白皮书

Kata Containers 在百度智能云的应用实践

作者: 张宇

目录

概述

本白皮书介绍了百度如何通过利用创新云技术(如 Kubernetes、Kata Containers 和 OpenStack) 提供大规模人工智能云计算和边缘计算服务的精彩历程。白皮书首先介绍百度智能云和其容器相关产品的背景，并阐述了百度在函数计算、容器服务、边缘计算等场景中如何高效应用 Kata 所提供的更大隔离性；之后，白皮书就百度如何将 Kata Containers 运行时集成到其 OpenStack 基础架构，使之具备更高性能等内容进行了详细描述，同时分享了一个 OpenStack Neutron 网络应用实例；随后，文章讲述百度边缘计算利用 Multus 容器网络插件为 Kubernetes pods 创建多个窗口网络接口 (CNI) 的应用情况；最后，文章简要提及新兴虚拟机管理程序创新以及 AI 加速硬件支持等课题，并就支持本白皮书撰写的相关人员致谢。

背景介绍-百度智能云及容器服务

百度智能云[1]

百度智能云是百度旗下面向企业及开发者的智能云计算服务平台，肩负百度技术能力对外输出的重任，于 2015 年正式运营。秉承百度以技术为信仰，将技术创新作为立身之本的理念，百度智能云致力于为各行各业提供以 ABC (Artificial Intelligence，人工智能；Big Data，大数据；Cloud Computing，云计算)为一体的技术服务。

到目前为止，百度智能云已经发布了超过 260 款产品和近 40 个解决方案，不仅提供计算、存储、网络、数据库、安全等基础云全线产品，也提供全面的大数据、人工智能解决方案，云服务形态囊括了公有云、私有云、混合云等。

在人工智能领域，百度智能云继承了百度 AI 持续领先的技术优势；目前，百度 AI 已经具备语音技术、图像技术、人脸与人体识别、自然语言等 210 多项技术能力，并且在无人车、智能家居等领域成功实现落地应用。此外，百度还自主研发了中国第一款云端全功能AI芯片--“昆仑”，小度助手成为了中国首个达到 “亿级” 装机量的对话式人工智能操作系统等等，作为百度生态的一部分，这一系列产品与服务都将助力百度智能云打造领先优势。

植根于百度厚沃的生态资源，百度智能云发展迅速，在知名咨询机构 Forrester 2018 年 Q3 发布的《2018 年中国全栈公有云开发平台 Wave 报告》[2]中，其凭借营收增速和市场占

有率的大幅增长，强势入围了卓越表现者 (Strong Performers) 象限；2019 年 5 月，权威市场研究机构 Synergy Research Group 发布的《2019 年第一季度亚太公有云市场报告》[3] 显示，百度智能云稳居中国市场第一阵营，跻身前四。

百度智能云容器服务

百度是业界容器技术最早的践行者之一。早在 Kubernetes v0.8 版本推出之际，百度已经过二次优化，实现了生产级别的服务实践。

2017 年 6 月，百度智能云发布了云容器引擎服务 (Cloud Container Engine，CCE)，正式开始将百度在容器技术上的多年积淀推向商业应用；同年 12 月，百度宣布以金牌会员身份加入云原生计算基金会 (Cloud Native Computing Foundation，CNCF)；次年 3 月，智能云容器引擎服务 CCE 通过 CNCF 首批 “Kubernetes 一致性认证”；2019 年 4 月，百度智能云获得 Kubernetes 认证服务商资质。

目前，百度智能云提供的容器服务包括：

■ 云容器引擎服务 (Cloud Container Engine，CCE)

CCE[4] 提供 Docker 容器的生命周期管理、大规模容器集群的运维管理，以及业务应用的一键式发布运行等功能，可以通过无缝链接百度智能云其他产品，提供高弹性、高可用、高效便捷的平台服务，适用于系统架构微服务化、DevOps 高效运维、AI 应用深度学习容器化等业务场景。

CCE 具有灵活的容器集群管理，以及底层网络/存储/安全的基础设施整合能力，与高性能弹性伸缩等特点。

■ 百度智能云函数计算服务 (Cloud Function Compute，CFC)

CFC[5] 是最早基于容器引擎 CCE 的业务场景之一，它提供基于事件机制的高弹性、高可用、扩展性好以及极速响应的云端无服务器计算能力，有助于用户仅关注业务逻辑的代码部分，而无需关注和配置服务器资源，并可支持多种函数触发器，适用于多样化的事件触发场景。

CFC 具有低成本、快速启动、“零” 运维、自动伸缩等特性。

■ 容器实例产品 (Baidu Container Instance，BCI)

百度智能云容器实例 BCI[6] 提供 Serverless 的容器服务，让用户只需要提供容器镜像和运行所需的基本参数，即可创建和运行容器，而无需管理服务器和集群，且只需为容器实际运行消耗的资源付费。

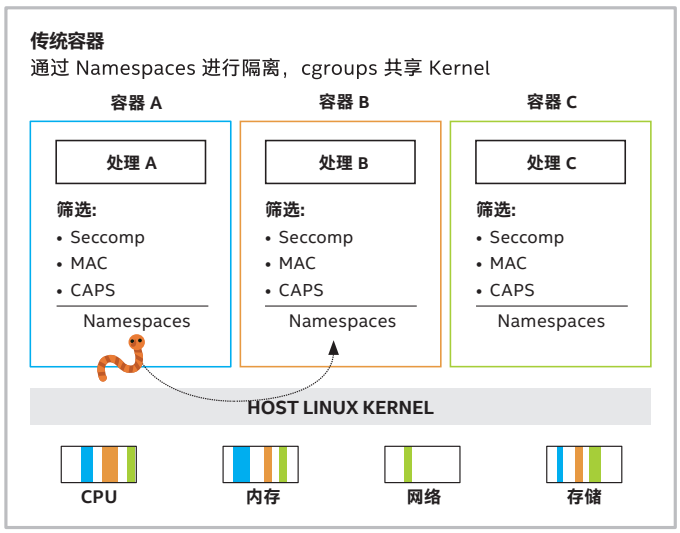
■ 边缘计算产品 (Baidu Edge Computing，BEC)

百度智能云边缘计算节点 BEC 基于宏大的 CDN 节点和网络构建，用于一站式提供靠近终端用户的弹性计算资源。该产品基于就近计算和处理，可以大幅度优化响应时延、降低中心带宽成本。边缘计算提供的容器服务包括容器实例 (BCI) 和 DuEdge[7] 百度边缘函数服务。DuEdge 百度边缘网络计算方案提供 “编程定制、内容分发、安全防护、人工智能” 四大核心功能，用以借助边缘网络计算的力量，破局云与端之间数据传输和网络流量难题，提升业务灵活性和运行效率。

容器服务面临的挑战及解决方案

百度智能云容器引擎经历了复杂业务、大流量、复杂部署等多方面的技术考验和淬炼，例如，单集群峰值每日网页访问量达 10 亿 +，单租户容器规模 50,000+ 等，其在不断地学习过程中，练就了以 Kubernetes 为中心的容器技术方向的强大掌控力。

基于容器技术实现资源共享在带来业务弹性和资源高利用率的同时，也会增加业务的安全风险几率。此类风险在于同一主机上的多个容器需共享同一个主机内核，而同一宿主机上可能运行不同租户的容器，这就可能会威胁到整个云基础架构和租户业务及数据的安全。因此，如何在充分发挥容器轻量化和敏捷性的同时，提高其容器隔离性来保障资源共享的安全，就成为百度团队必须探索的重要课题。



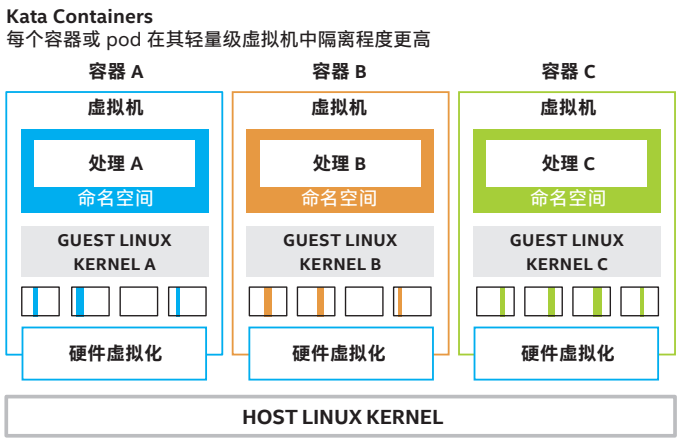
图一 传统容器

Kata 容器方案

Kata Containers[8]是一个开源项目，它采用轻量化虚拟机作为容器的隔离来构建一个安全容器运行时，而其虚拟化技术作为容器的二层保护为负载提供了更好的隔离性，这使得 Kata Containers 兼具传统容器的形态和虚拟机的安全性。

早在 2015 年，来自英特尔开源技术中心的工程师就开始探索采用英特尔® 虚拟技术(英特尔® Virtualization Technology，英特尔® VT)来提高容器的安全隔离性，并以此发起了英特尔® Clear Containers 开源项目[9]，与此同时，来自 Hyper.sh（一家中国的高科技初创公司）的工程师也发起了 runV[10] 开源项目，这两个项目采用的技术和目的都非常相似，都是为了将容器置于一个安全“沙箱”，以便进一步促进该技术发展和成熟。随后在 2017 年，英特尔和 Hyper.sh 团队将这两个开源项目在社区合并成了一个新的项目 Kata Containers。

传统虚拟机 (VMs) 可提供硬件隔离，而容器可快速响应，且占用空间相对较小，Kata Containers 将这两者的优势完美结合了起来。每个 container 或 container pod 都在自己单独的虚拟机中启动，并不再能够访问主机内核，杜绝了恶意代码侵入其它相临容器的可能。由于 Kata Containers 同时具备硬件隔离，也使得互不信任的租户，甚至于生产应用或前生产应用都能够在同一集群内安全运行，从而使得在裸机上运行容器即服务 (Containers as a Service, CaaS) 成为可能。



图二 Kata Containers

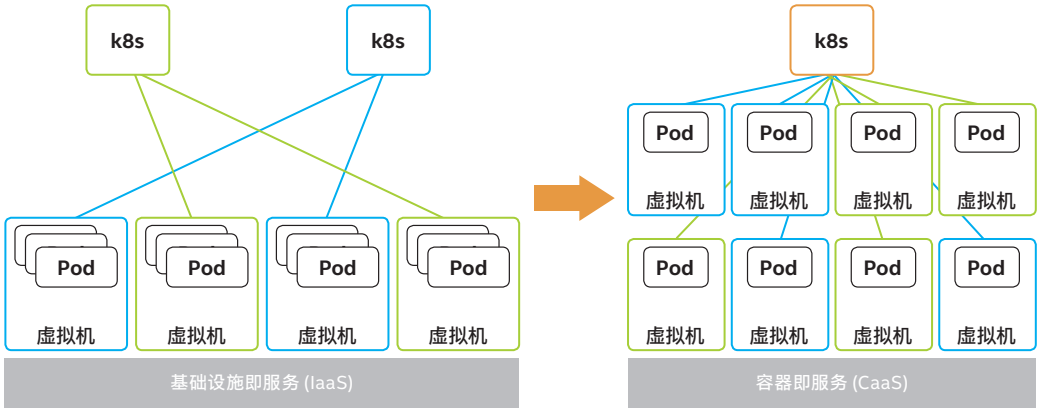
百度团队在 Clear Containers —也就是现在的 Kata Containers —发布之后，就开始关注这一安全容器技术，并和相关技术社区进行了多次交流，来调研和测试这一技术。基于广泛的安全容器技术调研，百度团队认为 Kata Containers 是目前具备高度安全和实践性的一项安全容器技术，主要考量因素如下：

- Kata Containers 是目前高度安全的容器技术。它使用轻量级虚拟机作为容器的安全隔离，而过去十几年的实践证实，虚拟化技术采用的安全模式，是人们共享资源的有效手段；
- 和传统容器相比，Kata Containers 的性能较高，且更有保障，同时又兼容传统安全技术。英特尔® 虚拟化技术 (英特尔® Virtualization Technology，英特尔® VT)[11] 及英特尔® 定向 I/O 虚拟化技术 (英特尔® Virtualization Technology for Directed I/O，英特尔® VT-d) 为虚拟化提供了硬件支持和加速，确保推动虚拟化的性能得以不断的提升，并成为前沿云计算的基础支撑技术；
- 由于 Kata Containers 遵循 OCI 容器实现规范，可以无缝对接容器编排方案，比如 Kubernetes 等。OCI 是一个 Linux 基础项目，旨在为操作系统级虚拟化（特别是为 Linux 容器）设计开放标准；
- Kata Containers 有着较高的成熟度。自从 2015 年成功推出的两个安全容器项目——英特尔® Clear Containers 和 Hyper runV——合并以来，相关技术有了重大进展，其功能性和开发基础设施也得到了验证。得益于 ARM，IBM 及 NVIDIA 的卓越贡献，Kata Containers 得以采用多层架构。之后更是多次发布了新的特性，修复了诸多漏洞，并被百度内部测试证明具有较高的成熟度；
- Kata Containers 获得业界的广泛支持，具备完善活跃的开源社区。蚂蚁金服、ARM、华为和英特尔都是 Kata 架构委员会的成员，该委员会负责指导项目的技术方向。另有其它众多有影响力的公司和组织参与并为 Kata Containers 社区的发展做出了重大贡献，其中包括了阿里巴巴、AMD、百度、Canonical、Google、IBM、NVIDIA、Red Hat、SUSE、腾讯、Vexxhost 和中兴等,且由于项目托管在 OpenStack 基金会，因而有着透明的社区治理、完善的 CI/CD 维护基础设施及项目文档。

总而言之，Kata Containers 拥有的多项特性，可支持百度智能云容器产品的设计要求，其中包括功能计算、云容器实例和边缘计算，所有这些都将在下面的章节中加以讨论：

函数计算

具有 Serverless 特性的函数计算为开发者提供了巨大的便利、弹性及性价比，而将所有的运维工作和安全保障留给服务供应商。容器技术是非常适用于函数计算的基础架构，利用容器轻量级、快



图三 基于 Kata Containers 实现 CaaS

速启动、易于编排等特性，百度智能云 CFC 可以通过一个弹性资源池为众多开发者提供共享的计算资源——在开发者函数被触发时，启动容器以承载函数运算；在函数运算结束后，释放资源以满足其它开发者的计算需求。这一模式既能快速响应终端用户请求，也可以通过资源共享的形式降低使用成本。

实现函数计算的首要需求就是底层基础架构对应用的不可见及解耦和，因此，资源的隔离也需实现对应用和用户的不可见。Kata Containers 的虚拟机隔离模式，既保障了容器在多租户环境中的安全隔离，同时也实现了对应用和用户的不可见。

云容器实例

在许多轻量级场景中，函数计算已经得到了较广泛的应用与实践，比如小程序、智能设备、自动化数据处理等。但是函数计算同样不能解决企业的全部问题，原因在于它要求用户的业务以函数为粒度进行拆分，函数之间完全通过接口通讯，且对业务的无状态化要求很高。对于复杂的业务系统而言，要满足这些要求并不容易，而且即便用户的业务架构可以真正实现函数化改造，囿于现有的大规模函数协作、编排和管理技术也还不够成熟，很难完全投入生产。

在面对容器化和 Serverless 时，用户通常需要对业务进行拆分，根据业务属性选择更合适的架构，有时会因此而面临两难的技术抉择。因此百度智能云一直在探索，是否有一种产品形态可以在充分发挥容器编排引擎优势的同时，又让用户享受到 Serverless 方案带来的低成本和高弹性？百度智能云容器实例服务 BCI 就是在这一需求下催生的最完美的产品。BCI 为用户提供了可以直接启动的容器化资源，不需要预先购买服务器和集群，用户可应业务需要随时启动一个或多个容器，并可以在业务完成后随时将其释放。

相比虚拟机，BCI 充分发挥了容器化的优势，其使用标准的 Docker Image 进行启动、细粒度的资源划分，并可在数秒内完成启动或停止，以及随时进行多副本复制。

对于小型应用的开发者，BCI 可以体现出媲美函数计算的 Serverless 特性——将应用打包到 Docker image 中，即可启动或停止容器，且开发者只需要为应用运行实际消耗的资源付费，完全不需要关心底层资源的管理和调度。

Kata Containers 则把虚拟机隔离作为容器实现的一部分，同一容器集群可以被多个租户共享，如图三所示。传统的云容器服务需要用户首先自己申请虚拟机作为租户隔离，然后在虚拟机上构建和维护容器集群及服务，Kata Containers 由于使用了虚拟机隔离使得云服务提供商可以向用户提供一个 Serverless 容器集群，实现真正的 CaaS，而用户只需关注自己的应用程序。

边缘计算

边缘计算产品形态是以一个边缘节点为集群部署服务，因为节点分布在不同地理位置。在同一个节点内，允许互相不信任的租户使用同一节点内资源，因此对安全性和多租户隔离要求极高。传统的容器架构涉及主机操作系统和访客容器之间的共享内核，如果一个容器出了问题，集群中的其他容器工作负载就会容易受到攻击。此外，由于边缘区域缺少中心云完备的网络虚拟化以及安全服务，且边缘直接暴露在外网环境中，使得产品的网络安全性面临巨大挑战。

同时，由于 BEC 的用户使用的是边缘的算力资源，性能要求也是用户考虑的第一要素。BEC 的产品形态决定了其需要在物理资源

有限的前提下，把产品性能发挥到极致，以免造成成本过高及资源浪费。Kata Containers 恰恰是轻量级虚拟机类容器，能够在保障安全性的同时，实现高性能，使得安全性和性能不必是一个折衷，而是鱼和熊掌完美兼得。

在 Kata Containers 方案上的实践

Kata Containers 虽然具有较高的成熟度和性能，但在实现具体的产品形态和实际生产部署中仍有一些关键技术点需要突破。

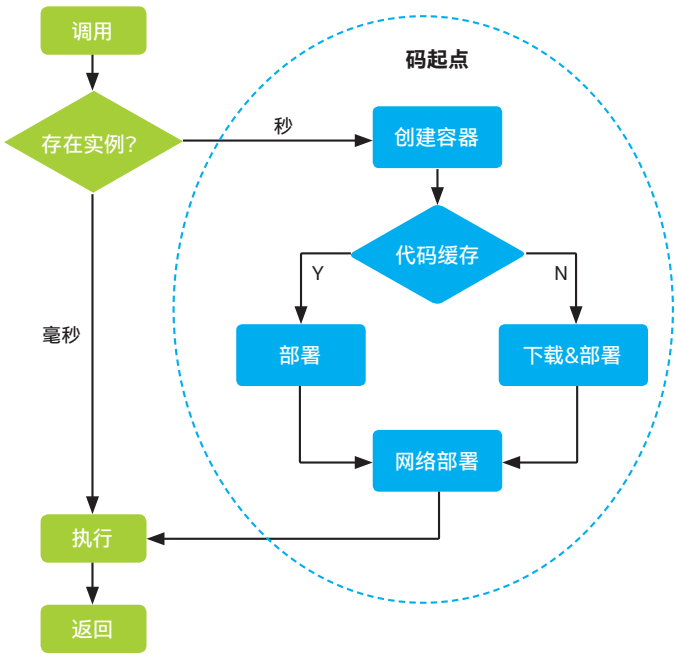
本节主要从函数计算与 OpenStack 结合，以及边缘计算方面分享百度智能云基于 Kata Containers 的应用实践。

基于 Kata Containers 的函数计算实践

由于 Kata Containers 采用虚拟机作为隔离，导致容器的启动速度难以满足函数计算场景的要求。为解决这一问题，百度开发出了许多高效的解决方案，在提升启动速度的同时，有效保持了 Kata Containers 的隔离性能。百度在这方面的深入探索取得了丰硕的实践成果。

■ 函数冷启动与热启动性能

函数的冷启动，如图四所示，是指函数首次被调用时，必须先准备用于运行的容器，再把用户函数部署上去，然后启动语言 runtime。



图四 函数冷启动

冷启动性能根据用户代码体积会有不同程度的浮动。如果不考虑 VPC 的切换时间，启动时间需要控制在百毫秒量级。那么为何冷启动性能这么重要呢？原因在于上层业务代码的访问超时值是统一设置的，不会为了某次 CFC 函数调用是冷启动还是热启动而去做动态调整。假如业务的超时设置较小，冷启动的长耗时很容易导致业务有损。根据测试，Kata Containers 的启动时间在 1s 左右，难以满足函数计算的需求。

目前，百度给出的解决方案是通过预先启动容器来实现快速响应。为提高准确性一部分容器会被预先启动成为 pre-warm 容器，在需要运行用户函数的时候只需进行一些初始化操作即可变成计算容器，来执行计算任务，从而大幅缩短容器的启动时间。下表比较了不同模式下容器的启动时间。请注意，数值越低，意味着性能越好。

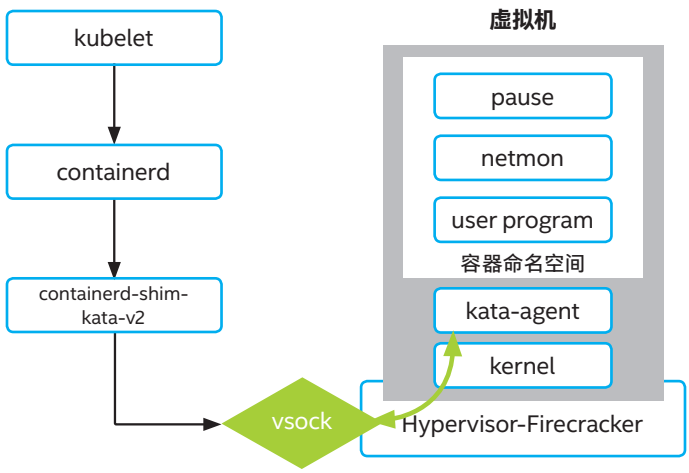
容器启动时间 ms	RunC	Kata
启动容器 runtime	526ms	1501ms
预启动容器 + docker cp runtime	441ms	463ms
预启动容器 + 动态挂载 runtime	155ms	98ms

表一 容器启动时间优化对比

■ 调度层启动加速

CFC 的调度层面必须和 Kubernetes 整合，另外，上层接口也需要一个与 Kubernetes CRI 有良好兼容性的方案。

Kata- 运行时会为 Kubernetes 的每个 pod 启动一个虚拟机，且多个容器可以同时置于虚拟机的同一个 pod 中。为了提高函数容器的启动速度，百度团队把创建 pod 的流程提前，采用了先启动 pod，再插入容器的方案。该方案先通过 Kubernetes 部署运行时为 Kata 的 pod，然后通过 CRI 接口，在 pod 中创建一个额外的业务容器。



图五 Kata Containers 预热 (pre-warm) 启动

如图五所示，初始化时，只会启动虚拟机内部的 pause 和 netmon 容器。当用户流量到来时，才准备代码目录与创建 user program 程序所在的容器。

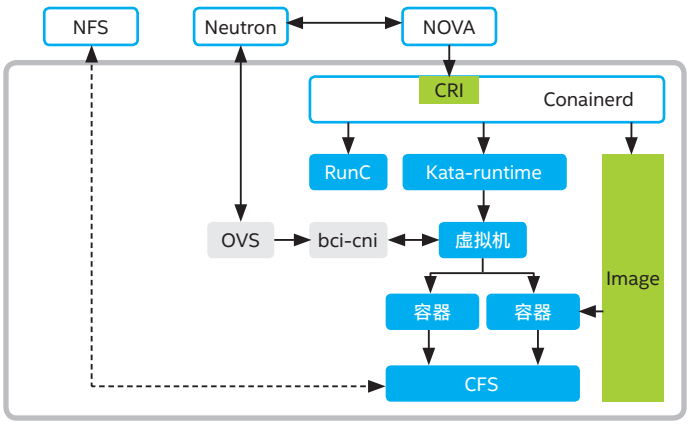
■ 支持动态挂载用户代码

挂载用户代码是函数计算的一项基本要求。而做加速启动的常用措施，使用 pre-warm 容器存在一个弊端，那就是 pre-warm 容器中未加载任何用户的代码，相反，用户代码是加载在计算容器中的，这就需要将用户代码挂载到正在运行的容器中。在 docker runc 场景下，宿主机和容器共享同一个内核，可以简单的使用 -v 参数进行映射。或者，在容器启动后，使用 nsenter 等手段，进行跨 namespace 挂载。然而在 Kata 容器场景下，宿主机和 Guest 机不在同一个内核，如何在 Kata Containers 中动态挂载，就成为一大挑战。

我们通过 mount bind 方式动态将用户代码加载到 Kata Containers 中，详见 github 上的讨论[12]。

Kata Containers 与 OpenStack 架构的融合

百度的大多数企业客户都有已有基础设施，且其中大部分是基于 OpenStack[13] 来构建，为了让 BCI 能够对接企业已有设施，BCI 针对企业内部系统的复杂性，以及网络架构的特殊性，自定义了一套能无缝兼容客户内部现有系统的新系统，如图六所示，实现了既能复用现有设施，同时又兼具轻量级虚拟化优点的改进方案，达成了轻量级虚拟机与重量级虚拟机的混布。通过混布，一方面节约资源，另外一方面也可降低开发与维护成本。



图六 Kata Containers 与 OpenStack 的融合架构

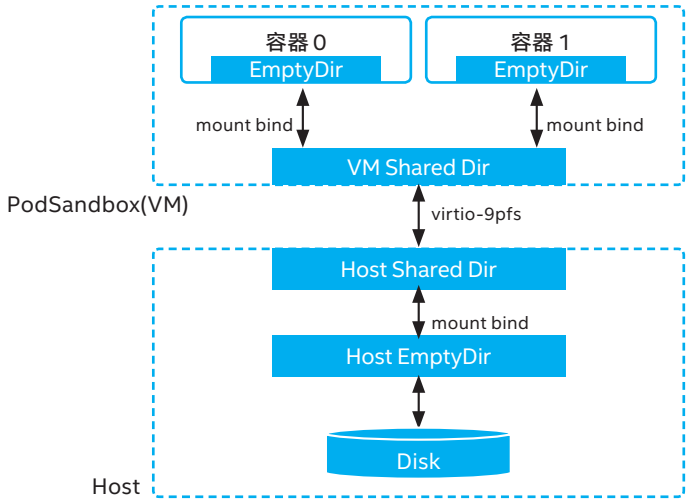
BCI 与 OpenStack 架构的融合技术特点在计算、存储和网络方面均表现鲜明：

■ 计算 - 整体架构基于 NOVA 构建

针对原有的 OpenStack 虚拟机，开发出适合在 NOVA 端使用的轻量级虚拟机（安全容器），并能做到和现有的虚拟机产品混布，NOVA 端统一调度资源，在下层通过 containerd 调度，同时在上层支持容器运行时插件 (Container Runtime Interface, CRI) 接口，方便用户通过自定义的 Kubernetes 节点进行调度。

■ 存储 - 优化 EmptyDir 存储速度及 Device mapper 加速

EmptyDir 是 BCI 产品的非持久化存储。它提供给容器组一个空目录，这个空目录会同时被容器组中的所有容器共享。在 Kata 中如何来实现这个空目录，是 EmptyDir 设计方案的关键。社区提供了一种实现方式，即通过 virtio-9pfs 这种文件系统虚拟化技术，将宿主机上的空目录共享给容器组，来作为 EmptyDir 的载体，如图七所示：



图七 Kata 社区的 EmptyDir 实现

但是，这套方案存在两个很明显的缺点：

a. 对于公有云这种多租户的场景，该方案需要文件系统提供 quota 管理，否则就容易造成宿主机的本地磁盘被某个用户的容器组写满，而导致其他用户的容器组无法正常工作。而现有的文件系统 quota 管理不够灵活，对于扩容、多用户管理等特性的支持不够。

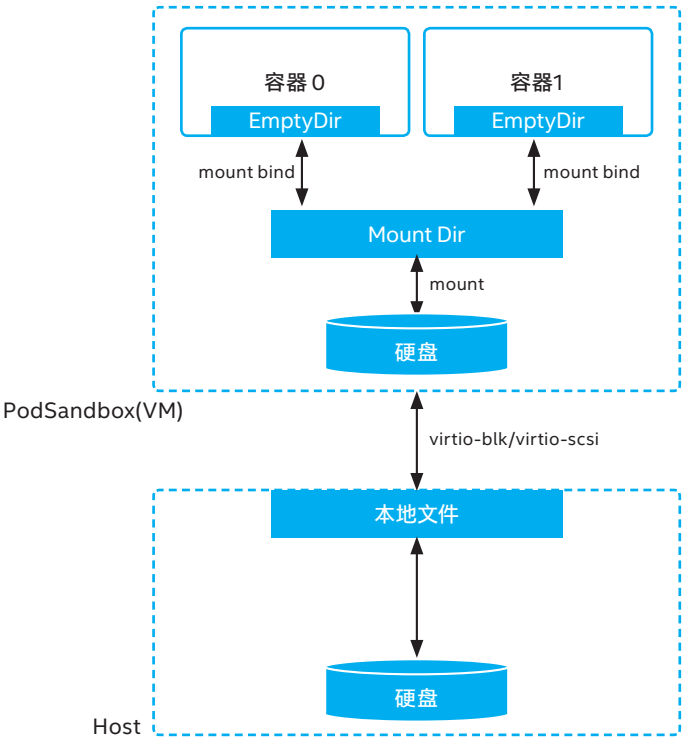
b. 9pfs 并不是专为虚拟化设计的文件系统协议，因此相对于宿主机，其性能会有很大程度的降低 (约 70%)。

请注意：Red Hat 和其他公司正在开发一种替代性的、更优化的共享文件系统协议 virtio-fs，它可以允许虚拟机访问主机上的目录树。它比 9pfs 性能更高，更安全。然而，在本文编写之时，

virtio-fs 仍在开发中。virtio-fs 的早期版本计划用于即将发布的 Kata Containers，用户可随后开始试验。

鉴于这些缺点，结合百度虚机产品的本地盘存储方案，百度团队通过块设备虚拟化技术将宿主机上的一个文件虚拟成虚机的块设备。由于宿主机上的文件大小是可控的，因此可以很好地控制 quota。而性能方面，virtio-blk/virtio-scsi 是非常成熟的虚拟化方案，相对宿主机，虚机性能不会有太大损耗。基于这两大优势，该方案可尝试应用到 BCI 的 emptydir 实现中。

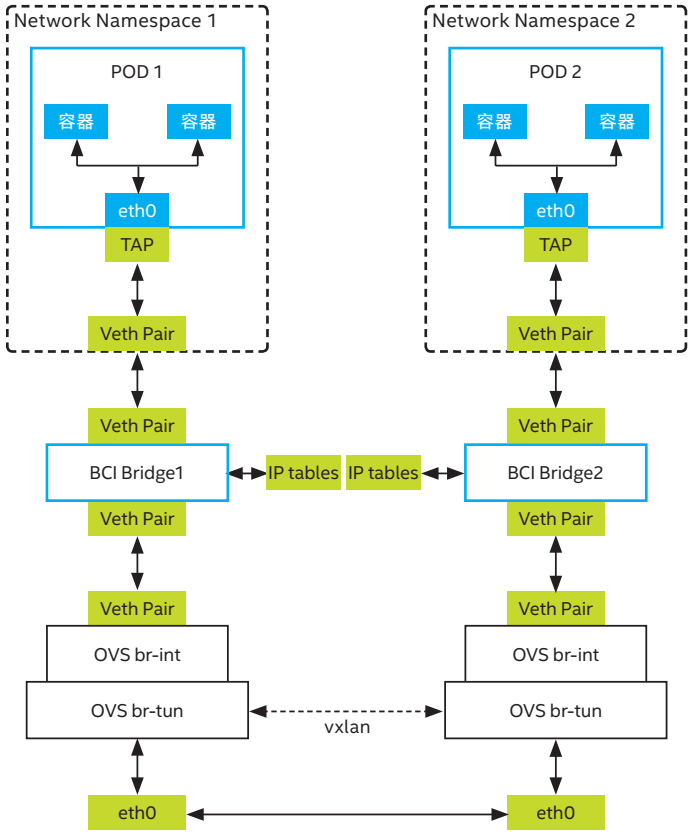
BCI 的 emptydir 方案实现如图八所示。经过测试，发现该方案的性能接近宿主机性能的 80% (单队列单线程)。



图八 基于 virtio-blk 的 EmptyDir 实现

■ 网络 - 自定义网络插件支持 OVS

网络属于轻量级虚拟化里比较复杂的部分，一方面对性能要求很高，另外一方面也对可维护性有很高要求。针对这一特点，百度自研方案复用了在 OpenStack 的 Neutron 组件，开发出兼容 Neutron + Open vSwitch 的网络方案，实现了网络隔离和网络限速功能。依托 Neutron + Open vSwitch 强大的功能及性能，复用之前的网络架构，即可以更低成本快速实现网络要求的功能。整体架构如图九所示：



图九 Kata Containers 与 Neutron 网络对接

其主要特点如下:

- 利用原有的 ovs + iptables 规则实现虚拟机和物理机隔离;
- 流量经过 ovs，与原有虚拟机的基本一致，不需要大改拓扑结构，因为多 host 网络通信由 ovs 提供 overlay 网络即可实现;
- 容器的 ip 由 neutron 注入，通过 containerd 和百度方案的 bci-cni 插件交互，实现指定 ip 的功能，eip 模块由原有 neutron 支持。

其中，该方案新开发一个 bci-cni 插件，适配 containerd 的 CNI 部分，并能创建指定的网络架构，最后接入 OVS。为了便于 neutron 和 containerd 之间的通信，在创建 pod 的时候，通过 nova 传入特定的注释: annotation，来携带指定的 ip 信息，方便 bci-cni 插件 ip 等相关信息进行设置。

(有关网络和卡式容器的更多详细信息，请访问 https://github.com/kata-containers/documentation/blob/master/design/architecture.md#networking.)

Kata Containers 在边缘计算的实践

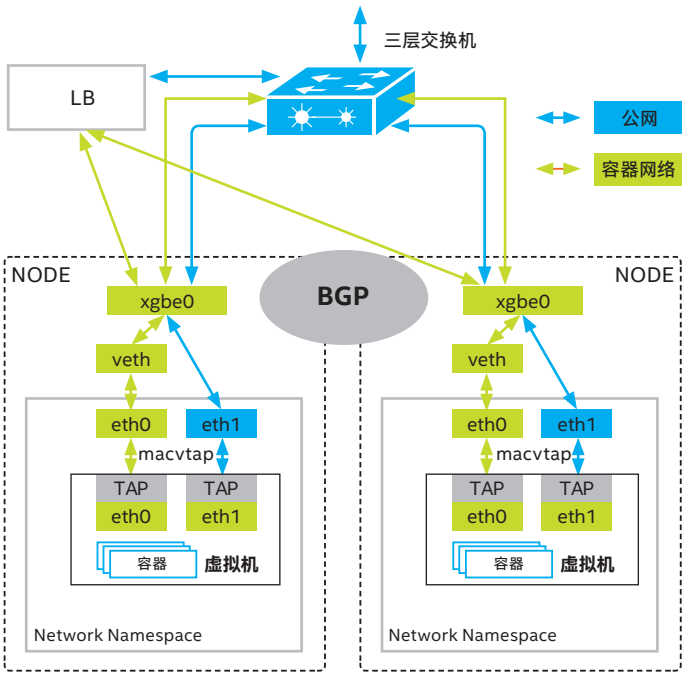
■ 边缘计算 BEC

与中心云不同，BEC 基于边缘节点部署运行，比如在 CDN 节点。由于缺少中心云完备的网络虚拟化以及安全服务措施，且边缘服务又直接暴露在外网环境中，产品安全面临巨大挑战，而网络安全又是 BEC 的重中之重，因为其一方面关乎计算区域与公网的安全性，另一方面又与计算区域内多租户的隔离紧密关联。另外有客户还需在容器中创建双网络设备，用于客户自主可控的、更精细化的流量控制，这无疑让网络架构更加复杂。

BEC 与 BCI 的一些性能需求也是呼应的，比如在存储的性能，以及可挂载 GPU 等方面，这方面优化在前述 BCI 实践中有提及，而在 BEC 的技术实践阐述部分将重点关注网络安全性。

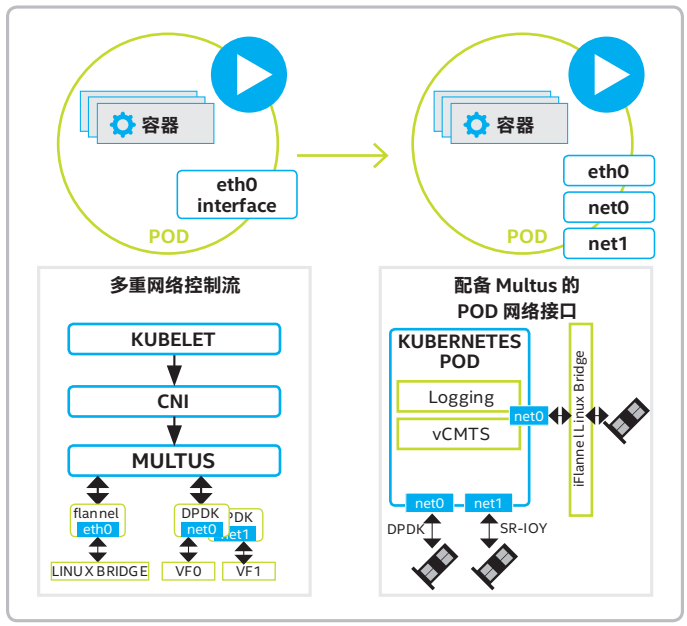
在 Kata 的场景下，容器的 Network Namespace 中还有一个虚拟化层，需要使用一个 Linux tap 设备，无法直接使用 veth 网卡。而百度智能云边缘计算采用的是 Macvtap 模式：创建母设备为 veth 的 macvtap 供 VM 使用。

1. 在双网卡的支持方面，BEC 定制了 CNI 实现方案，通过两种插件，分别实现内网和外网网卡的管理。其内网方案采用了 Calico 工具，通过 BGP 协议实现二层网络；外网方案采用了 macvlan，以最大限度减少过多网络设备带来的损耗，进而使得网络效率更高，整体架构如图十所示：



图十 BEC 双网卡架构

原生的 Kubernetes 中的 pod 仅支持一个网络接口，Multus[14] 是由英特尔开发的一个容器网络接口插件，可以用来为 Kubernetes 中的 pods 创建多个网络接口，如图十一所示。在其开发过程中，Multus 项目被贡献给 Kubernetes 网络工作组[15]，并成为该工作组成立的基础，也使得它成为标准多网络接口规范的开源版本。



图十一 Multus 多网口插件

Multus 作为其他 CNI 插件的代理，调用其他插件来完成 CNI 网络接口创建。为了配置 Multus，需要制定其它某个插件作为主插件，该主插件被用来配置和管理主网络接口。图十一中右下方展示了一个带有日志功能的容器化虚拟防火墙，它被部署在一个拥有3个网络接口的 POD 中。其中，eth0 作为 POD 的管理接口，使得该 POD 可以与 K8S 集群中其他 POD 通信，它也是 K8S 中主网络接口。

除此之外，Multus 还允许使用接口和软件栈，比如 SR-IOV、DPDK 等，图十一中显示该 POD 还有 2 个 SR-IOV 的 VF 接口，也就是 net0 和 net1。创建这些接口是用来加速网络数据平面，比如一个虚拟防火墙要求 2 个网络，以便使得防火墙规则彼此隔离。

双网卡带来的不仅仅是网络架构上的挑战，也使控制面的复杂度大幅提升。控制面 BEC 通过定制 CNI 实现，支持 Calico 和 macvlan 两套插件。而且为了保证内外网流量的分离以及回源的准确性，为 Kata Containers 自动注入了路由表规则。

2. 在网络隔离上，采用在边缘上部署中心云整套 VPC 虚拟化方案显然是不现实的。百度方案采用了云原生的设计思想，采用

Kubernetes 中的 network policy 机制，通过定制隔离策略，实现租户间的隔离。隔离默认作用于容器内网，外网网段不做隔离。且内网容器网络使用了 Calico 方案，原生支持 network policy。

3. 容器外网虽然不需要进行隔离，但是需要进行限速。BEC 采用在 CNI 中加入限速模块，通过 TC 工具实现。在实现过程中，为解决当创建 PodSandbox 时，TC 与 Kata 默认的 tcFilter 网络模型产生设备冲突，导致创建失败的问题，BEC 将 Kata 的网络模型改为 macvtap，保证了功能实现。

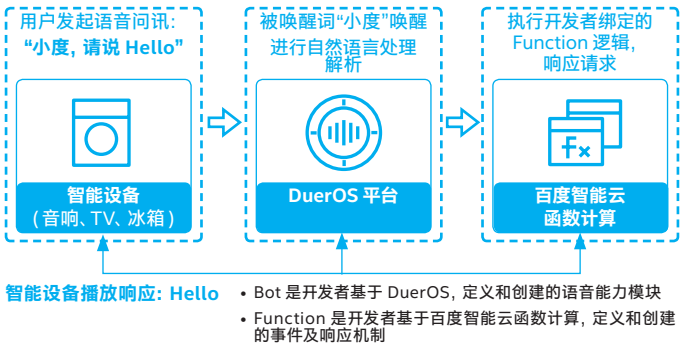
实践成果

函数计算应用 Kata 后的效果

百度智能云函数计算 CFC 在许多领域已经得到了广泛应用，例如，小度助手 (DuerOS) 基于函数计算为小度智能硬件的技能开发者提供部署平台，手机百度 App 基于函数计算为智能小程序开发者提供一站式云开发平台，多家在线内容媒体通过函数计算进行海量流式内容的预处理和分类，游戏发行商通过函数计算进行游戏安装程序的自动化打包和分发等。

■ DuerOS 技能开发

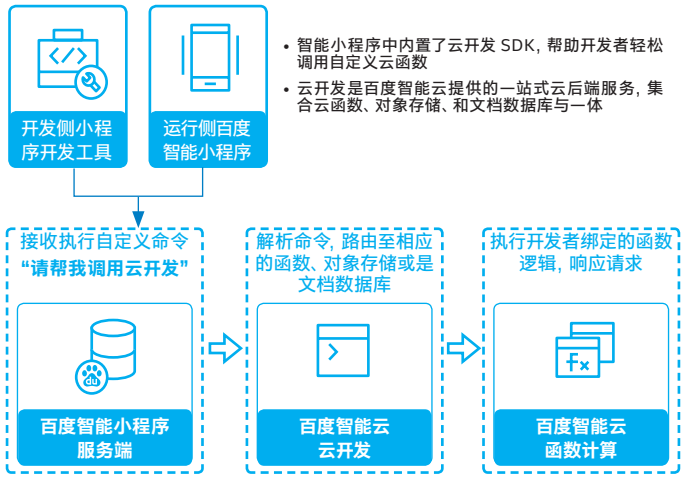
在 DuerOS 技能开发场景中，函数计算为超过 **3,000 名开发者、近 20,000 个技能**提供了计算能力，每天响应超过百万次来自智能终端的用户请求。由于资源共享带来的更低成本，使每个技能开发者可以享受**每月 100 万次函数调用和 400,000GB 秒的函数**运行时间，极大降低了使用门槛，从而让开发者能够更好聚焦在业务逻辑的开发中。



图十二 DuerOS 开发技能

■ 智能小程序开发

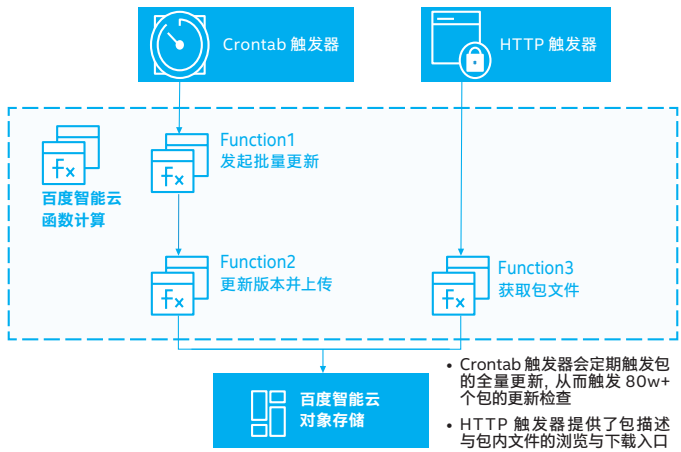
CFC 还支撑了百度智能云云开发服务，支持用户通过自定义函数，建立开发侧和百度智能云资源侧的连接，使用户能够更加灵活地使用百度智能云的计算、存储和网络资源，从而**帮助 50% 以上的智能小程序开发者**实现更高的开发效率和更低的使用成本。



图十三 智能小程序开发

■ NPM 包源站托管

CFC 也承担百度智能云的 NPM 包源站托管业务，即通过定时触发的函数自动检测 NPM 包的版本变更情况，当出现版本升级时，自动触发下游函数，进行 NPM 包的批量更新和下发。目前，CFC 已经支持了百度内外部各类应用超过 80W 个包的更新检查，健康稳定地服务于企业 DevOps 流水线自动化集成服务。



图十四 NPM 包源站托管

BCI 的实践应用成果

百度智能云容器实例 BCI 为百度内部的大数据业务提供了强大的基础架构支撑，帮助大数据部门构建起面向多租户的 Serverless 数据处理平台。对于大数据部门的平台研发人员，BCI 通过与 Kubernetes 的集成，满足平台研发者使用成熟的 Spark 社区技

术的诉求，而无需额外的学习成本，且不需要关注资源的管理和运维；对于使用平台的大数据开发者，基于 BCI 的大数据任务处理平台提供了更加便捷的 Spark 使用模式，开发者只需要打包数据处理任务，提交至平台，即可等待任务自动运行并获取结果，完全无需关心资源的调度过程，并且不需要自行维护任何 Spark 基础架构。

在这种多租户的大数据平台场景中，Spark 服务的提供方希望尽可能提升资源利用率，同时提供满足用户大规模提交计算任务时的算力。然而，传统的 Kubernetes 集群模式很难具备足够弹性，满足这一需求。因此，无需维护服务器，且能在数秒内快速启动的容器实例就成为了平台方的首选方案。通过使用 BCI，平台方的任务编排服务可以运行在自己的 Kubernetes 集群中，而用户提交的任务则通过 Virtual-Kubelet，调度用户账号中的 BCI 资源来运行，一方面平台方的使用模式和传统 kubernetes 没有差异，另一方面平台方无需为用户任务提前准备资源，这无疑提升了资源利用率。而 Kata Containers 则基于底层技术，让用户作业可以运行在隔离的系统内核和隔离的网络空间中，从而为任务执行和数据提供了更高的安全性。

目前，基于 Spark 的百度流式计算服务 BSC 已经集成 BCI 的 Serverless 容器能力，可以将原本搭建 Kubernetes 集群的资源成本降低 40%~60%，同时还可以获得大量宝贵的平台搭建实践经验。此外，百度大数据团队也正在与 BCI 团队合作，来推动更多大数据技术平台 (如 MapReduce、机器学习等) 的 Serverless 化，为开发者提供更多、更全面便利的大数据服务。

BEC 上的场景落地

百度智能云边缘计算节点 BEC 作为公有云标准产品，对所有客户提供开放式服务，并基于 Kata Containers 的特性，使多用户使用互相隔离、互不影响；目前，BEC 已承载了多类型对资源规格和性能要求高的业务，具体使用场景如下：

■ 互动直播/点播

提供就近收流合流、转码、弹幕分发能力，在大幅度降低网络延迟、优化用户体验的同时，降低客户媒体中心机房带宽压力 (**带宽成本可下降 80% 左右**)。另外，还可为本地化直播，提供链路网络质量保障。

■ 弹幕分发

弹幕指用户在观看视频时，在屏幕上实时滚动展示的评论内容。弹幕属于用户自定义数据，不支持通过 CDN 分发，传统的弹幕分发方式由 IDC 数据中心来做处理和分发；边缘计算节点在弹幕分发的场景下，主要由边缘计算节点来处理和分发，大幅度提高了处理、发送和接收的成功率，节省了中心带宽成本。

■ 智慧安防/城市

边缘计算节点可提供就近视频汇聚及处理功能，在同区的距离内解决视频流、图片的合流、存储、分析，从而实现低成本、高质量的智慧安防监控能力。在此场景下，其能够很好的满足智慧安防、智慧城市建设的要求，且 BEC 方案的系统扩展性强，易改造，可由边缘来做统一纳管。

■ 打造全新 CDN 架构

这款全新的架构提供边缘计算节点搭建用户的 CDN 服务，提供全区域弹性的轻资产算力资源，可帮助用户快速搭建 CDN 服务。

■ 云游戏、AR/VR

云游戏、AR/VR 等应用对延迟极度敏感，边缘计算节点可提供高实时性的 GPU 计算资源，有效应对游戏场景的渲染需求。

DuEdge 百度边缘网络计算服务

能够安全适用 Kata 的百度边缘场景包括：

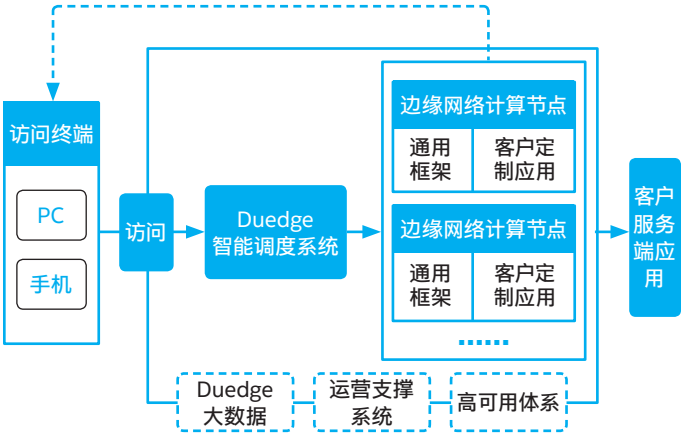
1. 用户开发和测试代码的计算容器：开发测试代码中通常存在一些 bug，如死循环、内存泄露、错误的网络请求调用等，如果没有 Kata 提供的隔离，受信任的容器业务可能就不会在同一台宿主机上运行，因为它们会以未经授权/或不受信任的形象出现；

2. 可能存在高危风险代码的计算容器：一些代码未通过代码自动检查，或者在代码自动检查过程中发现疑似恶意代码的情况；

3. 无安全沙箱语言的计算容器：一些编程语言实现安全沙箱难度比较大，比如 Python 和 go，为解决这一问题，这种语言的代码都在 Kata Containers 容器中运行；

4. 单独隔离业务的计算容器：一些业务希望有更独立、安全的计算容器执行；

5. 对内核有特殊要求的计算容器：一些业务应用需要特殊的内核版本支持，与宿主机的系统并不相同，因而宿主机的内核无法满足，需要通过替换 Kata Containers 的 kernel 来实现。

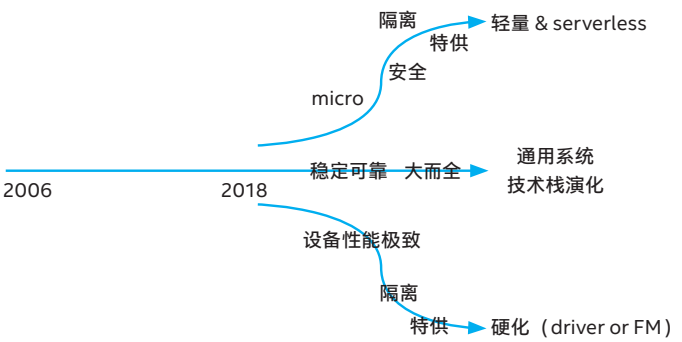


图十五 Duedge 边缘计算函数计算基础架构

展望

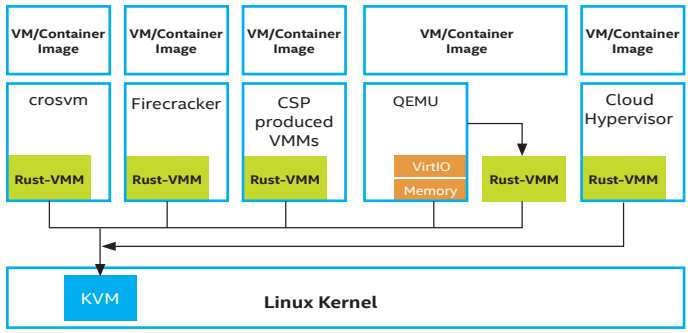
轻量虚拟化

如图十六所示，虚拟化技术从通用虚拟化向两极进化演；其中轻量虚拟化的进化，就是为了满足容器更多的需求，而针对隔离性、性能、特殊需求等场景进行分化，安全隔离性是首要需求。在百度智能云的业务需求上，也同样演进出多款适用不同场景的轻量虚拟化产品，在轻量的层次上，从函数计算、百度容器实例到边缘计算产品，设备模型逐级加重。



图十六 虚拟化技术演进

Kata Containers 作为一种安全容器解决方案，在百度的容器服务中扮演重要角色，并通过在不同场景下将虚拟机监控器（Virtual Machine Monitor, VMM）做替换来满足客户多样化的需要，这也和社区的 rust-vmm[16] 思路高度契合 (rust-vmm 是一个虚拟化组件库，便于用户创建自定义的 VMM 和 hypervisors；英特尔® 云管理工具 (英特尔® Cloud Hypervisor)[17] 项目是构建在 rust-vmm 之上的，其借助于更安全的 RUST 语言，且只采用 modernhypervisor 特性，用户可同时降低攻击面和记忆痕迹)。



图十七 一系列虚拟化 Rust Crates 用于虚拟机的快捷部署

AI 加速硬件支持

AI 推断边缘化有广阔的市场前景，AI 加速硬件也在推动这个市场前进，AI 硬件的虚拟化且支持容器内访问也是目前容器技术的一个研究方向。用户也希望支持 GPU 在多租户环境下共享，以最大化利用资源。此外，AI 加速还有专用的 FPGA，以及面向嵌入式设备的插卡 AI 加速芯片，同时能最好支持百度的 AI 计算平台，如 PaddlePaddle[18]。

可信赖容器计算环境

在高安全等级的边缘计算场景中，用户不希望将自己的代码或者密钥交给云计算服务商，这就要求云计算服务商可以提供可信赖的计算容器。英特尔® 软件防护扩展 (英特尔® Software Guard Extensions，英特尔® SGX)[19] 技术可提供一个可信计算环境，相应缓解此类问题。而如何将其应用在边缘计算场景，也是目前一个很有价值的研究方向。

致谢

本白皮书的撰写不仅得到了百度智能云各线同事的支持，也离不开百度智能云副总经理谢广军在新技术落地方面的长期关注，尤其是对 Kata 在百度落地的推动；感谢产品团队孙丽、周岳骞、吴秋材等对本白皮书需求部分的撰写、编辑，感谢产品团队负责人宋飞的审稿；感谢何方石、倪勋、沈迦勒、韩丁、周倜、白宇、王辉、谢永吉、李雨轩等对研发材料的撰写与支持。在本白皮书撰写过程中，也得到了智能云战略规划管理部陈寿送以及百度开源技术委员会谭中意的支持。

感谢英特尔团队 Eric Ernst、杨继国、沈晓晨、胡志明、吕荟晶、贾培自 Kata 发布以后，积极跟百度进行技术交流，从百度的应用场景入手，解决一些社区版本的技术难点，参与系统性能调优，共同探讨 Kata 的技术发展方向。感谢英特尔徐立冰、梁冰、金运通、陈科对于百度申请超级用户以及白皮书撰写方面给与的支持与帮助。

参考资料

[1] 如欲了解更多详情, 请参考链接: https://cloud.baidu.com/

[2] 如欲了解更多详情, 请参考链接: https://www.forrester.com/report/The+Forrester+Wave+FullStack+Public+Cloud+Development+Platforms+In+China+Q3+2018/-/E-RES142373#figure3

[3] 如欲了解更多详情, 请参考链接: https://www.srgresearch.com/articles/chinese-companies-now-account-40-apac-public-cloud-market

[4] 如欲了解更多详情, 请参考链接: https://cloud.baidu.com/product/cce.html

[5] 如欲了解更多详情, 请参考链接: https://cloud.baidu.com/product/cfc.html

[6] 如欲了解更多详情, 请参考链接: https://cloud.baidu.com/product/bci.html

[7] 如欲了解更多详情, 请参考链接: https://duedge.baidu.com

[8] 如欲了解更多详情, 请参考链接: https://katacontainers.io

[9] 如欲了解更多详情, 请参考链接: https://github.com/clearcontainers/runtime

[10] 如欲了解更多详情, 请参考链接: https://github.com/hyperhq/runv

[11] 如欲了解更多详情, 请参考链接: https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html

[12] 如欲了解更多详情, 请参考链接: https://github.com/kata-containers/runtime/issues/808

[13] 如欲了解更多详情, 请参考链接: https://www.openstack.org

[14] 如欲了解更多详情, 请参考链接: https://github.com/intel/multus-cni

[15] 如欲了解更多详情, 请参考链接: https://docs.google.com/document/d/1Ny03h6IDVy_e_vmElOqR7UdTPAG_RNydhVE1Kx54kFQ

[16] 如欲了解更多详情, 请参考链接: https://github.com/rust-vmm

[17] 如欲了解更多详情, 请参考链接: https://github.com/intel/cloud-hypervisor

[18] 如欲了解更多详情, 请参考链接: https://github.com/PaddlePaddle

[19] 如欲了解更多详情, 请参考链接: https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html