OrionX AI 算力资源池化解决方案技术白皮书

发布时间：2021 年 11 月

商标声明

本文档提及的商标为北京趋动科技有限公司的商标。

免责声明

本文档可能含有预测信息。由于实践中存在很多不确定因素，可能导致实际结果与预测信息存在差别。因此，本文档的信息仅供参考，不构成任何要约和承诺。趋动可能不经通知即修改本文档信息，恕不另行通知。

联系我们

电话：010-62560919

邮箱：BD@virtaitech.com

地址：北京市海淀区中关村大街 1 号海龙大厦十层北

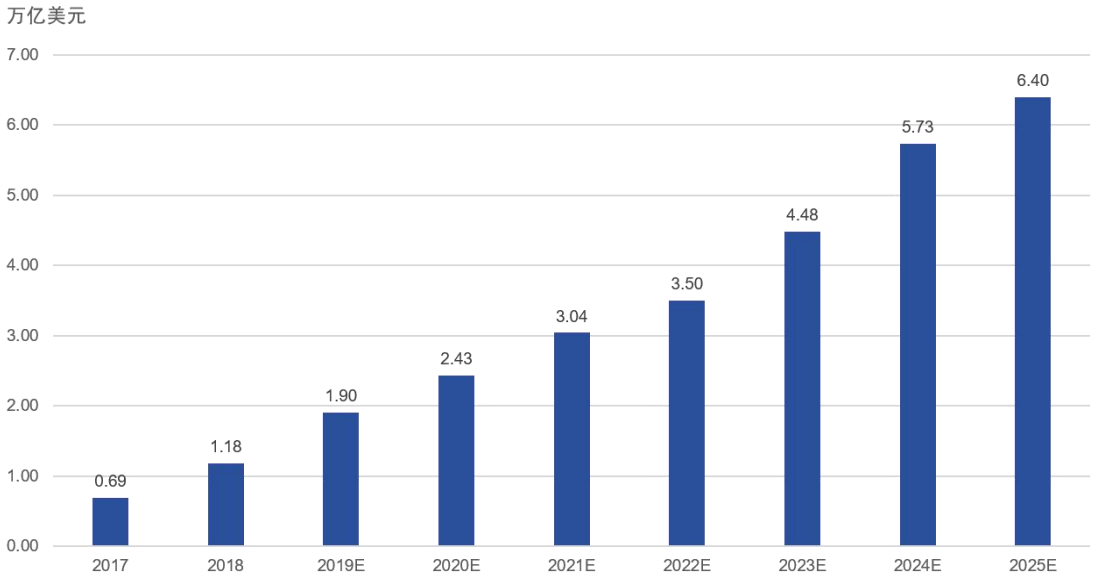
目录

图表目录

1 引言

当下，全球各国都在加速人工智能布局，将其作为战略性技术之一。作为较早发布人工智能战略的国家，中国政府将人工智能技术视为产业变革的核心力量，人工智能不仅是技术创新，更是推动经济发展、社会进步、行业创新的重要驱动力。“十四五”规划纲要更是将新一代人工智能作为要攻关的七大前沿领域之一，鼓励加速人工智能前沿基础理论突破、专用芯片研发、深度学习框架等开源算法平台构建，促进学习推理与决策、图像图形、语音视频、自然语言识别处理等领域创新，加速人工智能与诸如大数据、物联网、边缘计算等数字信息技术的融合发展，促进产业优化升级、生产力整体跃升。

德勤在 2020 上半年发布的《全球人工智能发展白皮书》预测数据表明：2025 年世界人工智能市场将超过 6 万亿美元；中国人工智能核心产业规模到 2020 年将增长至 1600 亿元，



数据来源：德勤研究

带动相关产业规模超过一万亿元。

图表 1- 1 全球人工智能市场规模走势图

作为 AI 市场中的重要组成，以 GPU、FPGA 等为主的 AI 加速器市场发展也随之水涨船高。根据 IDC 中国加速计算市场报告，预计 2021 年人工智能加速服务器市场规模将达到 56.9 亿美元，相比 2020 年增长 61.6%，到 2025 年，中国人工智能加速服务器市场将达到 108.6 亿美元，其五年复合增长率为 25.3%。

与此同时，由于缺乏高效经济的 AI 算力资源池化解决方案，导致绝大部分企业只能独占式地使用昂贵的 AI 算力资源，带来居高不下的 AI 算力使用成本；由于缺少对异构算力硬件支

持，用户不得不修改 AI 应用以适应不同厂商的 AI 算力硬件。这会加剧 AI 应用开发部署复杂性、提高 AI 算力投入成本并导致供应商锁定。

2 GPU 资源池化技术的演进

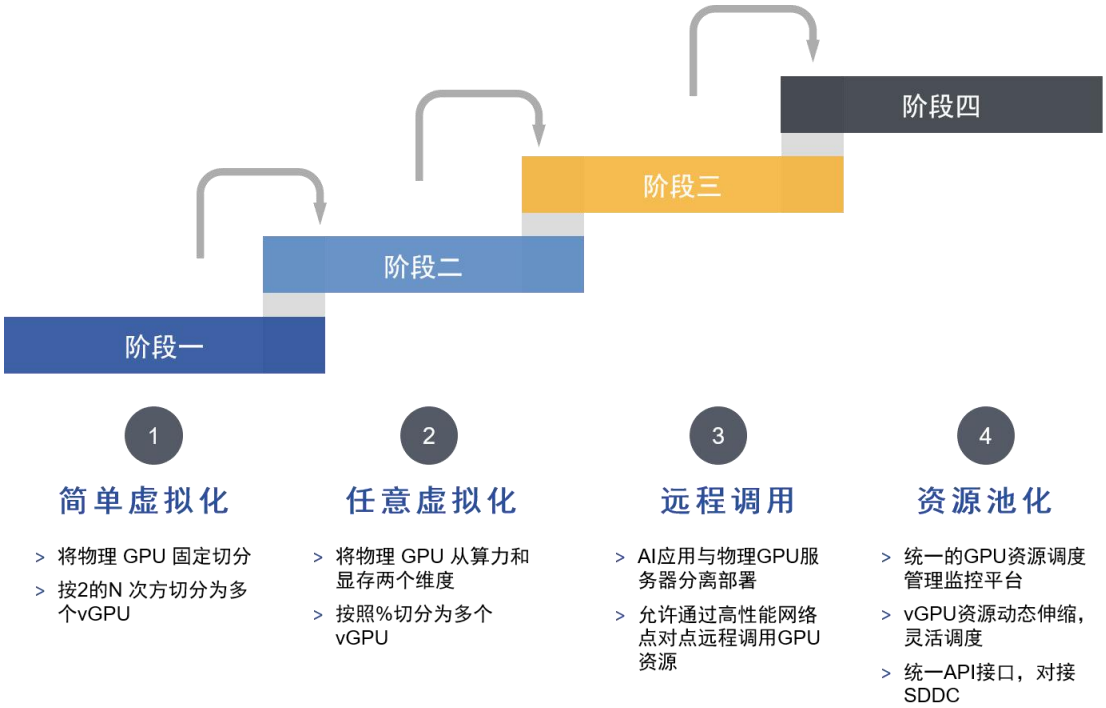
GPU 资源池化技术从初期的简单虚拟化，到资源池化，经历了四个技术演进阶段。

- **简单虚拟化**。将物理 GPU 按照 2 的 N 次方，切分成多个固定大小的 vGPU（Virtual GPU，虚拟 GPU），每个 vGPU 的算力和显存相等。实践证明，不同的 AI 模型对于算力、显存资源的需求是不同的。所以，这样的切分方式，并不能满足 AI 模型多样化的需求。

- **任意虚拟化**。将物理 GPU 按照算力和显存两个维度，自定义切分，获得满足 AI 应用个性化需求的 vGPU。

- **远程调用**。AI 应用与物理 GPU 服务器分离部署，允许通过高性能网络远程调用 GPU 资源。这样可以实现 AI 应用与物理 GPU 资源剥离，AI 应用可以部署在私有云的任意位置，只需要网络可达，即可调用 GPU 资源。

- **资源池化**。形成 GPU 资源池后，需要统一的管理面来实现管理、监控、资源调度和资源回收等功能。同时，也需要提供北向 API，与数据中心级的资源调度平台对接，让用户在单一界面，就可以调度包括 vGPU 在内的数据中心内的各类资源。

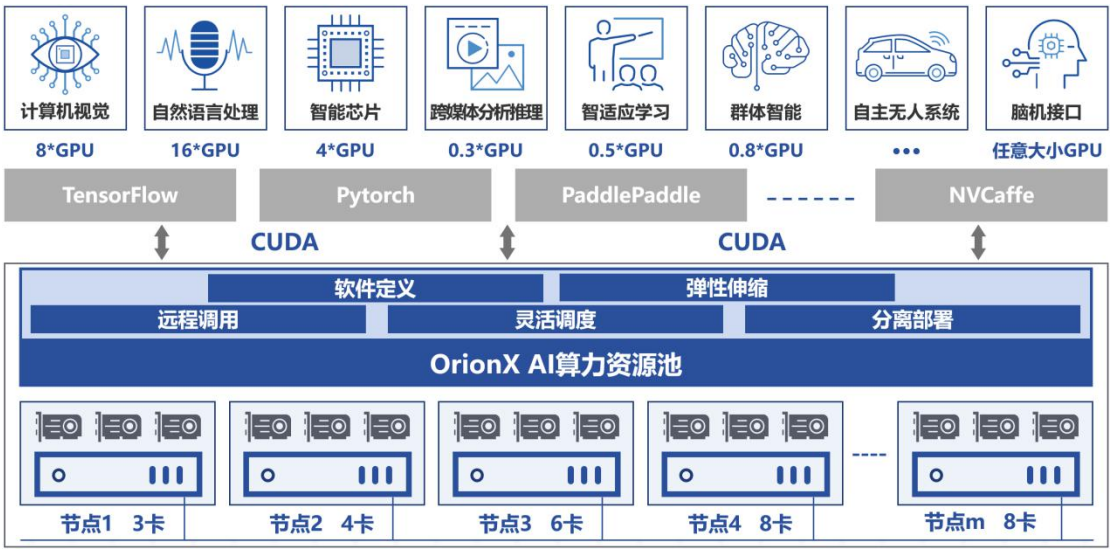


图表 2-1GPU 资源池化技术演进图

3 OrionX 产品概述

趋动科技的 OrionX（猎户座）AI 算力资源池化解决方案已经实现了上述四个阶段的技术功能，可以为用户提供 GPU 资源池化的整体解决方案。

OrionX 帮助客户构建数据中心级 AI 算力资源池，使用户应用无需修改就能透明地共享和使用数据中心内任何服务器之上的 AI 加速器。OrionX 不但能够帮助用户提高 AI 算力资源利用率，而且可以极大便利用户 AI 应用的部署。



图表 3- 1OrionX 架构图

OrionX 通过软件定义 AI 算力，颠覆了原有的 AI 应用直接调用物理 GPU 的架构，增加软件层，将 AI 应用与物理 GPU 解耦合。AI 应用调用逻辑的 vGPU，再由 OrionX 将 vGPU 需求匹配到具体的物理 GPU。OrionX 架构实现了 GPU 资源池化，让用户高效、智能、灵活地使用 GPU 资源，达到了降本增效的目的。

4 OrionX 产品优势

OrionX 通过构建 GPU 资源池，让企业内的 AI 用户共享数据中心内所有服务器上的 GPU 算力。AI 开发人员不必再关心底层资源状况，专注于更有价值的业务层面，让应用开发变得更加便 捷。OrionX 产品有如下优势：

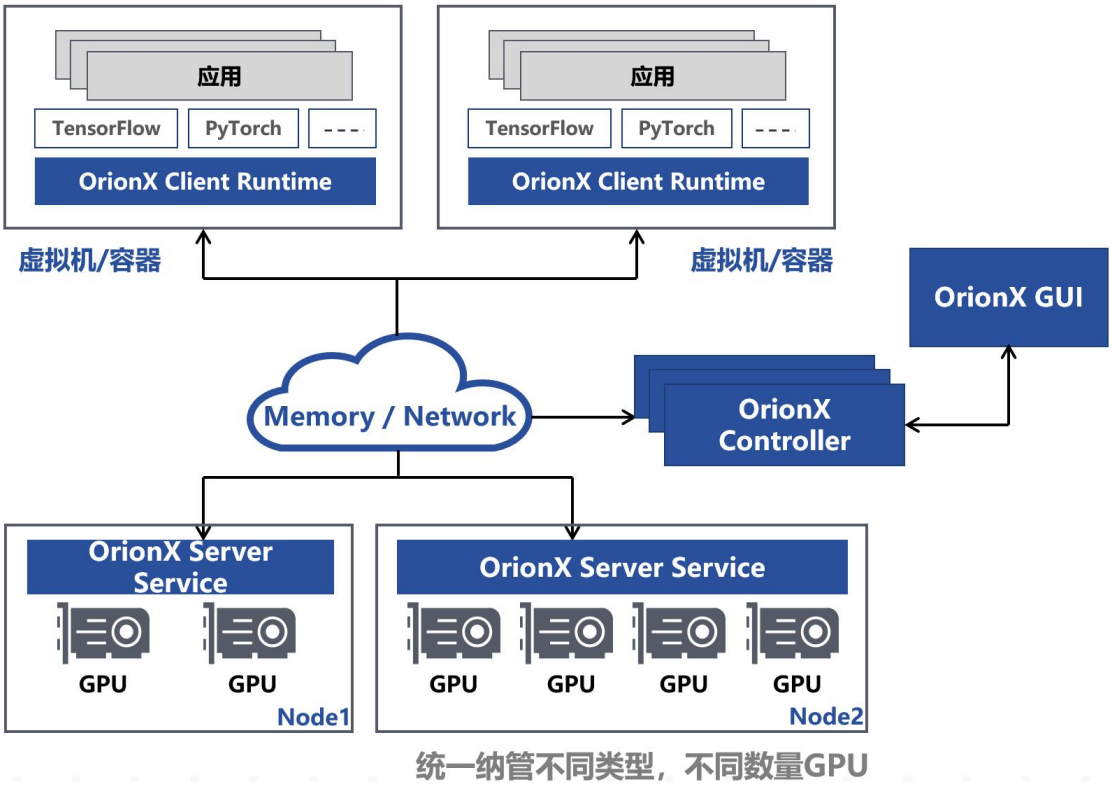
- **提高利用率**
    - 支持将 GPU 切片为任意大小的 vGPU，从而允许多 AI 负载并行运行，提高物理 GPU 利用率。
    - 提高 GPU 综合利用率多达 3-10 倍，1 张卡相当于起到 N 张卡的效果，真正做到昂贵算力平民化。
- **高性能**
    - 相比于物理 GPU，OrionX 本地 vGPU 性能损耗几乎为零，远程 vGPU 性能损耗小于 2%。
    - vGPU 资源隔离，并行用户无资源互扰。
- **轻松弹性扩展**
    - 支持从单台到整个数据中心 GPU 服务器纳管，轻松实现 GPU 资源池的横向扩展。
    - 全分布式部署，通过 RDMA（IB/RoCE）或 TCP/IP 网络连接各个节点，实现资源池弹性扩展。
- **灵活调度**
    - 支持 AI 负载与 GPU 资源分离部署，更加高效合理地使用 GPU 资源。
    - CPU 与 GPU 资源解耦合，两种服务器分开购买、按需升级、灵活调度，有助于最大化数据中心基础设施价值。
- **全局管理**
    - 提供 GPU 资源管理调度策略。
    - GPU 全局资源池性能监控，为运维人员提供直观的资源利用率等信息。
- **对 AI 开发人员友好**
    - 一键解决 AI 开发人员面临的训练模型中 GPU/CPU 配比和多机多卡模型拆分问题，为算法工程师节省大量宝贵时间。

5 OrionX 软件架构

5.1 OrionX 的逻辑架构

一个典型的 OrionX GPU 资源池的逻辑架构中包含了 OrionX Controller（OC）、OrionX Server Service（OSS）、OrionX Client Runtime（OCRT）、和 OrionX GUI（OG）等功能组件。

OrionX 的各功能组件可以根据用户环境需求被部署在单服务器上，也可以被分布式地部署在数据中心的多个物理机、虚拟机或者容器环境中。在分布式的部署环境中，各功能组件可以通过多种类型的网络建立连接，从而把数据中心的 GPU 资源管理起来，形成一种可以被全局共享的计算资源，对 AI 应用提供可远程访问的、可灵活切分的、可聚合的弹性 GPU 算力。OrionX 的逻辑架构如下图所示。



图表 5- 1OrionX 逻辑架构图

CUDA(Compute Unified Device Architecture)是由 Nvidia 公司定义且公开推广、维护的一种 GPU 编程接口。从 2007 年推出之后，经过十几年生态培育，已经成为 GPU 编程的一

个事实标准。大部分流行的 AI 框架，例如 TensorFlow、PyTorch、MXNet 和 PaddlePaddle 都是基于 CUDA 编程接口开发。

OrionX 在管理物理 GPU 之后，通过模拟 CUDA 标准接口，为各种 AI 应用提供一个与 Nvidia CUDA SDK 接口功能一致的运行环境，从而使得 AI 应用透明无感知地运行在 OrionX GPU 资源池之上。OrionX 不仅在单服务器上模拟了 CUDA 标准接口，并且通过分布式部署各功能组件，能够提供分布式的 CUDA 运行环境。

5.2 OrionX 的功能组件

5.2.1 OrionX Controller（OC）

OrionX Controller 是 GPU 资源池的核心管理调度模块，其他所有 OrionX 的功能组件都直接或者间接通过网络连接到 OrionX Controller，并与其保持信息同步。为了实现 OrionX GPU 资源池的统一管理以及资源调度，节点 IP 地址、物理 GPU 信息、虚拟 GPU 信息以及应用任务信息等都会汇总至该组件。

一个 OrionX GPU 资源池可以只部署一个 OrionX Controller。为了提高 OrionX 的可靠性，可以进行 2+1 冗余备份。OrionX Controller 提供如下功能：

- 各个分布式功能组件的服务注册、服务发现功能。
- 弹性虚拟 GPU 的调度分配功能。
- 多副本高可用的元数据存储和管理。
- License 管理。
- 提供运维所需要的各种 Rest API。

5.2.2 OrionX Server Service（OSS）

OrionX Server Service 发现并管理物理节点上的 GPU 资源，同时把物理 GPU 的计算能力通过 OrionX 的高性能私有协议提供给数据中心内的各个物理节点，以及各个物理节点上的虚拟机、容器。

OrionX Server Service 部署在 OrionX 资源池内的每一个节点上，包括 GPU 节点和应用所在的节点。OrionX Server Service 提供如下功能：

- 发现和管理物理 GPU 资源。
- 把物理 GPU 资源抽象成弹性的 vGPU。
- 执行 AI 应用的 GPU 计算任务。
- 支持虚拟机、容器的网络隔离。

5.2.3 OrionX Client Runtime（OCRT）

OrionX Client Runtime 是一套兼容 Nvidia CUDA 编程环境的运行环境，模拟了 CUDA 的运行时接口。当 AI 应用在使用 Nvidia GPU 进行计算的时候，会自动调用 OrionX Client Runtime。由于 OrionX Client Runtime 提供和 Nvidia GPU 兼容的 CUDA 接口，因此应用无需修改，可以透明无感知地运行在一个虚拟的 GPU 环境下。

OrionX Client Runtime 部署在每一个应用环境下，替代原有的 Nvidia CUDA SDK。OrionX Client Runtime 提供如下功能：

- 兼容 CUDA 接口。
- 自动完成虚拟 GPU 资源的申请、释放、弹性伸缩等功能。
- 支持虚拟机、容器和宿主机的网络隔离。

5.2.4 OrionX GUI（OG）

OrionX GUI 给运维提供一个友好的 GUI 界面，方便管理员对 OrionX 整体资源池进行全面管理。OrionX GUI 提供如下功能：

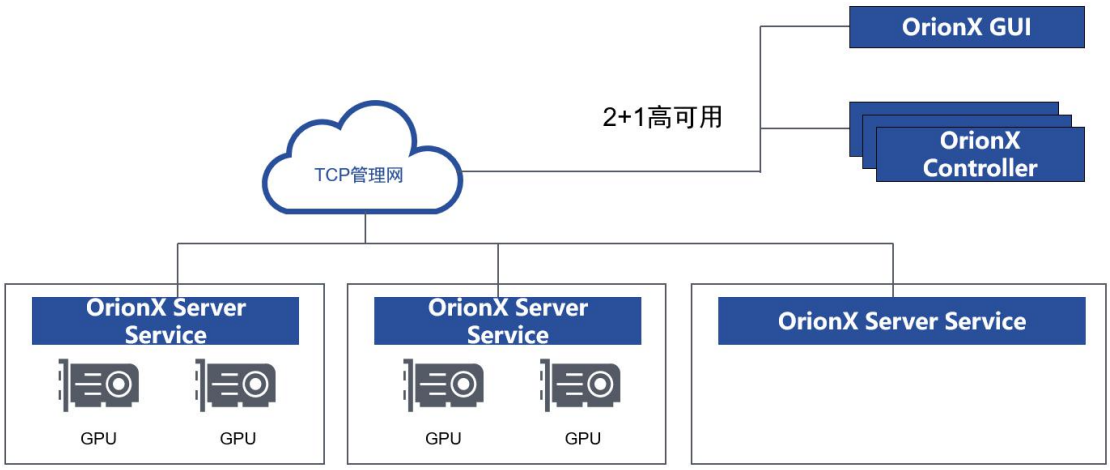
- 分级的运维账号登录与管理。
- 查看各组件的部署拓扑以及详情。
- 多维度、多角度查看资源池的资源使用情况。
- 细粒度管理资源池内的资源。
- 日志、监控及告警功能设置。

5.3 OrionX 组件间通信

OrionX 的各个功能组件通过管理平面网络和数据平面网络进行通信，共同完成 GPU 资源池的管理以及 GPU 资源的调度等功能。

5.3.1 管理平面

在部署 OrionX 时，使用基于 TCP/IP 网络的管理平面，来承载整个系统的管理工作。通过管理网络，分布在各个节点的功能组件都保持和 OrionX Controller 同步。管理平面逻辑结构如下图所示。



图表 5-2 管理平面逻辑结构图

通过私有的同步协议，分布式部署的各个功能组件具有如下特点：

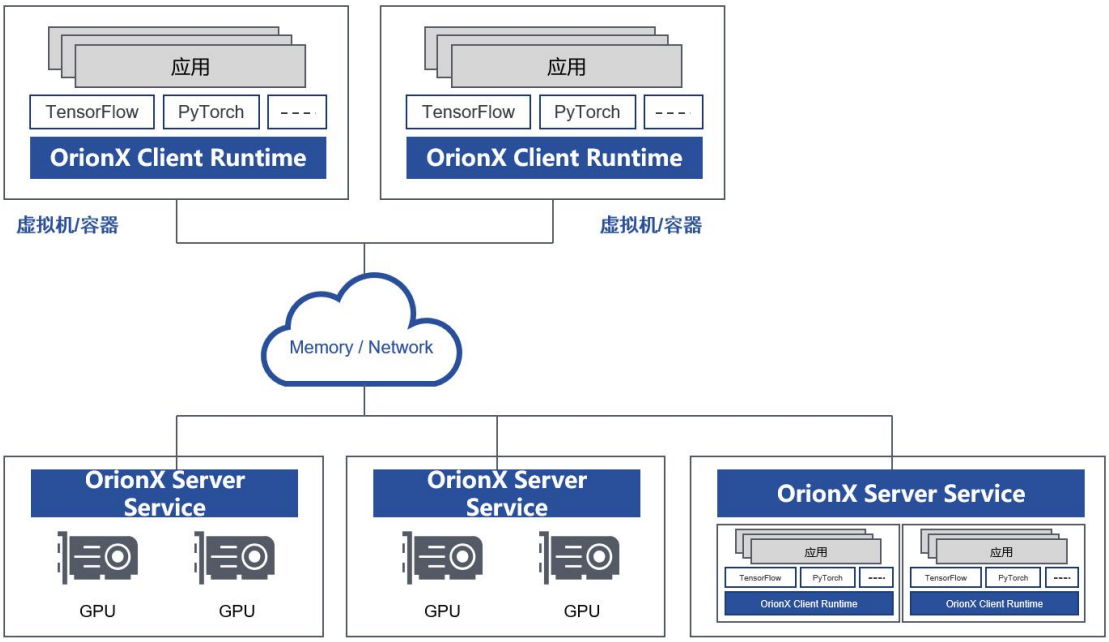
- OrionX Controller 支持多副本、高可用的部署模式。
- 各个功能组件启动的次序无要求。
- 当某一个功能组件从错误中恢复之后，可以自动同步到正确的状态。

5.3.2 数据平面

在应用运行的过程中，应用所在环境和 GPU 物理节点之间的数据传输使用的是 OrionX 的数据面。该数据面支持多种后端数据传输载体，包括 TCP/IP 以太网络、RoCE RDMA、

Infiniband RDMA、Share Memory 等。数据面具有如下的特点：

- 高带宽、低延迟。
- 同时支持多种传输协议，根据优先级自动使用高性能的传输方式。
- 支持虚拟机、容器和宿主机之间的 TCP/IP 网络隔离。



图表 5- 3 数据平面逻辑结构图

6 部署形态

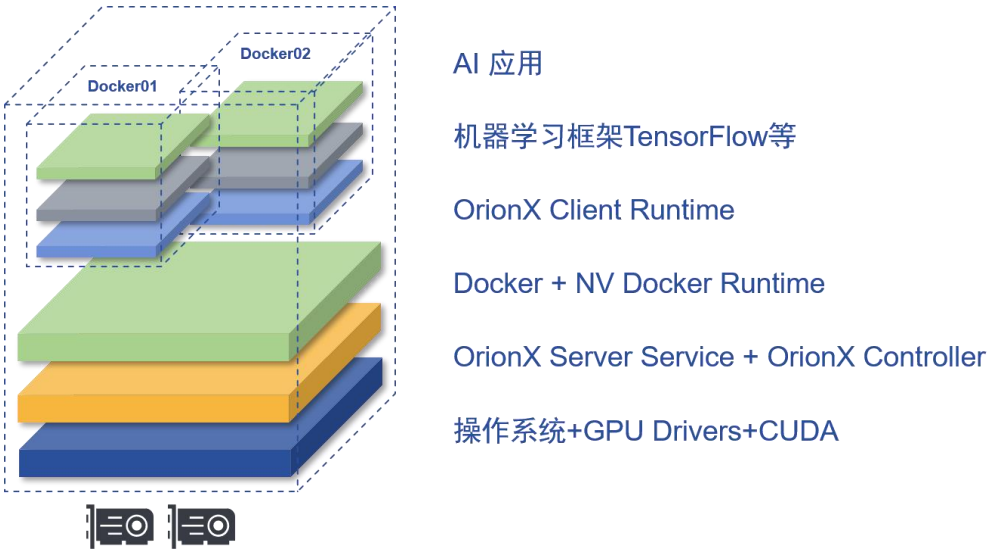
OrionX 的各个组件，支持直接部署在裸金属服务器上，即安装操作系统后，直接以 Binary 形式部署，也支持容器化部署。OrionX 具备适配多种 Linux 操作系统和云平台的能力，因此，OrionX 具有多样化的部署形式。

OrionX 支持 CentOS、Ubuntu、Debian 等 Linux 发行版本，同时支持基于 KVM 的虚拟机云平台和基于 Docker 的容器云平台。尤其是支持原生容器，并实现了和 Kubernetes 的平滑对接。

6.1 OrionX 与容器云平台集成

OrionX 支持原生容器，各个组件都可以通过容器镜像方式部署。在容器环境中，客户只需要使用 OrionX 组件提供的启动脚本，就可以一键完成 OrionX 的组件安装，轻松实现 GPU 资源池化。

OrionX 的容器部署方式，将 GPU Drivers、CDUA、CUDNN 和 NCCL 等软件栈都下沉到宿主机上，容器内部只需要安装 OrionX Client Runtime 和机器学习框架，即可运行 AI 应用，



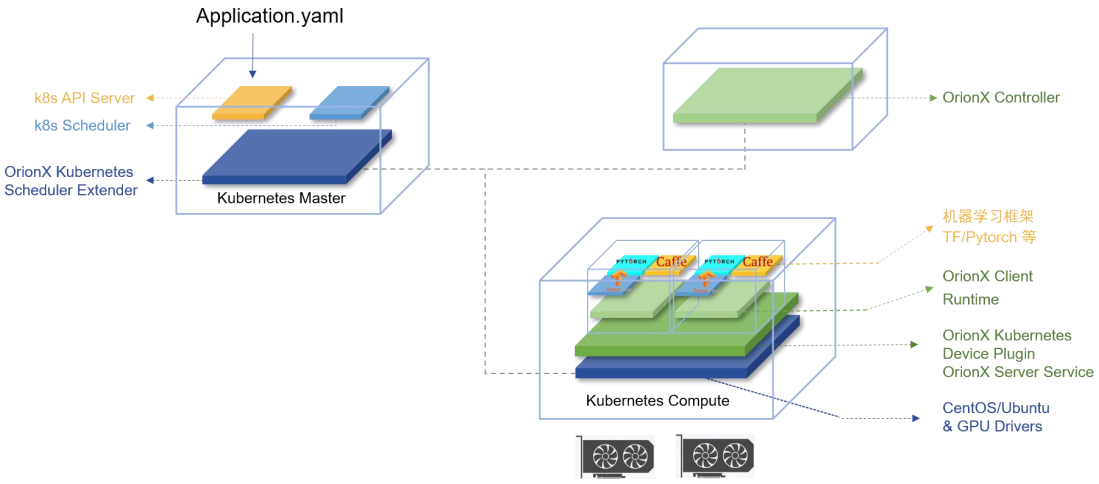
大大简化了客户算法工程师运维、管理 AI 基础架构的工作。

图表 6- 1OrionX 与容器云平台集成

6.2 OrionX 与 Kubernetes 集成

OrionX 为 Kubernetes 提供两个插件，实现与 K8S 的集成对接。集成后，系统管理员只需要在 K8S 中，即可完成对 GPU 资源池中 vGPU 资源的配置和调度管理。并且，允许系统管理员通过单一接口调度全部数据中心资源，实现 SDDC（Software Defined Data Center，软件定义的数据中心），这样就简化了运维工作。OrionX 为 Kubernetes 提供的两个插件是：

- **OrionX Kubernetes Device Plugin**
    - 通过和 OrionX Controller 通讯，获取 OrionX GPU 资源池信息。
    - 通过 Kubernetes 定义的 Device Plugin 标准向 Kubernetes 注册名字为 *virtaitech.com/gpu* 的资源。
- **OrionX Kubernetes Scheduler Extender**
    - 提供基于 HTTP API 通讯的松耦合调度扩展功能。
    - 通过配置文件向 Kubernetes 注册名字为 *virtaitech.com/gpu* 的资源敏感字，使其指向 Orion Kubernetes Scheduler Extender 的 HTTP 服务地址。



图表 6- 2OrionX 和 Kubernetes 集成

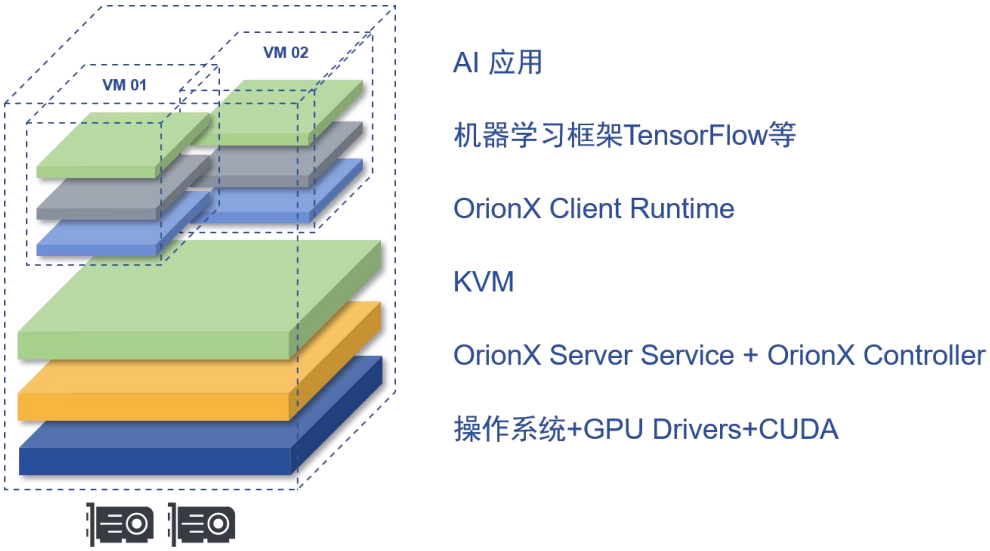
6.3 OrionX 与 KVM 集成

OrionX 支持原生 KVM，各个组件都可以通过 Binary 方式部署。在 KVM 环境中，客户使用

OrionX 组件的安装脚本，就可以完成 OrionX 的基础部署，轻松实现 GPU 资源池化。

OrionX 的 KVM 部署方式，是将 OrionX Controller 和 OrionX Server Service 以 Binary 方式部署在宿主机上，将 OrionX Client Runtime 部署在 VM 中，将 GPU Drivers、CDUA、CUDNN 和 NCCL 等软件栈都下沉到宿主机上。这样，VM 内部只需安装 OrionX Client Runtime 和机器学习框架，即可运行 AI 应用，大大简化了客户算法工程师运维管理 AI 基础架构的工作。

基于 KVM 的云平台，例如 OpenStack，OrionX 提供全开放的 Rest API 接口，与 Nova 组

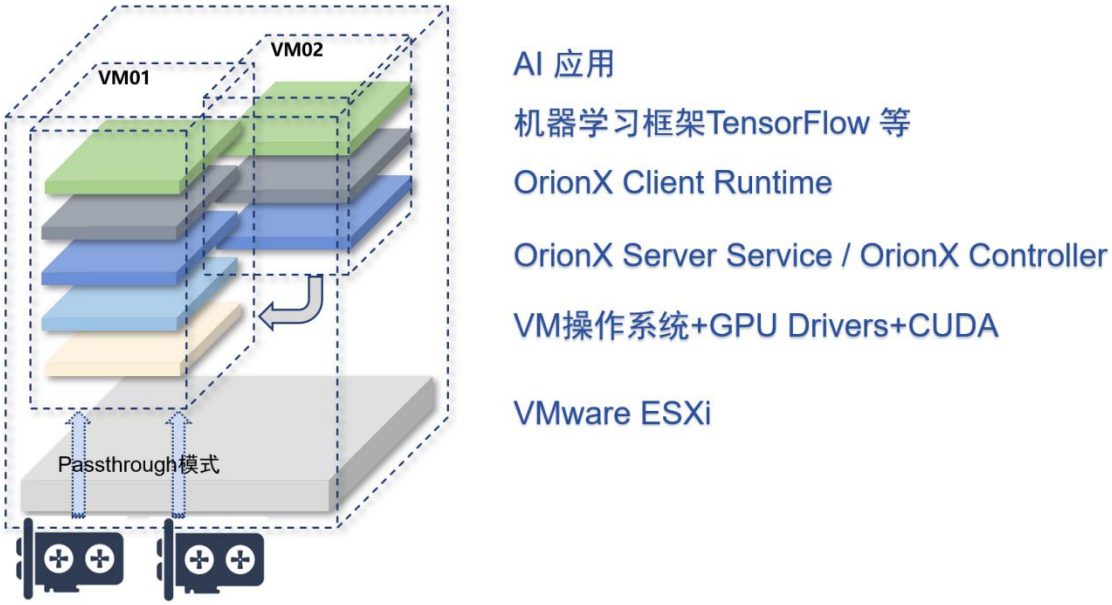


件对接，实现 GPU 资源池中的 vGPU 资源在 OpenStack 中的集中配置和调度管理。

图表 6- 3OrionX **和** KVM **集成**

6.4 OrionX 与 VMware 集成

OrionX 支持 VMware vSphere，各个组件都可以通过 Binary 方式在 VM 内部署。在 vSphere 中，将物理 GPU 通过直通方式全部透传给一个 VM，在该 VM 内部署 OrionX Controller 和 OrionX Server Service 组件，即可轻松实现 GPU 资源池化。该 VM 上或者其他 VM 上的 AI 应用即可通过 OrionX Client Runtime 组件调用虚拟 GPU 资源，大大简化 GPU 资源在 vSphere 环境中的分配难度，提高 GPU 资源的调度效率。



图表 6-4OrionX 和 VMware 集成

7 OrionX 应用场景

7.1 OrionX 支持大模型场景的典型应用

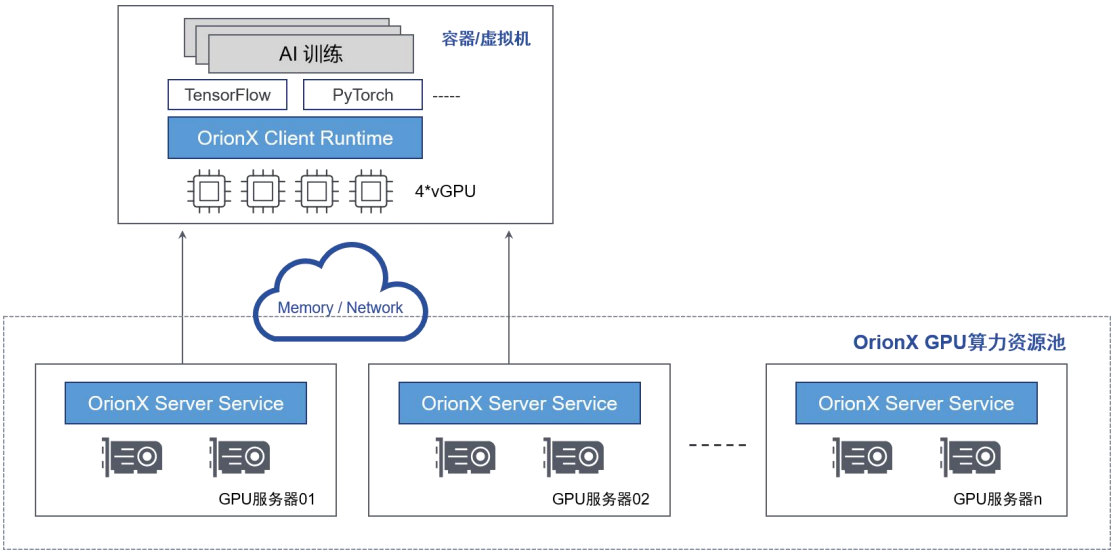
大模型场景如训练场景，对算力资源需求量大，通常会使用一张或者多张 GPU 卡资源。作为 AI 算力资源池平台，OrionX 既可以支持单台服务器上的单卡、多卡训练，也可以支持跨设备的多卡训练。

7.1.1 通过“化零为整”功能支持训练

OrionX 支持将多台服务器上的 GPU 提供给一个虚拟机或者容器使用，而该虚拟机或者容器内的基于分布式训练框架（Horovod 或 DistributedDataParallel）的 AI 应用无需修改代码。通过这个功能，用户可以将多台服务器的 GPU 资源聚合后提供给单一虚拟机或者容器使用。“化零为整”支持训练等大模型场景，为用户的 AI 应用提供数据中心级的海量算力。

Horovod 是 Uber 开源的分布式深度学习框架，旨在使分布式深度学习变得快速且易于使用，使模型训练时间从几天和几周缩短到数分钟和数小时。使用 Horovod，可以将现有的训练脚本扩大规模，使其仅用几行 Python 代码就可以在跨设备的多个 GPU 上运行。一旦配置了 Horovod，就可以使用相同的基础结构来训练具有任何框架的模型，从而随着机器学习技术堆栈的不断发展，轻松地在 TensorFlow、PyTorch、MXNet 和将来的框架之间进行切换。

DistributedDataParallel(简称 DDP)是 PyTorch 自带的分布式训练框架，支持多机多卡和单机多卡分布式训练。DDP 属于 Data Parallel，可以通过提高 batch size 来增加并行度。DDP 通过 Ring-Reduce 的数据交换方法提高了通讯效率，并通过启动多个进程的方式减轻



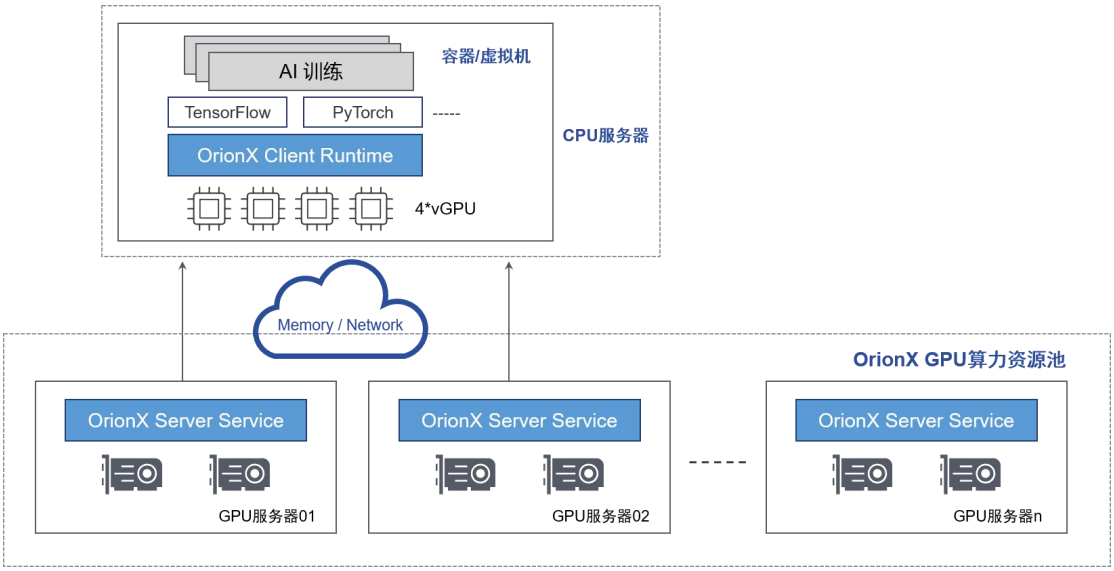
Python GIL 的限制，从而提高训练速度。

图表 7- 1 通过化零为整功能支持训练

7.1.2 通过“隔空取物”功能支持训练

OrionX 支持将虚拟机或者容器运行在一台没有物理 GPU 的服务器上，通过计算机网络，透明地使用其他服务器上的 GPU 资源，该虚拟机或者容器内的 AI 应用无需修改代码。通过这个功能，OrionX 帮助用户实现了数据中心级的 GPU 资源池，实现了 AI 应用和 GPU 物理资源的解耦合，AI 应用在一个不满足训练条件的纯 CUP 服务器上，也一样能够快速调集多个 GPU 卡完成训练任务。

“隔空取物”支持训练等大模型场景，既可以调取单台设备的多卡资源给容器或者虚拟机，实现类似单机多卡训练的场景；也支持调取多台设备的多卡资源给容器或者虚拟机，实现类似分布式多机多卡训练的场景。



图表 7- 2 通过隔空取物功能支持训练

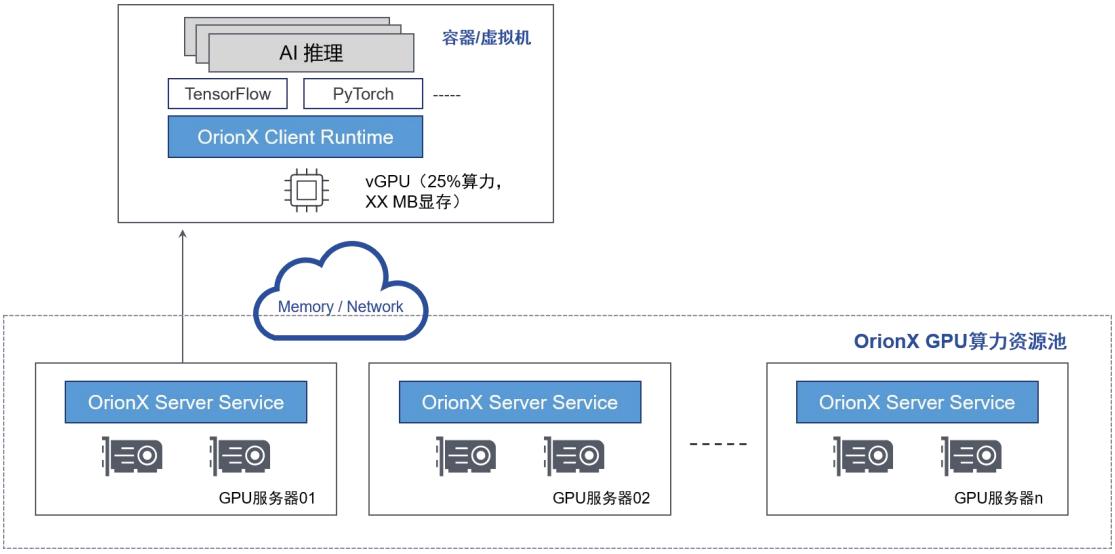
7.2 OrionX 支持小模型场景的典型应用

小模型如推理、开发、教学实训等场景，对算力资源需求量小，通常不能占满一张 GPU 卡资源。作为 AI 算力资源池平台，OrionX 可以从算力和显存两个维度，切分 GPU。支持将多个小模型任务调度到一张卡，有效提高资源利用率。

7.2.1 通过“化整为零”功能支持推理

OrionX 支持将一块物理 GPU 细粒度切分成多块 vGPU，然后分配给多个虚拟机或者容器。每一块 vGPU 的显存和算力都能被独立设置和限制。通过这个功能，用户可以高效地共享 GPU 资源，提高 GPU 利用率，降低成本。

算力切分的最小颗粒度为原物理 GPU 算力的 1%；显存切分的最小颗粒度为 1MB。

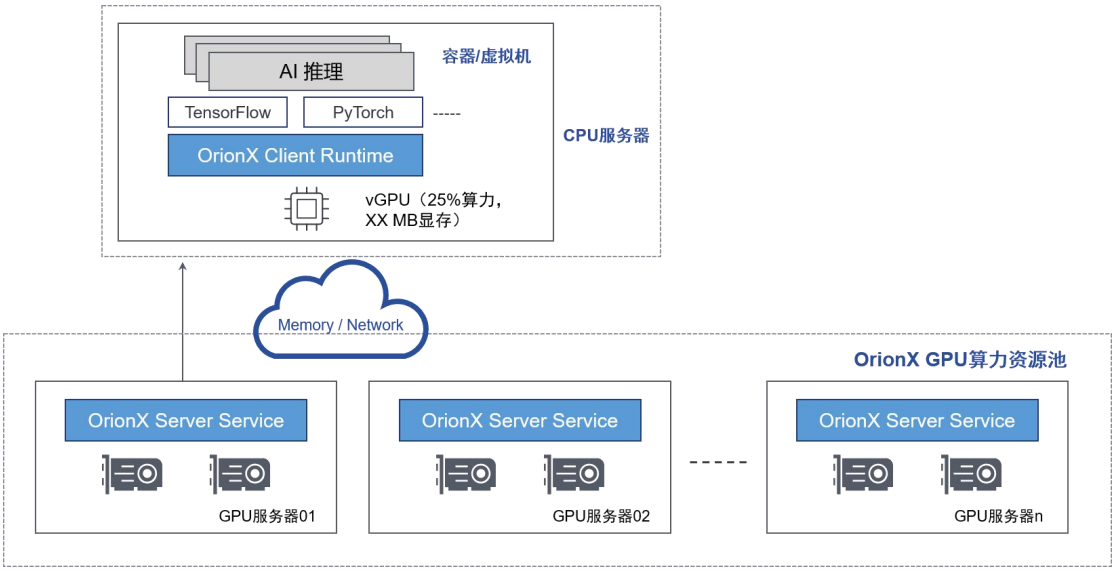


图表 7- 3 通过化整为零功能支持推理

7.2.2 通过“隔空取物”功能支持推理

OrionX 支持将虚拟机或者容器运行在一台没有物理 GPU 的服务器上，通过计算机网络，透明地使用另一台服务器上的 GPU 资源，该虚拟机或者容器内的 AI 应用无需修改代码。通过这个功能，OrionX 帮助用户实现了 CPU 和 GPU 资源的解耦合。在一些 CPU 与 GPU 需要双向平衡的推理场景下，OrionX 可以更好的平衡 CPU 和 GPU 资源的分配，减少短板资源的发生，拉平数据中心的利用率。

“隔空取物”支持推理、开发、教学实训等小模型场景，可以调取单台设备的细粒度卡资源给容器或者虚拟机，将多个小模型应用调用到一张物理 GPU 中，并严格限制 vGPU 资源间的隔离，实现资源利用率最大化。



图表 7-4 通过隔空取物功能支持推理

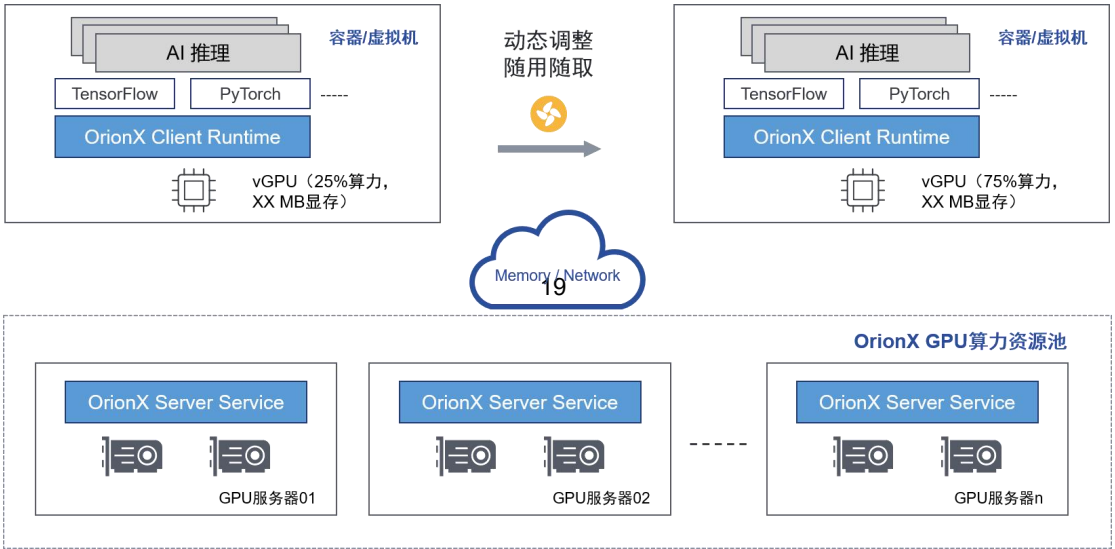
7.3 OrionX 支持大/小模型场景的典型应用

7.3.1 通过“随需应变”功能支持训练/推理

OrionX 支持用户在虚拟机或者容器的生命周期内，动态分配和释放所需要的 GPU 资源。通过这个功能，OrionX 帮助用户实现了真正的 GPU 资源动态伸缩，极大提升了 GPU 资源调度的灵活度。

OrionX 支持 vGPU 资源按需分配、随用随取，最大限度的利用算力资源。不论是大模型训练，还是小模型推理的环境中，用户都可以按照 AI 模型需求，动态的调整算力资源大小，而无需重启挂载 vGPU 的虚拟机/容器。OrionX 支持 vGPU 资源预留模式和获取模式：

- **预留模式：**和使用物理 GPU 类似，客户申请的 vGPU 是独占的，不可被其他用户使用。
- **获取模式：**客户申请的 vGPU 是动态的，只有在客户的 AI 应用运行时，vGPU 资源才锁定到具体的物理 GPU，一旦 AI 应用结束，物理 GPU 资源及时释放。

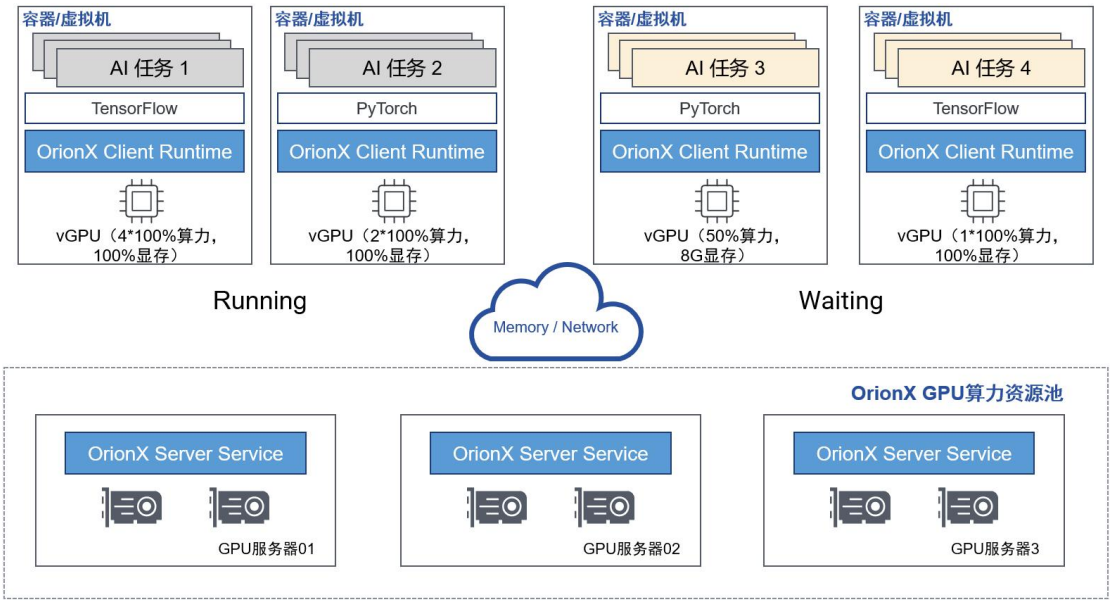


图表 7- 5 通过随需应变功能支持训练/推理

7.3.2 通过“任务队列”功能支持训练/推理任务自动排队

当请求 vGPU 资源的训练/推理任务遇到物理 GPU 剩余资源不足，无法分配的时候，OrionX 支持任务排队能力。OrionX 会将任务放入等待队列中，直到队列中前面任务跑完，可调度等待任务所需资源时，任务进入运算状态。

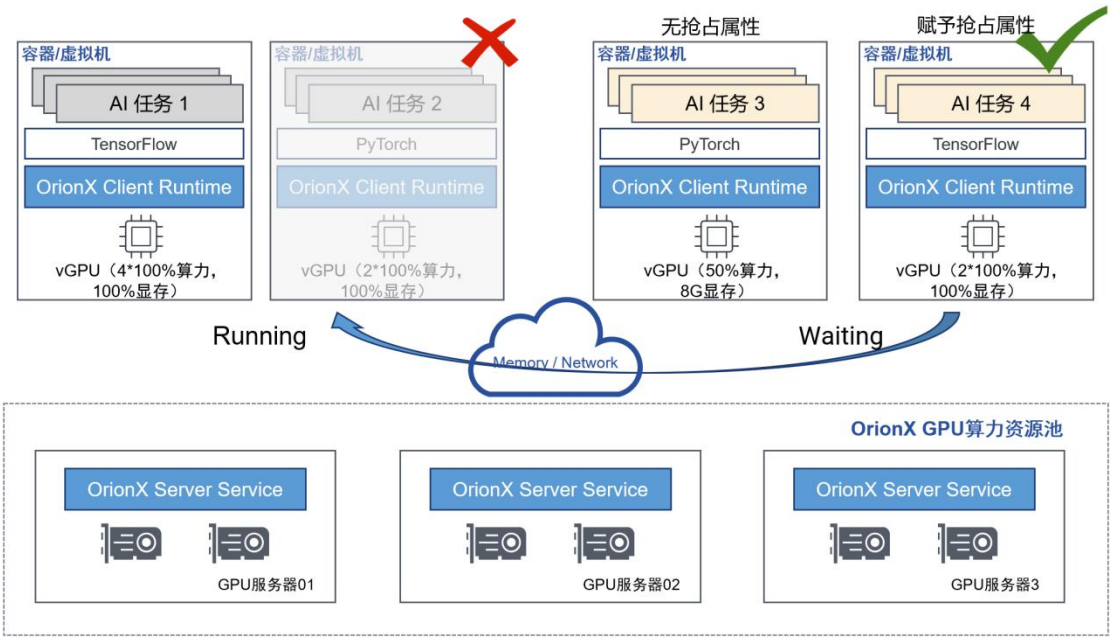
OrionX 允许对任务进行队列优先级预设，一旦资源缺乏导致任务进入等待队列，按照优先级进行全局排列，优先匹配重要任务。



图表 7- 6 通过随需应变功能支持训练/推理

7.3.3 通过“抢占”功能支持任务抢占资源

上述“任务队列”中，OrionX 允许对某些任务赋予抢占属性，一旦资源不足导致任务进入等待队列，按照抢占属性的特点，该任务将立即分配到可用资源。赋予抢占属性的任务，在 OrionX 集群剩余资源不足而导致无法分配到资源时，可以通过抢占正在运行的任务，让一部分任务提前退出，这样集群可能就会空出足够的资源分配给该任务。

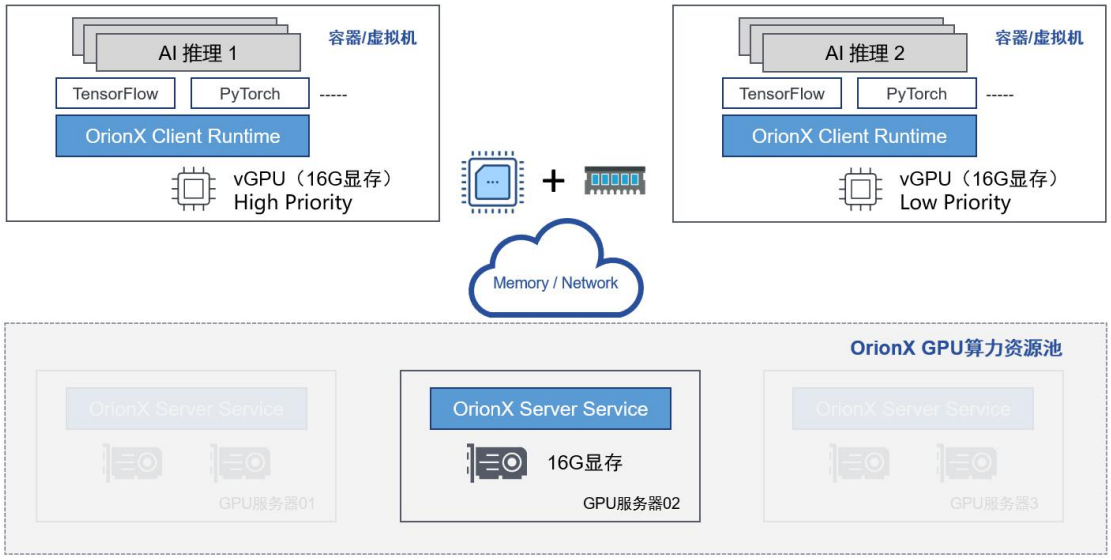


7.3.4 通过“显存超分”功能支持多任务叠加常驻

通常推理任务为满足最佳用户体验，会将推理模型常驻显存，24 小时不中断，以便拥有最快响应速度。但是这类常驻任务一般算力利用极低，而且潮汐效应明显。

OrionX 支持多任务潮汐叠加。通过“显存超分”，OrionX 会调用系统内存补充 GPU 显存，在逻辑上扩大 GPU 显存的承载容量，从而支持多个常驻显存的长尾任务叠加在同一个物理 GPU 上，提高单个 GPU 的承载量，充分利用 GPU 闲置算力。

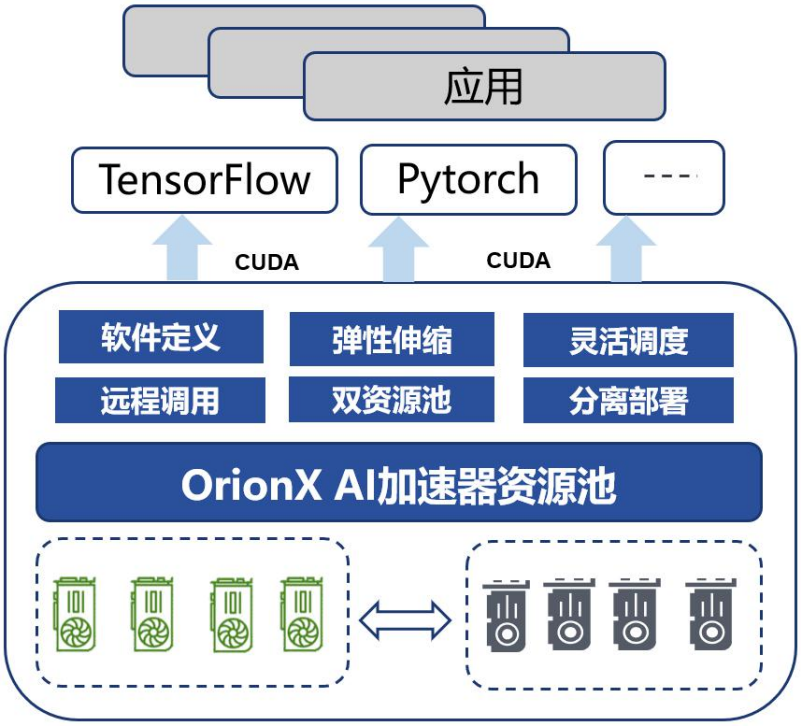
根据业务特点，OrionX 还支持不同任务设置不同优先级，从而保证突发高优先级任务的服务质量。



图表 7- 7 通过随需应变功能支持训练/推理

7.3.5 通过“双类资源池”功能支持物理/虚拟切换

个别 AI 任务由于程序本身自有的特殊性，需要直接使用物理 Native GPU 资源，OrionX 支持同时纳管 OrionX GPU（即经过 OrionX 池化管理的 GPU，可以被虚拟化为多个 VGPU），和 Native GPU（即原生 GPU，不会被虚拟化）。OrionX 能够在一个界面上方便的控制哪些 GPU 卡初始化上报为 OrionX GPU, 哪些 GPU 卡被初始化上报为 Native GPU。在初始化上报结束以后，依然能够灵活的在 OrionX GPU 和 Native GPU 之间安全的做切换。通过“双类资源池”功能，OrionX 可以同时支持 AI 任务申请 Native GPU 或者申请虚拟化之后的 OrionX GPU 两类不同资源，以应对不同任务资源申请的需求。


应用
TensorFlow
Pytorch

CUDA
CUDA
软件定义
弹性伸缩
灵活调度
远程调用
双资源池
分离部署
OrionX AI加速器资源池

8 性能测试

8.1 测试环境

- **硬件配置**
  - 双路 Intel Xeon Gold 6132
  - 128GB 内存
  - Tesla P40
- **软件配置**
  - KVM 环境：libvirt 1.3.0，qemu 2.5.0
  - NVIDIA 显卡驱动 418.43
  - CUDA 9.0
  - cuDNN 7.4.2
- **测试用例**
  - TensorFlow v1.12、官方 benchmark、使用 synthetic 数据

8.2 测试结果

测试结果说明：

- Native GPU：表示将测试用例运行在物理 GPU 之上，不使用虚拟机或者容器的性能。
- Orion Local Container：表示将测试用例运行在安装了 OrionX 软件的容器之中，使用本机 GPU 的性能。
- Orion Local KVM：表示将测试用例运行在安装了 OrionX 软件的 KVM 虚拟机之中，使用本机 GPU 的性能。

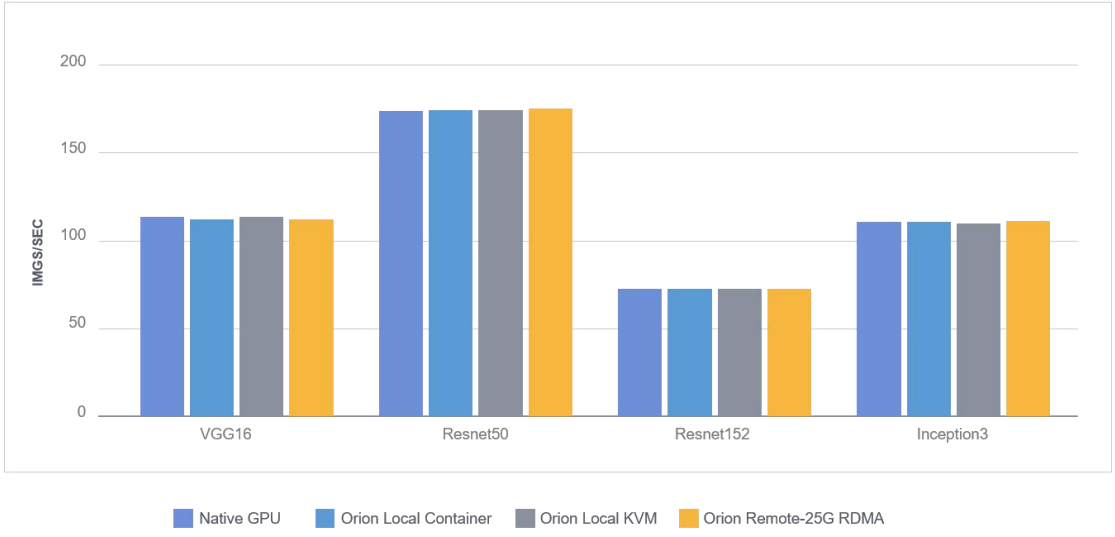
- Orion Remote – 25G RDMA：表示将测试用例运行在一台没有 GPU 的物理机之上，

性能数据 – 模型推理 – nVidia Tesla P40

通过 25G RDMA 网卡使用远程 GPU 的性能。

图表 8- 1 模型推理测试结果

性能数据 – 模型训练 – nVidia Tesla P40



图表 8- 2 模型训练测试结果

9 兼容性列表

- **网络**
  - TCP/IP 以太网络
  - RDMA 网络（InfiniBand 和 RoCE）
- **NVIDIA GPU**
  - Ampere 架构：A100,A10,A30,A40
  - Turing 架构：T4,RTX8000,RTX6000,RTX5000
  - Volta 架构：V100s, V100
  - Pascal 架构：P100,P40,P4
  - Maxwell 架构：M60,M40
  - Kepler 架构：K80,K40
- **NVIDIA CUDA**
  - CUDA 11.0, 11.1, 11.2, 11.3, 11.4
  - CUDA 10.0, 10.1, 10.2
  - CUDA 9.0, 9.1, 9.2
- **操作系统**
  - 64 位 CentOS 7.X
  - 64 位 Ubuntu 18.04 LTS、16.04 LTS、14.04 LTS
- **云平台**
  - 容器环境：Docker 1.13 及以后版本
  - Kubernetes 环境：Kubernetes 1.10 及以后版本
  - KVM 环境：QEMU-KVM(QEMU 2.x)
- **深度学习框架**
  - TensorFlow 1.8-2.4.1
  - PyTorch 1.0-1.8
  - PaddlePaddle 1.5-1.6，2.0-2.1
  - MXNet 1.4.1
  - Xgboost 0.72、0.8、0.9
  - Deepsheech V1.1、V1.3、V2.0
  - NVCaffe 1.0
  - TensorRT 5、6、7

注：兼容性测试基于 OrionX-CS-2.8.2 + OrionX-Controller-2.5.0 版本验证。