心

云原生（Cloud Native）的概念最早是由Pivotal公司的Matt Stine于2013年提出，并在《迁移到云原生架构》一书中总结为——12因素、微服务、自敏捷架构、基于API协作、扛脆弱性等。而云原生计算基金会（CNCF）成立后，将云原生代表技术定义为容器、服务网格、微服务、不可变基础设施和声明式API。企业数字化转型实践角度出发，依然将云原生体系（而不只是云原生技术架构）的定义浓缩为传统三驾马车——DevOps+微服务+容器，这样既能囊括技术解决方案，也能包含企业研发管理建设思路，使其能真正成为解锁企业数字化的“他山之石”。



云原生三驾马车

从云计算发展再来重新认识云原生

云计算技术从1999年VMware公司推出商业虚拟化软件VMaware Workstation起，到现在，仅仅二十多年时间，便已历经虚拟化、公有云、云原生三代。以往云计算对企业IT的支撑仅仅停留在IT资源的运维管理层面，对企业研发管理变革的影响力有限。而云原生体系，基于资源全面虚拟化、应用运行高度标准化、业务研发全线拉通，以IaaS、PaaS、SaaS的高要求高收益反向推动企业研发管理变革，从而全面驱动数字化转型车轮快速向前运转。

1 云+容器质变性推进软硬运行环境的标准化

当前第三代的云计算技术，首先是基于第二代公有云的大规模实践发展而来，即便沉淀到私有云领域，基本沿用公有云架构再做的To B能力下沉，其在IaaS层基础资源管理方面也充分体现了公有云的特点。

首先便是基础硬件标准化思路的变革——第二代云对于基础硬件的标准化思路与第一代云截然相反。第一代云计算技术企图以软件虚拟化技术去全面兼容市面参差不齐的异构PC机，也正由于在普适兼容方面投入过多，导致性能损耗过大，而第二代云则在此方面做了减法，其基于大规模公有云实施经验，反向要求入云纳管的硬件设备，特别是计算服务器首先必须是标准化的（甚至需根据云厂商要求进行定制），要求满足云厂商的硬件兼容标准，由此来大幅降低硬件兼容适配范围，再进行针对性调优、推动虚拟化核心技术沉入OS内核、推动硬件辅助虚拟化技术发展，以此来大幅降低虚拟化性能损耗。目前头部厂商均宣称其基于KVM的虚拟机技术，性能损耗均已能降低到5%以内（优秀者可以到3%以内）。可预见，全行业级别的云基础硬件生产标准必然在不远的将来出台（国家主导成立的相关云计算产业协会来牵头），以此逐级实现上游全产业链的深度协同标准化生产。

第二点，随着AWS的Nitro硬件卡、阿里神龙Moc卡等智能网卡技术的出现，结合英特尔的DPDK/SPDK技术，服务器上部分网络或存储处理指令可以移到用户态处理、且将流量直接卸载到智能网卡中，此方案能大幅提升云上网络传输性能、全云存储性能（陆续出来的还有腾讯云的黑石、华为云的擎天卡等），应用此类智能卸载技术在特定云上领域（例如高性能计算、云存储、裸机容器、零信任安全保护等）将拥有无可比拟的性能优势与更体系化的软件定义能力。凭借此类革命性技术优势，定制版智能网卡、定制版服务器的市场占比必将不断提升，从而进一步提升云厂商在云服务器领域的标准话语权。

第三点，第一代云计算技术的资源全局调度能力、弹性伸缩能力，均是依托单机虚拟化技术（不论是VMware的软件虚拟化，亦或基于Intel VT-x/AMD-v的硬件辅助虚拟化技术）来实现，缺乏数据中心级的大规模自动化调度能力。而从第二代云计算技术开始，特别是SDN技术的出现使得软硬协同一体化的数据中心级(甚至跨地域级)全云资源调度管理成为常态，此举必然大举推动企业网络环境的标准化进程。

第四点，传统云下容器运行模式是宿主机+虚拟机+容器的三级运行模式，一般就意味着容器网络传输需经过三级转换，故传输性能不如人意，难以大规模商用。而容器运行在云上，通过云网络技术（例如VxLan、SDN）可将容器网络平面与云上(Overlay)网络平面拉平，再将流量卸载到智能网卡上加速，既方便做灵活的统一网络规划与安全策略控制,也能减少一次网络转换、加速容器网络性能。

当前开源容器网络解决方案如flannel、calico等，均是基于过往Internet模式下海量分散单机寻址路由的思路进行设计，且仅仅站在容器层来设计网络方案，故而不论性能还是实际落地代价均不理想。而云上容器的网络解决方案，从调研来看，均是直接融合了云端网络虚拟化能力，将容器网络与Overlay网络拉平，依托VPC、智能网卡等先进技术来简化传输链路、提升性能，且方案出发点便是以全数据中心云上云下网络一体化拉通为落地目标，故不论方案深度还是可落地性均非开源纯容器网络方案可望其项背。这也是笔者坚持“不能脱离云单以容器来论云原生”的另一关键原因!

2 云+容器+微服务，拔高基础设施上限至PaaS层

依托云平台的资源软件可定义能力，搭建统一大集群规格、可弹性伸缩的PaaS服务，可为传统企业瞬时提供能比肩一线互联网企业的高性能、高可靠性的中间件服务与应用开发框架能力，并大力推动企业内开发框架标准化进程。

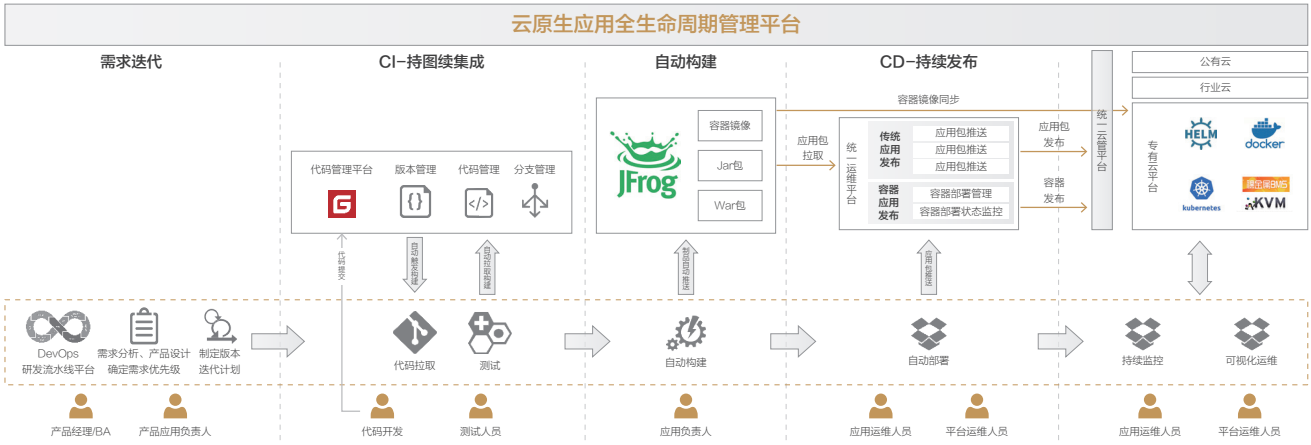
微服务的核心思想便是能力拆分复用、功能解耦，其与容器是天然一对——服务的并发弹性支撑可通过容器的水平伸缩能力来匹配实现、服务的高可靠可通过容器的自动恢

复特性来自动化满足。同时，基于应用服务的全面容器化之后，应用开发框架中的大部分非业务相关的框架性能力便能与业务应用逐步解耦，以SideCar方式伴生运行——此举既能实现应用框架独立于业务应用的自主升级，也能使业务开发团队更加聚焦于业务功能开发，同时为跨系统、跨技术栈的真正全链路服务追踪、全局服务治理提供技术落地可能性，这便是ServiceMesh。

而即便不用容器技术，仅基于云+微服务使用场景，应用依赖的中间件服务（例如常见的Redis、MQ等）也能以云端标准PaaS服务的方式提供，并支持大规模弹性、分租户申请、实例级隔离使用，为业务应用的性能提升、稳定性提升提供统一、规模化、专业化技术服务保障。故而，在云原生时代，PaaS服务也已成为云端基础设施的一部分。

在云原生时代，PaaS服务也成为了云端基础设施，在短时间内大幅提升传统企业系统的运行性能、稳定性的同时，也能大幅降低企业应用的中间件建设成本、开发框架的维护成本、架构人员能力要求与精力投入。同时，借助业务上云，也能轻松实现应用开发框架收敛、中间件服务标准化，将其版本迭代管理、能力演进节奏都依托云端实现，其企业架构治理前景也是非常可观的。

3 云+容器+DevOps，依托运行标准化，降低流水线对接成本，真正实现产研运全流程自动化



云原生产品生命周期研发体系

基于云+容器来进行DevOps敏捷化运作，其核心收益点在于——容器依赖自包含特性带来的CD环节（持续发布）的对接改造成本的大幅降低、基础运行环境大幅软件可定义之后系统资源的自动化申请与创建使得研发全流程自动化有了真正落地可能。

DevOps这个概念并不新鲜，但是在云原生时代以前，除了一线互联网企业，鲜有成功案例，其一大原因在于应用依赖环境的繁杂，特别传统企业，历史遗留系统多、技术栈庞杂、框架版本碎片化严重，若要在CD环境对接改造所有系统的各类运行环境，对接成本高、技术复杂度高、发布后监控反馈环节对接难度大，最终导致改造后系统运行风险反而不可控。故而只能在Java等少数“先进且标准”、对接成本更低的领域试点，全面推广几无可能。而互联网企业恰恰因为历史包袱少、技术栈相对单一，便有了成功运作的相对标准化技术环境基础。

而容器技术出现后，业务应用统一打包成依赖自包含的容器镜像，对运行环境的依赖程度进一步降低，DevOps的流水线系统只需要对接云上统一的容器发布系统，即可轻松实现各类技术栈应用的自动化发布，再依托云平台全栈的容器运行状态监控机制、ServiceMesh流量劫持等能力，在应用发布后状态监控环节，也有了可低成本实现的标准化解决方案。整体自动化发布方案也更加立体，更具备生产落地可行性。

第二，在传统云下环境，运行资源调度、网络策略下发、安全策略配置等均是相互割裂，相关配置工作多为手工操作，故即便在编码环节能做到持续集成、自动化构建，应用包也能自动化推送与部署，但中间的应用配置变更依然是手工操作，终究不能真正实现全流程线上化、自动化，也就依然无法实现快速迭代发布。而在第二代云计算技术之后，有赖云上计算、存储、网络、安全等资源的软件可定义，程序运行所需配置信息通过调用云端标准接口即可开通或下发，使CI/CD各节点均可线上化打通实现，全流程自动化发布才真正成为可能。

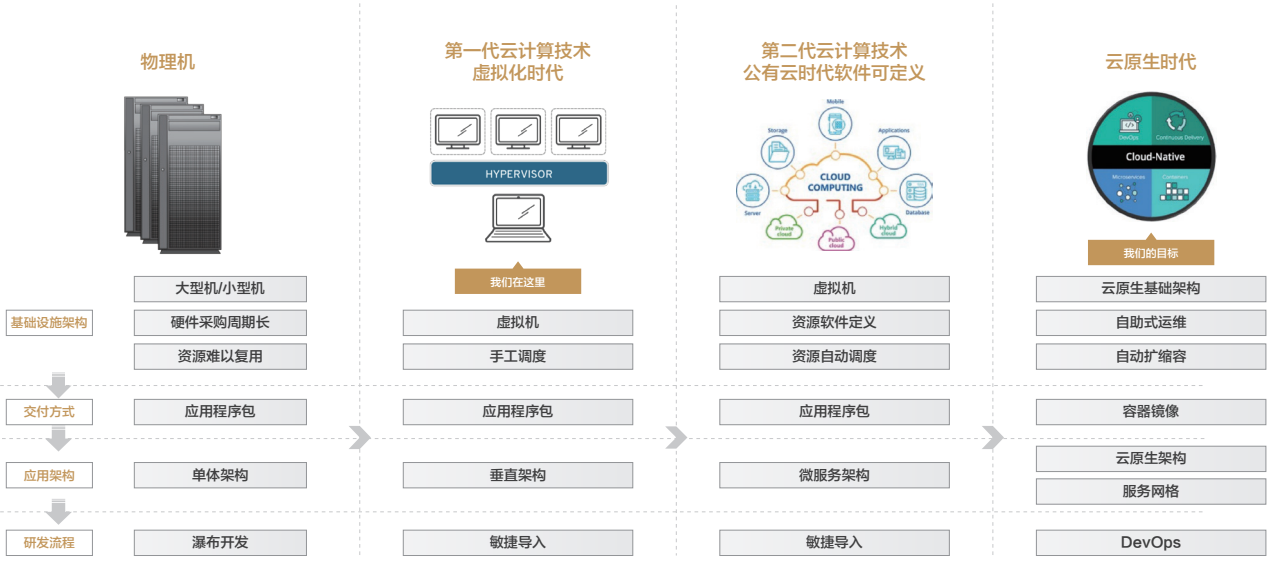
第三点，在ServiceMesh出现以前，大部分应用运行保障相关的框架级非功能特性（如弹性、韧性、安全、可观测性、灰度等）均依赖不同技术栈的应用开发框架能力来保证，甚至如在开发框架上述能力缺失的情况下，还需要业务开发团队自建，这就导致了此类特性涉及的运维自动化、运行监控等工具能力也需要重复建设多套，很难做标准统一的同规格高质量建设，因此即便前面CI/CD做到了全流程自动化，自动化发布后也不敢在生产无值守运行。这一点在容器环境，伴随着ServiceMesh技术的出现，才真正有了标准化解法。

云原生体系迭代加速业务敏捷开发

在云原生时代，通过云上环境的软件可定义能力、标准化能力，已经可以使大量研发管理操作、运维监控操作由过去的手工处理变为自动化、标准化的系统动作，从而整体性提升企业IT研发运维自动化水平、质变性跃升软件开发架构能力与PaaS服务稳定性，最终加快业务研发敏捷迭代速度，是一场彻底的科技引领业务的转型变革。

一方面云上应用开发框架、PaaS服务充分结合了云的特性，其持续迭代、运营运维的能力要求，对平台框架开发人员、业务团队架构师的能力要求变得更高了，以往上述人员只需要精于软件开发框架，而云原生体系下，还需要懂云、懂容器，技术能力需要往全栈方向延展；另一方面，传统IT部门中研发与运维是分离的，部门并行设立、工作独立开展，而上云之后，因敏捷迭代要求，开发与运维的协作需要更加紧密，不论是平台开发框架、PaaS服务、业务系统，均需要以产品化思路来运作，这就要求管理者、架构师、业务人员、开发人员、运维人员、运营人员均需要从过往工具化软件开发思维提升到产品化研发思维中来，并形成配套的产品型研发团队，即便业务部门、研发部门与运维部门依然并行设立，但“人”与“事”也要开始做分离，以“事”聚“人”，如此才能适应云上快速迭代要求，业务也才能真正“敏”起来。

总的来说，云原生时代的三驾马车，不仅是针对IT的技术侧能力表述，还蕴含了业务上云后的研发全生命周期管理含义与研发团队组织思想变革。云原生体系是企业管理模式向互联网模式进化的核心关键技术与管理思想、组织流程的深度融合，应该也能成为实现国家“互联网+“战略的一条重要路径，也必将大幅推动中国乃至世界企业的互联网化进程！



云技术变革逐层驱动企业研发与管理模式的变革

采用Node Proxy提升服务网格转发性能的技术实践

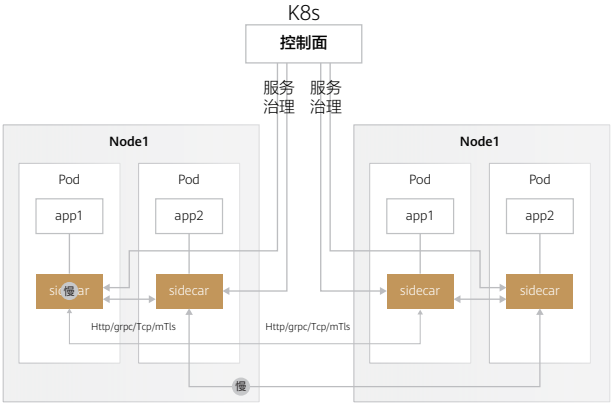
张伟
华为云容器网格数据面技术专家

在云原生场景下，微服务应用的规模和调用复杂度快速增长，Kubernetes是作为云原生应用的标准标准编排平台，提供了微服务运行调度、资源管理的底座，如何在持续运行阶段高效的治理微服务、降低运维成本，成为云原生技术演进的重要问题之一。

ServiceMesh（服务网格）技术将微服务架构下各应用中非功能性的服务治理逻辑从业务进程剥离到独立的边车进程（Sidecar）中，以无侵入的方式提供服务间的连接、安全、流控、灰度发布、观测能力，实现业务轻量化和服务治理基础设施化，基于传统IP网络之上的应用网络技术，服务之间不再是直接基于IP地址的发现与路由，而是基于服务的元数据信息，如：服务名、版本等，正因为如此，Service Mesh 也成为下一代微服务技术的代名词。

然而ServiceMesh的这种方式导致数据面转发路径增加，对于金融及电商这类对于延时很敏感的业务会带来较大影响，以Istio项目为例，其数据面采用Envoy作为Sidecar，支持热重启、动态配置、插件结构、且具备完善的服务治理、流量控制和可观测能力，相对于原有服务调用或传统基于SDK模式的微服务治理框架（SpringCloud/Dubbo），在数据转发面额外在节点内增加了两跳。

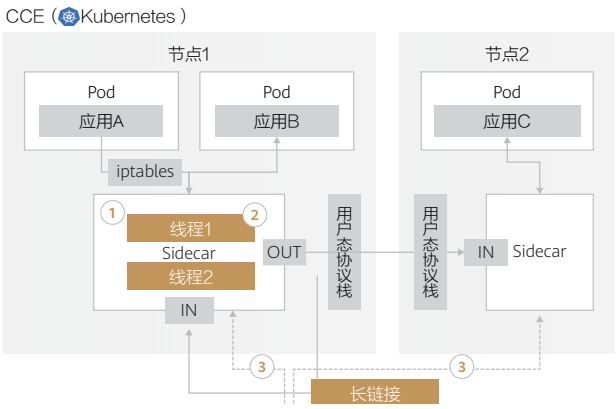
同时，Istio每个Pod内自动注入一个数据面代理，单个Pod内应用容器的数据面代理通常将占用50~70MB内存空间，并且



Sidecar的内存占用还将随着集群内服务数目与实例数目的规模快速增长；经验上，当集群内Pod规模大于50W实例时，除非引入命名空间隔离等限制，否则集群中实例的动态变化将对系统的稳定性造成较大冲击。

另一方面，动态Sidecar代理无法有效的与物理CPU内存资源进行绑核操作，造成系统线程调度上下文切换带来较大额外开销，造成节点内用户购买资源的利用率较低。

为解决Istio存在的以上问题，华为云在基于Istio开发商用的应用服务网格（ASM）产品时对其作了优化，实现了Node Proxy模式，简单来说将分散在每个Pod内的Sidecar集中到各个节点上，实现主备、多活的高可用，每个Sidecar将负责本节点内所有业务请求的出方向和入方向流量。



这样，首先将极大降低非业务容器额外的系统内存占用。另外，Sidecar的生命周期与Pod进行分离，可预先对Sidecar进行CPU绑核操作，一方面降低系统线程调度上下文切换开销，另一方面保证Sidecar的CPU处理能力，避免当系统内业务量非常大时，业务Pod抢占了Sidecar的CPU资源，使得Sidecar转发效率下降形成瓶颈，反过来影响整个业务的转发能力。

为了达到这个目标，并不是简单将Sidecar从业务容器内移动到节点内启动就可以的。还需要进行多处改造：

» 删除原始Pod内Sidecar模式的iptables流量拦截策略，在主机侧iptables内针对出方向且目标服务为clusterip的地址进行拦截，支持vpc模式及trunk port组网模型下的微服务透明代理的同时，保证非微服务流量不受干扰。另外，由于需要支持高可用，节点iptables拦截规则基于权重将新连接指向某个节点内当前可用的Sidecar中；

» Istio内webhook过程将不再自动注入Sidecar到用户业务容器中；

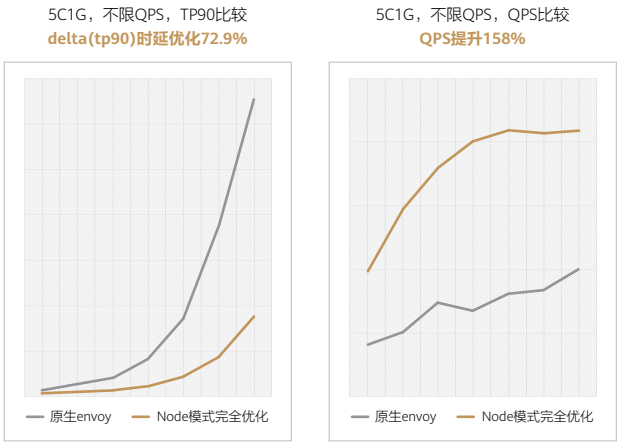
» 对比原始Pod内Sidecar模式下，每个Sidecar启动时已经确定了其固定身份为业务容器，这样实现简单，且后续由Sidecar代理透明拦截的流量都可以直接被当做以其宿主业务容器的身份发出的，并且在Sidecar整个生命周期内不会改变。但在Node Proxy模式下，Sidecar与用户业务容器生命周期不同，将在运行中动态的决定请求的身份，用于进行SSL身份证书的验证及后续观测数据及调用链追踪数据的上报，因此Node Proxy模式也对这些功能点进行了较大部分的代码适配。

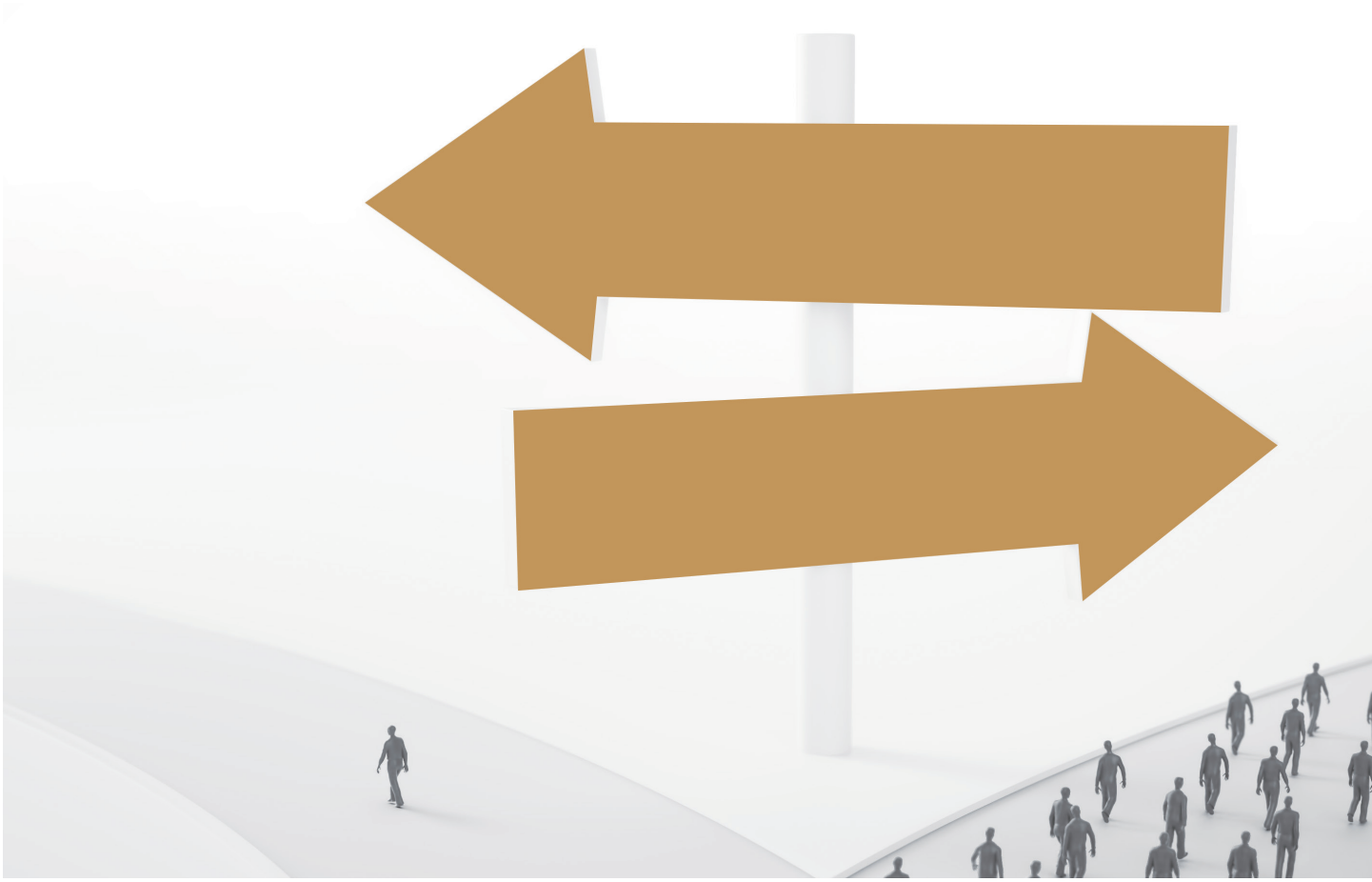
» 透明服务网格代理在实现应用服务治理灵活的同时，由于引入额外两跳导致转发性能的快速下降，并且由于Sidecar本身线程模型对每条新连接处理的不均衡性，导致虽然在系统CPU资源满足的情况下增加工作线程数量，但整体tp90和Qps提升效果不明显。在这一点上，我们通过增强Sidecar本身线程模型对新连接处理的均衡性、提升新连接网络事件处理优先级、降低新连接接收线程与处理线程间锁冲突等手段使得在多核多工作线程配置下比原始方式更均衡的处理每线程连接，对于Http1.x等协议的性能提升非常明显。尤其在Node Proxy模式下，可以对集中代理配置多线程更好的达到线程均衡下对系统内所有业务Pod的转发性能。

» 在性能提升的其他手段上，访问日志的记录及调用链整理上报等非关键流量治理模块对于端到端时延的占用通常达到15%以上，并且对于金融、游戏等某些对于时延敏感的客户来说，其已经拥有一些物理或四层流量监控系统只需要服务网格的流量治理能力。因此，可以在Sidecar运行中，动态开关访问日志、调用链监控等模块来降低端到端时延及整体吞吐是有价值的，但原始Sidecar不支持此动态针对访问日志及调用链的底层开关。针对这个问题，华为增强了runtime部分控制能力，使得支持在Sidecar运行中，不重启的前提下可以动态关闭访问日志及调用量，换取性能部分的提升。

» 对于欧拉4.19新内核的操作系统，节点内Pod访问可以通过Ebpf短接技术绕过操作系统协议栈得到更好的性能，经过内部POC测试，在保持现有Sidecar透明代理功能不变的情况下，Http1可以得到14%左右的端到端提升。

经测试，采用NodeProxy前后，TP 90时延优化72.9%， QPS提升158%，如下图所示：





数据分析平台演进探索

——从三大市场需求目标看数据分析演进方向

李昆
华为云EI软件总工程师

陈杰
华为云大数据MKT专家

关于大数据业务的市场需求变化

日前，工业和信息化部发布《“十四五”大数据产业发展规划》，提出到2025年，我国大数据产业测算规模突破3万亿元，并强调坚持数据要素观，以释放数据要素价值为导向，从国家政策层面为推动大数据产业高质量发展提供指导。

作为领先的云服务厂商，华为云在为客户提供大数据等技术时发现，用户对数据分析平台的需求有了新的变化，诸如“算法以及机器学习工程师花费了很多的时间在特征处理和数据预处理上，期望数据和AI作业开发编排能够有机结合在一起，进行作业流的编排衔接和数据版本控制”、“市场热点实时在变，搜索推荐模型每天重新训练，AI开发者需要将机器学习算子作为数据开发的一个节点统一编排训练”、“AI和数据需要高速流转，达到低时延实时推理，支持分析的时效性”、“AI开发者在开发和优化机器学习模型时，无法直接从数据湖中1000多个制程参数，灵活快速挑选适合的字段进行建模，必须配备懂大数据技术和制造业务的数据工程师来帮忙准备优质的数据”等。

同时，作为数据分析平台的需求方（华为流程IT和消费者的大数据已全面上云），华为数据治理在实现数据实时可现（例如通过报表描述发生什么）的基础上，亟需进入下个阶段，达到诊断预警（例如提前感知业务问题，自动预警风险）和智能决策（例如分析问题根因，推荐方案），以应对日益复杂的内外部环境，提升企业的韧性。

出现上述变化的本质原因是：数据价值的充分发挥需要AI的加持，而AI模型的精度依赖大量高质量的数据，这两者的技术需要有机结合。

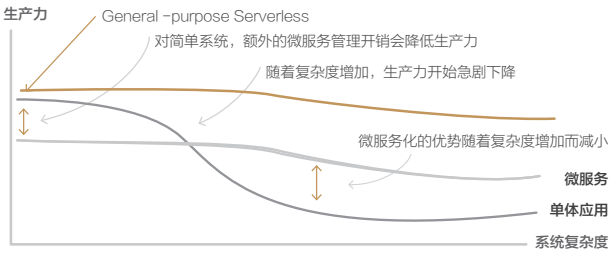
数据分析平台演进探索

政企面向未来数字化、智能化转型升级，需要以云原生的思维构建数智融合的数据分析平台架构，把原本散落在各个部门和组织的数据统一汇聚到数据湖中，省去开发者关注各种底层的琐碎文件管理，以及大量、复杂的分析引擎、AI引擎和管理运维工作，支持开发者在集成的开发平台上，便捷地使用最新的算法模型挖掘各种数据的潜在价值。概括来说，未来演进要满足以下三个目标：

1 降低成本。

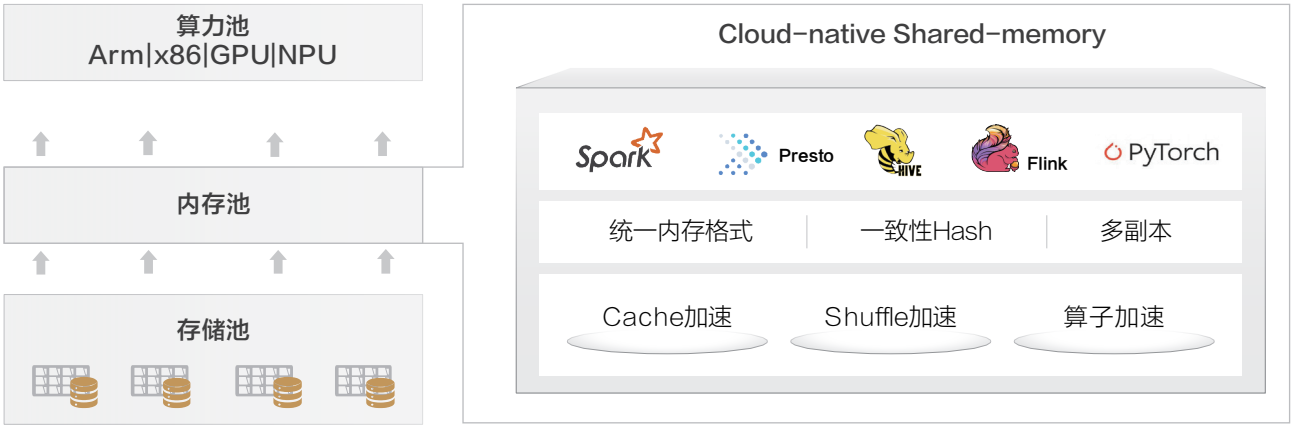
用云原生技术降低存储和处理大规模数据的成本，减少运维开销。

平台基础设施和能力的全面云原生化、轻量化、Serverless，是云原生的演进形态。Serverless技术本身会从专用走向通用：支持有状态、程序能够自动并行、可以在多云执行、高效利用云原生的计算和存储，能够让所有的应用都可以基于通用Serverless开发。



Serverless发展从专门到通用

存算分离大幅降低了数据的长期持有成本，提升了扩缩容的弹性，但是被拉远的计算服务层和存储服务层间的“数据墙”会由此引发性能损耗。因此，需要基于“内存中心架构”推动公有云大数据架构演进，在保持成本和弹性优势的同时，减少数据搬运/拷贝工作、提升性能和故障解耦能力，即CPU和内存的故障不相互影响的能力。



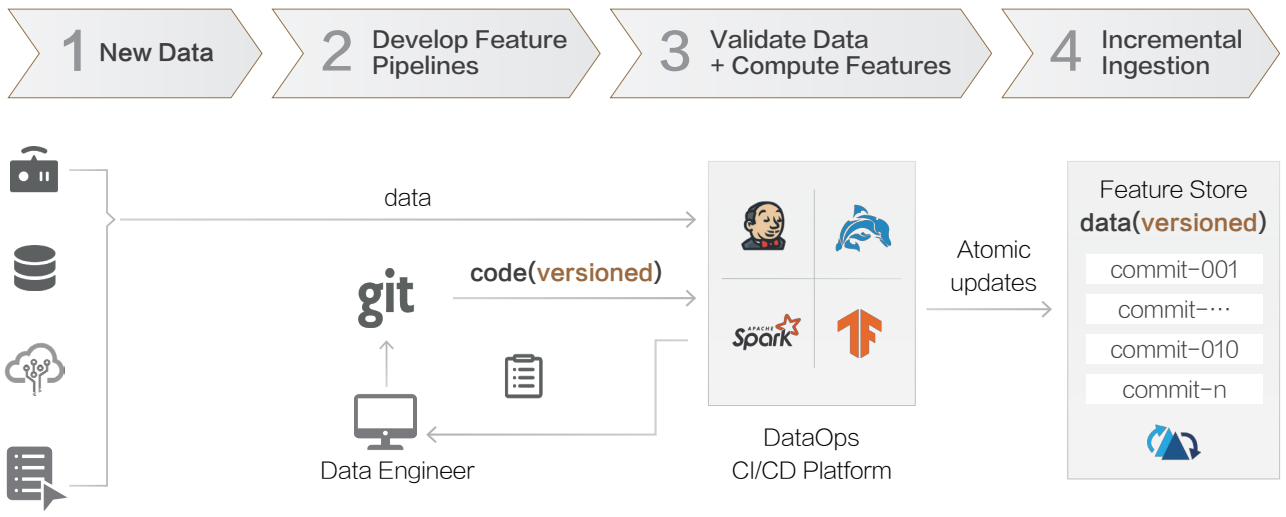
云原生共享内存

2 提升价值。

融合机器学习技术，让用户可以从数据中回答更多的问题、做更好的决策。

首先，要确保企业基于唯一的事实来源进行分析，通过对大数据、数仓、AI等各种分析引擎对元数据的统一管理，解决传统数据分析和AI模型之间“数据搬家”的问题，实现数据在不同引擎间的自由流动，以及权限的细粒度管理和版本管理，打通大数据分析和AI模型引擎，基于一份数据进行不同的分析，避免不同团队基于不同数据分析造成结果的不一致，提升数据驱动决策的准确性和可信性。

其次，要让数据分析师可以便捷地进行模型、特征训练，极大释放数据的潜在价值，让DataOps和MLOps互通，像管理代码一样管理数据，实现数据与AI开发高效无缝互通。



某互联网客户DataOps & MLOps实践

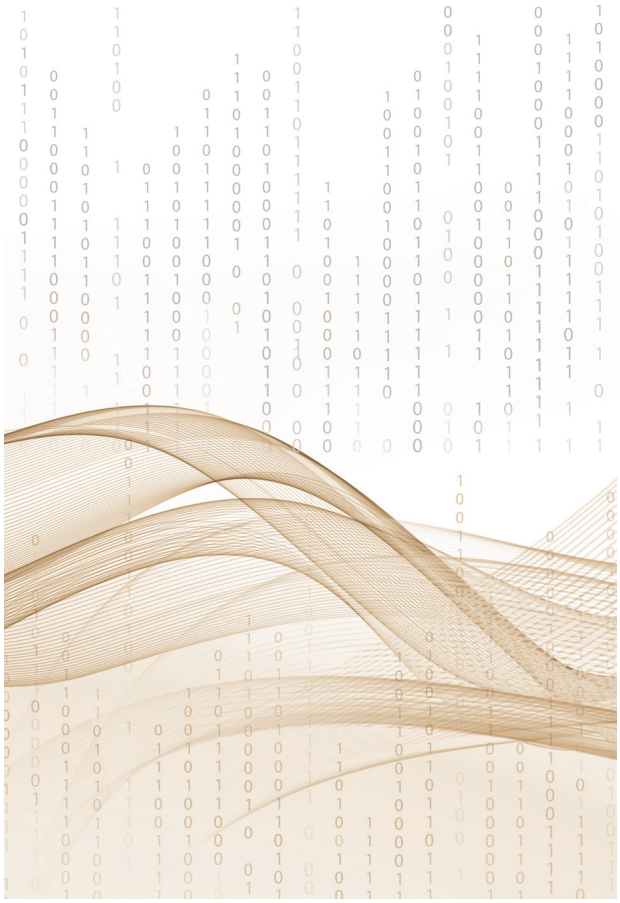
3 降低门槛。

基于SaaS、low-code/no-code等技术，让人人都可以完成数据分析任务。

大数据产业仍存在使用门槛高、碎片化等技术瓶颈约束。在企业的转型升级过程中，业界已经有非常领先，而且成熟的企业通用和行业通用SaaS服务，这些软件都是先行者经过多年研发积累出来的智力资产，重新开始自研很难在短时间内达到业界先进的水平。所以，如果所需技术能力在业界已经有成熟服务支撑，那么在成本可接受的情况下，应该考虑优先引入，通过先进技术构筑主干平台。那么在架构设计中，需要将能力服务化、技术组件化，通过分层解耦和复用，像搭积木一样，即插即用，促进敏捷交付，并降低长期开发与运维成本。

数据生命周期长，每个步骤都涉及各种技术分支，而且还在不断的变化演进中。需要数据分析厂商，提供低（无）代码的集成开发平台，为使用者屏蔽底层技术，可以基于一套平台完成数据分析的全流程。

以上是华为云对数据分析平台演进的一些想法和探索，其出发点是从根本上遵循大数据的自然特性和发展规律，整合数据全生命周期的先进技术，降低大数据使用成本，充分激发数据要素价值潜能。

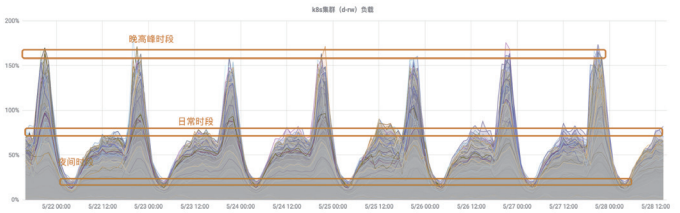


陌陌K8s集群在离线分时混部的优化实践

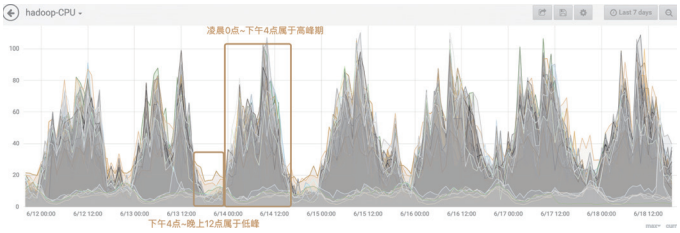
周峰
陌陌云基础设施团队
虚拟化研发负责人

业务场景描述

K8s在线业务集群在不同时段的资源使用情况差别较大，具有明显的潮汐现象：夜间时段的资源利用率最低，日常时段相比夜间有2倍以上的增加，晚高峰期间相比日常时段有1倍左右的增加，以某个集群为例：

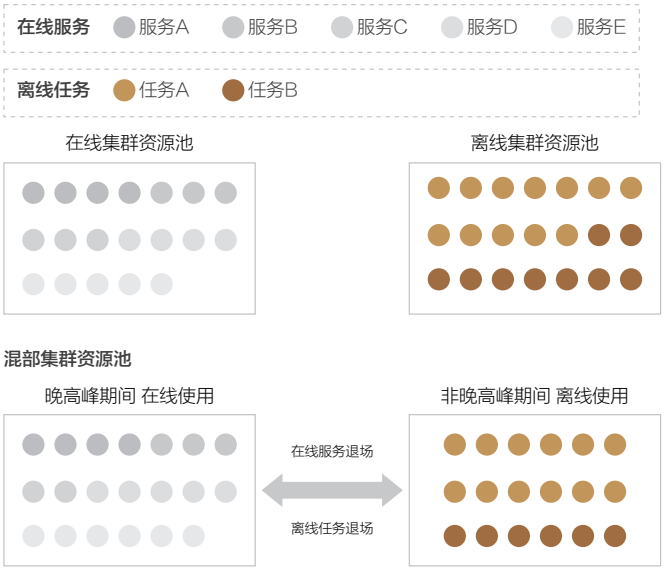


为了应对晚高峰期间的流量洪峰，每个业务都要申请较多实例，这就导致其它时段业务具有较大的资源冗余，集群整体的CPU利用率均值（按天计算）只有20%~30%，存在严重的资源浪费情况。而离线集群的情况刚好和我们相反，在凌晨和下午4点之间属于高峰期，下午4点到晚上12点属于低峰期，如右图所示：



结合上面在线和离线集群的情况，为了提升集群的资源利用率，我们有了希望在线业务和离线任务能够混部的想法。如果有个资源池，在线业务晚高峰期间使用，离线任务在夜间和日常时段使用，那资源的利用率将会提高不少。

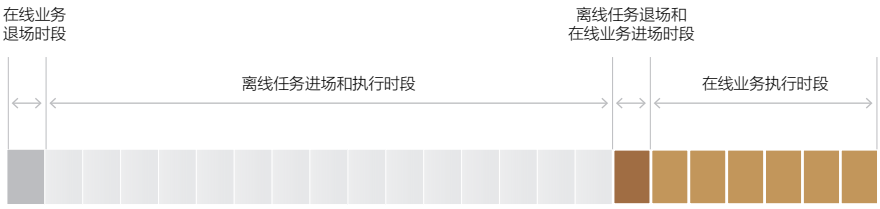
为什么不让在线业务和离线全时混部？一方面需要系统底层提供较强的隔离特性，比如IO、CPU、cgroup参数和cache参数的选择等，另一方面需要融合K8s在线调度和Yarn离线调度，提供统一的调度层，这些关键技术目前还处于调研阶段，所以短期内我们选择了分时复用，把全时混部作为长远目标演进，如右图：



在线服务既部署于在线集群，又部署在混部集群，在线集群的实例常驻，混部集群的实例仅晚高峰期间存在，夜间与日常时段都退场给离线任务使用。

解决方案

要实现上面的想法，我们首先把完整的一天划分成下面四个阶段：



各阶段描述：

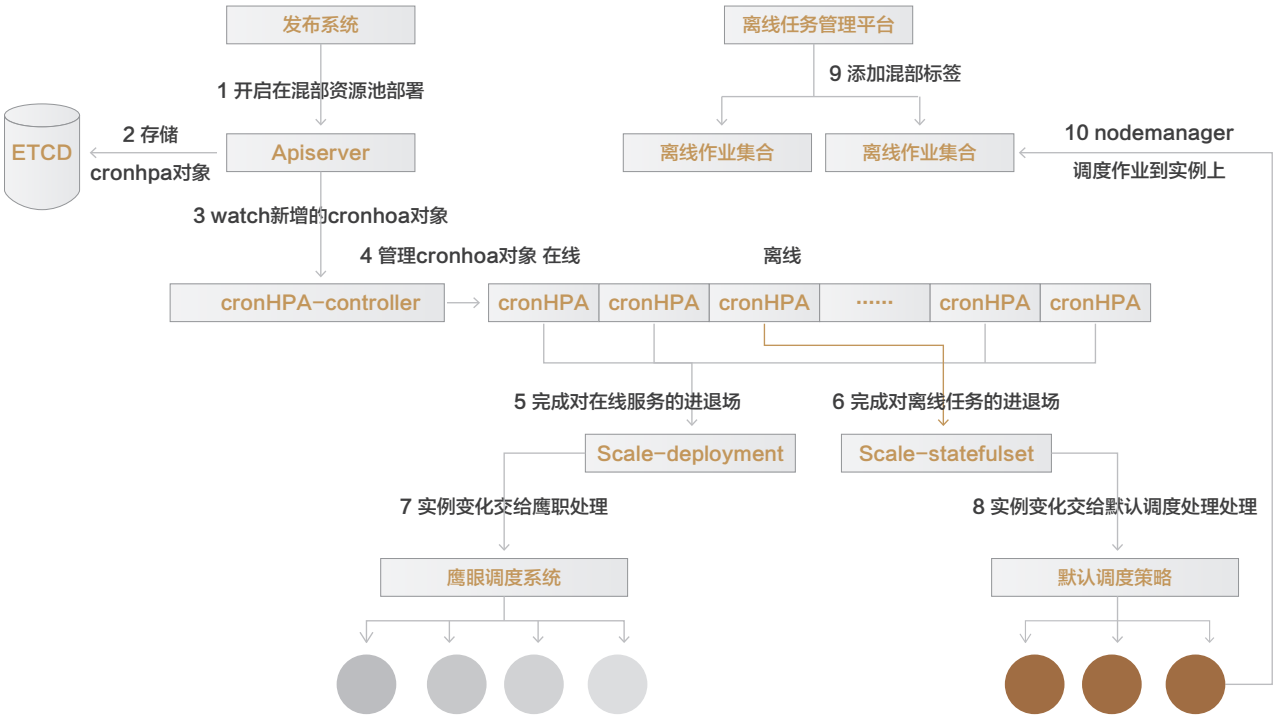
» 在线业务退场时段：在线业务开始分批次退场，并在1点之前完成全部退场。

» 离线任务进场和执行时段：离线实例在凌晨1点开始进场并开始调度离线任务。

» 离线任务退场和在线业务进场时段：下午16点离线任务开始退场，并在16点30分内完成退场，然后在线业务开始进场。

» 在线业务执行时段：包含在线业务的晚高峰，持续运行到当天结束。

系统总体架构：



大致过程：

» 业务在发布系统上开启服务在混部集群的部署。

» 发布系统通过Apiserver的cronHPA API新增cronHPA对象。

» cronHPA-controller watch到新的cronHPA，并进行托管，控制服务的进场和退场。

» 离线任务具有单独的cronHPA，控制离线实例的进场和退场，实例中包含NodeManager和agent服务负责离线任务的调度。

» 在线业务的实例由鹰眼负责调度，离线任务的实例由默认调度策略负责。

我们主要从以下几个方面进行详细讲述：

» 在线服务的进场和退场。

» 离线服务的进场和退场。

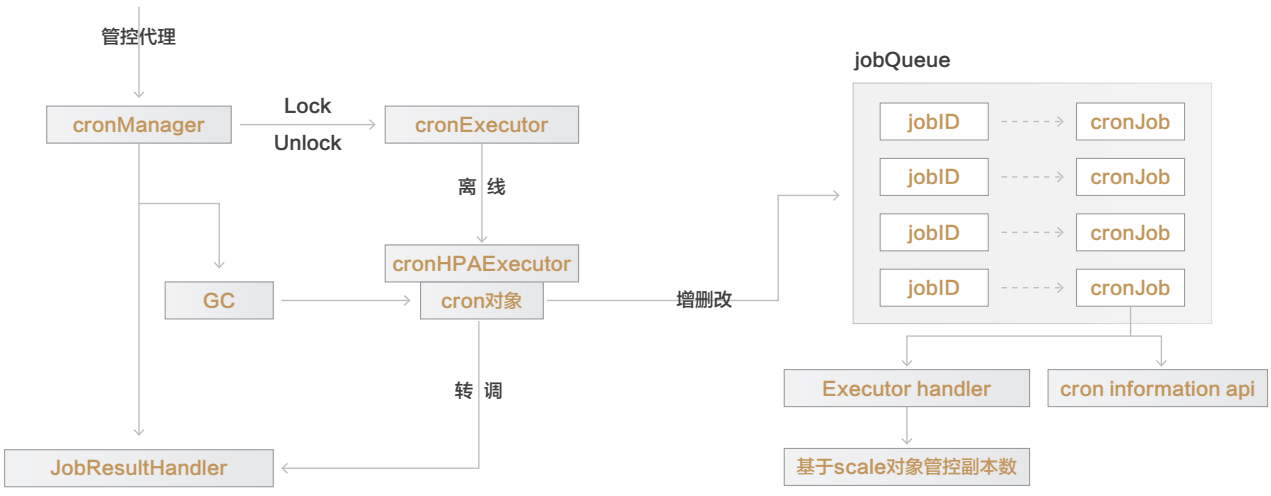
» 在线服务的调度。

» 离线任务的调度。

在线服务的进场和退场

一方面，由于在线服务的退场可能需要多次分批进行（如果一次缩掉混部集群的所有实例，可能会造成服务的在线集群请求量瞬间增加许多，为了保证服务缩容过程中的稳定性，我们采用分批缩容的方式），如果通过deployment API实现的话，需要和apiserver频繁进行交互。

另一方面，对实例的管控局限于deployment现有API，为了保证更大的灵活性，我们最终选择了Operator扩展的方式实现，新增一种crd，命名为cronHPA，实现对实例的定时管理，cronHPA的底层架构如下图所示：

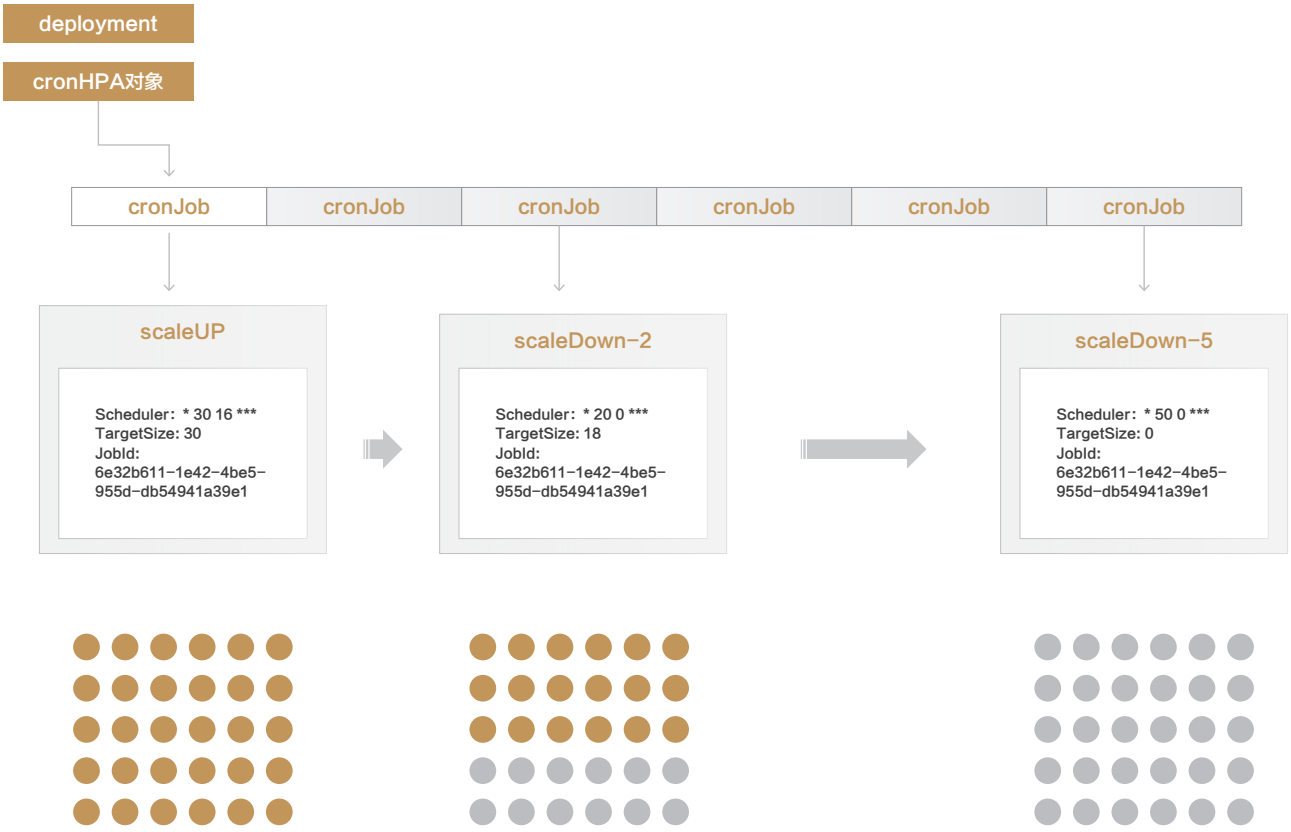


关于cronHPA如何实现单个服务的进场和退场?

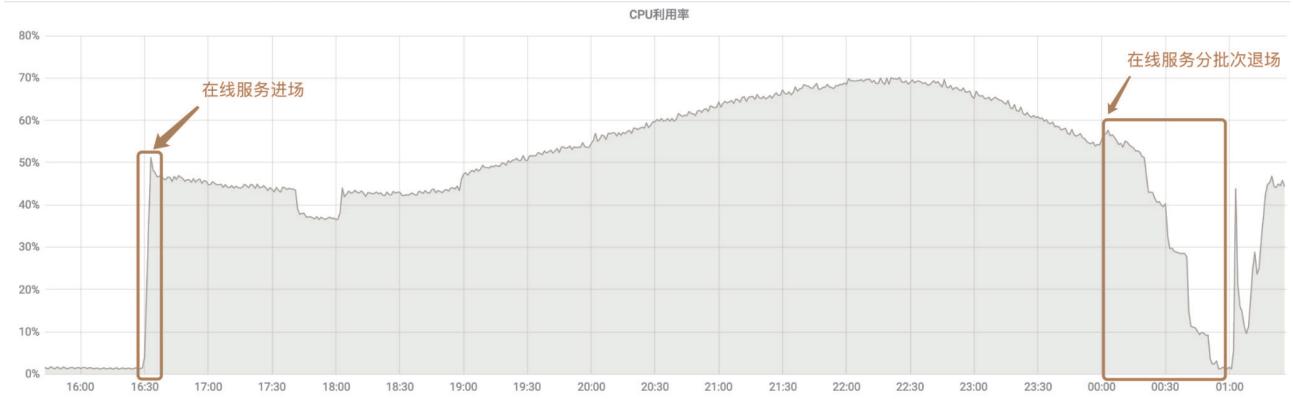
如下图所示，每个服务具有一个单独的cronHPA对象，而一个cronHPA对象包含多个job，其中job主要分为两类：

» ScaleUp：负责该服务的进场，按目标实例数进行扩容。

» ScaleDown-X：负责该服务的退场，分批次完成。

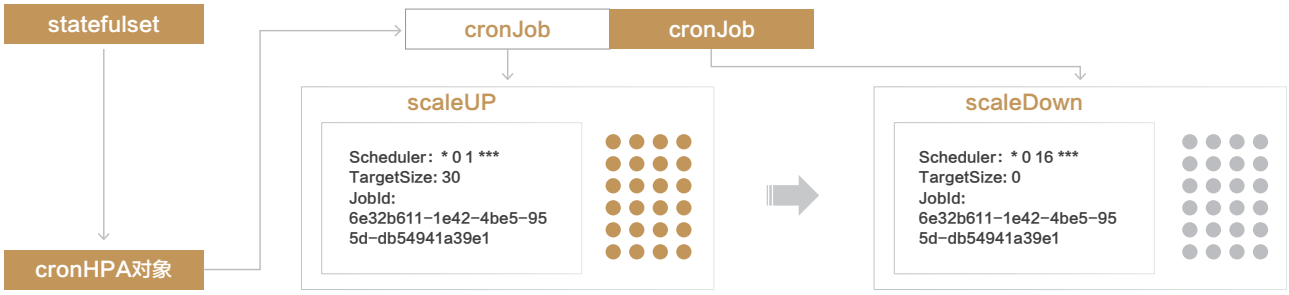


以某个服务器上利用率的变化看服务进场和退场的效果：

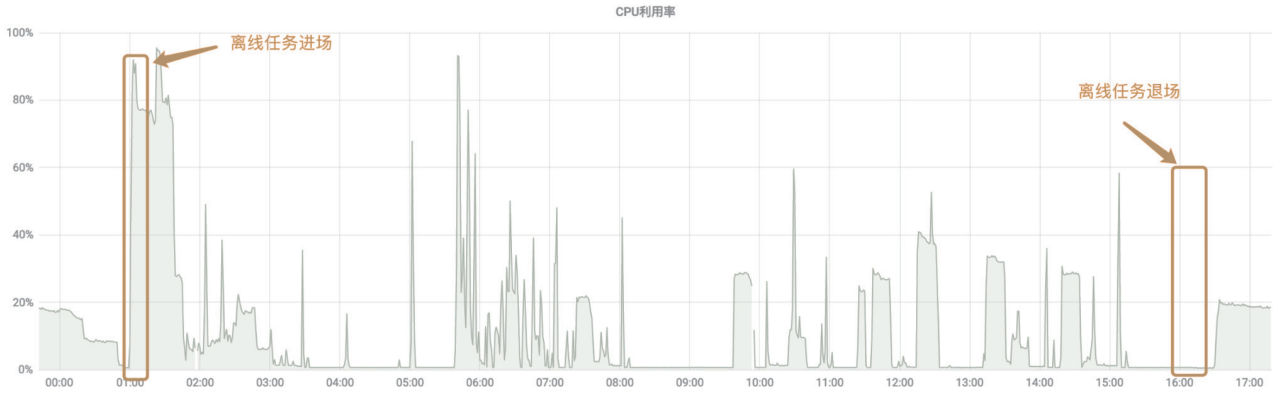


离线服务的进场和退场

这个也是由cronHPA管理，不过由于离线NodeManager在注册时需要认证，所以使用statefulset管理实例，保证实例的hostname不变。另外，由于离线实例的缩容是不需要分批进行的，所以这两个操作都是分钟内完成，具体如下：

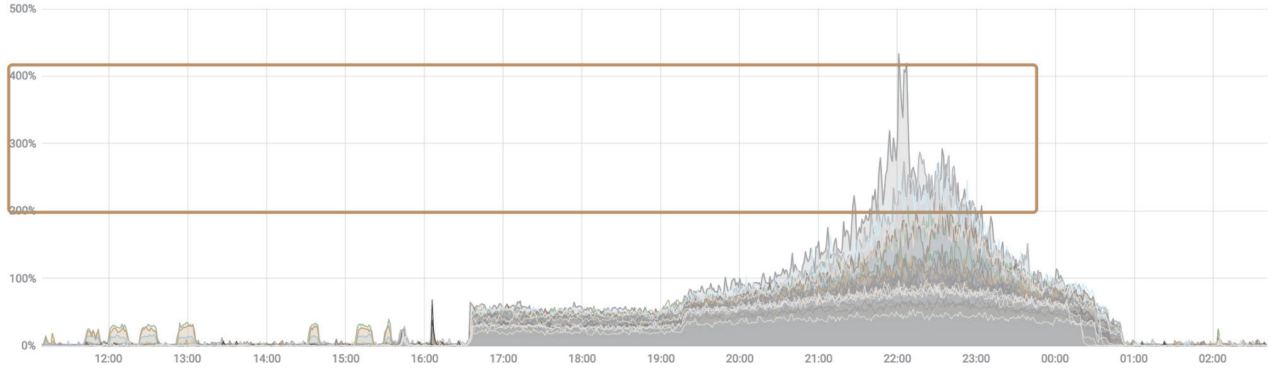


实际效果如下：

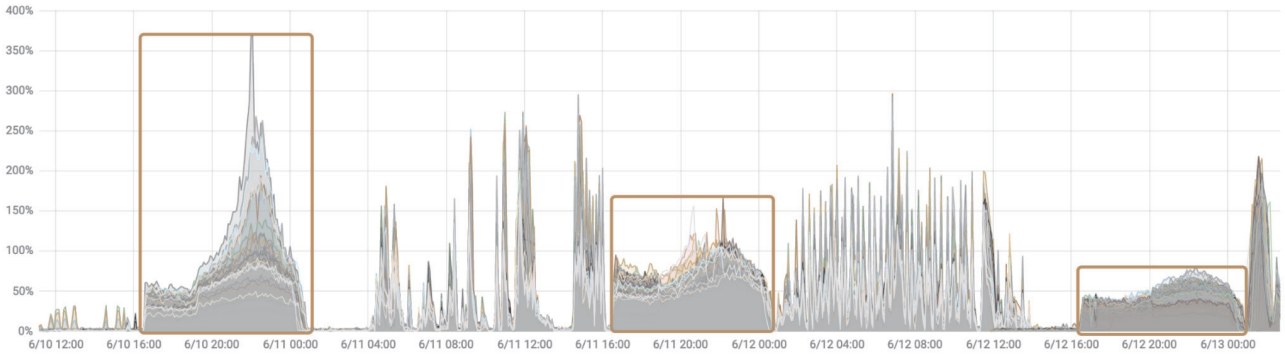


在线服务的调度

在进场时段，涉及到很多实例的调度，如果使用默认调度策略，会存在个别节点负载的长尾问题，如下图所示：



而且随着部署实例的增多，这个问题会越发严重，所以混部在线服务的调度转由鹰眼负责，基于动态调度和重调度机制保证当前的实例调度组合足够稳定，开启鹰眼调度后，在线服务晚高峰期间的稳定性有了明显改善，如下图所示：



离线任务的调度

主要基于标签对任务和NodeManager染色，完成离线任务在混部集群的调度，架构简图如下：

关键逻辑描述：

- 容器启动后，Agent通过环境变量识别是否是容器环境，并完成容器环境的相关初始化工作。
- 用户在DPW集群管理系统新增离线任务，然后对要在混部执行的任务进行染色提交到ResourceManager。
- 容器中的NodeManager启动后，识别到自身是容器环境，会带标签进行注册。
- ResourceManager基于标签区分任务，在离线集群和混部集群之间分派。
- NodeManager将容器的配额细分给收到的任务，并进行执行。
- Agent除了初始化工作，还会对本地NodeManager进行管控。

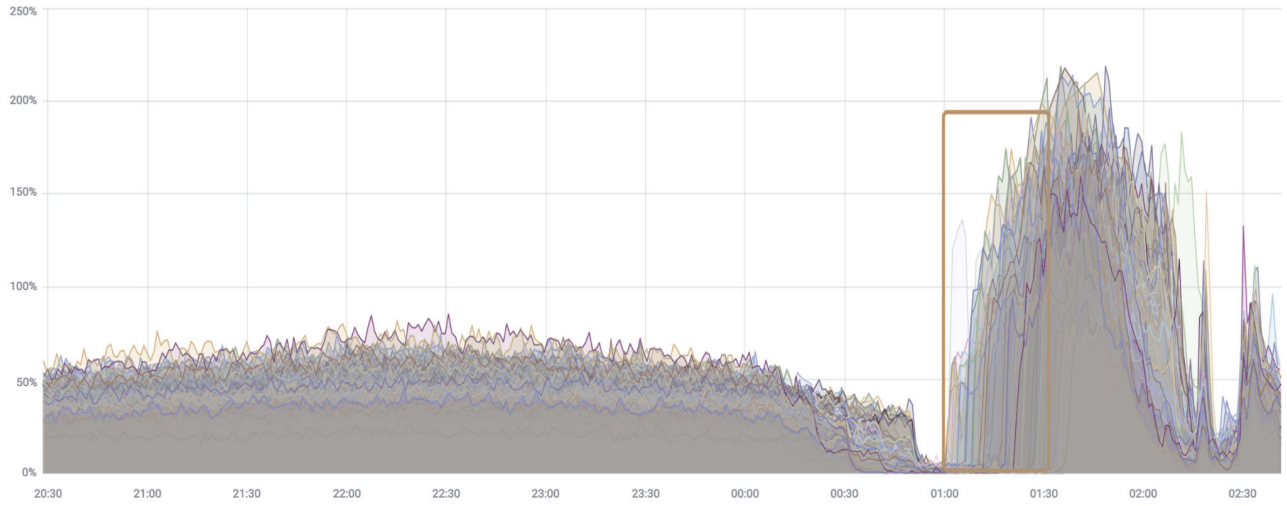
实践过程中遇到的问题

在实践过程中，我们也遇到了很多问题，这里选择几个典型的问题分享给大家。

离线任务进场慢?

在对集群扩容后，发现凌晨的离线实例进场很慢，如下图所示：

大概耗费了30分钟左右才完成全部进场，通过分析，一方面由于statefulset在启动实例时是串行执行的，而离线实例数又比较多，导致总体上有一定耗时，但也应该会收敛到分钟级别。通过进一步排查发现，是由于新机器拉镜像慢导致，在第二天的凌晨，就自动恢复了，为了应对这种情况，对于扩容以及有镜像更新时，我们会在离线进场前进行预拉取操作，保证离线进场的速度足够快。

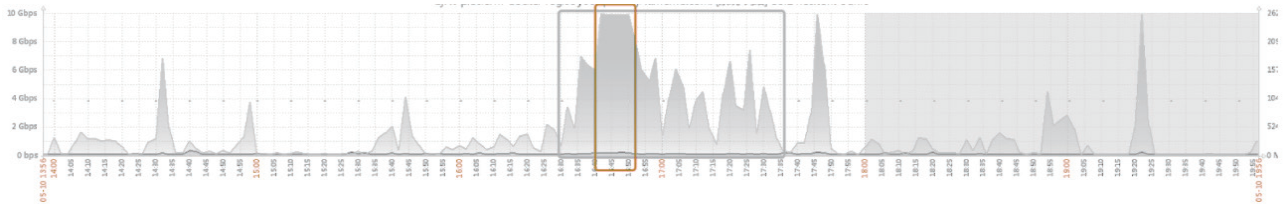


在线服务集中进场导致镜像机带宽满载?

主要通过下面两个方式进行优化：

» 引入错峰机制：调整不同服务的进场时间点，避免进场拥挤。

» 调度策略开启ImageLocalityPriority：调度优选阶段优选本地已经有镜像的节点。



多种部署标签不一样的服务如何处理?

在普通集群不同服务的部署标签可能是不一样的，如果要在分时混部增加相应的标签，不仅会导致混部集群内的资源出现相互隔离，而且会增加运维成本。

为了解决这个问题，我们引入了基于Deployment的Webhook用于动态调整不统一的标签，自动完成统一标签的工作。

如何提前发现跨机房引入的耗时问题?

由于分时混部集群所处的机房与迁入的服务原先部署的机房可能不一样，所以迁移环节，我们会基于灰度实例自动分析耗时对比，发现不适合迁移到分时混部集群的服务，如下图所示:

服务迁入分时混部后，耗时明显增加。

通过自动分析可以提前发现不适合迁入分时混部的服务。

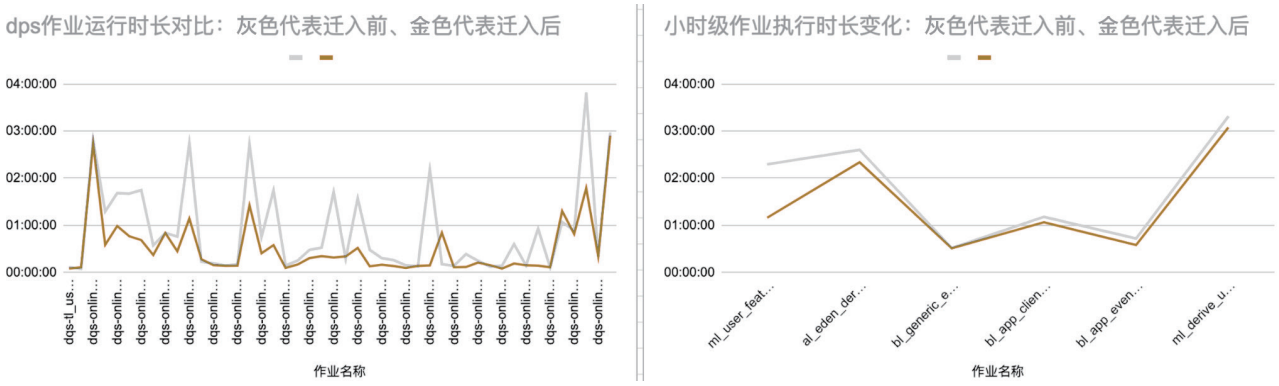
实际效果

在线业务的晚高峰尖刺基本抹平

业务晚高峰的流量比平时更高，晚高峰期间混部集群自动扩容分摊部分业务流量（35%~50%），这样业务晚高峰的单实例CPU损耗相比日常时段的变化就会很小，以某线上服务为例，晚高峰的尖刺在开启混部后基本消失。

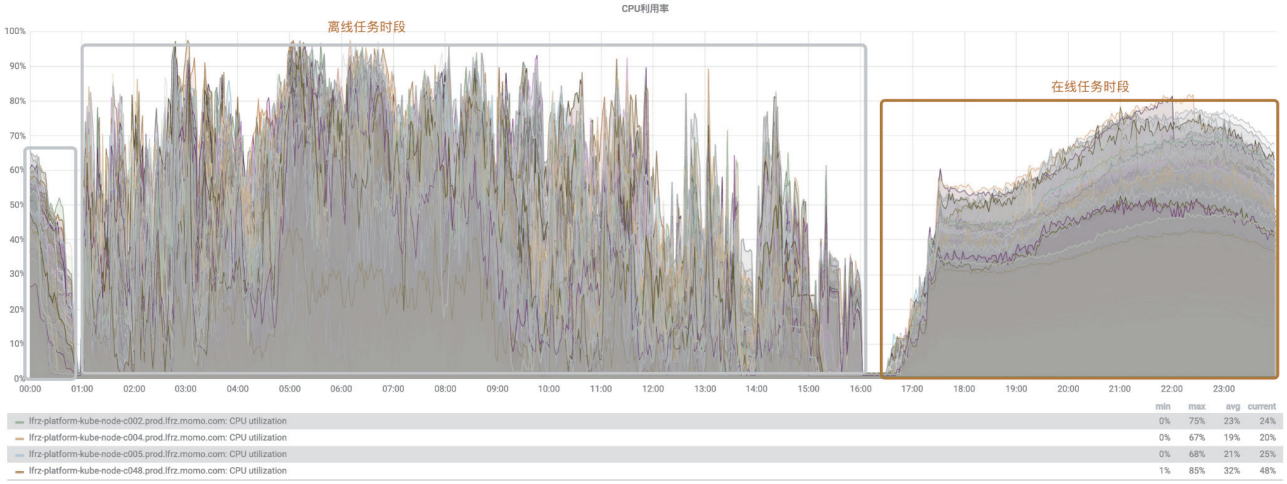
离线任务的执行时间变短

离线集群节点上除了部署NodeManager，还包括其它组件（比如HDFS的DataNode等），所以机器资源并非离线独占，迁入混部后，离线独占机器资源，以dps和小时级作业为例，dps作业接入混部后所有任务执行总时长为原来的60%，小时级作业的总时长为原来的90%，都有不同程度的减少，如下图所示



整体资源利用率提升

混部集群天粒度的CPU均值从20%~30%提升到40%~50%，如下图所示：



未来规划

» 离线任务调度优化：离线实例被强制退场导致任务执行失败，离线调度时评估下该任务是否能在退场前完成，对于无法完成的情况调度到离线集群执行，避免任务执行失败。

» 在离线全时部署：分时复用的进退场间隔会有服务器空跑现象、一些发布不平滑的服务也无法迁入混部集群，后续考虑基于cgroup实现底层的资源隔离和研发统一的调度框架，支持在离线同时部署在一台机器，实现全时混部，进一步提升资源的利用率。

《基于区块链的数字藏品研究报告》解读

可信区块链推进计划

随着信息技术不断发展，高新技术与数字文化创意领域深度结合，数字文创产业的快速发展成为数字经济发展的重要引擎，然而，传统数字作品存在的版本确权难、版权追溯难、维权取证难等问题日益凸显，极在影响了数字文创产业的健康发展。

区块链作为一种新型的信息技术，具备账本一致性、链上可追溯、不可篡改性等特性，能为数字文创作品的版本保护提供较好的解决方案，因此基于区块链的数字藏品应运而生。非同质化通证（NFT，Non-Fungible Token）就是一种使用了区块链技术的新型数字作品权属凭证，通过将文创作品的权属存储在去中心化的区块链网络上，复用链上信息的可追溯、可查询特性，让创作者更便捷地实现作品的确权，并对作品的权属转移和使用过程进行跟踪追溯，能有效解决数字文创作品在版权确权难、版权追溯难、维权取证难等方面的痛点，进一步规范数字文创版权保护，激发数字内容市场活力。

然而，基于区块链的数字藏品是一块创新的业务，目前业务规范尚未建立、监管仍不明朗，区块链推进计划联合了国内的一企业，讨论并编写了基于区块链的数字藏品研究报告，希望为行业提供一些参考，各企业需要审慎来开展相关业务。

区块链数字藏品的定义

基于区块链的数字藏品是使用区块链技术来表明某个数字作品、艺术品、商品的权益归属，它能在区块链网络中标记出数字藏品的所有者，并对后续流转进行追溯，包括但不限于数字图片、音乐、视频、电子票证、数字纪念品等各种形式。基于区块链数字藏品是基于联盟区块链技术生成和发行的，只能被通过实名认证的区块链用户所拥有，是具备唯一标识的非同质化数字商品。

区块链数字藏品的特性

基于区块链的数字藏品具备唯一性、不可分割、不可篡改、可验证、稀缺性等技术特性：

» **唯一性：**每个数字藏品都在链上都具备唯一标识 ，代表数字或现实世界中的某个对象。

» **不可分割：**每个数字藏品自身都不可分割，可代表特定的数字藏品。

» **不可篡改：**数字藏品本身属性及所有权信息、历史交易记录等信息在防篡改的链上存储记录

» **可验证：**区块链上信息公开透明 ，所有用户均可查询 、验证数字藏品的所有权信息

» **稀缺性：**区块链数字藏品独一无二、权属明确，可永久保存，具备稀缺性

区块链数字藏品的核心作用

区块链技术为数字藏品的创作生态带来很多核心作用，开创了一种新型可确权、可追溯的文化消费，相比传统的数字作

品管理，其优势十分明显：

» 在版本登记方面，区块 链技术标记了作品的所有者和唯一性，基于区块 链上信息不可篡改的特性，能实现线上作品中可能性的存证确权，降低作品确权成本。

» 在所有权追溯方面，区块 链上的信息不可篡改、公开 透明 、可追溯，让藏品的所有权信息更便于查询、更可信。

» 在侵权维权方面，标记了数字作品唯一性，从代码层面上限制了它的可复制性，难以侵权。

» 在创作者收益保障方面，通过区块 链可以记录作品未来的流向，创作者保留版权的同时，还能从作品转卖中获得分成。

区块链数字藏品的应用场景

区块链数字藏品的应用场景大体上可以分为两类，一类是基于区块链数字藏品的应用场景、一类是基于区块 链的数字藏品衍生品的应用场景。

针对第一类场景，常见的应用有区块链艺术收藏品、区块链数字音乐藏品、区块链数字游戏藏品、区块链数字体育藏品、区块链数字票务藏品。

» **区块链艺术收藏品：** 将数字或实体的艺术收藏品制作成“区块链数字藏品”，确保作品真实性，持久性。

» **区块链数字音乐藏品：** 将数字音乐的专辑或单个音乐制作成“区块链数字藏品”,确保数字音乐的版权和归属，避免盗版和侵权。

» **区块链数字游戏藏品：** 将游戏中的资产，包括土地、人物、服装、道具、游戏积分等上链，确保游戏资产的稀缺性和流通。

» **区块链数字体育藏品：** 将体育类周边的产品上链制作成区块链数字藏品后发行，满足新一代体育迷的消费需求。

» **区块链数字票务藏品：** 将各种类型的数字化门票铸造为数字藏品，有助于门票的确权和流转，同时降低票务成本。

在第二类数字藏品衍生品的场景中，常见的应用有区块链数字教育、区块链数字科研、区块链元宇宙。

» **区块链数字教育：** 将网络上的教育资源上链做成数字 教育产品，可以助力教育资源的确权和流转。

» **区块链数字科研：** 将实体的生物样本上链，记录和追踪生物样本的来源和使用信息，提升了科研领域资源的开放性。

» **区块链元宇宙：** 通过将元宇宙中的虚拟物品的“数字版权”上链，保障虚拟物品的确权和交易，解决虚拟物品资产化的难题。

区块链数字藏品的潜在风险

我们同时还要注意区块链数字藏品的一些潜在风险，如：技术风险 网络欺诈风险、价格炒作风险 侵权维权风险、金融安全风险 监管合规风险。

» **技术风险：** 数字藏品接口不规范、元数据中心化问题、智能合约安全漏洞等技术层面的风险。

» **价格炒作风险：** 数字藏品文创作品价值没有公认衡量标准，因此容易引发市场过高的心理预期，成为炒作或投机者的牟利目标。

» **金融安全风险：** 谨防变相ICO、洗钱和跨境资产转移等金融方面的安全风险。

» **网络欺诈风险：** 市场上的大多数消费者和投资者对于数字藏品的概念并不熟悉，因此很容易被黑灰产利用，成为网络欺诈工具。

» **侵权维权风险：** 数字藏品数字资产的版权保护缺乏明确的法律约束，因此要谨防知识产权的卖方侵权和买方侵权。

» **监管合规风险：** 当前法律法规和监管政策都不完善，业务形态多种多样，需进行合规检测和监管。

数字藏品的形态是多种多样的，需要与数字经济生态相结合，赋能数字文创、影视、娱乐、体育等业务场景，推动区块链数字藏品技术与产业融合，通过技术创新推动模式创新，进而 引领数字经济变更，但是，针对数字藏品创新过程中的技术及市场等风险，亟需主管监管部门和行业主体共同发力，一方面，主管监管部门可以加快完善监管体系，从政策出台、标准制定、市场监督等方面发力，促进行业健康有序发展；另一方面，运营主体应加强行业自律，通过申请合法经营资质、加强风险防范机制、加强技术合规能力等手段主动规避相关风险，合规、合法开展相关业务，充分发挥区块链技术在版权保护方面的价值。

有关更多内容细节，请下载并参见《基于区块链的数字藏品研究报告》。

数字化转型下我国分布式数据库应用挑战及发展建议

冯柯
华为云数据库GaussDB
首席专家

一、发展情况

过去三十年，以金融业为代表的核心信息系统架构依托IOE（即IBM、Oracle、EMC）技术，构建了一套集中、专用、封闭的稳态技术体系。但是，随着互联网及云化时代的到来，企业业务架构产生巨大变化，以银行为代表的金融业需加速构建敏态体系，推动底层数据库的分布式改造和互联网金融业务创新。分布式数据库具有满足行业关键应用的高扩展性、高性能、高可用性及软硬件解耦等特性，是金融等重点行业信息系统数字化转型的基石。

产品成熟度提升

随着分布式数据库在金融等重点行业的不断应用，产品成熟度得到很大提升。

一是新技术的不断发展使得分布式数据库在自身固有的优势领域，如扩展性、高可用等方面进一步强化，已有多个应用在重点行业核心业务中落地。

二是国产分布式数据库的性能已经实现了与其他商业数据库持平甚至超越，这在多个大型企业机构产品准入测试中得到充分证明，可对行业核心业务起到重要的支撑作用。

三是更多厂商开始提供对主流国产分布式数据库的功能支持，产品的兼容性取得显著进展。管理控制软件、迁移工具等配套设施逐渐完善，极大地降低了数据库的使用门槛和迁移成本。

生态逐步完善

一是加快推动分布式数据库在重点行业落地，主流分布式数据库厂商纷纷与众多大型银行、企业等开展联合创新活动，取得了许多突破性的成果。以GaussDB分布式数据库为例，在与大型商业银行的联创过程中，已完成10个以上业务系统的分布式数据库替换，覆盖银行A类到D类全场景业务。

二是通过一站式的迁移解决方案，实现以较小的业务改造工作量从传统数据库向分布式数据库转型，迁移成本相对较低。而且使用分布式数据库后，业务系统运行稳定，可靠性和扩展性有所增强，从各项指标看，已基本具备承接Oracle及DB2大机下移的能力。三是分布式数据库相关的行业标准和评价体系逐步健全，对产品发展起到较强的规范引领作用。

总体发展情况向好

当前国产分布式数据库已经渡过了“能用”阶段，正在迈向“好用、易用”阶段。

横向来看，我国分布式数据库的发展基本与国际同步，性能、可靠性等部分技术指标甚至具有优势，在应用领域也取得些许领先。

纵向来看，以金融业为例，分布式数据库应用取得较大进展，不管是在互联网新核心业务，还是传统核心业务中，分布式数据库行业应用落地数量大幅增加，有逐步替代集中式数据库的趋势。

二、面临的主要问题

主体改造意愿不强，行业实践尚不充分

一方面，原有数据库系统改造为分布式数据库，对用户及应用单位提出了较高的要求。改造所面临的成本问题，以及改造完成后分布式运维实施的复杂性，使得部分金融机构对于全面应用分布式还存在有一定的疑虑，主动改造意愿不强。另一方面，分布式数据库在行业典型应用场景中的落地仍处于摸索阶段。由于部分项目中存在一定的需求定制化，应用解决方案与产品的边界不够清晰，产品的规模化复制能力仍有待加强，行业最佳实践相对缺乏。这些因素也影响了金融机构对迁移采用分布式数据库技术的积极性。

分布式数据库的生态建设仍需加强

生态建设是当前我国基础软件相对薄弱的一环，特别是对基于自研的分布式数据库厂商而言，虽然对实现技术和产品更加自主可控，但在生态建设方面仍需积极应对投入转化慢、门槛高、市场接受程度低等挑战。一方面，部分产品的技术体系相对封闭，用户无法从市场快速获取合格的开发运维人员，导致业务改造及生产运维仍严重依赖原厂，规模化复制效应较差。另一方面，部分产品的开放性仍有待提升，与其他平台数据互联互通的能力不足在客观上造成了业务“上车容易下车难”的现实困境，增加了用户被锁定的风险。

可持续发展的盈利模式需进一步探索

我国数据库的发展可以追溯到30多年前，在这样一个相对较长的发展周期内，技术和产品都取得了显著进展，但在产业化方面，知识产权的保护不够充分等诸多问题造成部分参与主体的市场化盈利能力较弱，产业整体规模难以做大。分布式数据库虽然已取得了一定进展，但“池子深才能养大鱼”，如何依托当前政策窗口，真正形成可持续发展的商业模式，还需进一步探讨。

三、行业的应用建议

尽管存在一些问题，但我们坚信分布式是数据库未来的发展趋势。如果将分布式数据库和单机数据库类比为“高铁”和“轿车”，因两者定位不同，期望“高铁”像“轿车”一样简单易用既不现实也不科学。所以应避免将分布式数据库的应用简单地理解为对单机或者集中式数据库的一对一替换，而要深入考虑如何充分发挥分布式数据库的技术优势。遵循以

上思路，我们对于分布式数据库在金融等重点行业的应用提出以下几点建议：

通过技术创新和最佳实践，推动行业应用不断深入

一方面要探索利用人工智能等新技术提升产品服务效能。人工智能技术可实现自动数据分区规划、故障自动诊断和自愈、自动负载均衡、面向混合负载的自调优等功能。目前人工智能技术在分布式场景已经有了一些单点突破，但距离全场景落地、实现整体成本的全面降低还有很长的一段路要走，需要继续加以积极的行业引导，推进技术交流和产业落地。另一方面，需充分发挥好示范项目效应。在金融等重点行业典型应用场景如分布式架构设计、多地多中心容灾等，形成最佳解决方案，并在行业推广落地。在此过程中，提炼出更适合分布式数据库的开发、运维、硬件建设等相关要求，研究制定数据库应用方面的标准规范，提高行业的标准化水平，引导各参与主体规范应用分布式数据库，推进行业转型。同时应约束不必要的定制化需求，减少无序竞争，实现技术聚焦。

积极推进生态建设

发挥产业引领作用从软件发展历史看，生态建设是基础软件产业化的重要一环。任何一款商业上真正成功的软件产品，无一不是生态建设上获得广泛认可的成功案例。

首先，充分发挥产业联盟桥梁纽带作用，推动产业发展。在行业内积极进行资源引流，逐步提升技术营运效率及影响力，搭建高端对话平台，促进分布式数据库应用方、应用开发方及厂商更好地交流，共同面对分布式转型下的业务及技术挑战，推进行业生态繁荣；加强与分布式服务框架的合作与交流，通过开源、社区等形式建立广泛的赋能体系；鼓励应用软件厂商全面向分布式架构转型，建立相应的培训体系和检测认证体系。

其次，完善技术生态，鼓励引入第三方软件垂直提供商。在运维管控、工具端以及解决方案层面实现更多差异化的平台能力，加厚行业整体的技术底盘；鼓励第三方产品服务化和上下游集成，推进各产品的互联互通，打造良好技术生态，促进行业健康发展。

再次，建立基础软件开放生态体系，推动开源建设。应鼓励有研发实力的厂商基于国产开源数据库做发行商，有运维能力的厂商基于优质的国产数据库打造适用于自主可控要求的数据库解决方案。数据库厂商和合作伙伴应基于数据库代码开源、产品开放等形式，使数据库产业从封闭商业生态走向产业共赢的开放生态，共同打造开放的数据库生态体系。最后，进一步推进政产学研合作，加强人才储备。明确人才发展战略，梳理多层次行业人才资源地图。加强厂商与各科研院所合作，推进高校在包括数据库在内的基础软件方面专业投入，鼓励有条件的厂商和高校开展课程共建、实践共建，为联合推进分布式数据库关键技术在理论和实践层面的难点问题攻关储备智力资源。

全面拥抱云，开展行业可持续发展的尝试与探索

数据库上云已逐步成为产业共识。根据Gartner的统计，到

2021年，云数据库在整个数据库市场中的占比将首次达到50%；而到2023年，75%的数据库将会跑在云平台之上。

发展云数据库，不仅是对技术和产品的重要升级，更是对数据库良性健康发展的商业模式有益探索，对于实现主体可控、支撑行业长期稳定发展具有重要的现实意义。分布式数据库与单机数据库不同，需要更大的集群规模才能实现资源的更有效利用。分布式数据库与云计算是天然伴生关系，通过云化部署，能够帮助分布式数据库扬长避短，充分发挥分布式数据库在扩展性、资源调度方面的灵活性和优势，在提升资源利用效率的同时，显著降低运维成本，实现真正业务价值。

一是云化基础设施可以通过智能调度、运维系统高效管理更为丰富的应用，并通过多云及边缘计算将应用扩展到多种场景中。

二是软硬协同可为应用提供更好的性能，提升应用隔离性等。

三是云数据库和云基础设施结合，如利用云基础设施本身的能力实现数据库的跨数据中心访问等，可使存储具备理解、预处理数据库语义的能力。

基于以上，一是建议扩大云数据库在金融行业的应用规模。云数据库已经在互联网、电子政务等各行业得到了广泛应用，在金融行业的应用及推广也在稳步推进中。应引导重点用户单位与厂商尝试在行业落地云数据库及云平台，鼓励技术共创，共同探索基于现代云平台的分布式数据库运维及业务开发体系。

二是建议推进行业云发展以提高行业标准化程度。在满足合规营运的前提下，应实现底层基础设施共享，降低中小用户对于分布式数据库的使用门槛和人才需求，减少重复投资，实现集约化营运，充分发挥分布式数据库的规模化优势。厘清各参与主体运营职责与边界，依托业内现有的成熟云平台技术，形成一个或若干个云技术底座，鼓励传统非云数据库厂商根据自身产品技术特点完成与云平台的对接，最终形成行业的云上产品集市，逐步简化并统一运维及交付界面，降低行业应用门槛，提高行业标准化程度。

云原生时代的eBPF，以软件定义内核

杜顿康博士
金沙江创业投资副总裁

1eBPF，革命性内核技术

eBPF（Extended Berkeley Packet Filter）是一项应用场景广泛的Linux内核技术，凭借其强大的可编程性以及高速发展的活跃社区，逐渐成为“软件定义内核”的代名词。在现代数据中心和云原生环境中提供高性能网络和负载均衡，以非侵入式以及低性能要求的方式提取安全与可观察性数据，帮助开发人员跟踪应用性能表现，为故障排除提供数据支持。eBPF是DevOps中重要的“眼”与“耳”，在云原生的可观测性，安全以及网络三个百亿美金市场规模的领域发挥着越来越重要的价值，成为初创企业去挑战巨头的重要武器。

eBPF起源于Linux内核，可以在操作系统内核环境下运行沙盒程序，从而让应用开发工程师与内核的协作更为敏捷高效。内核具有监督和控制整个系统的特权能力，一直是可观察性、安全和网络功能的理想场所。但内核的迭代依托于社区的共识，需要对需求的共性进行很好的抽象与长期的讨论，迭代周期往往以年为单位，而eBPF可以在不改变内核源代码或加载内核模块的基础上，安全有效地扩展内核的功能。

2014年在Linux内核3.18版本中，Alexei Starovoitov第一次实现了eBPF，距今不过短短10年的时间，已经发展成为Linux最活跃且迭代速度最快的项目之一。2021年8月，由微软、谷歌、Meta等公司联合成立了eBPF基金会，大力发展eBPF技术，在国内也得到了广泛关注，很多大公司以及可观测性、安全公司也开始关注并采用eBPF技术。

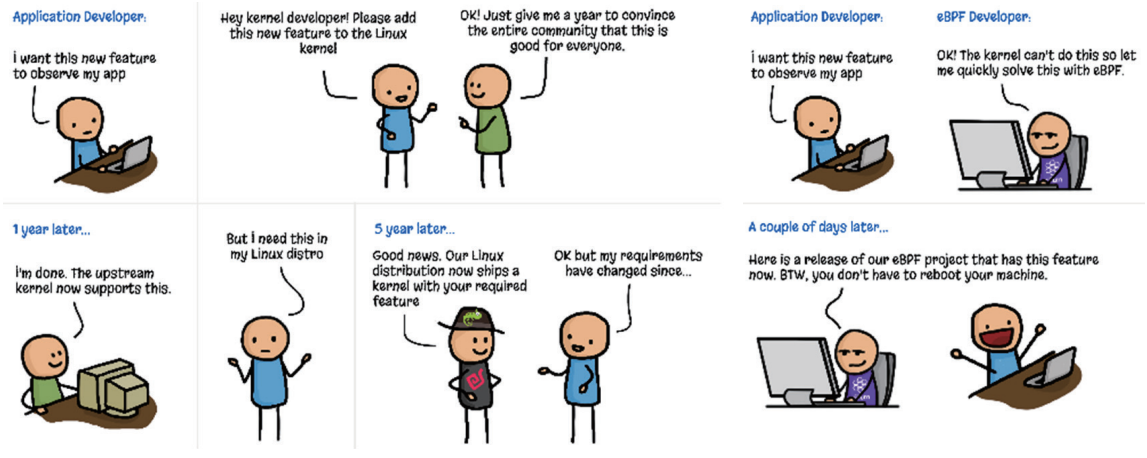


图1：有eBPF前后内核与应用工程师的协作流程（资料来源：网络）

1.1 eBPF的历史与优势

eBPF的前身是伯克利数据包过滤器（Berkeley Packet Filter, BPF），一个用在虚拟机中来编写包过滤器的工具，常用于做tcpdump格式的报文过滤，在eBPF技术出现后，传统的BPF又被成为cBPF。eBPF相比BPF通过map映射让内核空间与用户空间可以互相交互数据，同时改进了性能，极大地拓展了指令集的数量，进而拓展了更多的应用场景。同时相比Linux内核模块，eBPF能更好地保障内核安全，提供了更为丰富的数据结构，也降低了应用开发工程师调用的门槛。

1.2 eBPF的运行机制

在大多数场景中，eBPF需要通过类似于Cilium，bcc或者是bpftrace这些程序来进行一层抽象，从而将eBPF技术应用在某个特定的领域。这是由于Linux内核要求eBPF程序以字节码的形式来进行加载，而直接写字节码几乎是不可能的，因此需要依赖某个编译方案例如LLVM将高级代码编译为字节码。

同时与传统的cBPF一样，eBPF也是事件驱动的，需要钩子（Hooks）来进行触发，并通过运行辅助函数（Helper Function）来进行任务处理，通过上述对eBPF技术的分析，我们可以归纳eBPF的工作流程如下：

» **编译：** 通过LLVM等工具将eBPF程序编译为字节码

» **挂载：** 将程序挂在到内核特定事件触发的钩子上

» **验证：** 在被事件和钩子触发前，在虚拟机里头验证程序的安全性

» **运行时编译：** 在Runtime的时候进行编译从而获得最大的效率

» **辅助函数：** 当事件触发的时候，调用辅助函数来对数据进行处理与修改

» **同步：** 使用eBPF map在用户空间以及内核空间共享数据，并保持状态一致性

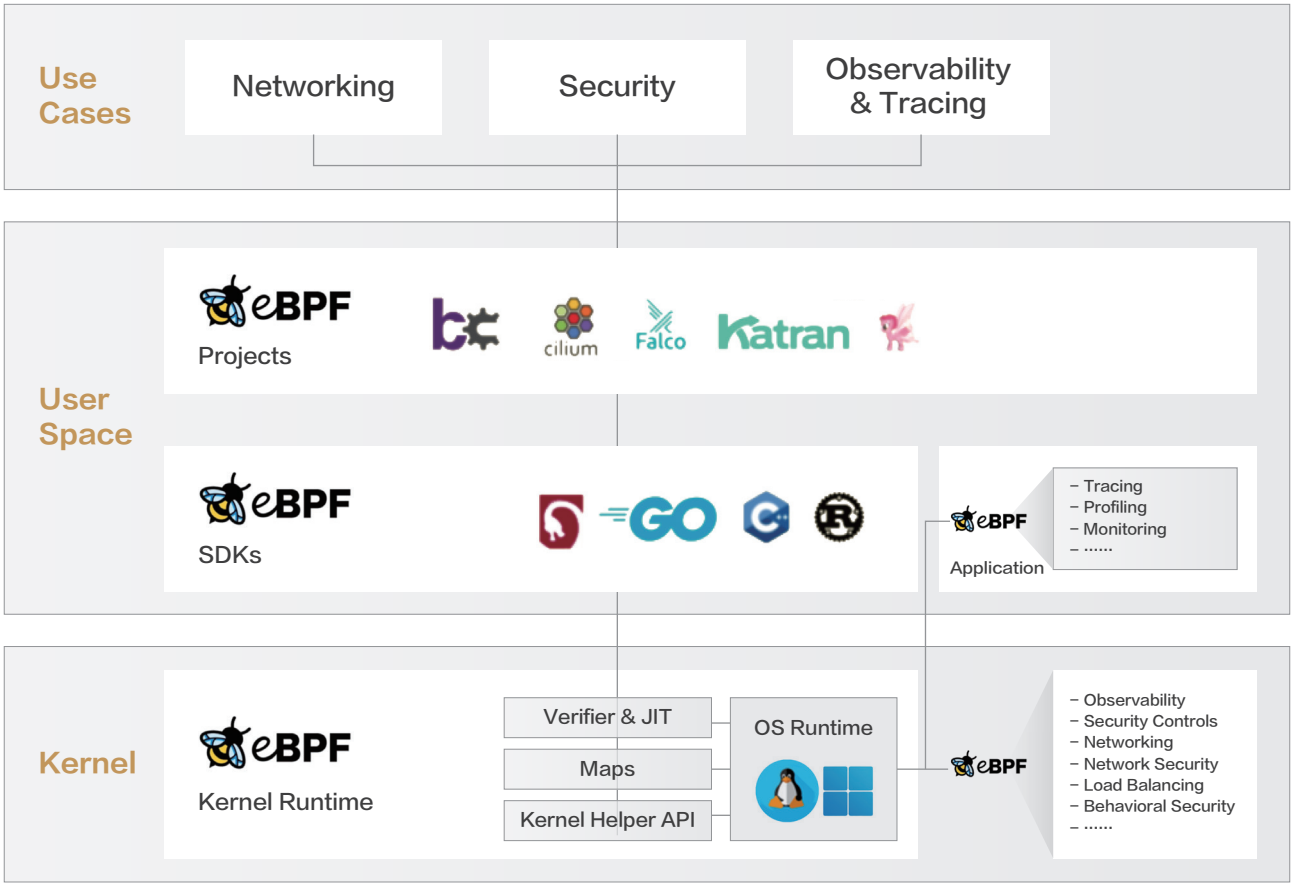


图2：eBPF的架构以及运行机制示意（资料来源：eBPF基金会官网）

2. eBPF丰富的开源生态

eBPF生态的标志性事件之一是2021年8月12日，在Linux基金会之下，由Facebook，Google，Microsoft三家巨头以及Netflix，Isovalent共同发起了eBPF基金会，将eBPF的开源工作推向了一个新的高度。

这些创始成员大多在自己的生产环境已经进行了大量的探索与验证，或者是开源社区的重要贡献者，例如Facebook开源了一个基于eBPF技术的负载均衡器Katran，并且在其IDC中服务多年，近期还讨论了更多基于eBPF的可扩展加密技术；Google则是将eBPF以及Cilium应用于Google的K8S引擎上；Isovalent则是eBPF里最活跃的开源社区Cilium的商业化公司，同时也是eBPF社区中最重要的贡献者与维护者。

2.1 围绕eBPF技术的开源项目

在eBPF的开源基金会中，收录很多eBPF技术的开源社区，涵盖了网络，安全以及可观测性等主要领域，并且也包含一些常用的开发工具：

2.2 Cilium：高速成长的eBPF开源社区

如果我们展开eBPF开源项目的star数的增长历史，最早的开源项目起源于2016年前后，而其中bcc以及cilium是最受关注的两个项目，其中bcc作为开发工具最早开源于2015年中，通过C语言编译eBPF程序降低了开发的门槛，也是其被更广大

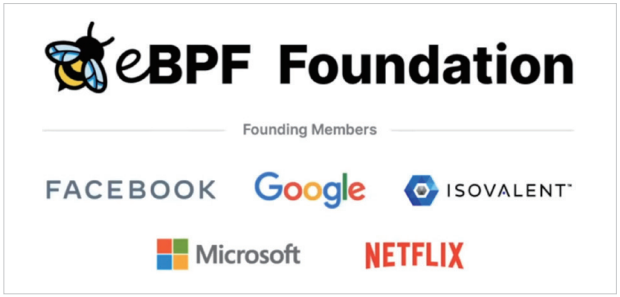


图3：eBPF基金会与创始成员（资料来源：eBPF基金会官网）

开发者使用的重要推动力，而Cilium则是应用与影响力最广泛的eBPF应用类项目。

目前Cilium是云原生CNCF基金会的孵化项目，从功能上，它为应用负载提供稳定，透明且安全的网络连接性以及负载均衡能力。常被集成在K8S中，功能简介如下：

» API保护的可编程性：这来自于Cilium在第7层的核心能力。借助eBPF的可编程能力，Cilium则为每一个独立的应用协议例如REST/HTTP，gRPC以及Kafka提供了独立管理的能力。例如可以仅对所有的HTTP访问中使用了特定method以及特定路径的进行放行。

» Cilium为共享相同安全策略的应用容器组分配了一个统一的安全身份，该身份会与应用容器的网络数据包关联，从而进行身份验证。此外也兼容了传统的基于CIDR的出入口安全策略，可以对来自外部的访问进行安全管理。

	开源方	类别	开源时间	stars	简介
BCC	ioVisor	工具	2015.06	14.3k	BCC - Tools for BPF-based Linux IO analysis, networking, monitoring, and more
bpftrace	ioVisor	工具	2018.09	5.2k	High-level tracing language for Linux eBPF
Cilium	Isovalent	网络	2016.08	11.6k	eBPF-based Networking, Security, and Observability
Katran	Facebook	网络	2018.05	3.6k	A high performance layer 4 load balanc
Hubble	Isovalent	可观测性	2019.11	1.9k	Hubble - Network, Service & Security Observability for Kubernetes using eBPF
Pixie	New Relic	可观测性	2020.02	3.2k	Instant Kubernetes-Native Application Observability
Falco	Sysdig	安全	2016.05	4.8k	Cloud Native Runtime Security
Tracee	Aqua Security	安全	2019.10	1.8k	Linux Runtime Security and Forensics using eBPF

表1：eBPF基金会推荐的开源项目（资料来源：eBPF基金会官网）

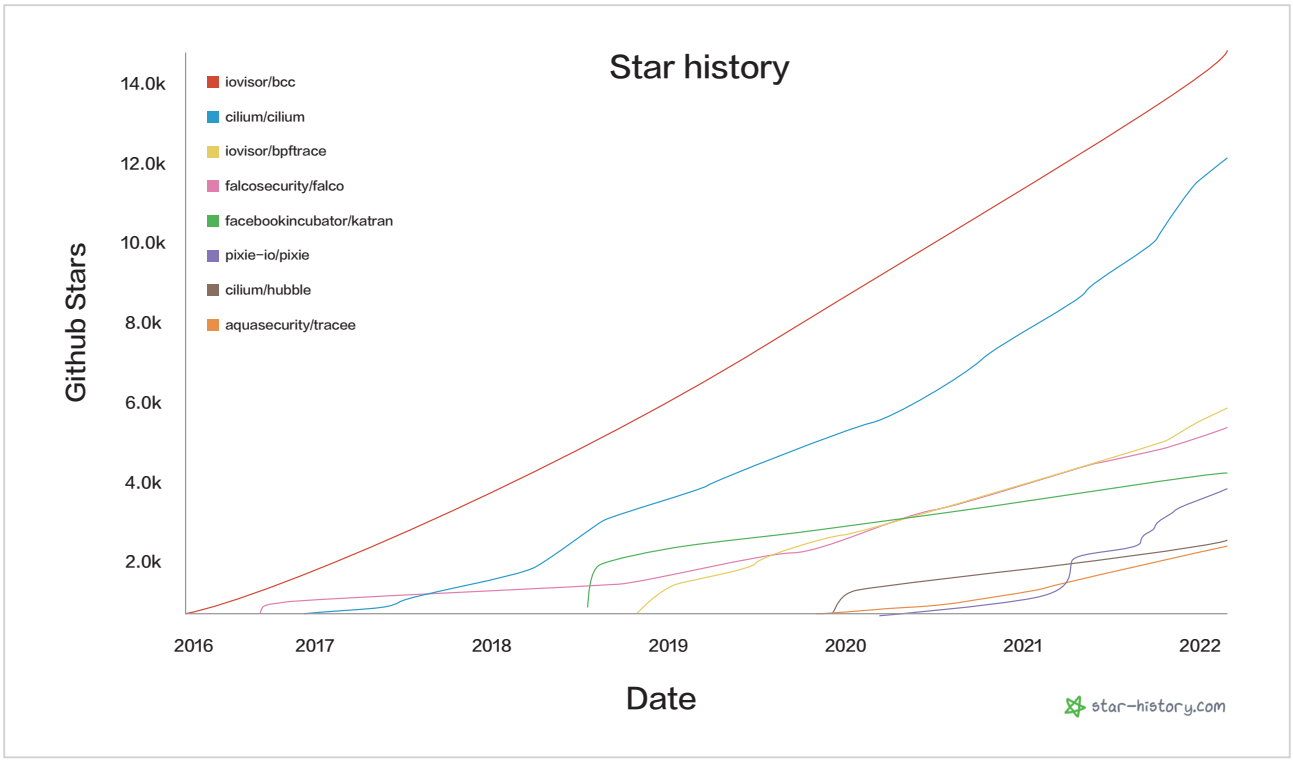


图4: eBPF开源项目与Star增长历史（资料来源: star-history.com）

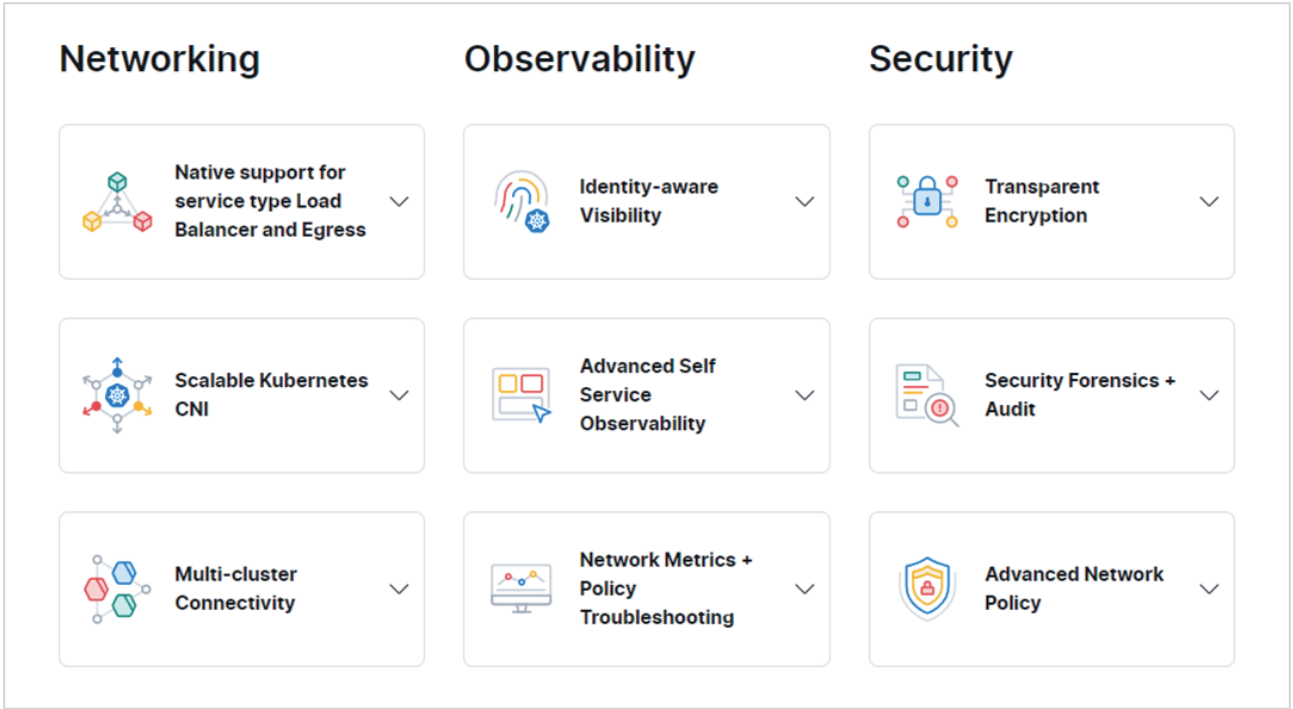


图5: Cilium的核心功能（资料来源: Cilium官网）

» 在网络层面，Cilium提供了一种简单的第3层网络模型，能够跨越多个集群来链接所有的应用容器，同时基于eBPF进行动态智能化的负载均衡以及EDT-based（Earliest Departure Time)带宽管理。

» 可观测性上，Cilium基于社区内的另外一个开源项目hubble来进行支持，提供服务之间的依赖性的监测，运营监控以及告警，同时提供基于流量日志的应用安全可见性管理。

» Cilium的架构如下：

在社区运营层面，Cilium也相当成功，自2016年8月开源以来，已经收获了11.5k的star，同时接近400位开发者为Cilium项目贡献过代码，其中有60多位在最近一个月保持活跃，Github的issues讨论也非常热情，接近6000条issue被处理。

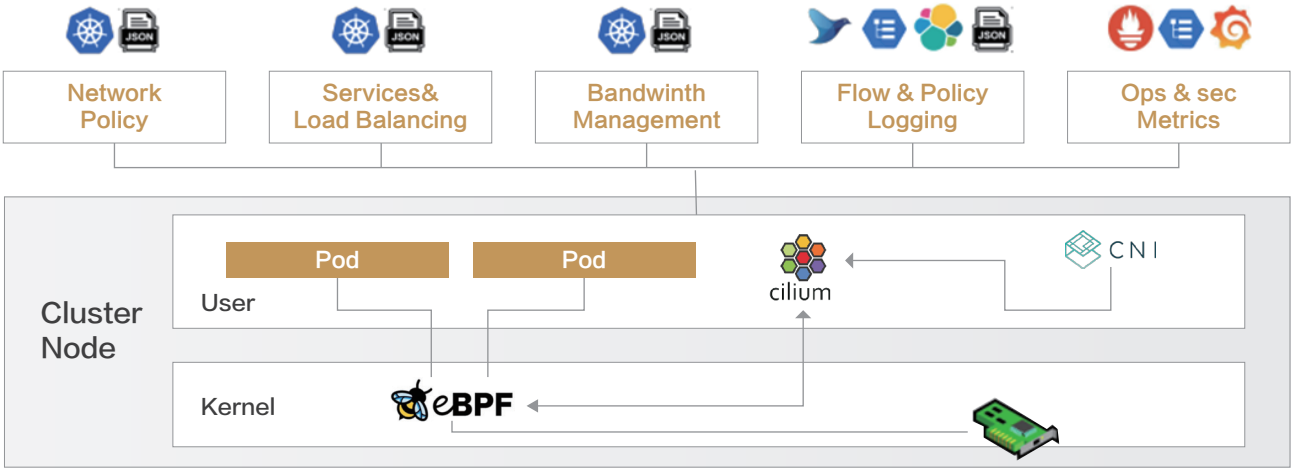


图6: Cilium的架构图（资料来源: Cilium官网）

3 eBPF广阔的商业化空间

除去Cilium主要关注的网络之外，eBPF的另外两大核心场景可观测性，安全也有着极大的市场空间，这也是为什么这项技术在近些年来获得了资本市场的广泛关注。

3.1极具想象力的市场空间

可观测性市场中已经出现了Datadog这样400亿美金的巨头，2021年的收入高达10.3亿美金并且仍然维持70%以上的增速。Gartner预估整个IT运营管理的市场规模在2023年将达到370亿美金，而Datadog也在其S1招股说明书自下而上地预测他们的TAM（Total Addressable Market）约为350亿美金。Datadog也在持续丰富和拓展产品矩阵，在2022年推出的云工作负载安全（Workload Security）中，也开始使用eBPF技术来提供内核级别的数据获取能力。除去Datadog以外，150亿美金市值的Splunk，88亿美金市值的Dynatrace以及30多亿美金的New Relic都是可观测性市场的重要玩家。

图7: Datadog的2021年与2022年产品矩阵对比（资料来源: softwarestackinvesting）
其中2022年推出的Workload Security就发挥了eBPF的价值

网络安全市场同样有着巨大的市场空间，根据Gartner的预估，整个安全市场的规模在2021年达到了1500亿美金，并且仍然维持12.4%的高增速，这个市场也诞生了市值接近500亿美金的CrowdStrike以及市值300多亿美金的Zscaler等公司。

这其中eBPF有很大的潜力将在云安全的市场中发挥价值，尽管目前云安全的市场规模在众多细分赛道中仍然是最小的，但同时也是增速最快的市场，如果维持40%的增速，那么将在2026年达到47亿美元的规模，这种增速也催生了云安全公司的快速发展。比如Sysdig就在2021年4月和12月连续完成了两轮融资，累积融资额也达到了7.3亿美金，他的竞争对手Aqua Security也不甘示弱，在2021年3月和11月分别完成了两轮融资，目前已经披露的融资额已经达到2.7亿美金。

Information Security & Rrisk Manaement End User Spending by Segment, 2020-2021 (Millions of U.S.Dollars)

Market Segment	2020	2021	Growth(%)
Application Security	3,333	3,738	12.2
Cloud Security	595	841	41.2
Data Security	2,981	3,505	17.5
Identity Access Management	12,036	13,917	15.6
Infrastructure Protection	20,462	23,903	16.8
Integrated Risk Risk Management	4,859	5,473	12.6
Network Security Equipment	15,626	17,020	8.9
Other Information Security Software	2,306	2,527	9.6
Security Services	65,070	72,497	11.4
Consumer Security Software	6,507	6,990	7.4
Total	**133,776**	**150,409**	**12.4**

Source: Gartner(May 2021)

3.2 融资与并购热潮

正是由于eBPF技术的快速迭代，发展展现出来的潜力，以及在大的市场中逐渐扮演越来越重要的角色，也出现了一批以eBPF作为公司核心技术与核心理念的公司，在2020年底，eBPF-centric的公司迎来了投融资热潮：

» 2020年10月，基于eBPF的可观测性开源项目Pixie Labs获得Benchmark 915万美金的融资

» 2020年11月，Splunk收购了网络可观测性的公司Flowmill

» 2020年11月，Cilium背后的公司Isovalent获得了来自a16z以及Google，Cisco的2900万美金的融资

» 2020年12月，Pixie Labs被New Relic收购，成为New Relic的产品线

这其中不乏来自上市可观测性公司Splunk，New Relic的并购，以及Google，Cisco等科技巨头的参与，也能看到benchmark和a16z等投资机构的身影，在2020年底，整个eBPF的生态在DevOps赛道可以说是风头无二。

3.3等待爆发的eBPF技术

2021年开始，eBPF-centric的公司需要逐渐在商业上去交出答卷，而eBPF技术的最佳代表，当数Cilium社区与背后的Isovalent公司，他们也同样是eBPF基金会，以及每一年的eBPF技术峰会的组织方。

Cilium项目的创始人Thomas Graf是一名瑞士人，他最长的一段职业经历自2005年到2014年一直在Red hat担任内核研发的工程师，在red hat的最后2年，他也参与了Open vSwitch项目的贡献，这也是“软件定义网络”的重要开源社区，这也让他发现了下一代网络基础设施的重要技术：eBPF，并在2016年初发起了Cilium。公司的CEO Dan Wendlandt则是卡内基梅隆大学的网络方向的博士，毕业后先加入了Nicira，负责Open vSwitch项目的开发并且于2012年因为收购加入了VMware，负责云原生，网络方向的基础设施，也持续在Open vSwitch的项目中积极贡献， 2016年底，看到Cilium的高速发展，他与Thomas共同发起了Isovalent这家公司，在Stealth模式下运行到2020年11月，直到宣布获得了a16z的融资。

在Cilium开源社区的基础上，Isovalent提供了更多企业级的支持，包括基于eBPF将网络和runtime行为与K8S身份认证相

结合，为SIEM以及SOC平台提供云原生的审计、合规检测；将零信任与网络结合起来，基于eBPF的网络实现安全且可以高效拓展的网络连接，在保障节点数增长的同时保障不同节点间的透明性与负载均衡；同时，基于eBPF技术，Isovalent还提供应用的故障排查，帮助用户更高效地收集多租户的metric数据，管理网络策略，自动化网络的监管行为等等。

4 eBPF引领“软件定义内核”

尽管在融资上陷入短暂的沉寂，在Linux内核的开源社区里，eBPF仍然是最热与迭代最快的主题与社群，每一代的Linux内核都逐渐支持更丰富的feature：自3.16版本支持x86_64架构之后逐渐支持了ARM，MIPS直到最近5.x的版本支持RISC-V的架构，eBPF已经逐渐覆盖了主流的计算平台；支持的辅助函数的数量也从3.x的仅仅3个，拓展到4.x的90个，而到了5.x的版本已经支持接近200个辅助函数；而指令数也从4096拓展到了百万量级，这些进步都极大地拓展了eBPF的价值深度与广度。

在eBPF 2021年的峰会上，Isovalent的创始人也对eBPF技术提出了下一代的展望：

1.软件定义网络的新代名词： eBPF会凭借其对于多云，混合云的多层网络的支持，成为云原生时代的SDN（软件定义网络）基础设施，从而更好地链接云-边-端的架构

2.下一代的服务网格： eBPF通过降低边车（sidecar）的复杂性，将成为服务网格（Service mesh）的制胜关键，同时Cilium也推出了其服务网格的Beta测试

3.智能化的可观测性： eBPF通过提供云原生场景下不同层的语义与信息，从而提供更完整与全面的安全 ，从而让可观测性无处不在且更加智能化

新的技术会有需要逐渐被接受，到广泛应用发挥能量的过程，正如同JavaScript对浏览器的影响一样，相信随着易用性，可编程性的逐渐完善，eBPF技术将成为引领并且代言“软件定义内核”的关键技术。

“供应链金融+云原生”化作产业金融数字化腾飞双翼

李洋
广东省CIO联盟会长

一、数字经济蓬勃发展的特点

» **数字基础设施的战略地位日益凸显:** 2020年，国家首次提出“新基建”这一概念，随后，央视列出了“新基建”主要包括的七大领域：5G基建、特高压、城际高速铁路和城市轨道交通、新能源汽车充电桩、大数据中心、人工智能、工业互联网，这些领域都与科技息息相关。

» **数字产业平稳发展:** 数字经济包括数字化产业和产业数字化，以人工智能、大数据、区块链、云计算等为内驱力的数字产业也在推动着产业数字化转型的进程。

» **产业数字化转型稳步推进:** 数字产业的平稳发展为产业数字化转型的推进奠定了基础，金融、教育、医疗、制造、互联网等产业无不随着潮流迈向数字化转型新征程。

数字政府走上发展快车道: 基于便民利民、保民安民的目的，国家大力推进智慧政府与数字政府的建设。

二、数字化转型的关键切入点

数字化转型是一个产业、企业的经营者们在数字经济时代，将其经营领域内的价值创造(含用户体验提升、商业模式创新、业务增长、组织/流程效率提升、风险控制加强等)核心诉求，诉诸新一代网络化、数字化、智能化技术，从而提升其竞争力和实现业务成功的科学、系统化工程。简言之，数字化转型是利用“数字化”工具和手段，通过“转”的战术和路径，实现“型”的战略目标的过程。企业的数字化转型一般可以从以下几方面切入：

» **确定数字化转型战略:** 数字化转型战略的制定要与企业的整体战略和业务战略相契合，有了高瞻远瞩、切实可行的数字化转型战略之后，就要从“1+3+1”这个大的切入点模型来推动数字化转型进程。“1”指数字化生态，“3”指业务数字化、数字化基础设施、数字文化与组织，另一个“1”指数字化治理。

» **业务数字化:** 业务数字化是数字化转型的核心，包括营销、研发、生产、服务、供应链、风控、运营七大方面。几乎所有的行业都可以从这七个方面切入，进行数字化转型。对于新零售行业，重点从营销方面切入；对于制造业来说，智能制造是不可逆转的行业趋势，研发生产环节可以作为业务数字化的切入点；对于地产行业来说，营销和服务至关重要，数字化营销、数字化客服、数字化售后是这一行业不可或缺的；对于产业金融来说，数字化供应链、风控、运营，都是重要的切入点。

近年来，随着国家对数字经济的重视，数字基础设施的战略地位日益凸显，以人工智能、大数据、区块链、云计算等为代表的数字产业平稳发展，同时驱动各行各业的产业数字化转型。为响应国家的号召、顺应时代的潮流，无论是国企央企，还是民营企业都将数字化转型放到了战略地位，数字化转型如火如荼，数字经济蓬勃发展。

» **数字化基础设施：**技术是数字化重要的基础设施，企业要重视流程组织和技术，达到降本增效与数字化转型的目的。

» **数字化文化与组织：**数字文化与组织习惯的养成有利于数字化理念的形成。

» **数字化生态：**各行各业的企业都不能“自扫门前雪”，不管“他人瓦上霜”。企业要平稳运营，也要协同上下游产业链，甚至跨产业构建数字化生态圈。

三、 产业金融的数字化

产业金融是在现代金融体系趋向综合化的过程中出现的依托并能够有效促进特定产业发展的金融活动总称。产业发展对资金的需求犹如人体对血液的要求，而金融在提供资金来源方面具有决定性作用。产业是基础平台，金融是作为催化剂和倍增剂而存在的，金融与产业互动满足客户需求，创造新的价值，大大加快财富累积。从资本的角度做产业，产业的财富放大效应会迅速增加；而金融只有与产业融合才能产生放大效用，产生更大的价值。以海尔集团为例，作为世界500强之一的海尔旗下拥有众多产业，为了使各产业有序健康发展，海尔搭建了“金融+科技”平台，发挥金融引进资金的作用，满足产业资产端的需求。

将新时代创新性关键技术与产业特色相结合，落实国家发展战略，提升产业链运转效率，构建多方参与的“区块链+产业链“分布式协同生态，对推动数字经济与新基建具有重要意义：

» **落实国家发展战略：**贯彻落实习近平总书记指示、落实《粤港澳大湾区发展规划纲要》、推动《广东省重点领域研发计划实施方案》；

» **深入研究区块链技术以及产业应用：**信息化培育发展新动能、研究区块链技术及应用，打造产业集群、大力发展数字经济，结合地方特色；

» **提升供应链管理标准化、规范化：**构建跨产业供应链管理标准化体系、促进供应链创新与应用，提高现代供应链管理的标准化、规范化；

» **推动供应链金融服务实体经济：**发展数字经济、连接金融与实体产业、让金融服务于实体经济、缓解中小企业融资困境；

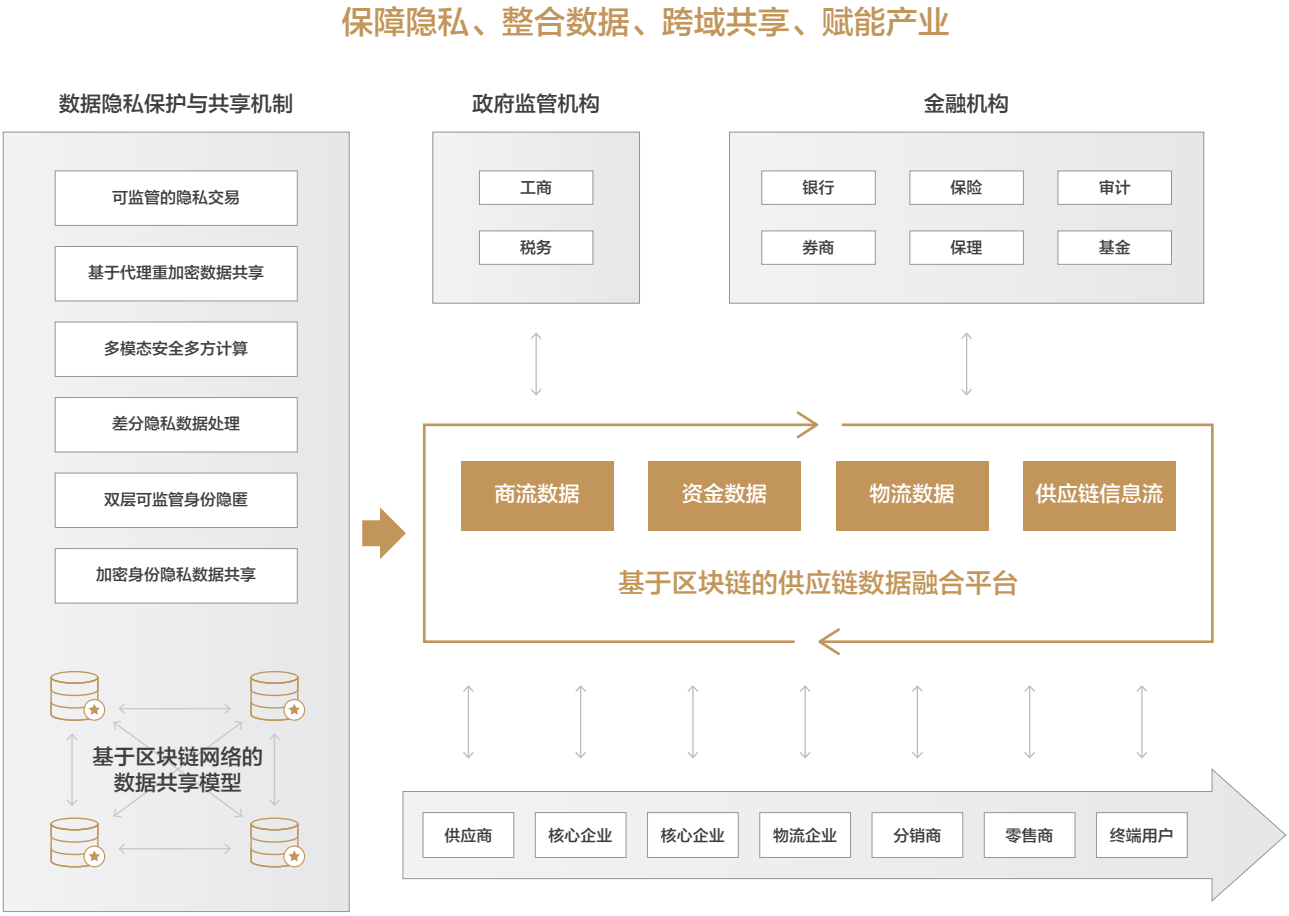
» **落实国家服务中小微企业的金融政策：**降低融资成本，减轻企业经营负担、供应链上实现“多流合一”、缓解中小微企业“融资难，融资贵”问题。

四、产业金融业务数字化实践：供应链金融平台

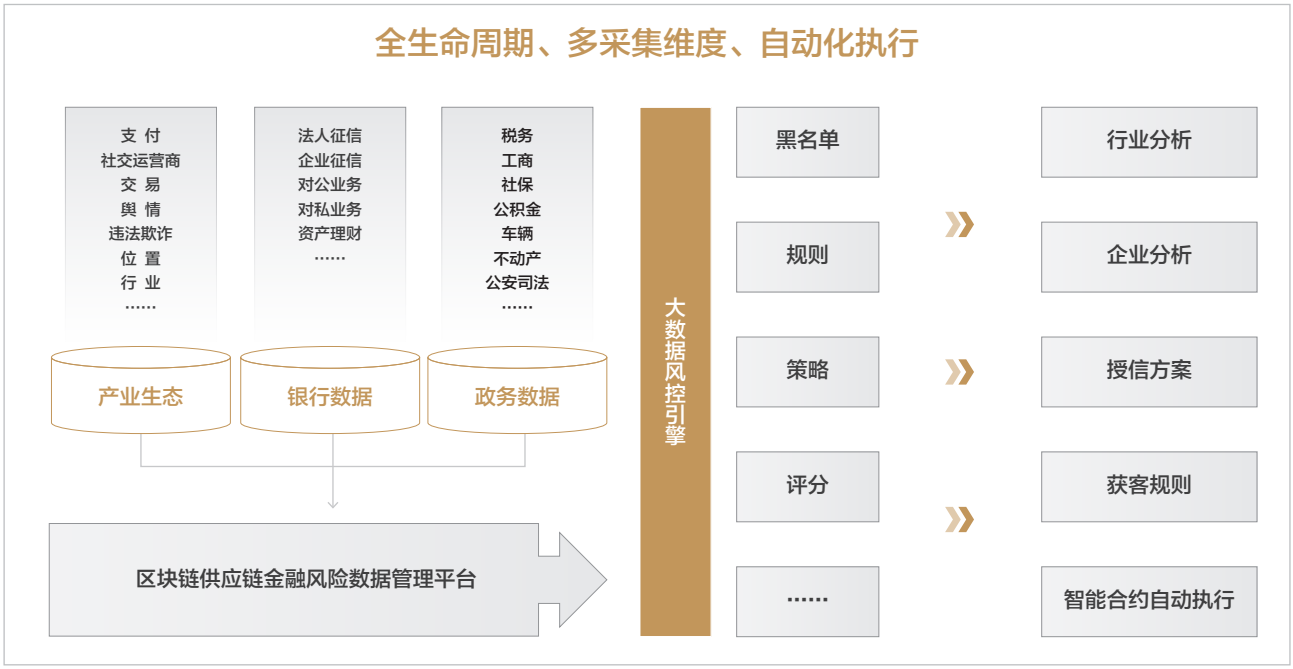
建立基于国产密码的安全自主可控的高可扩展、高性能区块链基础服务平台，此平台具备快速构建上层应用业务的能力，能够满足大规模用户的使用场景需求，增强区块链协议及工作机制的安全性。此平台运用安全高效的共识算法、并行多样的多链分片，使其具有诸多性能与可扩展性；拥有国产密码的安全保障、可靠的隐私保护与权限控制；应用开发友好的智能合约和可扩展的同构/异构跨链互操作，具有商用级的产品成熟度。

基于区块链构建供应链的四流合一

供应链金融平台是基于商流、资金流、物流、信息流四流合一搭建的，这四流代表了供应链的上下游，也就是所谓的利益攸关方。企业经营离不开延产、供销存、采购，可以借助供应商核心企业的信用背书将供应链上下游的资金、资产聚合到供应链金融平台。在这个过程中，货物需要存储、运输，这就要发挥物流企业、分销商、零售商、终端用户的作用了。在四流合一的场景里，还有包括工商、税务在内的政府监管机构以及包括银行、保险等在内的金融机构作为利益攸关方。构建基于区块链的供应链四流合一数据管理场景时的一个重要技术前提是建立数据隐私保护与共享机制。基于区块链网络的数据共享模型与基于区块链的供应链数据融合平台共同保障用户隐私，整合数据，跨域共享，赋能产业。



基于区块链的供应链全生命周期风控



基于区块链的供应链物流与仓储数据管理

供应链的采购、物料管理、生产、营销、配送、国际物流、生产外包等环节是在大数据、人工智能、物联网、区块链技术的基础上进行的，这些技术中心汇集了大量数据。在物流与仓储数据管理的过程中，可以打通多中心间的数据壁垒，实现数据实时共享同步；通过RFID及电子标签等物联网技术确保数据与物品一致；确保区块链数据不可撤销、不可篡改、真实可靠。

基于区块链的供应链全生命周期风控

供应链中存储了海量数据，包括产业生态、银行数据、政务数据等，不论对公对私，这些供应链金融风险数据都将放入大数据风控引擎，利用黑名单、规则、策略、评分等机制对供应链上下游核心企业进行信用评估，对上下游具有融资需求的客户进行行业分析、企业分析，制定合适的授信方案，衍生获客规则以及智能合约自动执行规则，此系统有效保障了供应链全生命周期风控。

五、云原生助力产业金融业务数字化实践

供应链金融平台是由科技线与业务线共同打造的，此过程需要云原生助力降本增效。为了搭建供应链金融平台，加快数字化转型，企业需要打造具有业务敏捷性的组织。类似于赛普提出的框架，这个组织要求企业做到业务敏捷，而不仅仅是科技敏捷。业务敏捷是指企业能够敏锐洞察业务需求的变化并有快速应对这种变化、交付业务所需的技术平台或系统的能力。云原生的数字化基础设施正是基于业务敏捷组织必备的核心能力构建的。

供应链金融打造了一个以贸易关系为枢纽的价值生态圈，其作为核心企业为疫情阶段的上下游客户提供了融资需求。同时，也可以以SaaS的方式对外出租供应链金融平台以实现平台利用最大化。云原生技术的运用则为其他产业数字化转型提供了样板。

梦饷集团：电商业务平台在云原生数据库方面的应用实践

许锋涛
梦饷集团首席DBA

制造端和供应链端的升级是新消费品牌诞生和快速成长的基础，得益于电商平台拥有的大流量和便捷交易的优势，社群新电商模式的梦饷集团也成为助力新消费品牌成长的重要力量。梦饷旗下拥有饷店、爱库存、爱豆学堂等业务，累计帮助10000多家品牌商销售近4亿件商品，触达数亿消费者，为超200万店主提供创业创收的机会。

为了满足日益增长的客户需求和实现更大的商业价值，梦饷集团联合华为云进行了一场平台数字化升级，基于华为云云原生数据库提供的智能运维、敏捷开发、高效分析等能力，全新打造领先的众包分销平台。

一、业务痛点：业务剧增带来升级契机

随着业务的品类升级，梦饷集团将持续加码全品类、全渠道、全开放，并深耕精细化运营。如何精准识别店主、KOC等渠道与品牌商家的销售需求，如何将上游制造商与消费者深度链接，让平台赋能效应最大化，以及如何安全高效应对平台日益剧增的数据量是当前需要解决的问题。日益增长的客户需求对数字化基础设施的电商平台提出了高要求，因此，数字化升级刻不容缓，作为数据底座的数据库，也有着更高的要求：

» **稳定：**数据层是电商的核心资产，必须提供稳定可靠的服务。因此，无论是在功能性能还是资源弹性，数据库必须在随着数据量增加的同时，始终确保系统能稳定运行。

» **高效：**梦饷集团的数据种类复杂，并发压力大，复杂维度下需要对海量数据进行单表的高并发查询。因此，新数据库要满足支持4TB以上的数据和1600以上并发的查询，响应时延控制在200毫秒以内。

» **敏捷：**电商大促或爆款抢购时，扣减库存无ACID事务，如

果扣减失败，只能由业务处理。因此，库存扣减ACID事务最好由数据库保证，业务无需处理。

» **智能：**后台监控数据写入频繁，同时需要基于时间线对监控数据进行分析，所以亟需一款快速写入和兼顾分析功能的时序数据库。

二、拥抱云原生，全面推进数字化

云原生是大势所趋，越来越多企业基于云原生能力创新服务，梦饷集团也希望通过云原生技术，实现更智能高效的人货匹配，因此选择了华为云云原生数据库GaussDB系列产品。

提供高效分析能力：百万级写入、毫秒级响应

梦饷集团新上业绩分析平台，需要数据库基于海量数据做高并发的多维度分析。华为云GaussDB(for MySQL)凭借最高可达145万+条/秒的写入速度，毫秒级的响应时间、分钟级的添加只读节点，以及小于50毫秒的主从实例复制延迟时间，完全满足梦饷集团的高效稳定需求。

为敏捷开发提供更多保障：支持ACID事务，数据强一致

梦饷集团业务发展日新月异，敏捷开发迭代是业务拓展的重要保障。华为云GaussDB(for Redis)支持ACID事务，保障库存扣减结果一致；支持3AZ强一致，所有节点均可读写，无主从不一致风险；存算分离架构确保内存数据可定期持久化，添加计算节点分钟级完成，扩充存储秒级完成，无数据再均衡过程，为敏捷开发提供了更多的保障。

智能监控和运维：万亿级写入性能、分钟级扩展

梦饷集团后台监控数据写入频繁，需要对时序数据进行实时监控和洞察分析，避免运维风险。华为云GaussDB(for Influx)提供了分钟级弹性伸缩，每天万亿条写入性能，优于开源InfluxDB 2-5倍性能的多维聚合查询，支持亿级时间线，海量时间线下性能更稳定，为梦饷集团提供智能高效的运维能力。

此外，华为云还针对梦饷集团对数据高可靠、高可用、高性能、高安全、高效率等方面的诉求，提供了RDS Proxy+RDS HA+跨Region备份的组合解决方案，全链路从写入到查询响应时间基本是秒级，故障场景下提供主库和备库秒级故障转移能力，支持跨Region容灾备份，提供更高的安全保障，用技术创新支撑去中心化的电商平台交易更稳定高效。

三、上线后效果

梦饷集团业务搬迁至华为云后，运维效率提升约30%，数据链路的时效性以及业务系统的吞吐能力升1倍，核心业务数据库访问平均耗时由1.5s降至1s。此外，梦饷集团每秒成交的订单数再创历史新高，订单峰值比历史最高峰值再次提高了几倍，成为了利用数字基础设施助力业务快速增长的标杆。

梦饷集团基于华为云云原生数据库的能力，构建了智能运维、敏捷开发、高效分析等平台优势，为业务大促保障、需求落地、风险识别、性能优化、定期巡检等场景提供了有力保障，平台的升级也让商家拥有了更大的自主运营空间、更灵活的销售与活动模式，有效的降低仓储物流等成本，为每个店家成就梦想。

极光: 推送业务无中断迁移上云实践

曾振波
极光推送后台技术专家

关于企业上云，业内已经有了非常多的讨论和论述。这里主要是从极光自身的实际情况阐述几个理由:

» 传统自建机房在扩充底层软硬件资源时，需要进行选型、采购、参数测试验证、实施部署等流程，整个过程需要消耗很多的人力和时间，对于快速发展的业务来说是很大的负担。云服务可以极大的缩减整个流程，对于部分云服务例如云主机可以实现分钟级别的资源交付。

» 自建机房需要投入高额的硬件资源准备，包括机房配套基础设施、服务器、网络、安全设备等，大量的冗余资源闲置，整体资源利用率不高。上云可以实现按需购买使用，实现更高的资源利用率。

» 基础设施建设和维护需要投入大量的人力和精力，往往还吃力不讨好。特别是虚拟化方面一直以来都是极光的痛点，资源隔离做得不好很容易受到其他虚拟机的影响，往往因为某个业务的突增影响同一个物理机上的所有虚拟机。云厂商有庞大的专业团队进行建设和维护，各方面相当可靠。

» 云厂商提供成熟稳定的PaaS层服务可以进一步释放我们的精力，让我们更专注在我们的业务，例如天然支持多AZ的RDS可以为我们在考虑同城双机房架构时提供很大的助益。

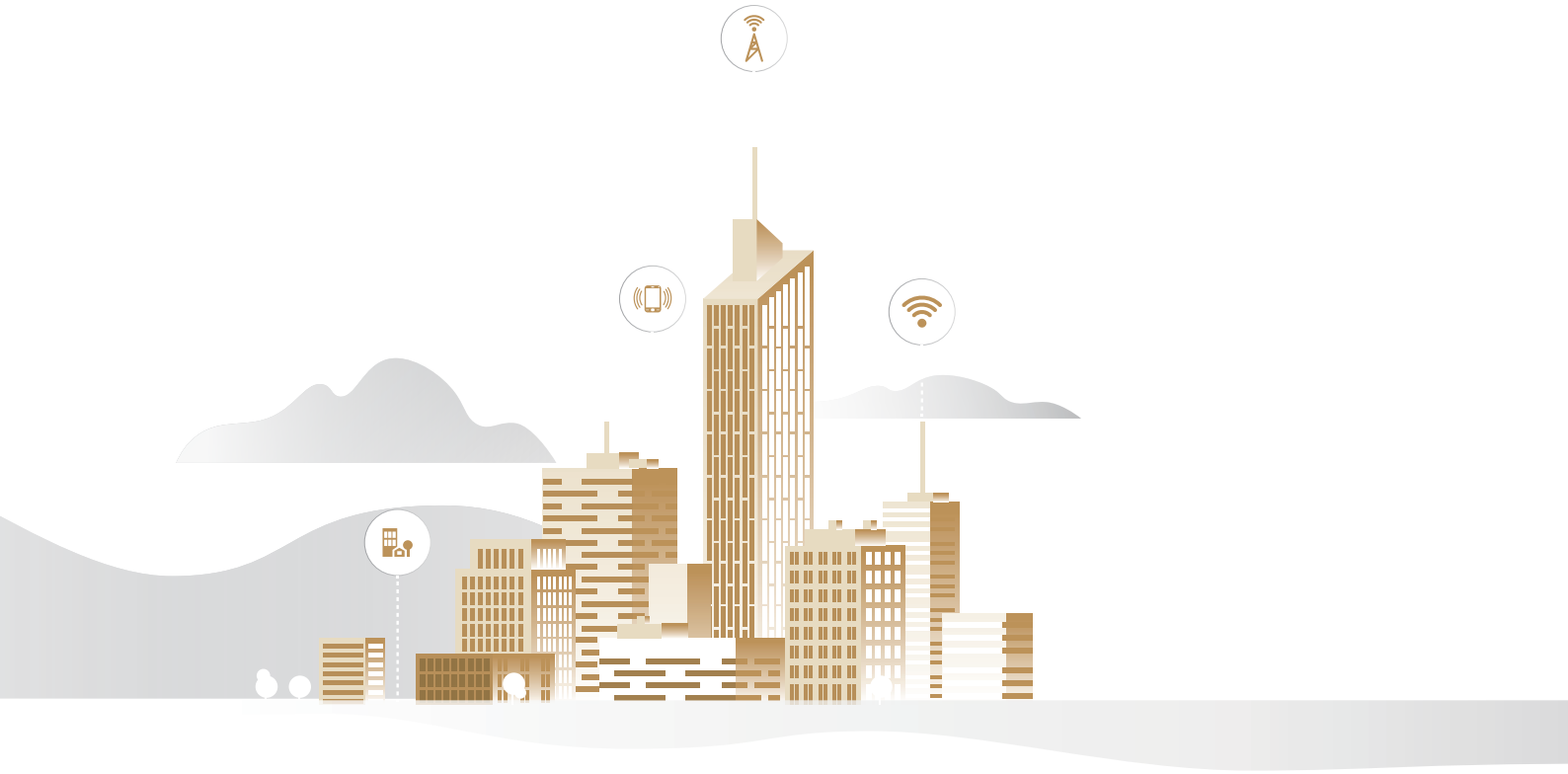
极光机房架构变迁历史

早期极光只有一个单一的机房，随着业务的发展，系统规模越来越庞大，单一机房的资源不足以支撑极光的业务。因此我们将业务系统迁移到了新建的机房，数据业务继续保留在原有机房，整个过程磕磕碰碰历时一年左右。

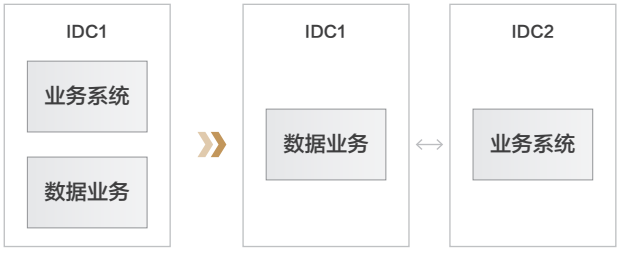
后来业务系统所在的机房再进一步优化，在同城增加一个机房，并用专线进行互通，迟延在1ms到2ms之间，并将部分子系统迁移到新的AZ。由于我们的业务量级非常大，部分子系统的QPS超过了百万，在业务峰值偶尔出现延迟增加的情况，

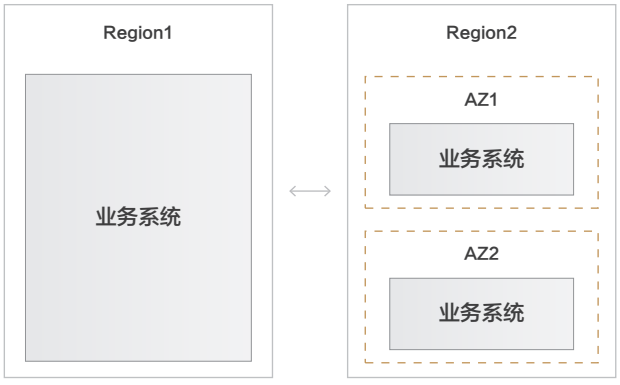
因此也做了相关的调整，访问量大的子业务系统尽量不跨AZ进行访问。此时的多AZ，并不是每个AZ都有完整的业务流程，仅仅形成一个大内网，在部署的时候进行优化处理。

由于机房仅有单一的网络出入口，带宽也有限，很容易受到同机房的其他客户的影响。曾经出现过出口带宽用满甚至整个出口中断的情况，业务受到严重的影响。我们也考虑了异地双机房、单机房多网络出口方案，但是这些方案仅仅是针对性的解决我们的一些问题，没有系统性的解决我们当时的困境，因此这两个方案并没有真正意义的实行。同时内部也



在考虑上云的方案，外加一些外部因素，上云的方案就推到了首位。至于云厂商的选型此处就不做陈述，最终选择了华为云。





极光推送的业务特征分析

极光推送为开发者提供服务，一个开发者可以有多个Appkey也就是多个应用，每个Appkey的全部数据互不相关，一个Appkey有多个终端设备用户。累计终端用户超过500亿，同时还有各个维度的数据，例如tag、alias等等，单副本数据总量超过80TB；月活跃终端用户超过5亿，各个API接口请求总量超过5万QPS。使用超过2万核CPU，超过2500台虚拟机来支撑这些业务。

对外有2类网络通信：极光推送业务和开发者服务的通信，主要形式是RestfulAPI；极光推送业务和应用的通信，主要形式是基于TCP长链接的自定义协议。

自建机房/华为云基础设施分析

推送业务部署在虚拟机和K8S上，这里主要分析对比虚拟机的CPU、网络、磁盘的相关指标，以及K8S网络的指标。

从物理机看，自建机房的物理机相对华为云目标AZ的物理机性能低一些，例如华为云的物理机使用更高主频更高配的CPU。在虚拟化层面，华为云的虚拟化做得更好，资源隔离

更加严格，提供各种规格的云主机和磁盘，从整体上来说计算能力更强，但是在网络和磁盘IO吞吐和QPS有严格的限制，需要做好规划。经过测试对比，选择了相关规格的云主机和磁盘。

自建机房-云环境架设专线，云主机和自建机房机器之间的RTT在5ms以内，常规情况下为2-3ms，自建机房内网机器之间RTT为0.2ms左右，同一AZ云主机之间RTT为0.2-0.3ms。

在K8S网络方面，自建机房做了相关的优化，通过专用的网络设备能够使用Underlay的路由模式，可以说是目前可用的原生网络模式中性能最好的模式。华为云自建K8S集群仅仅支持Overlay模式，性能相对差了一些；同时也提供了K8S服务，通过硬件加速等优化提供了较好的网络性能。

上云方案的选型

上云有几个需要考虑的要素：

» 业务无中断迁移，尽量不影响客户的使用，尽量不需要客户做任何变更。

» 迁移前后业务功能一致，需要保证数据和业务的完整性。

» 需要考虑切换过程中极端情况导致的回滚操作，并且需要保证数据和业务的完整性。

基于以上几点，在方案选型方面，我们主要考虑2个方案：

方案一：自建机房和云环境是2套独立、隔离的环境，关联的仅仅是自建到云环境的数据同步，业务上相互隔离。以Appkey为单位，迁移Appkey所有的数据和业务。数据通过专线进行迁移同步，同时尽量保证原有自建机房数据完整，最好能够数据双向同步/或者数据双写，方便极端异常情况下的业务回滚，至少保证能够回滚后业务正常。

方案二：自建机房和云环境通过专线连接起来后，形成一个大内网。将数据耦合度比较低的子业务单独切换到云环境；对数据耦合度比较高并且访问量/访问迟延要求高的子业务，需要都跟随数据一起迁移。内部业务系统逐渐迁移切换完成后再对入口进行整体切换。

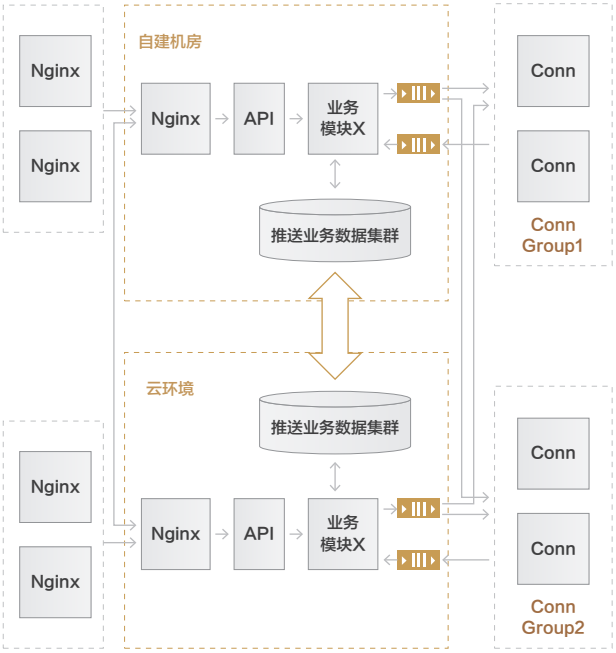
2个方案均能实现业务无中断，同时各有优缺点，方案一需要额外开发少部分数据同步/恢复工具，前期准备工作充分的情况下，可以比较简单快速的切换；方案二不需要开发额外工具，但是需要操作的模块多，操作时间长，切换相对复杂，容易出现差错；综合考虑下选择了方案一，尽量保证切换过程

	方案一	方案二
前期准备	相关业务数据的同步工具开发； 业务及数据的全面梳理，保障业务和数据整体迁移。	业务梳理，确定业务数据耦合度和访问要求， 确定迁移模块/子业务的顺序。
数据迁移	组件同步，业务工具同步	组件同步
流量切换	按照Appkey切换	按照模块/子业务进行迁移
业务中断	切换瞬间长链接会断开， 极光通道推送可能会 失败，有离线下发功能	无业务中断
切换复杂度	低	高
切换时间	中	长
切换数据 一致性/完整性	BASE	BASE
回滚	将Appkey修改切换回原来的机房即可。	重新将模块/业务原路切换回去即可。
回滚数据 一致性/完整性	BASE	BASE

简单无差错。

上云的详细方案

自建机房和云环境拉通专线进行数据同步，从业务层面来说，两个机房各自承载全部的业务数据，为了方便故障回滚，各个数据项尽可能的保持双向同步，保持数据最终一致性即可；两个环境的推送业务是相互独立的，先保证全量数据同步到云环境，以Appkey为单位进行流量迁移，将Appkey的流量迁移到云环境，迁移期间各自承担一部分Appkey的推送业务，最终将全部流量迁移到云环境。



部署方案

为了快速迁移，采用1:1对等资源部署的方式即云环境部署一套和自建机房同等资源的系统，涉及业务模块、存储集群、依赖组件、监控体系等。同时新建另一套内部域名跟原有域名作区分，对外域名不进行变更，在迁移的最后阶段再进行变更切换。

在系统入口的部署做了特殊的处理：

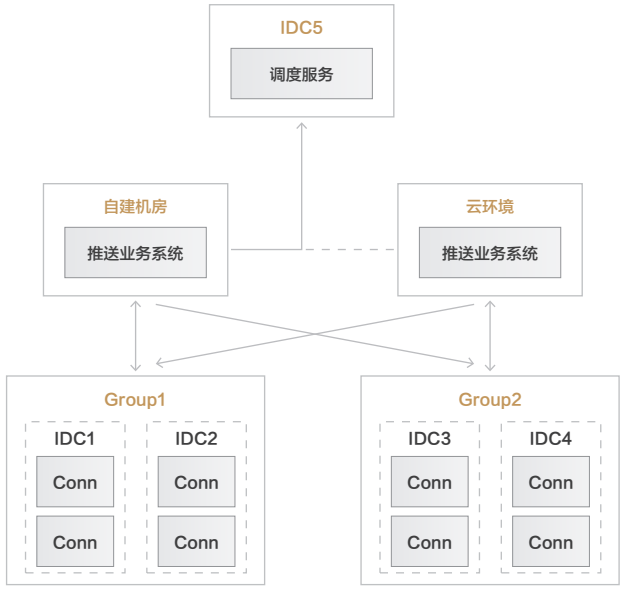
» **API入口** - 部署同等规模的API服务器以及前端Nginx，由于对外域名只有一套，只能在自建机房和云环境做二选一，请求流量都进入到自建机房，在Nginx的Lua代码中判断请求信息，根据Appkey归属信息决定是否转发到云环境；同时新建备用域名指向云环境的入口以备异常情况使用。

» **SDK接入网关入口** - 接入网关分成2个集群，各自服务自建机房和云环境，同时接收另一个机房的下行数据；SDK先连接到调度服务，根据Appkey归属信息分配到相应的接入网关集群，同时调度服务跟自建机房互通，最后再迁移到云环境。

为了快速部署，并且避免遗漏某些业务模块或者组件缺失，也为了避免配置错误，我们整理了所有的机器列表以及相关信息例如IP，将自建机房的机器信息和云环境的机器信息一一对应起来，当然还包括域名信息也进行一一对应，在部署的时候对着这些信息进行配置和部署。

数据迁移方案

推送业务的数据存储涉及ES、CouchBase、Redis、PIKA、



MySQL，需要把全部存量数据同步到云环境，同时建立实时同步通道进行同步增量数据，保证云环境的数据最终一致性。

数据同步方式为组件工具同步、业务双向同步，确保迁移整个过程数据在2个机房的完整性和最终一致性。专线的拉通，使得2个机房之间RTT为2-3ms，为数据全量迁移和增量同步提供了非常强的支撑。

根据各存储组件，我们先预研了通用的迁移方案：

» **ES同步方式1：** 云环境新建集群，拷贝源集群的数据文件到新集群，完成存量数据的迁移；增量数据由程序写入（即由业务实现数据在2个集群的双写）；使用脚本工具补充切换窗口的数据。

» **ES同步方式2：** 新增云环境节点加入到集群，逐步剔除自建机房节点，即云环境和自建机房当成同一个内网，简称大内网模式。

» **CouchBase同步方式1：** 使用自带集群同步工具XDCR进行同步。

» **CouchBase同步方式2：** 存量数据使用业务工具导入，增量数据由程序写入（即由业务实现数据在2个集群的双写）。

» **CouchBase同步方式3：** 大内网模式。

» **Redis同步方式1：** 存量数据使用redis-shake或者主从同步，增量数据由程序写入（即由业务实现数据在2个集群的双写）。

» **Redis同步方式2：** 存量数据使用业务工具导入，增量数据由程序写入（即由业务实现数据在2个集群的双写）。

» **PIKA同步方式1：** 存量数据使用主从同步，增量数据由程序写入（即由业务实现数据在2个集群的双写）。

» **PIKA同步方式2：** 存量数据使用业务工具导入，增量数据由程序写入（即由业务实现数据在2个集群的双写）。

» **MySQL同步方式1：** 存量数据使用主从同步，增量数据由程序写入（即由业务实现数据在2个集群的双写）。

» **MySQL同步方式2：** 双主复制

以上为几种通用的迁移方式，但是每个数据集群实例的特性不一样，从业务依赖程度、数据量、读请求量、写请求量几个维度评估，最终采取的迁移方案也不一样，我们梳理了自建机房的所有数据集群实例列表，对存量数据迁移、增量数据同步、切换方式、数据一致性需要的时间、切换操作、数据验证等都做了详细的评估和说明。

测试方案

测试分为功能测试和性能测试两部分，这两部分都使用我们自己的内部账号进行测试，先进行功能测试，在功能都完备的情况下再进行压力测试。

在平时的开发过程中我们积累了大量的测试用例，覆盖到了全功能和内部细节，整理这些测试用例构造测试数据并执行，从而实现功能测试的目的。

对于压力测试，原计划的压力测试方案是：

1. 在压力测试的过程中，切断云环境写入数据机房以及自建机房的数据链路，避免测试数据污染线上系统。

2. 执行压力测试，确认压力测试的结果满足要求。

3. 清理云环境由于测试过程导致的脏数据。

4. 恢复第一步被切断的数据链路。

5. 重新进行相关数据的数据同步。

6. 由于第四步和第五步造成了数据和系统的变更，需要再次进行功能测试。

这个方案比较复杂，并且执行时间会比较长，特别是数据同步消耗比较多的时间，因此我们根据业务特性重新调整了压力测试的方案。测试以Appkey为单位，并且系统中各个Appkey是相互独立的，基于测试Appkey进行测试产生的测试结果数据仅属于Appkey本身，并且只对系统的整体运营数据有影响；测试Appkey也当做正常的Appkey存在在系统中，相关数据也不需要进行清理，后续如果有需要可以继续使用这些测试Appkey。因此决定在功能测试之后，构造压力测试数据，然后直接进行测试，并且对整体运营数据做清洗过滤。压力测试场景和测试用例由业务团队和测试团队根据业

务特征和系统特性来构造，覆盖所有的核心功能和核心模块，压测结果数据至少不低于当前业务的峰值。

整个测试在存量数据同步完成并持续同步增量数据之后进行，主要是考虑在做压力测试的时候存储集群有等量的数据量才能使压测结果更加接近原有系统。

流量切换方案

在功能完备、数据完整的情况下，迁移操作非常简单，执行脚本，修改Appkey归属信息就可以了，具体内部逻辑如下：

» 修改Appkey归属信息。

» 在API请求入口判断Appkey的信息，将流量转发到云环境的入口，后续所有流程都在云环境执行。

» 调度服务器获取Appkey归属信息，SDK新的请求返回新的接入网关集群信息，连接到正确的接入网关服务器。

» 调度服务器通知接入网关服务器断开不属于该集群的Appkey SDK链接。

» API入口和接入网关入口变更有个时间差，2个机房的业务逻辑都能够完整执行，并且有数据同步，不管是SDK连接到哪个接入网关集群，都能够接收相关数据。

执行迁移操作后，需要进行验证，包括但不限于以下部分：

» 基础监控是否正常（网络/CPU/内存/磁盘等）。

» Prometheus业务监控是否正常。

» 推送业务运营数据是否正常。

编号	任务	开始时间	结束时间	关键子任务	结果
1	第1批迁移切到5% 活跃-非vip +5%	3月3日	3月3日	迁移脚本执行	完成
				pushapi确认迁移成功	完成
				sis确认迁移成功	完成
2	业务运行状况检查	3月3日	3月7日	基础监控是否正常（网络/CPU/内存/磁盘等）	完成
				Prometheus业务监控是否正常	正常
				iPortal推送业务数据是否正常（日新增/日在线/日推送量）	正常
				接入机器节点是否正常（是否能够承载响应等连接数）	正常
3	第2批迁移切到15% 活跃-非vip+++vip (6个vip) +10%	3月8日	3月8日	迁移脚本执行	完成
				pushapi确认迁移成功	完成
				sis确认迁移成功	完成
4	业务运行状况检查	3月8日	3月8日	基础监控是否正常（网络/CPU/内存/磁盘等）	正常
				Prometheus业务监控是否正常	正常
				iPortal推送业务数据是否正常（日新增/日在线/日推送量）	正常
				接入机器节点是否正常（是否能够承载响应等连接数）	正常
5	第3批迁移切到50% 活跃-非vip +35%	3月9日	3月9日	迁移脚本执行	完成
				pushapi确认迁移成功	完成
				sis确认迁移成功	完成
6	业务运行状况检查	3月9日	3月10日	基础监控是否正常（网络/CPU/内存/磁盘等）	正常
				Prometheus业务监控是否正常	正常
				iPortal推送业务数据是否正常（日新增/日在线/日推送量）	正常
				接入机器节点是否正常（是否能够承载响应等连接数）	正常

整个推送业务体量非常大，很难一次性全部切换，为了保证迁移过程有序稳定的进行，我们按照一定的策略和迁移比例制定迁移计划，分批次逐步迁移整个系统，每批次操作完成后都进行验证和观察一定的时间。优先测试账号并进行回滚验证，其次是非VIP，最后是VIP。

迁移实施管理

整个迁移过程，我们建立了实施过程跟踪，每天跟进当前的进度，下一计划步骤的工作任务，有哪些依赖工作，当前有哪些风险并且由谁来跟进解决等等，尽量确保迁移工作计划持续有序的执行。

尽管我们做了详细的方案，在实施过程中难免会碰到一些问题，我们尽量快速分析定位问题，直接解决问题或者方案微调，在风险可控的范围内解决问题，这里摘选几个问题陈述一下：

» 有一个CouchBase Bucket实例在实施过程中发现有分钟级别的数据不一致。经过分析发现不一致的数据都有主动删除的操作，CouchBase在删除时并不是真正的删除，仅仅是标记为删除，然后在后端线程异步执行数据删除。CouchBase采用XDCR进行跨集群数据同步，可能是在数据同步过程中，删除操作未能及时同步。考虑到该实例的数据访问量级并不大，跨专线的访问时延并不影响到业务，因此进行方案的调整，仅保留云上的实例，自建机房和云上的系统共同访问。

» 有一个Redis实例偶尔出现CPU负载增高的情况，自建机房的实例观察正常。在此期间Redis没有进行数据备份，业务访问量也相对平稳，虚拟机并未受到同台物理机的其他虚拟机的影响；分析日志发现CPU负载增高时，Redis有内存碎片清理的动作，比对相关的配置发现配置不一样。怀疑内存碎片清理消耗过多CPU，经过调整Redis配置，该情况不再出现。

» 业务操作MySQL时偶尔出现迟延增加甚至超时。通过监控发现期间有业务突增，但是业务量属于正常范围内，并且自建机房的MySQL访问正常；虚拟机/物理机负载正常，业务机器/MySQL机器的网络IO/磁盘IO均正常，MySQL各项数据也正常；再细看业务机器的基础监控发现网络丢包重传相对增加，怀疑网络链路有异常，经过华为云团队排查发现网络设备的光模块异常。更换光模块后业务恢复正常，该情况不再出现。

上云后的规划

上云不是终点，而是另一个起点。虽然已经将业务迁移上云完成，但是依然有很多工作需要做。云厂商提供了稳健的IaaS层，也提供了很多PaaS层服务，充分利用云资源的优势，拥抱云原生，持续演进优化我们的架构，为我们的客户通过更加优质的服务。以下是上云后的一些工作：

» 持续优化现有运行在虚拟机上的系统，提升系统资源利用率。

» 继续推进全面容器化，在尽量保证业务互不影响的情况下，采用混合部署的方式，进一步提升资源利用率。

» 采用云厂商的服务优化替换现有自建服务，例如文件存储等等。

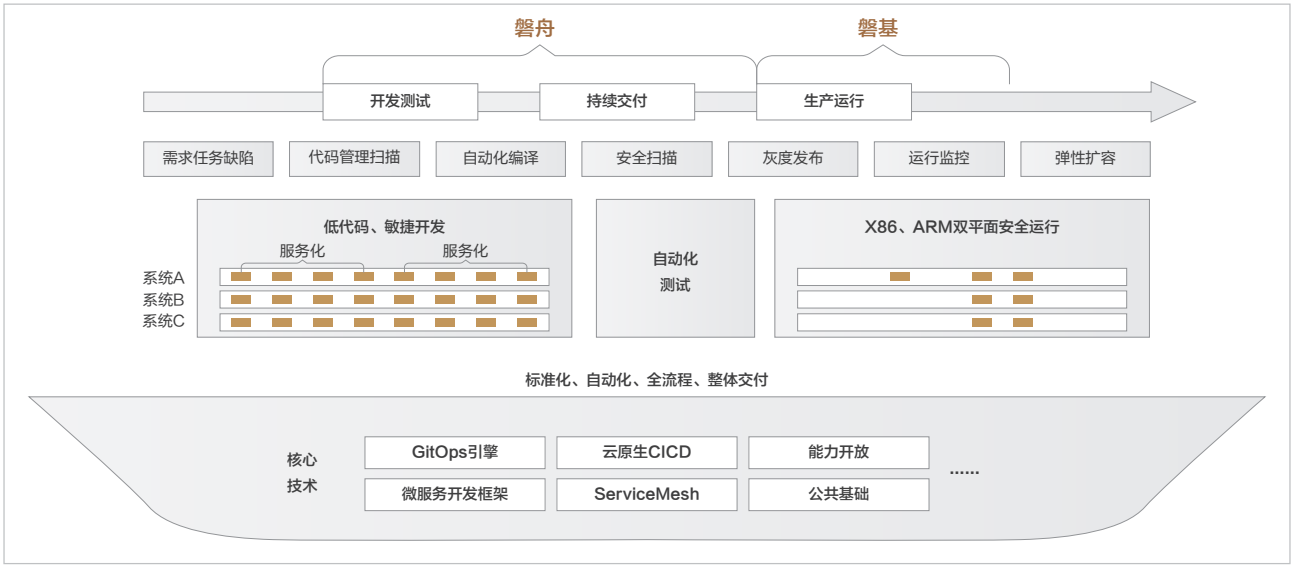
» 进一步考虑多AZ甚至多Region的架构。

中国移动：磐舟磐基平台基于KubeEdge的落地实践

中国移动磐舟磐基团队 & 华为iSula团队 & CNCF KubeEdge团队

磐舟一体化云交付平台是中国移动自主研发的面向开发人员的代码开发，自动部署的平台。磐舟一体化交付平台自研实现了一套GitOps驱动引擎，支持从需求设计、开发构建、测试部署的全部开发与运维功能需求，实现应用一键上磐基容器云平台。

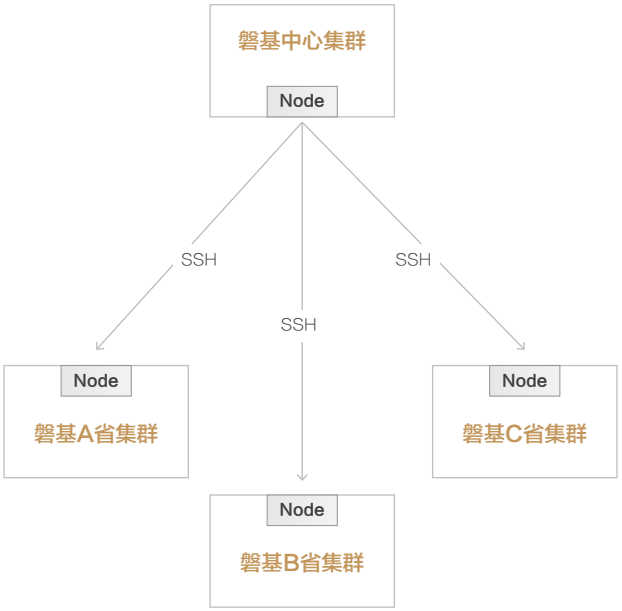
磐基容器云平台是中国移动信息公司基于Kubernetes构建的企业级PaaS解决方案，实现Kubernetes能力的标准化封装及调用，包括提供开发和运行环境、资源弹性伸缩、精细化微服务管理、便捷一站式服务、跨地域多集群调度和智能监控维护等六大能力。



磐舟和磐基是相互配合的，开发人员在磐舟集群上开发，部署到磐基PaaS集群上运行应用，也支持在磐舟上归档磐基集群ops配置，通过GitOps来管理、部署磐基集群。

随着国产化进程推进，中国移动建设了大量的国产化服务器集群，磐基磐舟如何实现国产化的容器云开发交付一体化体系？在某资源池我们需要统一管理近500台鲲鹏服务器，源码可以通过磐舟统一编译为X86/ARM双架构的镜像，但是集群的管理也需要实现ARM自动化支持，开发交付环节频繁使用Kubernetes集群，最近2个月已有800多次的集群创建回收动作，人工支撑显然已经跟不上云原生的发展速度了。

另一个场景是，移动的开发人员在集团磐舟Kubernetes集群上进行开发，制作好镜像后，不能直接推送到省测公司的Kubernetes集群，需要运维人员在磐基中心集群上通过多级ssh跳板机，手工登录到省公司磐基K8s集群进行部署。这一步没有实现自动化，操作流程十分繁琐。



想解决这些问题，我们进行了一些尝试：

首先是考虑是否可以将集群统一？答案显然是不行。因为集团k8s集群，由于业务不同，不能和省公司的k8s集群合为一体。

那么是否可以做k8s的集群联邦？目前集团集群与省公司集群之间可能是比较远的(跨省)，集群联邦的整体消耗会大一些，并且目前跳板机的场景，跳到省公司集群一台机器上就够了，不需要看到省公司的所有机器。

维持ssh现状，维护shell脚本？shell脚本需要人力维护，在省公司的节点逻辑很可能需要使用service来完整，继续维护shell，第一不是那么CloudNative，第二也背离了磐基磐舟轻松上云的初衷。

本着达到灵活、易用，提升集群部署时效，解决端到端开发运维效率，成就内部客户的目的，我们针对整体场景做了进一步抽象，抽象成“1+31+N”的典型网络模型。

1个中心+“31+N”个边缘集群的场景，中心与集群、集群与集群，集群与N之间，存在着网络隔离与网络不可预知的情况；在这些集群之间，保持网络隔离的情况下，在中心节点做到云原生体验的自动化运维，做到GitOps自动化。

带着抽象之后的这个模型，我们在平台管理上进行了深入调研，最终选用了CNCF的云原生边缘计算项目KubeEdge来解决完成以上所有集群的统一管理。

KubeEdge是什么？解决什么问题？

KubeEdge的特点是在云边通信的资源消耗小，使用方式基于Kubernetes，上手方便，比较适合我们当前的场景。KubeEdge项目是华为云开源的一个基于Kubernetes的开放平台，并为网络应用提供基础架构支持，提供云和边缘之间的部署和元数据同步。KubeEdge具有以下几点关键优势：

容器化应用封装

» Build once, run anywhere

» 轻量化基础镜像，降低资源占用

通用的应用抽象、定义

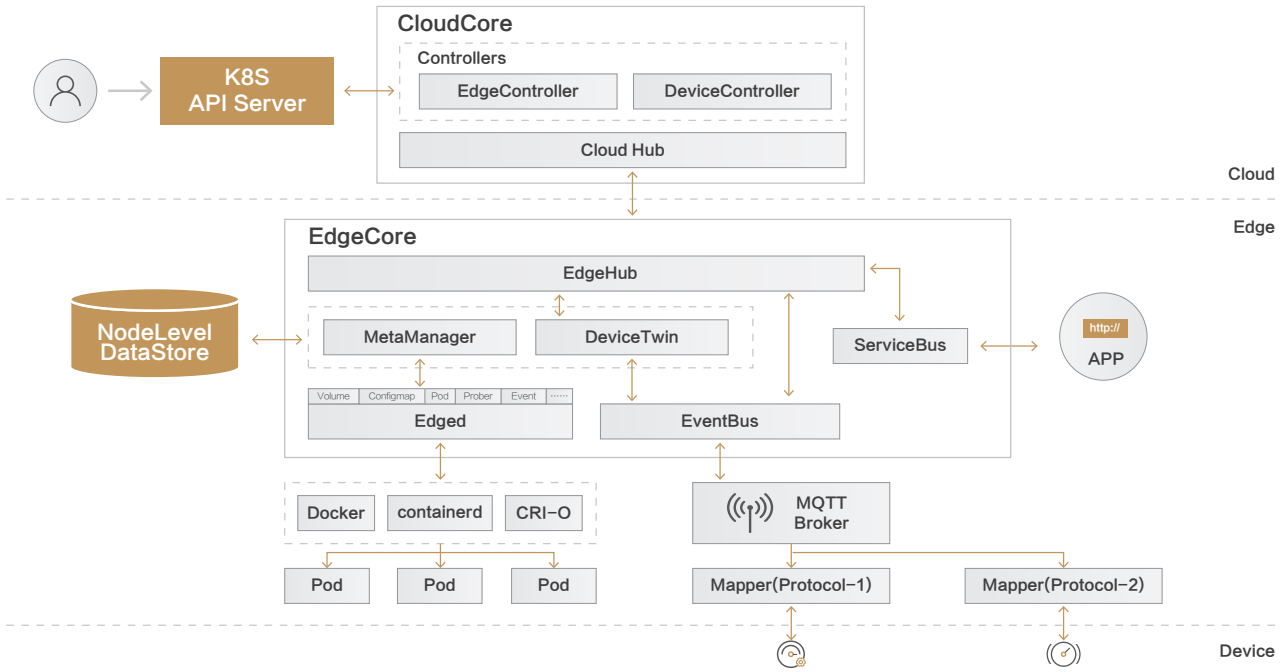
» 业界事实标准

» 云上、边缘统一管理

松耦合的架构

» 易扩展的API框架

» 易于定制平台组件



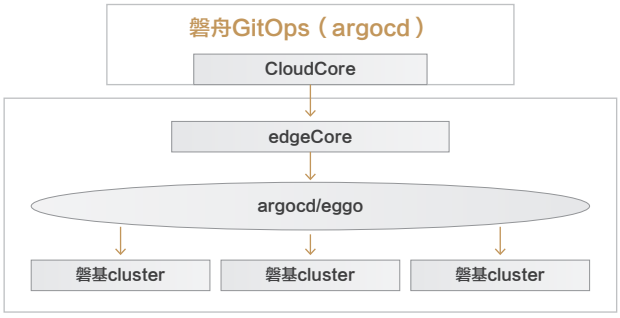
磐舟磐基平台的KubeEdge实践

通过对KubeEdge的应用场景分析，以及对移动内部1+31+N模型结合，我们可以将集团的“1”想象为KubeEdge的CloudCore节点、将各省公司的node节点想象为EdgeCore节点，从而就实现了1+31+N下的云边协同模型。映射到我们的具体场景是这样：

» **集群业务部署场景：**把集团的K8s master节点作为KubeEdge的CloudCore节点，省公司的node节点作为KubeEdge的EdgeCore节点，CloudCore节点与EdgeCore节点连接上后，在EdgeCore上启动磐舟GitOps业务中ArgoCD pod，统一下发CD一体化的元数据，从而将省公司资源池做到方便的集群创建、集群纳管，最终方便的达成自动化GitOps交付。

» 集群自动化创建场景：基于省公司的资源池来创建磐基PaaS集群，运维人员在master节点使用磐舟GitOps，通过CloudCore与EdgeCore的通信，部署来自openEuler社区的集群自动化部署工具-eggo的实例。之后在边缘侧，就可以通过eggo来自动化完成省公司磐基PaaS集群的创建。

综上，通过将KubeEdge集成至磐基PaaS平台，成功打通移动集团与各省公司的网络，实现“1+31+N”的K8S集群全部连通、



实现统一管理、简化多集群的运维系统、减少运营成本；同时也成功将前面提到的500台鲲鹏服务器以及它上面的BC-Linux for Euler集群纳入磐基PaaS平台的大家庭之中，运维效率大幅增加。

磐舟磐基平台在多集群管理的下一步计划

在完成KubeEdge集成到磐舟磐基平台这项专项工作之后，考虑到后续不仅是由master节点纳管单个edge节点，还会考虑在将南向集群的单个节点组成一个集群，实现控制面的自动化集群部署，支撑省公司集群的控制面自动化。磐舟磐基平台计划进一步集成CNCF社区最新的集群联邦方案Karmada来完成“1+31+N”的PaaS多集群统一管理工作。

VIPKID: 基于Karmada的容器PaaS平台落地实践

慈轶恒
VIPKID后端研发专家

VIPKID的业务覆盖数十个国家和地区，签约教师数量超过8万名，为全球100万学员提供教育服务，累计授课1.5亿节。为了更好的为教师和学员提供服务，相关应用会按地域就近部署，比如面向教师的服务会贴近教师所在的地域部署，面向学员的服务直接部署在学员侧。为此，VIPKID从全球多个云供应商采购了数十个集群用于构建VIPKID内部基础设施。

业务背景

无法躲开的多云，多Region

VIPKID的业务形态是将国内外的教育资源进行互换互补，包括过去的北美外教资源引入到国内和将现在国内的教育服务提供到海外。为了追求优质的上课体验，除了组建了一个稳定低延迟的互通链路外，不同形态的计算服务都要就近部署，比如教师客户端的依赖服务，要优先部署在海外，而家长客户端的依赖服务则首选国内。因此VIPKID使用了国内外多个公有云厂商的网络资源和计算资源。对多云的资源管理，很早就成为了VIPKID的IaaS管理系统的一部分。

K8s多集群策略

在VIPKID容器体系设计之初，我们首选的方案是单集群模式，该方案的优势是结构简单且管理成本低。但综合评估多云之间和多Region的网络质量与基础设施（网络和存储）方案，并综合我们的项目周期，只能放弃这个想法。主要原因有两点：

- 不同云之间的网络延迟和稳定性无法保证
- 各家云厂商的容器网络和存储方案均有差异

若要解决以上问题，需要耗费较高的成本。最后，我们按照云厂商和Region的维度去配置K8s集群，如此就拥有很多K8s集群。

集群的容灾

容灾对于容器而言，相比传统的VM资源已经友好得多。K8s解决了大部分Pod和Node级别的Case，但单个集群的灾难处理，还需要我们自行解决，由于VIPKID早期已经完成了微服务化的改造，因此可以利用快速创建新集群或者扩容现有特定集群的方式来快速进行计算服务的转移。

业务挑战及痛点

如何对待不同集群中的同一个应用?

在多云部署过程中，我们遇到了一个很复杂的问题：同一个应用在不同集群中的workload几乎都是不同的，比如使用的镜像、启动参数（配置）甚至有时候连发布版本都不一样。前期我们是依赖自己容器PaaS平台来管理这些差异，但随着差异需求增多，场景也越来越多，差异抽象越发困难。我们的初衷是让开发者可以直接在我们的PaaS平台上管理其应用，但因为复杂的差异越来越难以管理，最终只能依赖运维同学协助操作。另外，在某些复杂场景下，运维同学也难以快速、准确的进行管理，如此偏离了DevOps理念，不仅增加了管理成本，而且降低了使用效率。

如何快速完成故障迁移?

对于故障迁移，这里我从应用和集群两个不同视角来描述，应用视角看重关键应用的自愈能力以及能否在多集群状态下保障整体的负载能力。而集群维度则更看重集群整体级别的灾难或对新集群的交付需求，比如网络故障，此时应对策略会有不同。

应用视角——应用的动态迁移： 从应用出发，保障关键应用的稳定性，可以灵活的调整应用在多集群中的部署情况。例如某关键应用在A集群的实例出现故障且无法快速恢复，那就需要根据事先制定的策略，在同厂商或同Region下的集群中创建实例，并且这一切应该是自动的。

集群视角——新集群如何快速ready： 新集群的创建在我们的业务场景很常见。比如当某个K8s集群不可用时，我们的期望是通过拉起新集群的方式进行快速修复，再如，业务对新的云厂商或者Region有需求时候，我们也需要能够快速交付集群资源。我们希望他能够像启动Pod一样迅速。

为什么选择Karmada?

上述列举的痛点，若只试图满足暂时的需求是远远不够的，系统在快速发展过程中必须要适当的进行抽象和解耦，并且随着系统组成模块的角色分工逐渐清晰，也需要适当的重构。对于我们的容器PaaS平台而言，业务需求与集群资源耦合越发严重，我们将解耦的切面画在了多集群的管理上，由我们自研的平台管理应用的生命周期，另外一个系统管理集群资源的操作指令。

明确需求后，我们就开始在开源社区寻找与调研这类产品，但找到的开源产品都是平台层的，也就是与我们自研平台解决思路类似，并且大多是以集群视角来进行操作的，所有资源首先在集群的维度上就被割裂开了，并不符合我们对应用视角的诉求。以应用为视角，可以理解为将多个K8s集群作为一个大型集群来管理，这样一个workload就可以看做是一个应用（或一个应用的某个版本）而不是散落在多个集群中同一个应用的多个workload外一个原则是尽量低的接入成本。我们调研了开源社区的多种方案，综合评估后，发现Karmada比较符合我们的需求。

试用Karmada后，发现有以下几方面优势：

» Karmada真正意义的实现了以一个K8s集群视角来管理多集群的能力，让我们能够以应用视角管理多集群中的资源。另外，Karmada的OverridePolicy设计几乎所有差异都可以单独声明出来，简单直观且便于管理，这与我们内部对应用画像在不同集群之间的应用差异不谋而合。

» Karmada完全使用了K8s原生API，使得我们可以像原来一样使用，同时也表明我们在后续的接入成本会很低。并且Karmada的CRD相对来讲也更容易理解，我们平台的服务画像模块可以很容易的将分发和差异配置动态渲染成Propagation和Override策略，下发给Karmada控制面。

» 开源开放的社区治理模式，也是我们团队最看重的一点。在试用Karmada过程中，不论是我们自己还是社区方面对需求的理解和设想的解决方案，都可以在社区中开放讨论。同时，在参与代码贡献过程中，我们团队整体技术能力也显著提升。

基于Karmada的改造方案

我们的容器平台，承载了整个公司所有的容器化部署诉求，

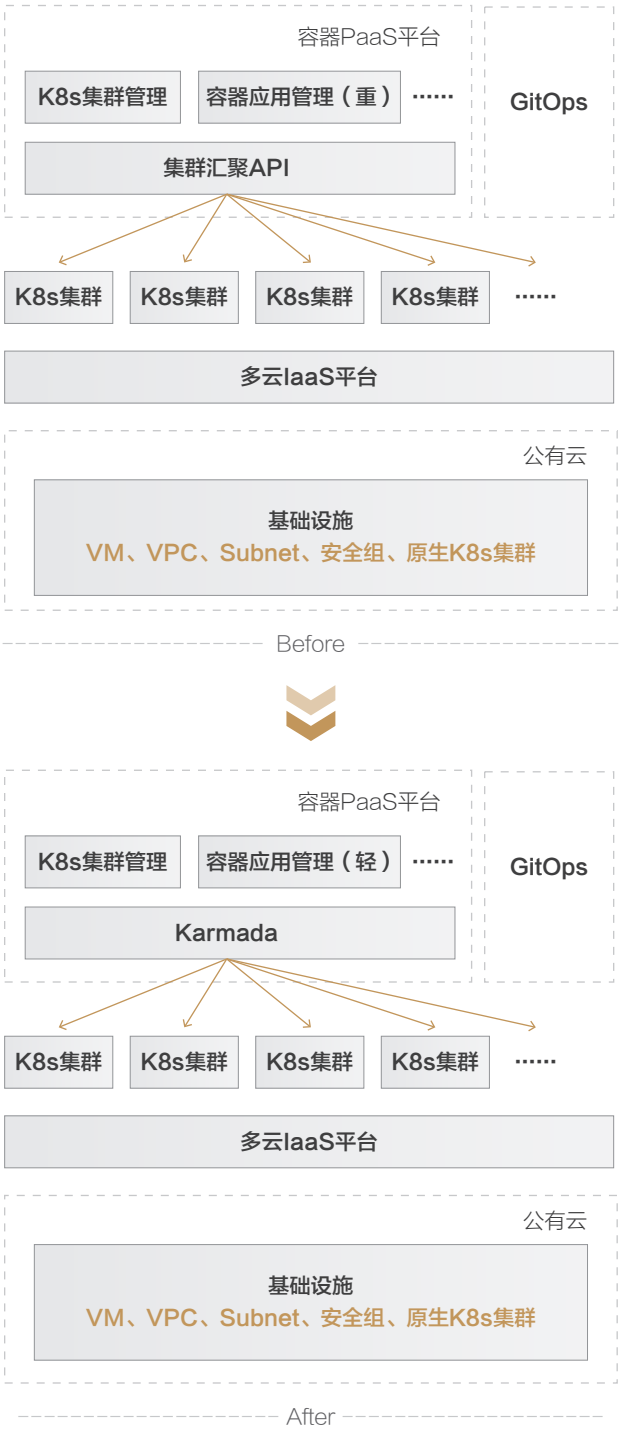
包括有无状态、在离线业务和AI大数据等。并且要求PaaS平台的设计和实施不会对某一家公有云产生任何依赖，因此我们无法使用云厂商封装过的一些产品。我们会依赖内部的IaaS平台去管理多家云厂商的各类基础设施，包括K8s集群的创建、扩容、VPC，子网以及安全组的配置。这个很重要，因为这让我们可以标准化多个云厂商的K8s集群，让上层PaaS平台几乎无需关心厂商级别的差异。另外，对于系统级别的应用和组件，我们为开发者创建了另外一条管理渠道，那就是使用GitOps。这对高阶开发者来说要更加友好，对系统应用的组件安装部署更为高效。

在平台落地之初，我们单独剥离了一个组件专门和K8s集群进行交互，并且向上保留K8s原生API，也会附加一些和集群相关信息。但在落地过程中，“容器应用管理”系统需要进行许多操作适配多集群下的复杂情况。比如虽然PaaS系统看起来是一个应用，但系统需要渲染不同的完整资源声明到不同的集群，使得我们在真正维护多集群应用时仍然是不相关的、割裂的，因为我们没办法在底层解决这个问题。诸如此类问题的解决还是占用了团队不少的资源，尤其是引入CRD资源后，还需要重复的解决这方面的问题。并且这个系统无法不去关心每个集群里面的细节状况，如此背离了我们设计“集群汇聚API”组件的初衷。

另外，由于GitOps也需要与集群强相关，在集群数量较大，并且经常伴随集群上下线的情况下，此时，若要正常运转就需要对GitOps的应用配置进行批量变更，随之增加的复杂度，让整体效果并未达到预期。

引入Karmada后，多集群聚合层得以真正的统一，我们可以在Karmada控制平面以应用维度去管理资源，多数情况下都不需要深入到受控集群中，只需要与Karmada交互即可。如此极大的简化了我们的“容器应用管理”系统。现在，我们的PaaS平台可以完全倾注于业务需求，Karmada强大的能力已经满足了当前我们的各类需求。

而GitOps体系使用Karmada后，系统级组件也可以简单的在各个集群中进行发布和升级，不仅让我们体验到了GitOps本身的便利，更是让我们收获到了GitOps × Karmada的成倍级的效率提升。



VIPKID引入Karmada之前和之后的架构对比

Karmada带来的业务收益

以应用为维度来管理K8s资源，降低了平台复杂度，同时大幅提升使用效率。下面以我们PaaS平台特性入手，来描述引入Karmada后的改变。

» **多集群应用的部署速度显著提升：**先前在部署时需要向每个集群发送部署指令，随之监测部署状态是否异常。如此就需要不断的检查多个集群中的资源状态，然后再根据异常情况做不同的处理，这个过程逻辑繁琐并且缓慢。引入Karmada后，Karmada会自动收集和汇聚应用在各个集群的状态，这样我们可以通过Karmada来感知应用状态。

» **应用的差异控制可开放给开发者：**DevOps文化最重要的一点就是开发者要能够完全参与进来，能够便捷地对应用全生命周期进行管理。我们充分利用了Karmada的Override策略，直接与应用画像对接，让开发者可以清晰的了解和控制应用在不同集群的差异，现已支持环境变量，启动参数，镜像仓库。

» **集群的快速拉起&对GitOps适配：**我们将基础服务（系统级和通用类型服务）在Karmada的Propagation设定为全量集群，在新集群创建好以后，直接加入到Karmada中进行纳管，这些基础服务可以伴随集群交付一并交付，节省了我们对集群做基础服务初始化的操作，大大缩短了交付环节和时间。并且大部分基础服务都是由我们的GitOps体系管理的，相比过去一个个集群的配置来讲，既方便又直观。

» **平台改造周期短，业务无感知：**得益于Karmada的原生K8s API，我们花了很少的时间在接入Karmada上。Karmada真正做到了原来怎么用K8s现在继续怎么用就可以了。唯一需要考虑的是Propagation策略的定制，可以按照资源名字的维度，也可以按照资源类型或LabelSelector的维度来声明，极其方便。

本期人物:

张水涛，联合交易中心（QME）科技创新部副总裁、创原会成员；13年行业耕耘见证了大宗商品场外市场行业的起步与发展，主持搭建QME和TJPME大宗商品电子交易平台。在大宗商品电子交易产品规划和架构设计、需求开发、标准化制度流程建设、风控体系建设、应急预案处理以及交易平台的运营和管理方面具有丰富的实战经验。主持搭建前海仓单区块链与物联网平台，实现了商品现货全生命周期的透明化及可追溯，为电子仓单注册、流转、融资、保险等操作提供了重要的业务前提和风控的核心基础。

10年金融科技人
如何在多重角色中自由切换

主要奖项

- **2020年，带领前海联合交易中心技术团队凭借科技赋能、切实服务实体的综合实力成功入围毕马威发布的《2020中国领先金融科技企业50》；**
- **2020年，IT业务全领域管理工作，因出色业绩获得总经理二等奖。**
- **2021年，QME前海仓单项目下，基于物联网，大数据和人工智能与区块链融合应用的供应链管理创新模式荣获十佳案例奖项；**
- **2021年，张水涛以金融科技专家身份参加深圳市地方金融监督管理局组织的《深圳市扶持金融科技发展若干措施》专家咨询论证会并提供意见。**

01 人生就像盲盒，处处都有惊喜

“人生充满了未知的惊喜，每个经历就是打开一处盲盒，开启新的起点。”

水瓶座的张水涛有着敏锐的思维特质，可以精准、感性的捕捉到行业动态，然后理性的规划完成它。

求学期间的他，是一个爱折腾的大男孩，进入大学不久，因对校内BBS不满主动向学校申请BBS和官网改造，凭借他的好人缘，快速组建了一支攻关团队，做出了令他自己满意的网站，这也使他萌发了最初的职业理想——成为一名用代码改变世界的程序员，而另外一次记者协会夏令营活动实践，则让他关注了很多实业信息，打开了他希望搭建一个信息平台的梦想。

2008年，北京奥运会成功举办的同时，张水涛也迎来了他的学业和爱情的双丰收。作为奥运志愿者，张水涛负责接待来自全球各地的奥运健儿，语言的障碍也没有阻挡他想为奥运会支持的热情，也正是因为“不地道”的英文，成功吸引了来自同校英文系的一位师妹。就这样，这一年，他即收获了学位，也迎来了爱情。

对于他现在的工作与生活，张水涛愿意称之为幸运，而对于他来说，人生最大的盲盒，就是他的一对儿女，所以谈到对子女的教育时，他鼓励他们可以去冒险，打开他们的人生盲盒，去寻找自己的梦想，就像小时候陪他们看的《银河补习班》，人生就像射箭，梦想犹如箭靶，定好自己的目标，勇敢的去飞吧!

02 从北到南 实现金融科技人的跨度

“父母从小教导我们，切莫眼高手低，学会做小事，才是放心做大事的开始。”

工作13年，张水涛形容自己是一个“拧螺丝”的IT小白。从码农开始，再到运维、数据、架构、安全、管理，几乎做遍了IT行业，对他所从事的每个岗位，他都会将工作架构、工作思路整理得清楚详细，他认为这是一种基本的工作态度和职业素养。

2016年对张水涛来说是转折性的一年，为了追逐梦想，举家从北京搬迁到深圳，实现地域的跨越，也实现了从技术骨干到QME科创主管的职场飞跃。入职QME后，张水涛干了三件重要事情，实现了公司和个人发展的快速发展。

第一件事情，实行QME IT“精益化”管理改革。随着信息时代的发展，企业运营和创新对于IT系统的依赖性更高，因此IT环境的可靠性，可用性和快速适应性越发重要，同时IT各类技术也在快速发展，在这样的大环境下如何使得IT技术可以快速顺应时代发展，为企业运营和创新带来便捷高效的IT系统，势必迫在眉睫，他认为精益化的IT管理是便是很好的解决这个问题。精益化IT就是精益思想和IT相融合，以最小成本，完成质量最好及速度最快的面向用户的价值交付和优化。作为张水涛进入QME后主持的第一件大事，现在回想起还记忆犹新：“第一次主持公司层面的流程改革，执行过程中也遇到了一些困难，我发现以往作为技术VP所沉淀的经验，似

乎无法适用于新的管理工作，只能逼自己花更多时间去系统性地学习、快速补齐管理的短板。”

张水涛及团队干的第二件事情，部署数字化转型全面云化，在QME“增收降本、管理创新”的经营战略指导下，同时国家及监管层面也鼓励企业IT系统上云实现自主可控，国家发改委、中央网信办联合印发《关于推进“上云用数赋智”行动培育新经济发展实施方案，工业和信息化部关于印发《推动企业上云实施指南（2018-2020 年）》的通知,他带领团队在保证系统安全下进行IT基础设施全面云化改造。

全面云化的优势主要体现在以下三方面:

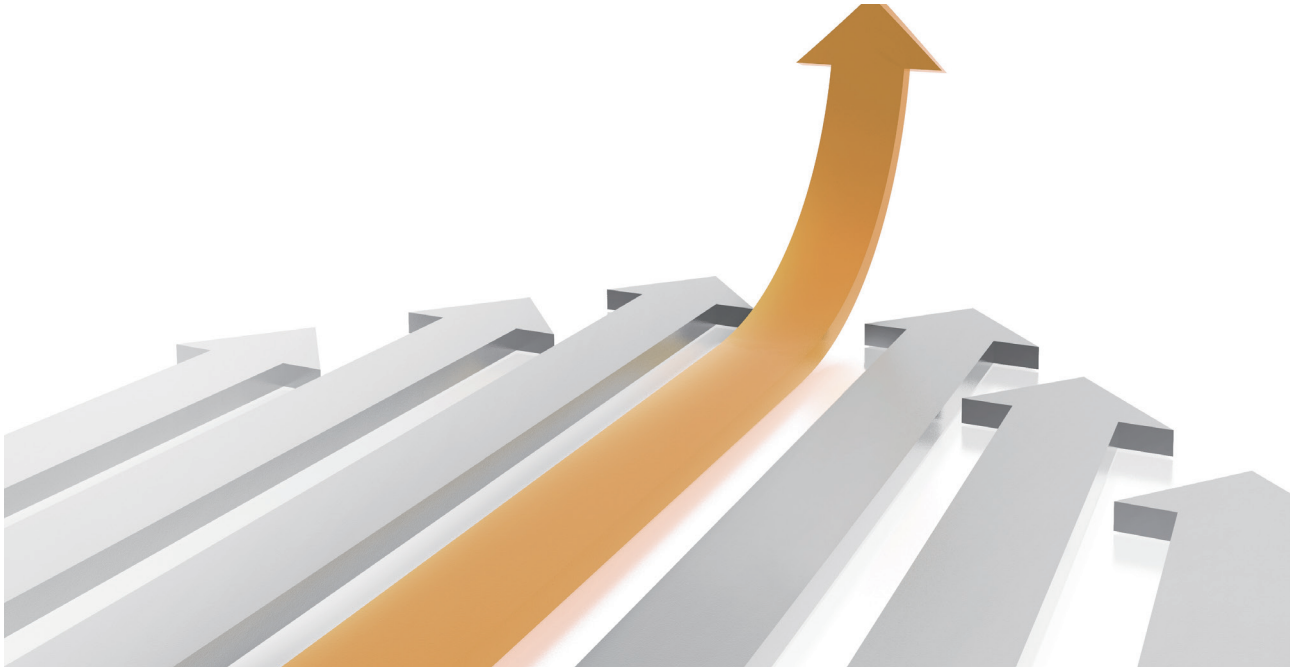
» **成本优化**，QME采用全云的基础架构后未来五年将节省约50%基础设施投入成本。

» **资源效率提升**，相比当前使用预配容量的模式来进行资源的采购，转型云架构模式后，可实现资源需求与预配容量之间的动态平衡；通过利用DevOps等技术开发实践，加快部署周期，同时消除差异和错误，提高运营敏捷性。

» **提升业务核心竞争力**，全面云化可以让企业内部运营数据化、智能化、精细化；同时让企业外部对客户的连接更加顺畅化、精准化、简易化，强大的云应用让企业在提升业务核心竞争力，在进行业务创新的同时打破技术投入瓶颈，提高业务创新能力。

第三件值得张水涛引以为豪的事情，就是2020年、2021年连续两年他带领团队以QME”前海仓单”入选毕马威中国领先金融科技“双50”企业。QME 是国内市场首家利用科技手段，将非标准化的大宗商品生产原料及产成品现货转换成标准化电子仓单资产的现货交易所，QME推出的前海仓单系统，利用物联网、区块链、智能识别等科技实现了商品现货全生命周期的透明化及可追溯，为电子仓单注册、流转、融资、保险等操作提供了重要的业务前提和风控的核心基础。秉着科技赋能，服务实体的理念，QME将区块链、物联网等技术应用于QME“前海仓单“，帮助中小微企业解决由于主体信用资质不足所面临的“融资难、融资贵”难题。

在谈到企业拥抱云原生的价值，张水涛表示通过上云，把传统的物理空间迁移到云空间，将云计算能力扩展到各个业务领域。上云的优势也立马显现出来，使用混合云的架构，它的安全能力，管控能力使得IT运维工作更加高效。通过使用云原生技术，提升了管理模式更轻便、灵活。他表示，互联网时代，他选择来南方发展，见证了企业转型，也实现了自己对于职业的理想，从产品基础研究，到业务应用落地再到市场积极求变的转型。



03 大宗商品交易平台 打造人民币国际化

对于未来，张水涛希望在他带领的IT管理团队，可以实现从改造信息科技到科技引领业务发展，这也会成为未来3-5年平台发展的关键核心。

中国是全球大宗商品的主要消费国，但由于现货市场发展滞后，短板突出，长期以来在交易市场规则约定、实物现货仓储交割、以及交易过程中的支付清算等诸多环节相较于国际上成熟的现货交易体系存在较大差距，导致产业优势无从发挥，主要商品定价权仍然基本集中在伦敦、纽约和芝加哥等传统贸易、航运与交易中心。此外，大宗商品市场全球化特征明显，对中国而言，与实体产业密切相关的大宗原材料和工业制成品也基本属于大进大出，要争夺定价权，需要有中国人自己的同时又是真正国际化的交易平台。

QME将利用集团境外市场的历史经验，并结合香港作为国际金融中心的成熟经验优势，发挥粤港澳大湾区特别是深圳的科技创新活力，向内辐射中国内地的实体产业腹地，对外连接“一带一路”沿线的资源市场，通过交易所业态的创新探索，补齐国内商品现货市场的短板，并与境外市场形成东西呼应，具体措施将包括：

» **仓储联通：** 基于QME仓单体系，打造与境外市场互认的仓储交割体系，在仓库、品牌认证及仓储管理系统等方面进行对接；

» **交易联通：** 以仓单互认体系为基础，便利客户开展跨市场交易、交割活动，减少实体企业套保成本，促进双边交易规模及市场活跃度；

» **融资联通：** 以仓单互认体系为基础，服务银行金融机构，便利金融机构对实体经济在跨境贸易中提供全流程融资支持。

基于境内外体系的全面打通，QME将充分利用集团境外市场完备的金融牌照资源，把基于实体经济真实供需水平而产生的商品价格向境外市场输出，在境外市场通过发行对应价格挂钩的金融衍生产品，形成我国对于特定商品在国际市场定价过程中的重要影响力。

经过三年的探索布局，QME已确立了起步于大宗商品现货、立足服务实体经济的发展方向，利用物联网、智能识别和区块链等前沿科技打造连接金融与实体的标准化资产体系，切实推动解决实体企业的融资难题，并借助境外市场形成的境内外、一带一路沿线的配合效应，分阶段逐步实现将前海联合交易中心建成全球大宗商品人民币定价中心的战略目标。

张水涛通过参与创原会•畅聊云原生（2）活动，他结识了云原生领域很多专家，通过线上研讨一些很“轻”的话题，他一直都还蛮受感染的，他们在这么高强度的生活节奏下，对正在发生的潮流、新的事物依然充满好奇心，然后不惜力的去实现它，他一直觉得学习是一件没有边界的事情。

创原会作为全球云原生的交流平台，张水涛觉得成为创原会的会员，与云原生圈子多交流，其实和现在的工作与资源是息息相通的。后期，打算借助创原会开启业务上的合作，在打磨平台的发展阶段，我们需要高效精准地接触到我们的目标用户和伙伴，创原会就是一个开发云原生与产业融合无限可能的平台。

注

1. QME为香港交易所在中国内地设立的大宗商品交易平台
2. 创原会•畅聊云原生是创原会为会员量身打造的“轻研讨”类线上活动，让会员能在自由、轻松的环境下，相互交流各自的云原生技术和实践经验。

云原生如何让车学会“社交”技能?

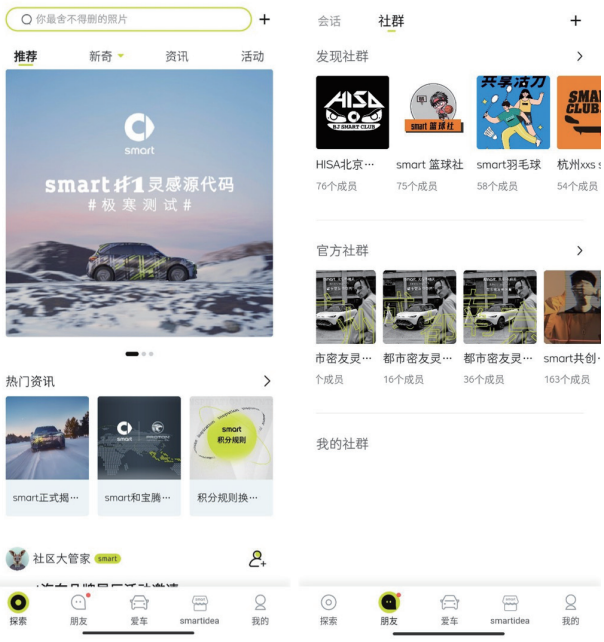
让车掌握“社交”技能，这样的时代会不会很有趣?身处于云时代，仿佛身边周遭的一切都变得智慧了起来。而智慧是什么?是人与连接。人与人连接，人与物连接，智慧的萌芽生于连接之上，蔓延万物。

刘克兴 smart品牌全球公司IT技术开发总监

smart APP，这款超级APP便是开创“人与车连接”未来智慧出行的先行者之一。如今，车在生活中早已不仅仅是一种交通工具，而是逐渐成为标榜个性的社交圈子。而smart APP正是为自己的品牌车友圈提供了一个全场景功能集成的“人-机-车”互联平台，让消费者在体验驾驶智能化的便利之余，还能享受一种全新的社交方式。这样的超级APP究竟是如何实现的呢? 本次我们采访了smart品牌全球公司IT技术开发总监刘克兴，来看看他为我们带来的精彩分享。

APP背后的云原生技术架构

从诞生之初，smart APP就成为了洞悉用户需求的重要抓手。在消费者整个购车生命旅程中，通过APP与用户的互动，了解掌握用户的真实诉求，进而反馈到研发端，推动产品功能的迭代。而在这个过程中，功能迭代效率与应用部署速度就成为了抓住用户注意力的机会窗口。因此，smart APP背后的云原生技术架构就成为关键的力量。



smart APP

smart APP携手华为云基于云原生2.0的技术体系，采用云容器引擎与服务网格的基础架构，实现了注册中心和配置中心的去中心化，构建了更轻快的微服务，无需构建任何基础架构服务，可以安全地对微服务进行微隔离，畅通无阻地进行内部通信；另一个则API网关的应用，将整个C端和B端交互的连接， 所有App小程序都会经过API网关，通过分组和独立的运维部署，使得项目之间、项目内部的开发测试人员可以基于同一上下文协作，协同效率较之前提高了50%以上。

云原生如何让用户教会我们做技术?

刘克兴表示：“smart APP与云原生有较好的匹配度，云原生能有效响应汽车营销领域快速变化的需求，并加速业务创新的落地。云原生的工具包解决了smart应用开发中的关键痛点 “。

首先，云原生提供的是一套热启动的工作方式，很多组件不需要重新构建，在去中心化服务和组件化服务的帮助下，让企业腾挪出更多的时间与人力聚焦于业务创新，业务也有更多机会去试错，使应用得以在最大化满足用户需求的基础上，实现快速开发。其次，面对多端部署的需求，服务端基于云原生的开发工具，以及APP跨平台开发技术，让应用可以快速的覆盖安卓、IOS以及其他应用生态上去，实现一次开发，多端适配，做到后端服务与前端开发1：2的分配方式，让整个开发过程，与传统开发模式相比，节约更多的投入成本。

什么样的组织才能用好云原生?

在刘克兴看来，对于云原生这样高产能的技术体系，只有为云原生架构而适配的组织结构才能最大化发挥云原生的技术效能。 生而云原生的团队，可以让人才梯度的培养变得十分简单，与过去好几个技术栈相比，云原生的单一技术栈将让整个组织变得敏捷高效起来。刘克兴说道：“基于云原生开发流程和工具，研发与业务的互动周期将极度压缩，实现应用功能‘两周迭代，月度发布’；DevOps团队也可以利用云原生平台进行开发和测试迭代，通过看板去管理和改善缺陷；同时在这样的组织结构中，我们还可以引入效能团队，对研发工作效率完成横向对比，辅助交付团队提高交付效率。”

而在云原生提升开发效率，锻造敏捷高效组织的背后，我们也看到在未来汽车行业中，云原生大放异彩的蓝图，在车联网、智慧门店以及智能网联的探索与演进中，云原生2.0必将带来更多创新的突破和改变。

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