DaoCloud

AI 进阶指南：GPU 集群管理的最佳实践

主讲人：卢传佳（船长）-- DaoCloud AI 产品经理

Content

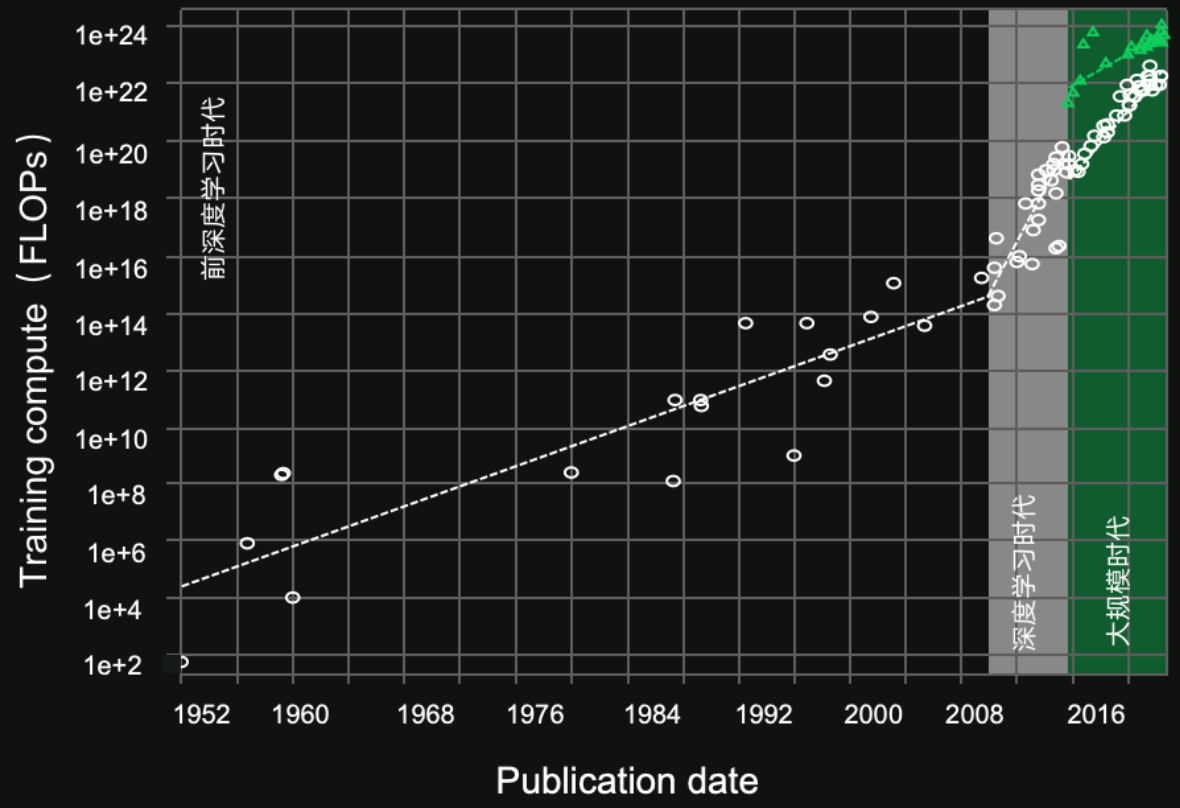
目录

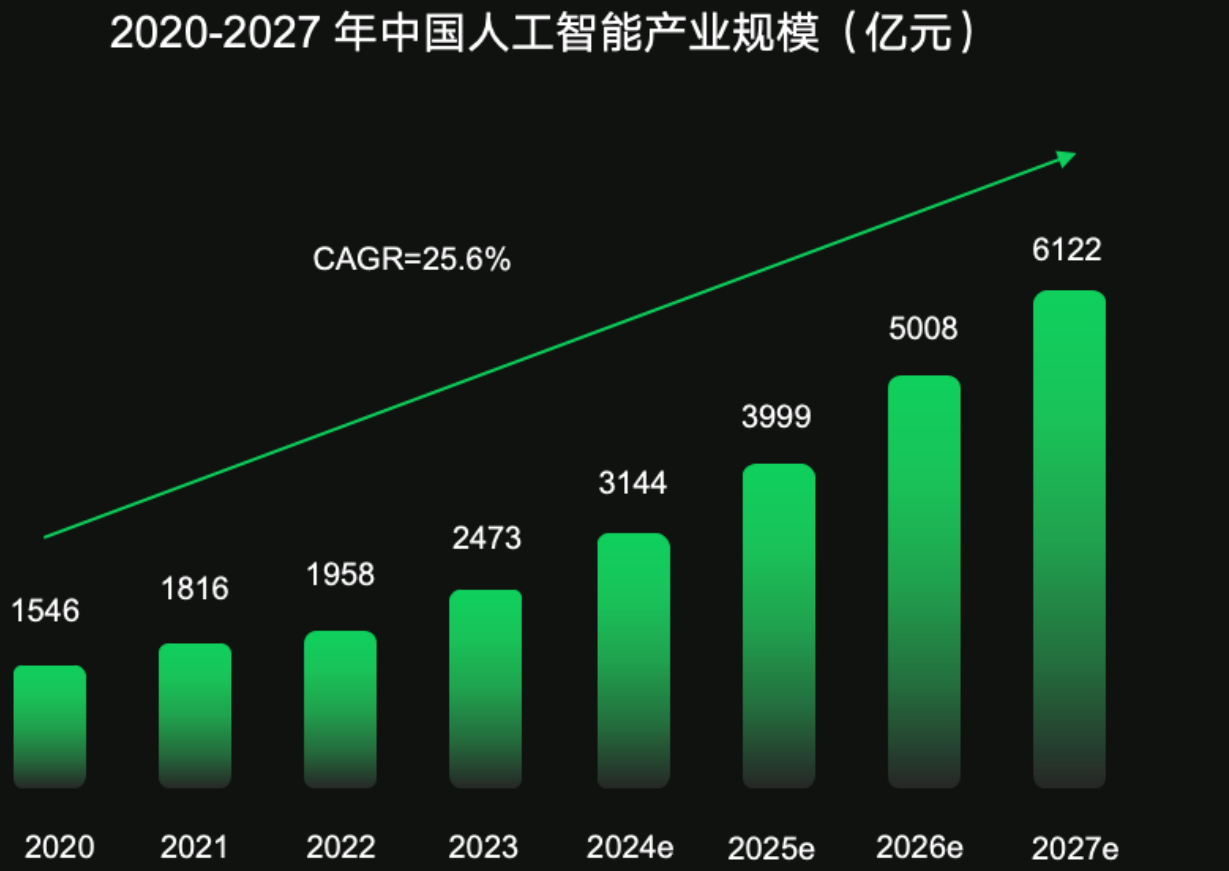
Part 01

引言

大模型引发的算力需求井喷式

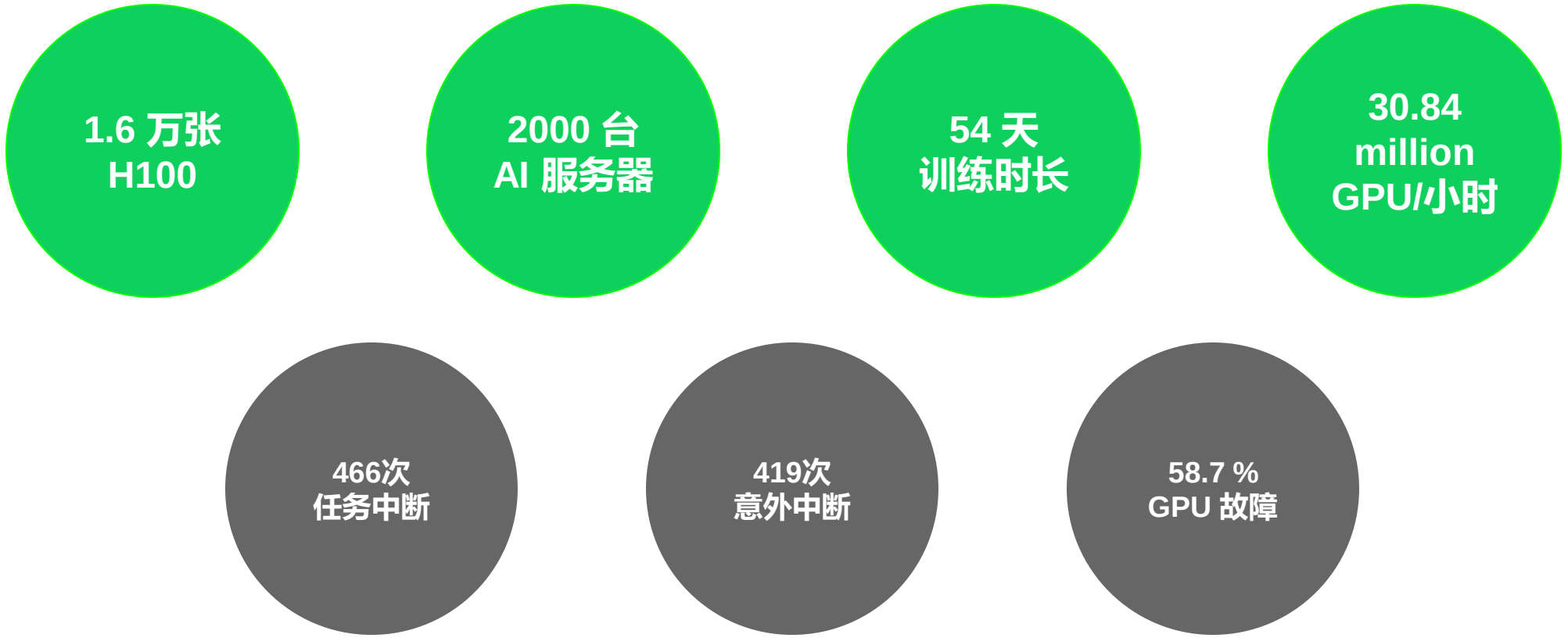
- SemiAnalysis：GPT-4 在大约 2.5 万个 A100 （GPU 利用率 32%-36%）上训练 90-100 天，单次训练成本 6300 万美元； GPT-5 的训练可能需要 3 万-5 万块 H100。
- Meta: LLama 3.1 405B 使用了 1.6 万张 H100（GPU 利用率达到了 90% 以上）
- Elon Musk： Grok 2 训练使用了 2.4 万张 H100。





举个栗子：LLama 3.1 405B 训练过程

Meta 在 7月 23 日正式发布了 Llama 3.1 系列模型，其中 405B 版本是目前最大的开源模型之一 。



Kubernetes 作为算力平台的优势

GPU 凭借其并行计算能力和专用架构,在AI训练和推理中发挥关键作用,显著提升性能和效率,推动AI技术快速发展,成为现代AI基础设施的核心组件。

Kubernetes 作为领先的开源容器编排平台，在容器化部署和自动化调度优化能力，可以高效管理 GPU 等昂贵的计算资源。

并行处理

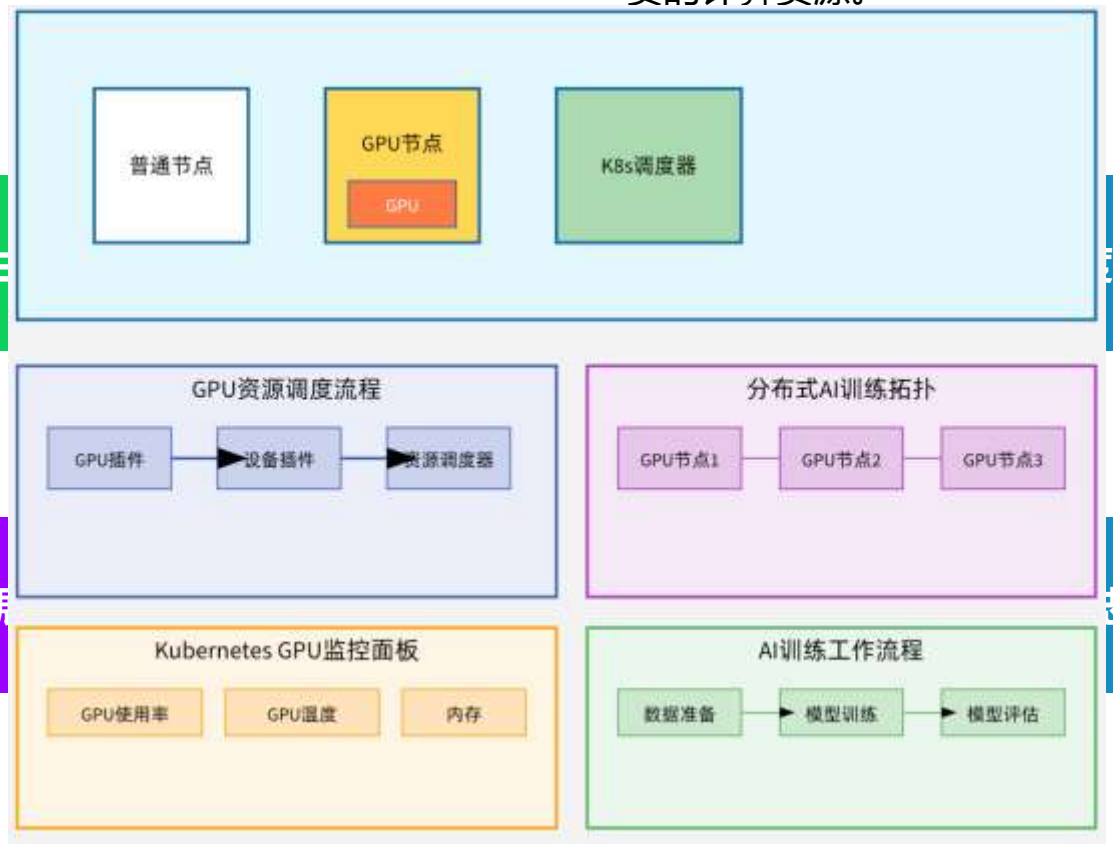
软件生

多集群管理

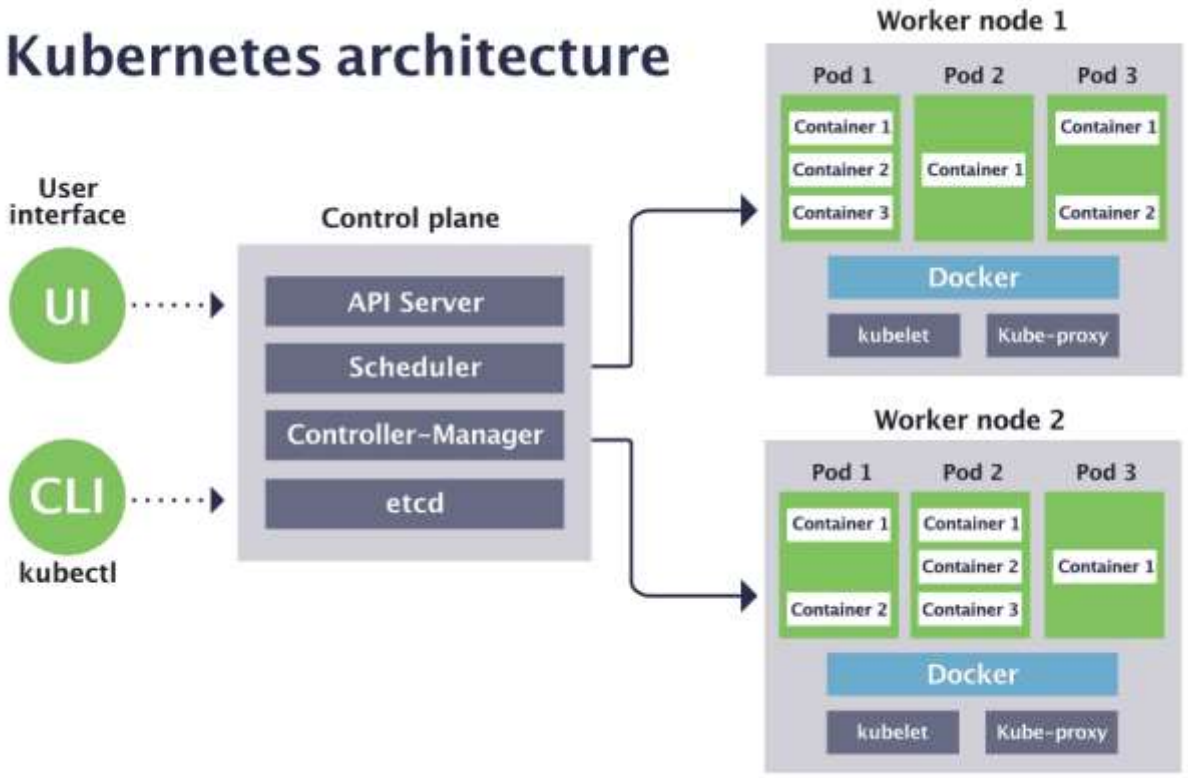
专用架构

性能发

高可用



传统 K8s 集群架构和 GPU 算力集群的不同

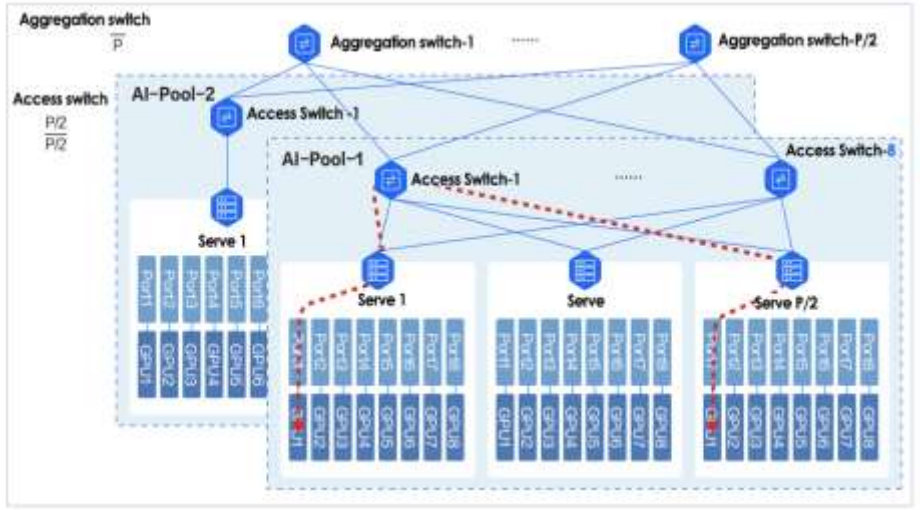


传统 Kubernetes 架构采用主从结构,控制平面负责全局决策和集群状态管理,而工作节点执行实际的容器运行任务,通过各组件的协同工作实现了容器化应用的自动化部署、扩展和管理。

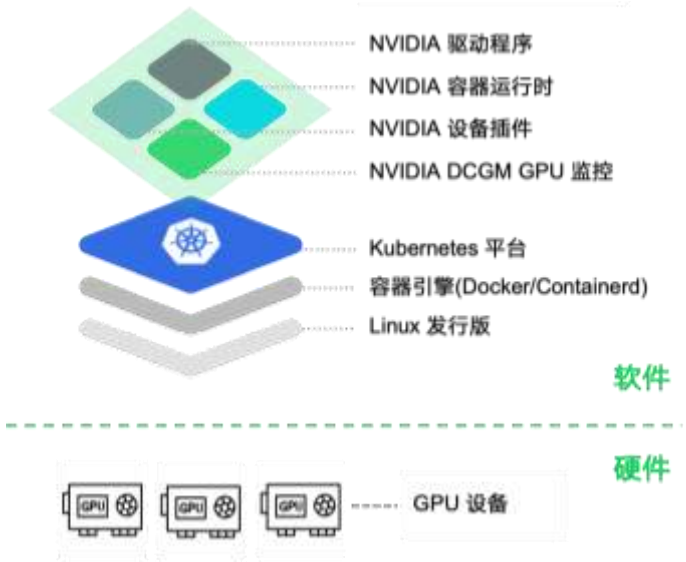
1. 控制平面(Master Node):
   - API Server: 集群的统一入口,处理内部和外部请求
   - etcd: 键值数据库,存储集群所有配置信息
   - Scheduler: 负责Pod的调度
   - Controller Manager: 负责维护集群状态
2. 工作节点(Worker Nodes):
   - Kubelet: 与容器运行时交互,管理Pod和容器的生命周期
   - Container Runtime: 如Docker,负责运行容器
   - Kube-proxy: 负责节点网络代理和负载均衡
3. 其他组件:
   - kubectl: 命令行工具,用于与API Server交互
   - 网络插件: 实现Pod网络通信

GPU 算力集群的特殊性

相较于传统的 Kubernetes 架构，组建一个大规模的 **GPU 算力集群**，需要从底层设计考虑整个集群的架构设计。



算力网络架构



驱动和插件依赖

特殊调度策略

Part 02

GPU 算力集群的挑战与解决思路

GPU 驱动和 CUDA 版本管理问题

现状

- 显卡驱动、CUDA 版本和操作系统，三者相互依赖，安装部署十分麻烦
- GPU 节点众多，人工维护成本很高
- 显卡驱动版本缺少统一管理，驱动差异会带来较多不可预置的问题，增加排查和修复成本
- 算法框架对 CUDA 版本的要求和适配性不一致，经常要求升级/降级 CUDA 版本
- GPU硬件和CUDA版本快速迭代,需要持续跟进和适配
- GPU节点出现问题难以自动发现

解决思路

利用 Kubernetes 管理节点

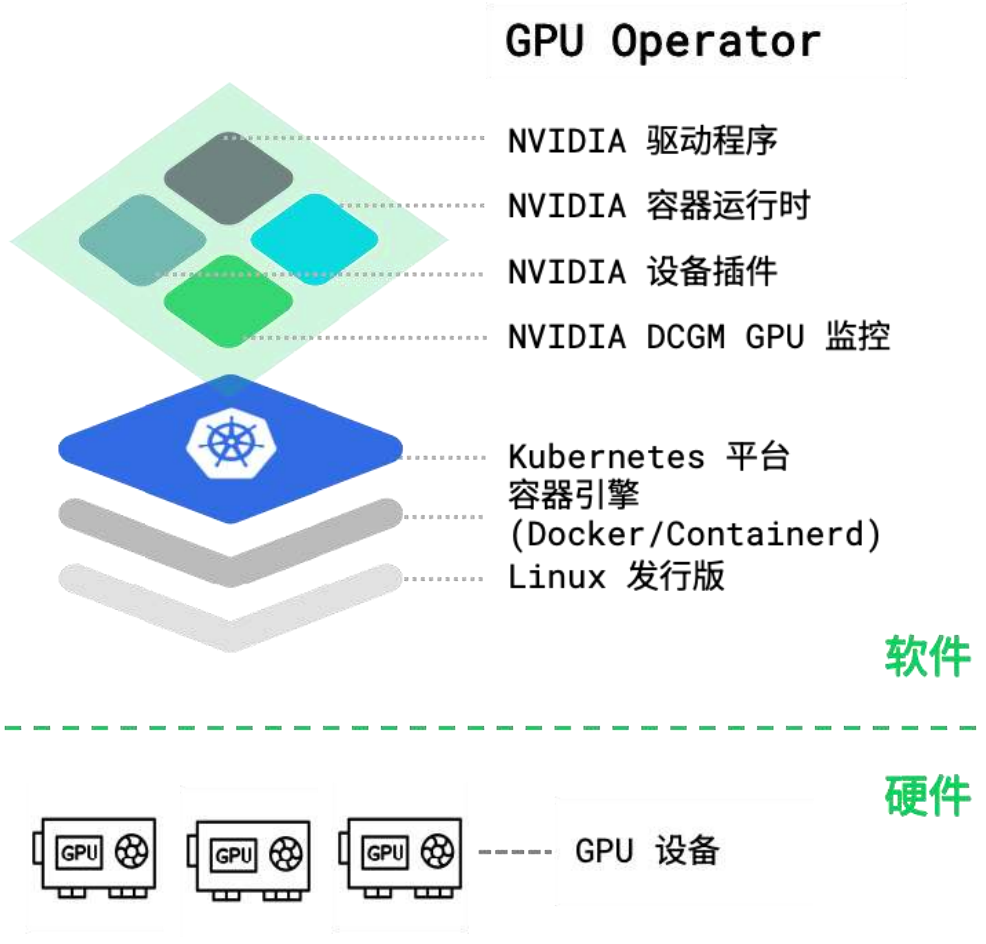
驱动自动安装

节点及卡信息可观测模块自采集

使用 GPU Operator 解决节点 GPU 纳管问题

GPU Operator - 自动管理并提供 GPU 所需的所有 NVIDIA 软件组件

- **NVIDIA 设备插件** - 通过设备插件机制将 GPU 公开给 Kubelet
- **NVIDIA 容器工具包** ：实现容器化环境中与 GPU 进行交互
- **GPU 驱动程序** ：Nvidia 驱动程序组件允许从容器进行驱动安装
- **NVIDIA GPU 功能发现** ：检测并标记启用 GPU 的节点
- **NVIDIA DCGM GPU 监控** ： 采集 GPU 指标



算力组网如何解决网络性能瓶颈

现状

- 网络性能直接影响分布式训练任务效率，在传统的 Kubernetes 主要以 以太网架构为主
- 大模型训练需要频繁交换大量参数和梯度信息
- 通常需要100Gbps甚至400Gbps以上的网络带宽
- 一次迭代的 Allreduce 通信量可达10GB级别
- 大模型 Checkponit 会瞬间产生海量数据传输压力

解决思路

高带宽

低延迟

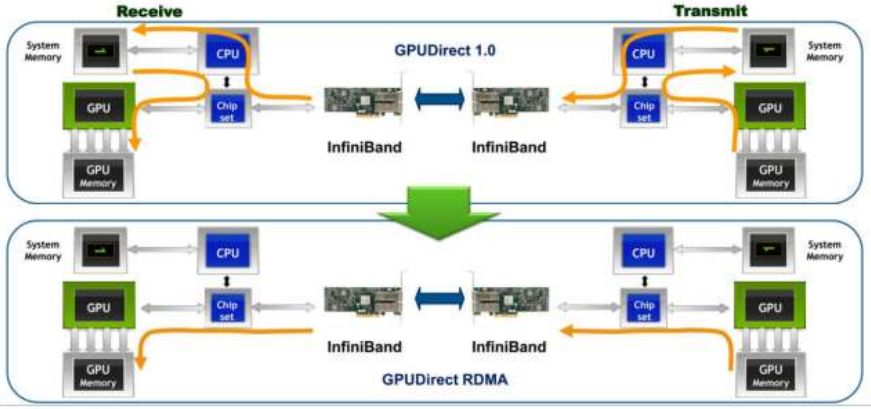
高吞吐量

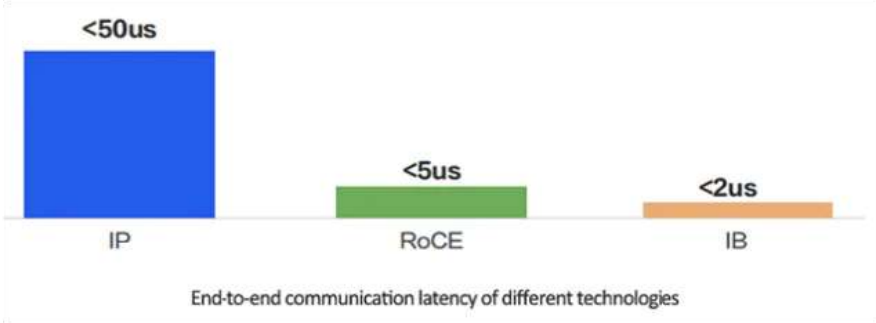
网络拓扑优化

- **专用硬件 IB 或 RoCEv2，提供高速网络**
- **专用的网络拓扑架构提供全互联架构**

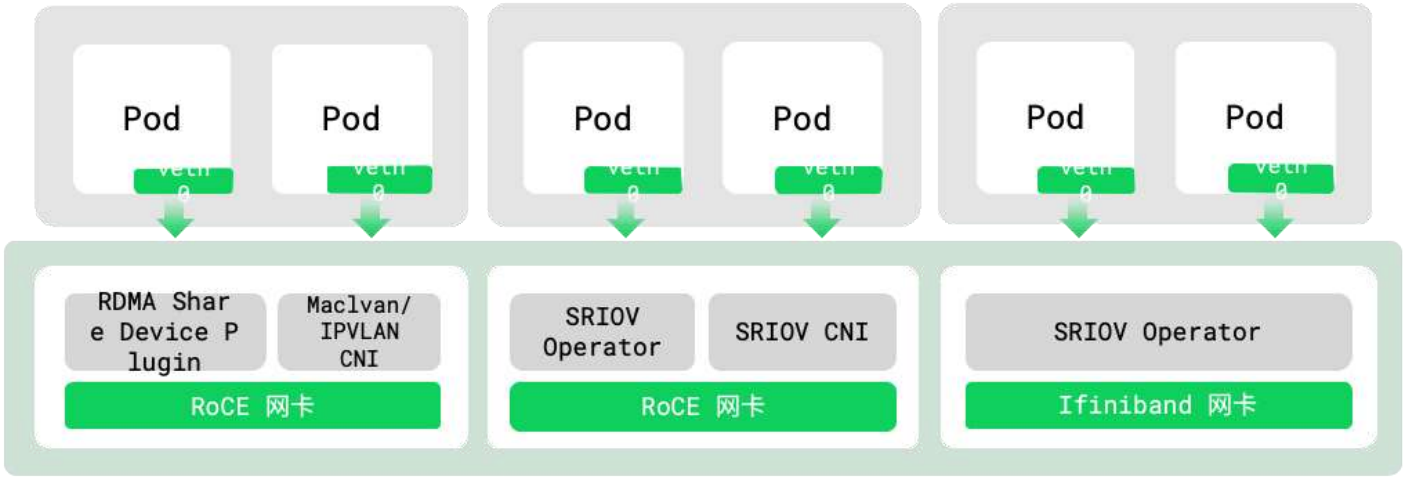
使用 GPU Direct RDMA 优化 GPU 的数据传输

GPUDirectRDMA 是一种特定于 GPU 的 RDMA 技术，允许 GPU 卡和 RDMA 网卡之间直接进行高性能的数据传输，无需通过 CPU，常应用于 HPC 和深度学习等场景，能够在数据传输过程中绕过 CPU，减少传输延迟，提高 GPU 的计算性能，尤其是在大规模数据集的训练和推理任务中。





1. 基于 Macvlan/IPVLAN 使用 RoCE/ Infiniband 网络直通
2. 基于 SR-IOV 使用 RoCE/ Infiniband 网络直通



基于 Macvlan/IPVLAN 使用 RoCE 直通

基于 SRIOV 使用 RoCE 直通

基于 SRIOV 使用 Ifiniband 直通

Sipederpool 提供 RDMA 解决方案

Spiderpool 是一个 Kubernetes 的 Underlay 和 RDMA 网络解决方案，它增强了 Macvlan CNI，ipvlan CNI，SR-IOV CNI 的功能，满足了各种网络需求，使得 Underlay 云原生网络方案可应用在 裸金属、虚拟机和公有云环境 中，可为网络 I/O 密集性、低延时应用带来优秀的网络性能，包括 存储、中间件、AI 等应用。 Spiderpool 是一个 CNCF Sandbox 项目。

GPU资源碎片化，如何有效提高利用率

现状

- 不同任务对GPU资源需求不一致
- 任务调度不合理，导致资源分配不均
- GPU硬件配置与任务需求不匹配
- 缺乏有效的资源管理和监控机制
- GPU利用率低下，造成资源浪费
- 高优先级任务可能因资源不足而等待
- 增加运维成本和复杂度
- 影响整体计算效率和性能

解决思路

任务资源调度优化

虚拟化技术

实时资源回收机制

使用任务调度策略，提高 GPU 利用率

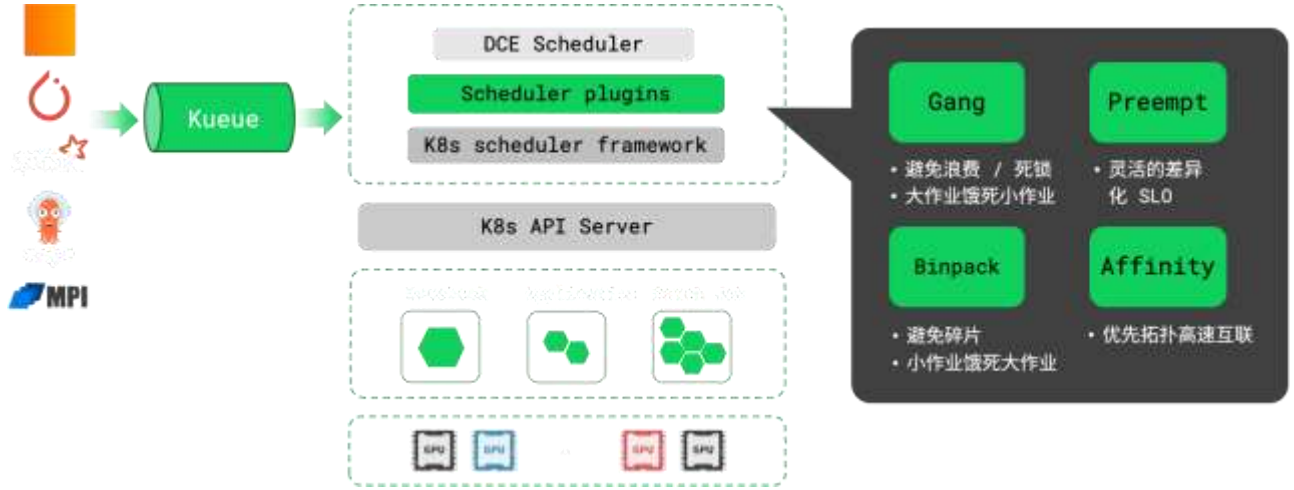
为 AI 训练任务以及不同场景下任务调度，提供丰富的调度策略，
利用专属 AI 训练开发场景下的专属调度策略，提高资源调度效率。

基于卡利用率的调度策略：

- 公平调度
- 紧凑调度

基于任务维度的调度策略：

- 队列调度
- 优先级调度
- Gang-Scheduler
- GPU 资源共享、抢占

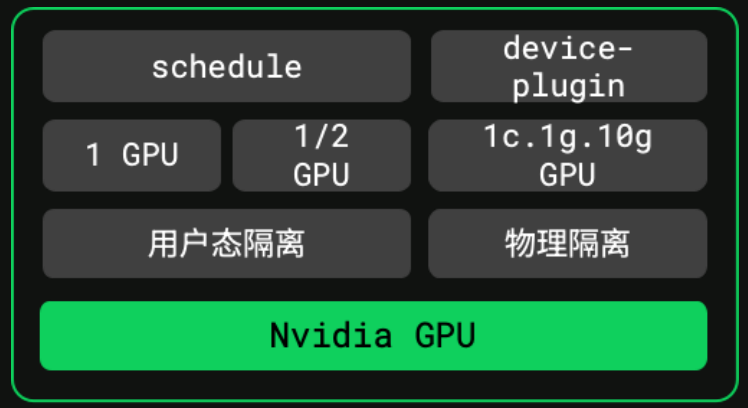


想要了解更多的调度策略请关注后续课程

使用虚拟化技术拆分卡，提高 GPU 利用率

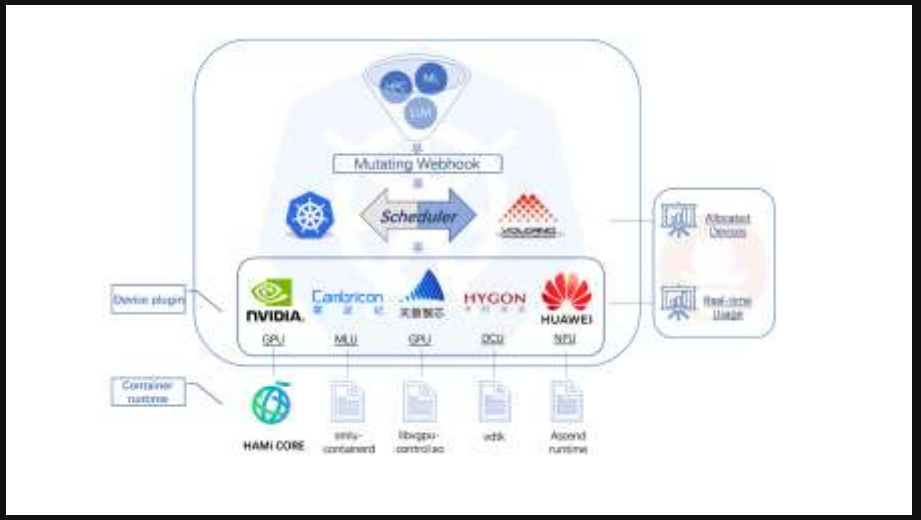
通过将 GPU 资源根据用量大小动态申请，将一张卡拆分多个用途，
可极大提高卡的利用率。

硬隔离： MIG （Nvidia)
软隔离： vGPU （通过软件技术，将GPU卡拆分到非常小颗粒），
同时可以接合一个超分技术



了解更多 vGPU 虚拟化技术，可以关注 HAMi, 以及后续课程。

（由 DaoCloud 和 4PD 联合开源的 GPU虚拟化技术）



https://github.com/Project-HAMi/HAMi

使用动态资源回收，提高 GPU 利用率

当**训练任务结束后，资源自动回收**；任务重启资源自动重新调度。

闲置资源自动回收，如 Notebook 可配置超时自动关机，及时回收算力资源。

国产 GPU 适配问题

现状

国产GPU普遍缺乏完善的软件开发工具链和框架支持。

与CUDA生态相比,国产GPU的编程环境和开发工具还不够成熟,导致开发者在使用时面临较高的学习成本和兼容性问题。

这不仅影响了开发效率,也限制了国产GPU在大模型训练等高性能计算场景中的应用。

解决思路

N 卡依旧是业界主流和最先进的硬件和生态

国产 GPU现在发展非常快，但短期还无法超越 ~

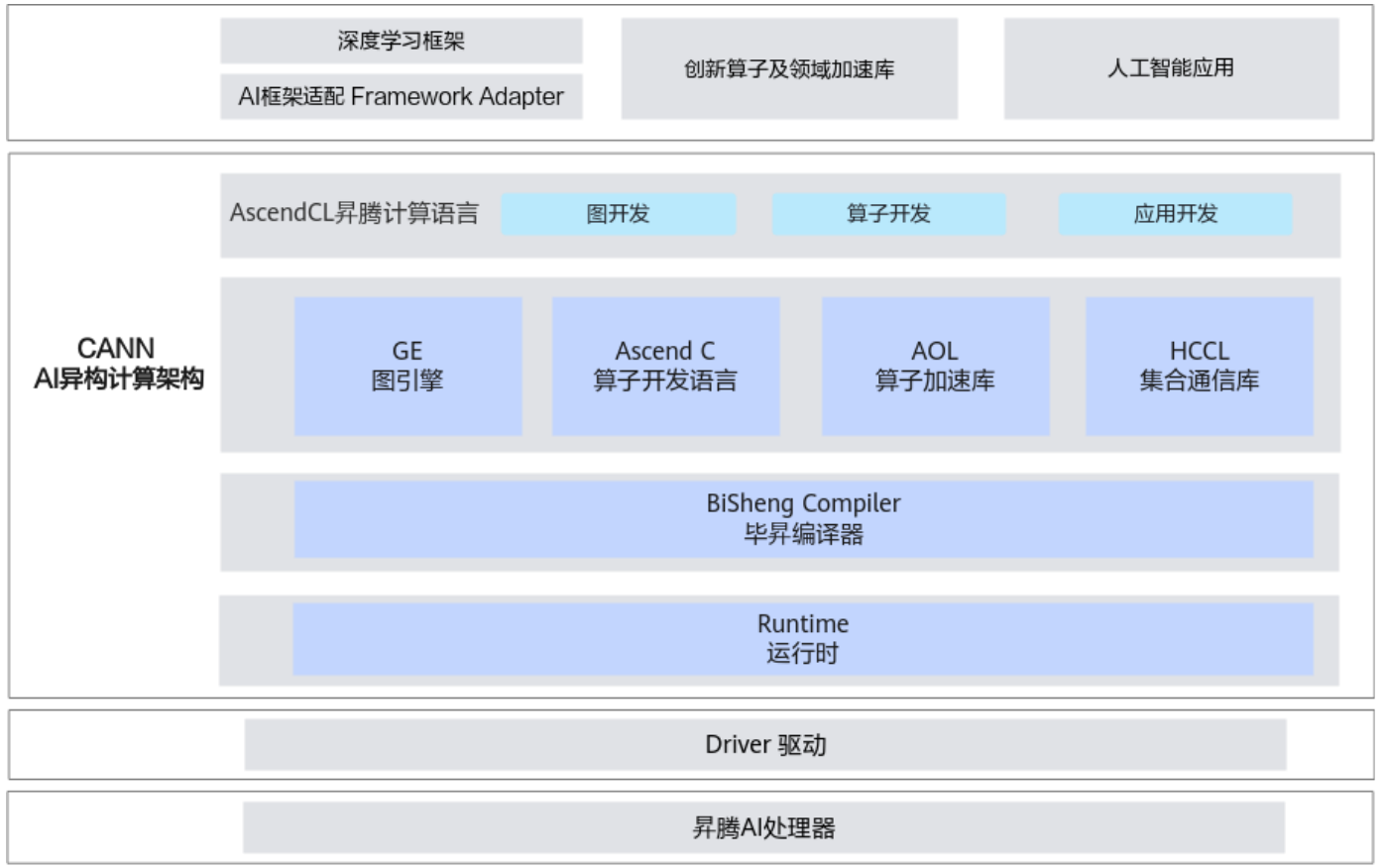
没有什么魔法，**“莫慢待”**，等待胜利的那一天。

国产 GPU 适配问题 - Ascend

异构计算架构CANN（Compute Architecture for Neural Networks）是华为针对AI场景推出的异构计算架构。

- 向上支持多种AI框架，包括MindSpore、PyTorch、TensorFlow等
- 向下服务AI处理器与编程，发挥承上启下的关键作用，是提升昇腾AI处理器计算效率的关键平台。

同时针对多样化应用场景，提供多层次编程接口，支持用户快速构建基于昇腾平台的AI应用和业务。



d.run 核心能力-高效纳管异构 GPU，实现算力资源灵活调度

在 d.run 体系上构建 GPU 资源池，让租户内的 AI 用户可共享数据中心内所有服务器上的 GPU 算力，通过高效的 GPU 调度算法使得 AI 应用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 AI 应用开发变得更加敏捷高效。

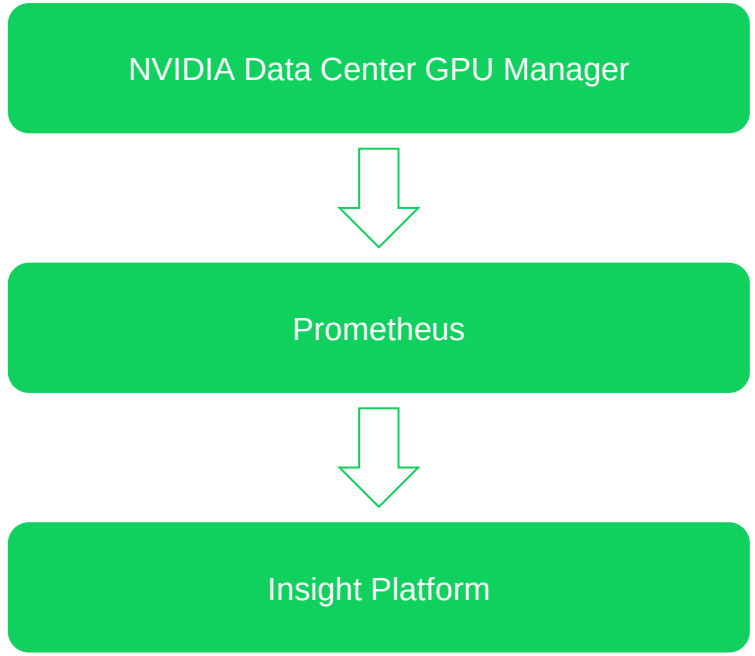
监控和故障排查复杂性（断卡重训）

现状

大规模集群中GPU分布在多个节点，增加了监控和故障定位的难度。

- 数据量巨大：高频采集的GPU指标产生海量数据，存储和分析成本高。
- 实时性要求：需要近实时的监控以快速响应异常。
- 指标多样性：需要监控计算利用率、内存使用、温度、功耗等多维度指标。
- 人力成本高：需要专门的GPU运维团队，培养周期长。
- 工具碎片化：市场上存在多种监控和故障排查工具，集成和使用复杂。
- 自动化程度不足：许多故障排查流程仍依赖手动操作。

解决思路



云原生可观测性自动化故障监控和排查

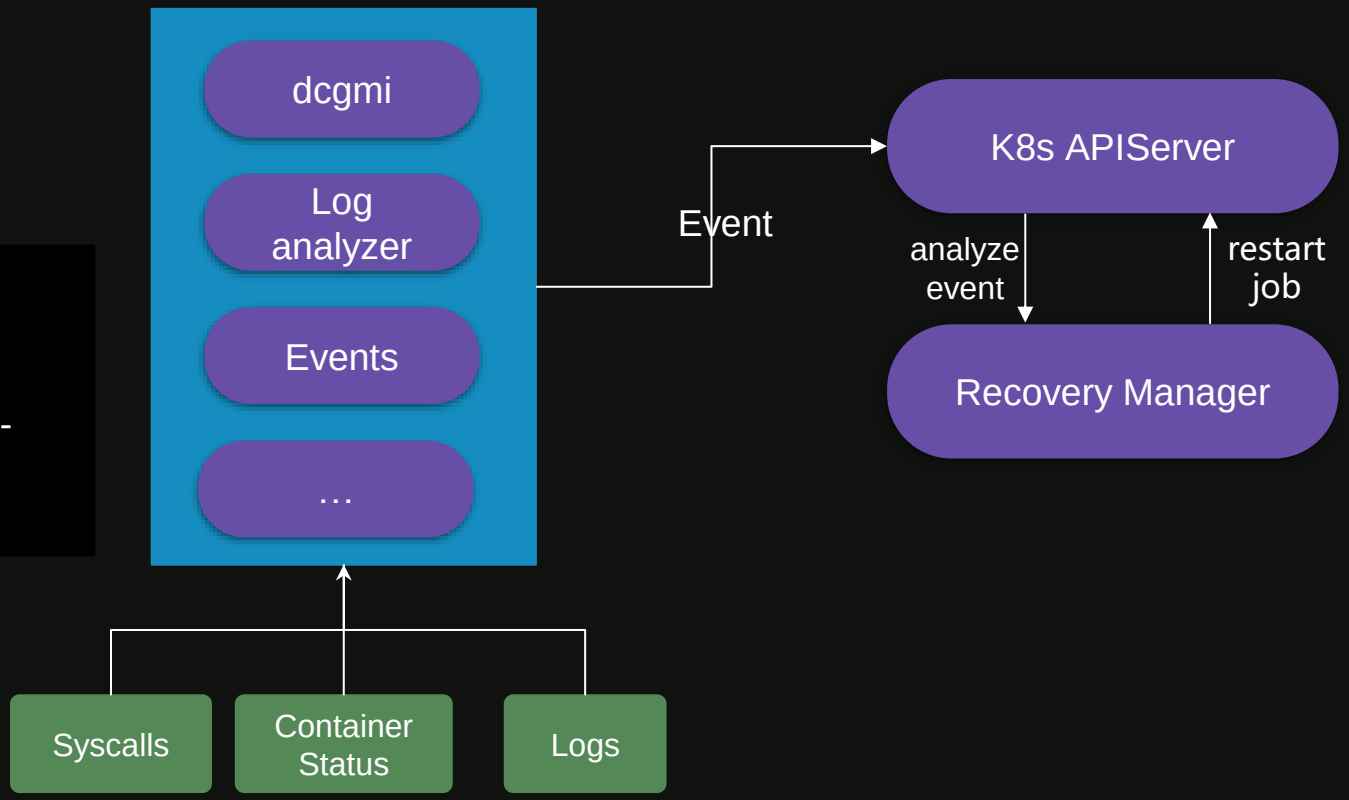
- **实时洞察**，内建丰富的仪表盘，直观了解 GPU 负载、内存和资源利用率，找到 GPU 使用中的瓶颈。并将模型服务指标与 GPU 指标相关联。
- 内建 **100+ 精选告警规则** 来主动识别性能和资源等问题，多种途径自动通知来低效负载，有效提高 GPU 利用率。
- 内置 **告警知识库**，知悉告警原因及后续处理方案，践行云原生运维最佳实践。
- 支持多租户自定义 **模型监控**，实现 指标采集，存储，分析，呈现全过程。

断卡重训 - DEMO

通过对采集到 Node、Pod 以及训练任务等的指标信息。

KCover 提供了一套全自动的断卡异常检测，帮助训练任务自动恢复的能力。

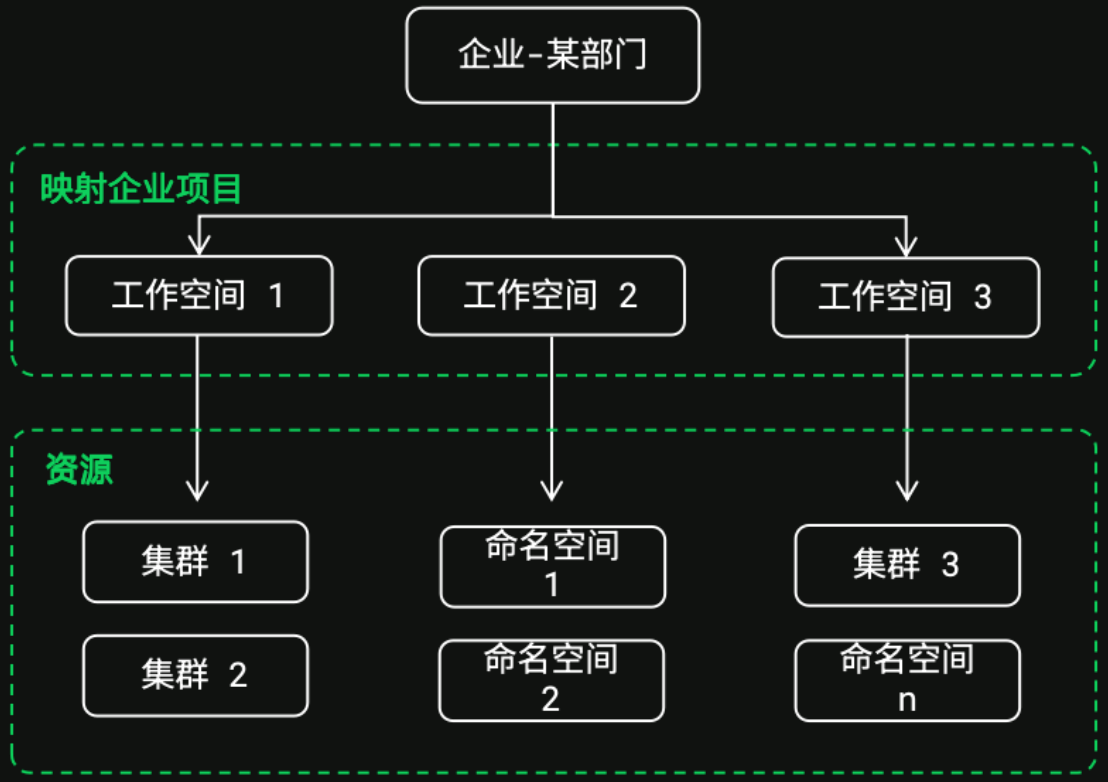
```
λ helm repo add baizeai https://baizeai.github.io/charts
λ helm repo update baizeai

λ helm install kcover baizeai/kcover --namespace kcover-system --create-namespace
```



https://github.com/baizeai/kcover
https://baizeai.github.io/talks/2024-08-21-kubecon-hk

使用租户管理实现算力资源隔离

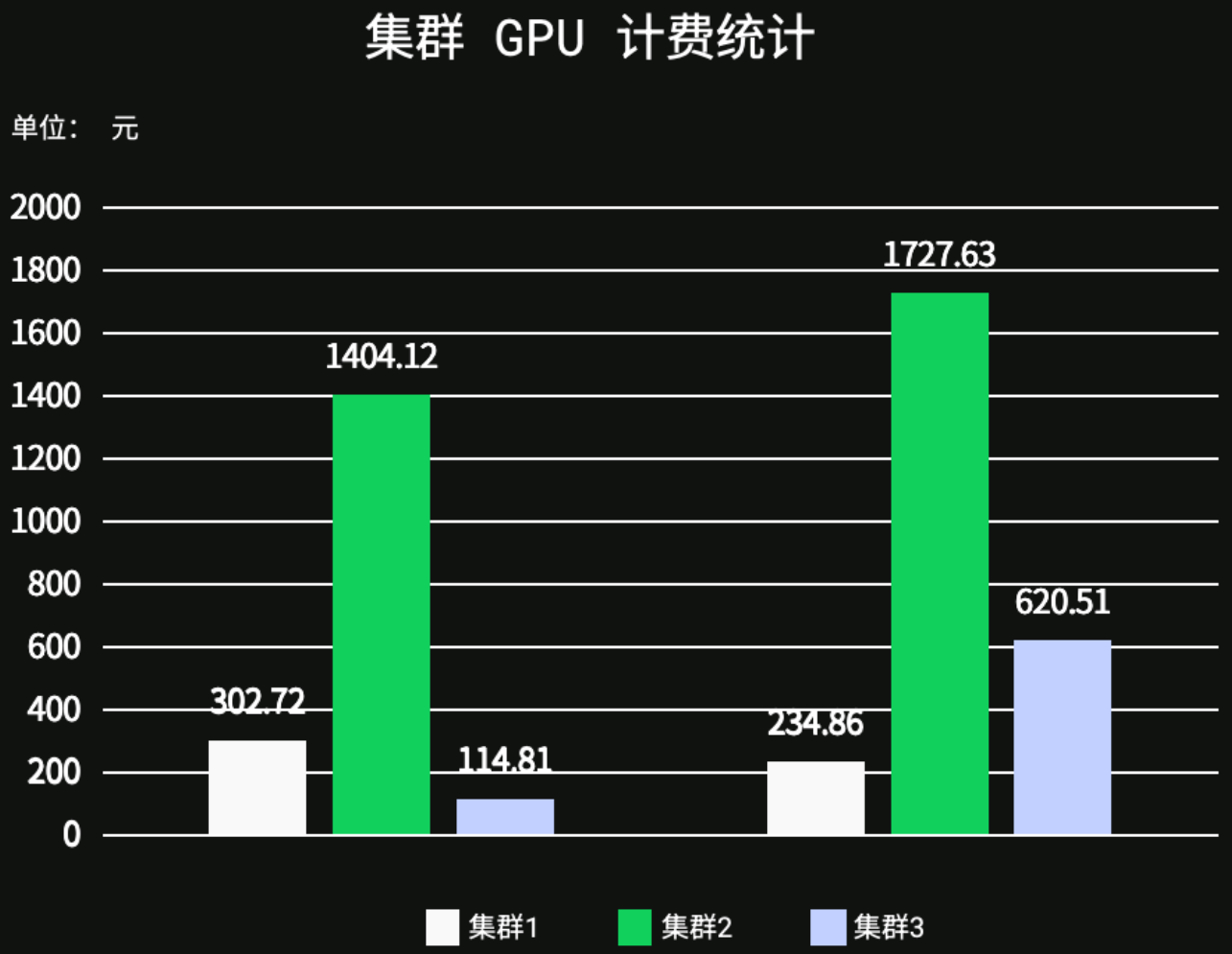
跨集群、跨命名空间资源隔离



细粒度的租户配额管理

工作空间	集群 GPU（卡），CPU（核）、内存（Gi）、存储（Gi）
命名空间	命名空间 GPU（卡）、 CPU（核）、内存（Gi）、存储（Gi）、PVC 数（个）…
容器（Pod）	GPU（卡），CPU（核）、内存（Gi）

运营中心对算力资源进行计量计费



支持对集群、节点、容器组、工作空间和命名空间 5 种资源类型分别进行 GPU 资源计量和计费，支持统计使用量、使用率、资源花费等内容。

支持自定义计费配置，根据不同类型自定义 GPU 单价与货币单位。可按照所选时间自动计算出集群、节点、容器组等在一段时间内的总计费用和 CPU、内存、存储、GPU 的各自使用费用。

支持通过 Excel、CSV 两种方式导出计费结果。支持计费报表中关联数据的快捷跳转，如查看同一时间段中集群下的节点计费等。


DaoCloud
产品
d.run
解决方案
客户案例
咨询与服务
合作伙伴
开发者资源
关于我们
活动中心
中文
d.run | 让算力更自由
立即使用
查看文档

打开 d.run，了解更多

Part 03

案例 & d.run demo

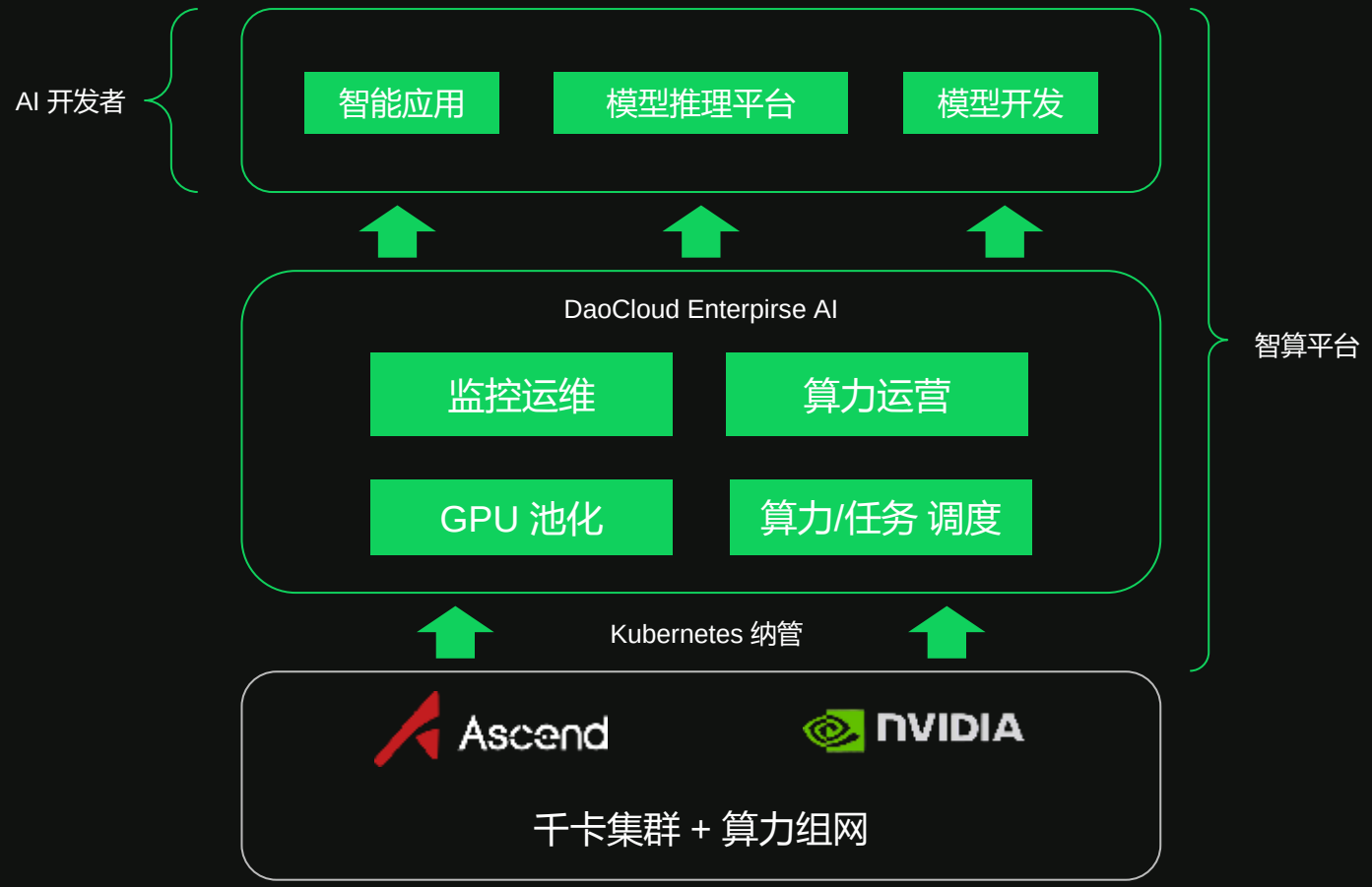
案例分析

客户资源：多数据中心，千卡规模，包含了 Ascend 和 Nvidia 多中算力资源

客户目标：基于现有算力资源，从 0 开始打造统一算力平台；对外提供算力租赁业务，主要服务大型 B 端客户。

解决方案：

- 基于云原生算力底座将客户所有算力资源统一纳管，提供丰富的算力运维、运营一体化平台
- 基于 d.run 丰富的算力应用生态，为 B 端企业提供便捷的大模型应用和开发套件



Thanks.

Q&A