DaoCloud

提升 GPU 利用率的 N 种方法

DaoCloud　刘爽

Content

目录

Part 01

什么是 GPU 利用率

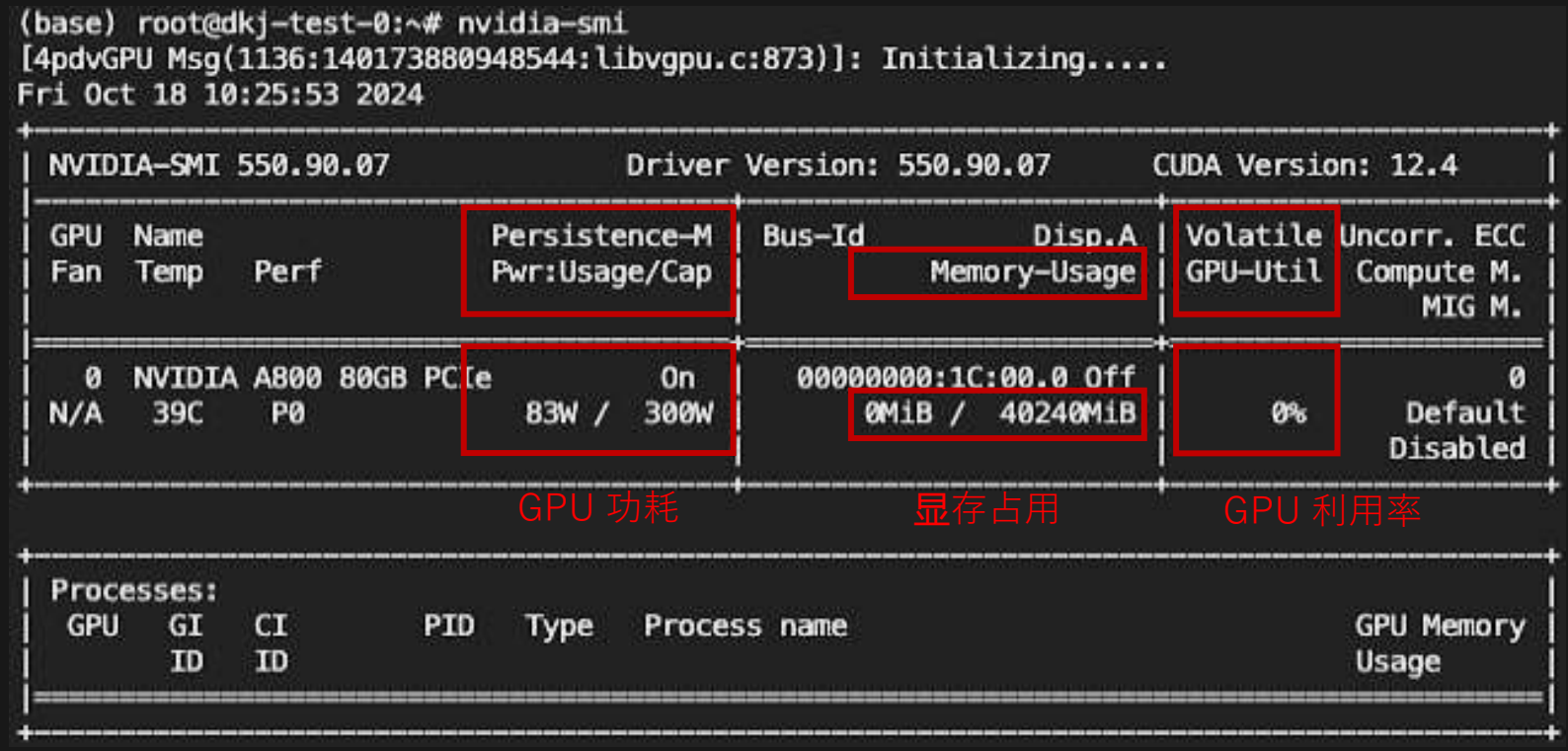
什么是 GPU 利用率

GPU 利用率是反馈 GPU 上各种资源繁忙程度的指标。

GPU 上的资源包括：

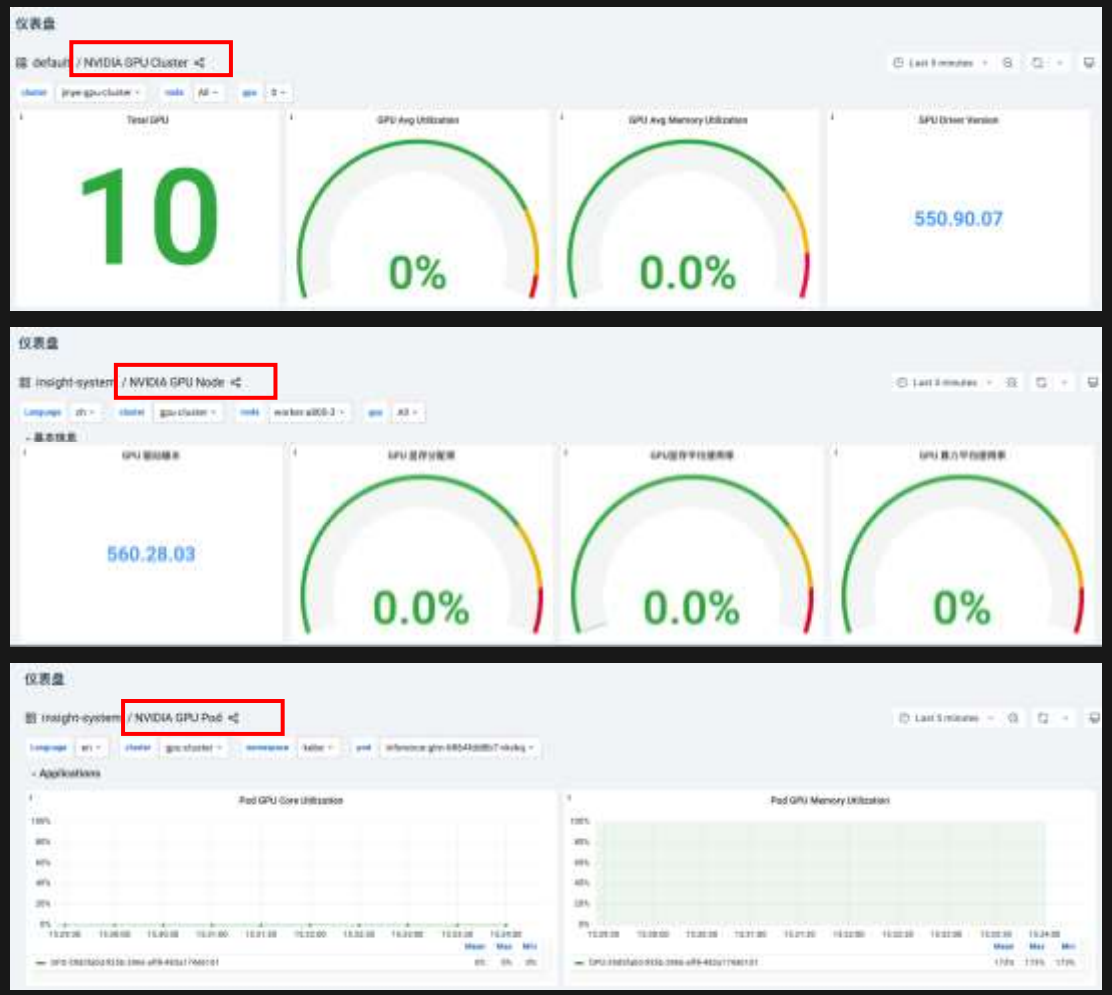
- GPU core：CUDA core, Tensor Core， integer, FP32 core， INT32 core 等。
- frame buffer：capacity, bandwidth。
- 其他：PCIe RX / TX, NVLink RX / TX， encoder 和 decoder 等。

通常，我们说 GPU 利用率泛指 GPU core 的利用率。

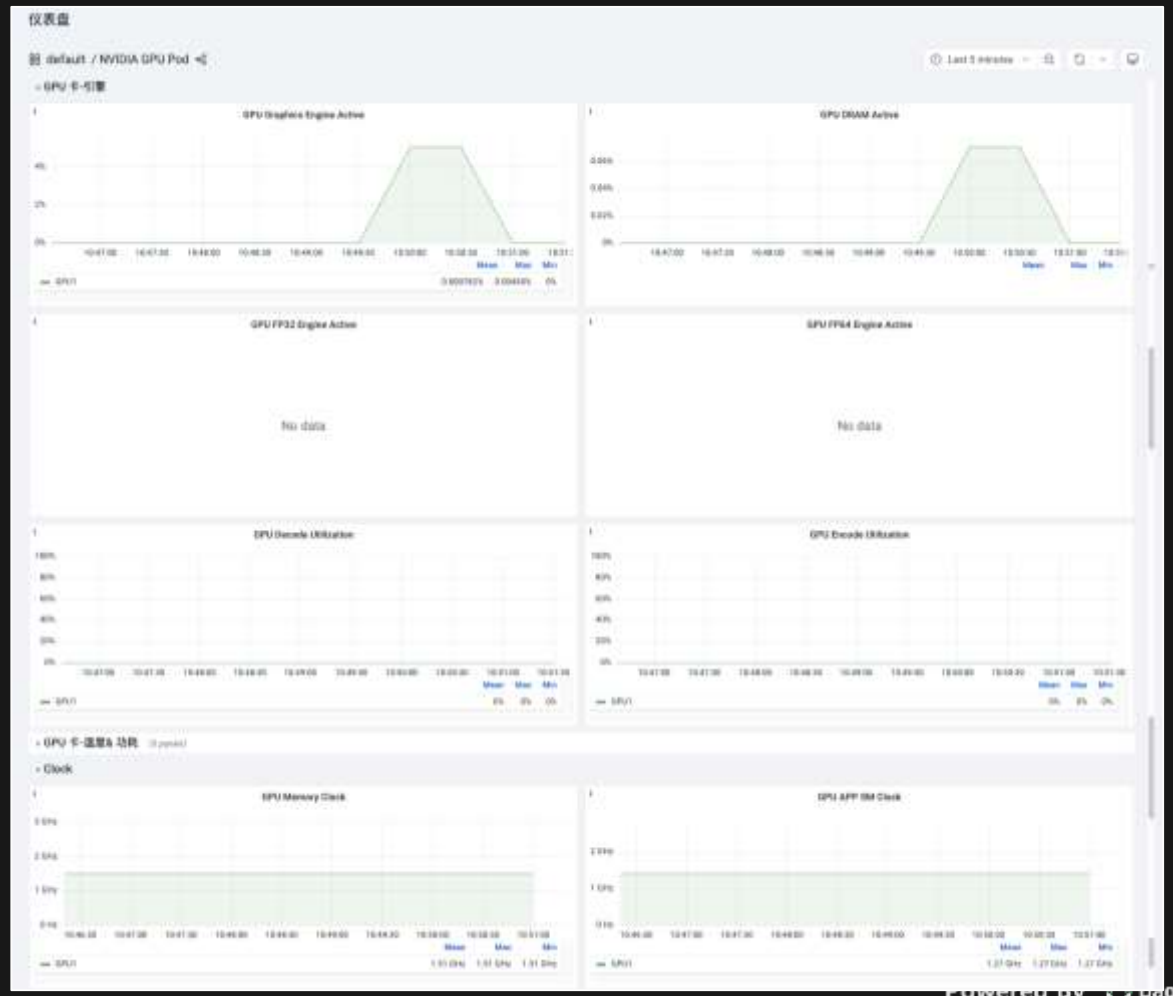


监控 GPU 利用率的方式

集群、节点、容器组



图形引擎活动、PCIe 传输速率/接受率、内存带宽利用率



GPU 利用率低的主要原因

资源分散，难以统一管理

无资源池，无整体调度，监控，管理平面。

资源分配不灵活

无法实现资源超分，不能指定任务所需的资源大小和种类。

资源利用率低

无虚拟化&资源池化技术，GPU使用为独占模式。

需求多样，造成任务堵塞

缺乏优先级排序，当多任务并行时，容易造成任务堵塞，或低优先级任务霸占资源，影响高优先级任务运行。

Part 02

提升 GPU 利用率的方法

GPU 池化

模型搭建

模型应用

智能问答

模型推理

智能算力

模型训练

Colossal-AI

AI 模型

资源调度层

异构算力	资源优化	运维及运营
快速部署	优先级抢占	多维可观测
异构卡纳管	Binpack/Spread 调度	计量计费
多GPU模式选择	算力/显存超分	资源配额

GPU 资源池

资源隔离层

schedule	device-plugin	
1 GPU	1/2 GPU	1c.1g.10g GPU
用户态隔离	物理隔离	
Nvidia GPU		

gpu-manager	gpu-schedule	
天数智芯	瀚博 FPGA	摩尔线程
国产 GPU		

device-plugin		
1 GPU	1 GPU	1 GPU
物理隔离		
华为昇腾 GPU		

提升 GPU 利用率的方式总览

异构算力

快速部署

- GPU operator
- NVIDIA-vGPU
- ascend-mindxdl

异构 GPU卡纳管

- 异构卡纳管
- 异构资源自动发现
- 异构操作系统部署

多模式选择

- 整卡模式
- 虚拟化模式
- MIG 模式

资源调度

调度策略

- 指定 GPU 型号
- 基于节点调度
- 优先级抢占
- 基于GPU卡调度

资源优化

- 算力超分
- 资源监控
- 显存超分
- 资源切片

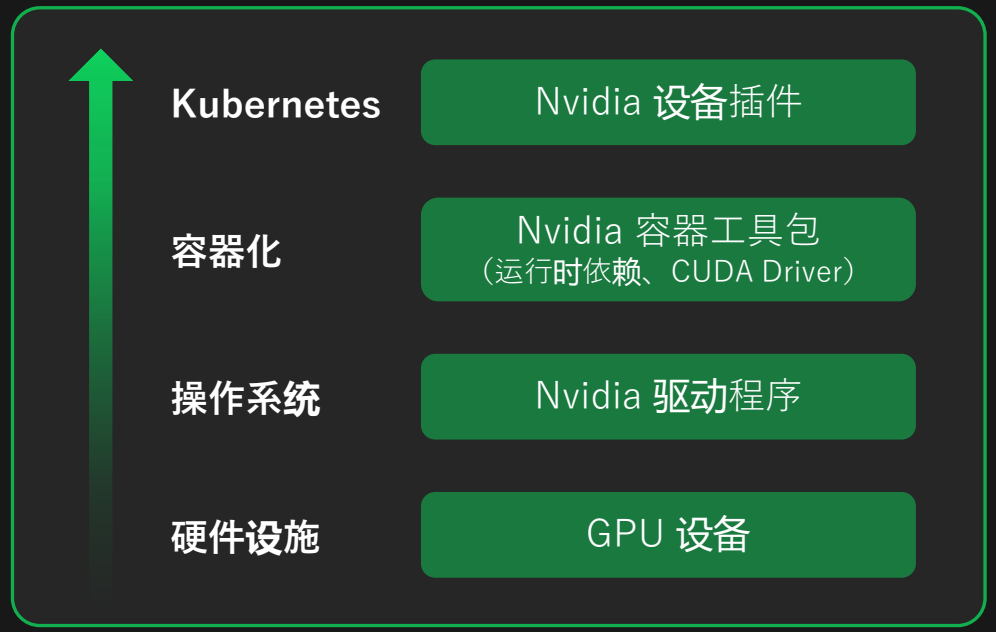
运维及运营

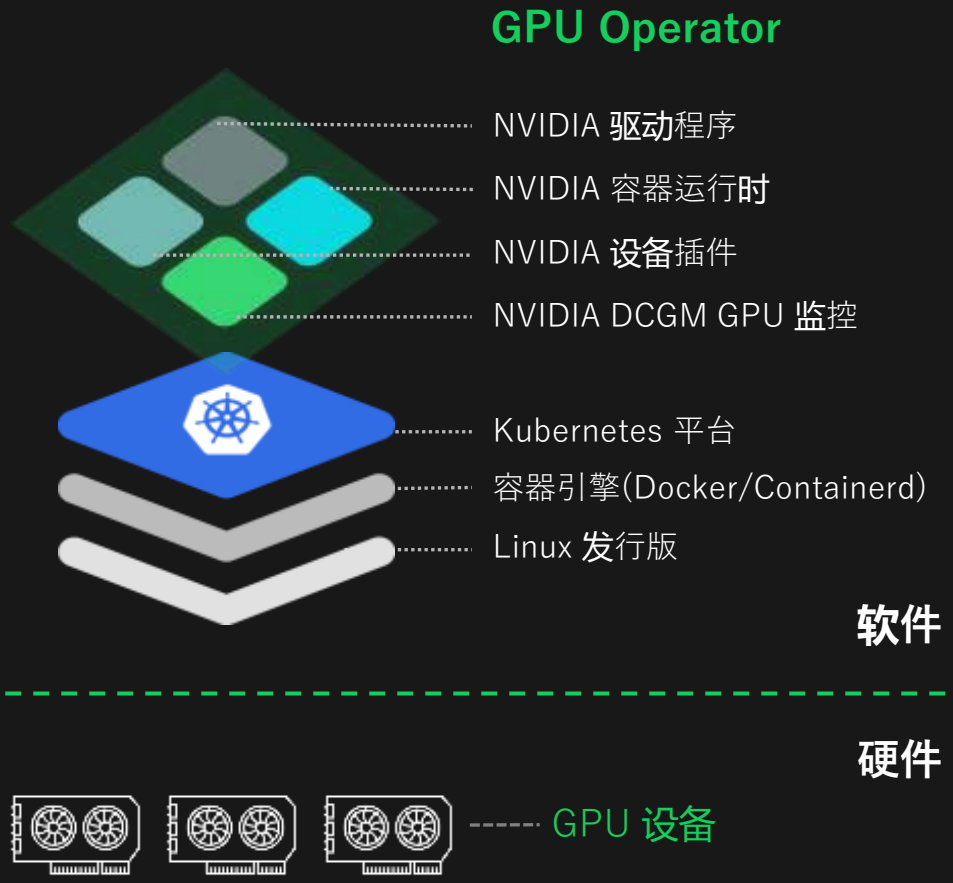
- 多维可观测
- 计量计费
- 资源配额

安装部署

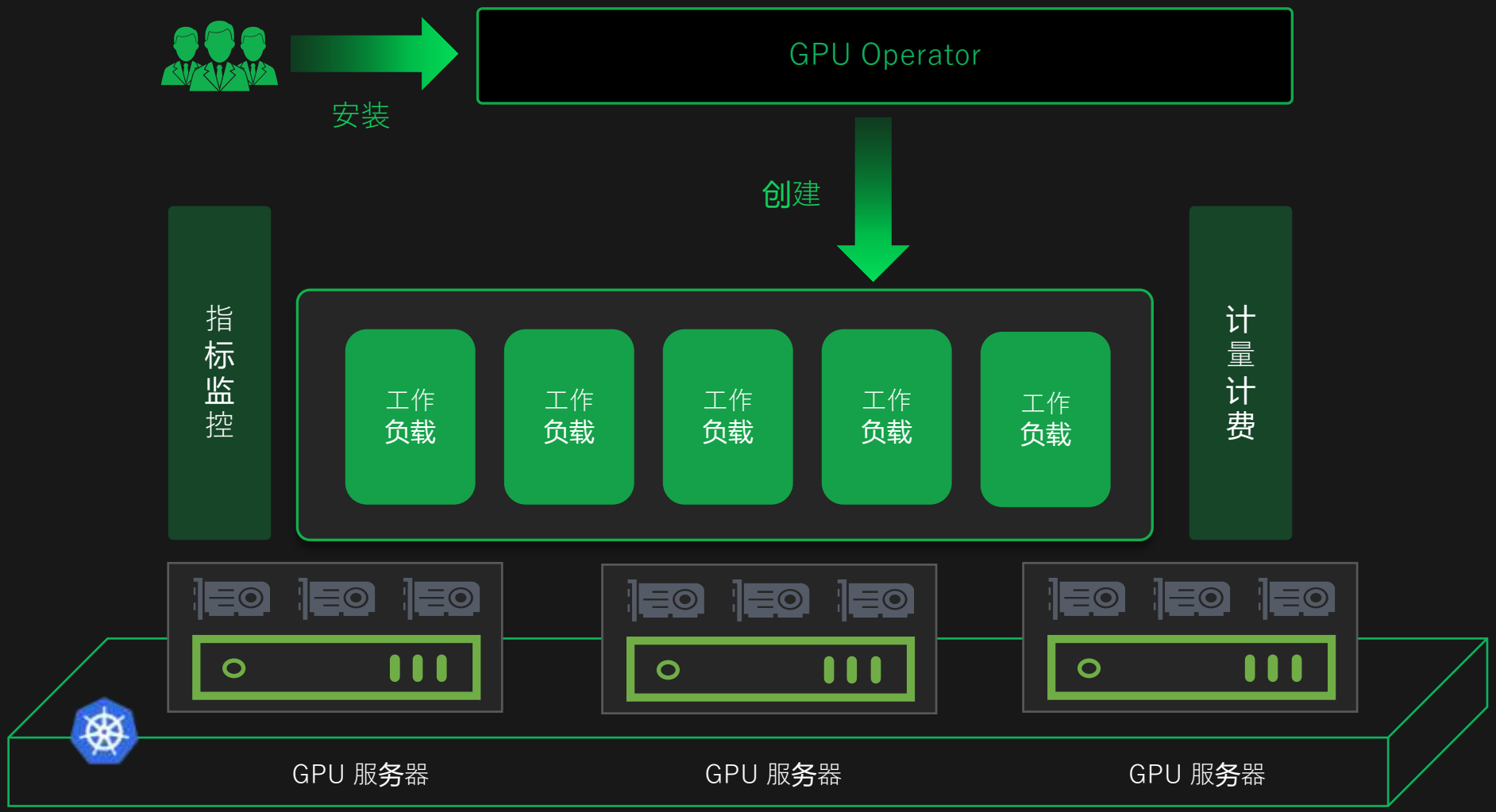
GPU Operator - 自动管理并提供 GPU 所需的所有 NVIDIA 软件组件

- **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





资源调度-整卡模式



GPU 资源调度-MIG 模式

Single 模式 | **Mixed 模式**

MIG 模式将一个物理 GPU 划分为多个 MIG 实例，每个实例可以独立分配给不同的工作负载使用。

资源强隔离

每个 MIG 实例具有自己的计算资源、显存和 PCIe 带宽，就像一个独立的虚拟 GPU。

自定义切分规格

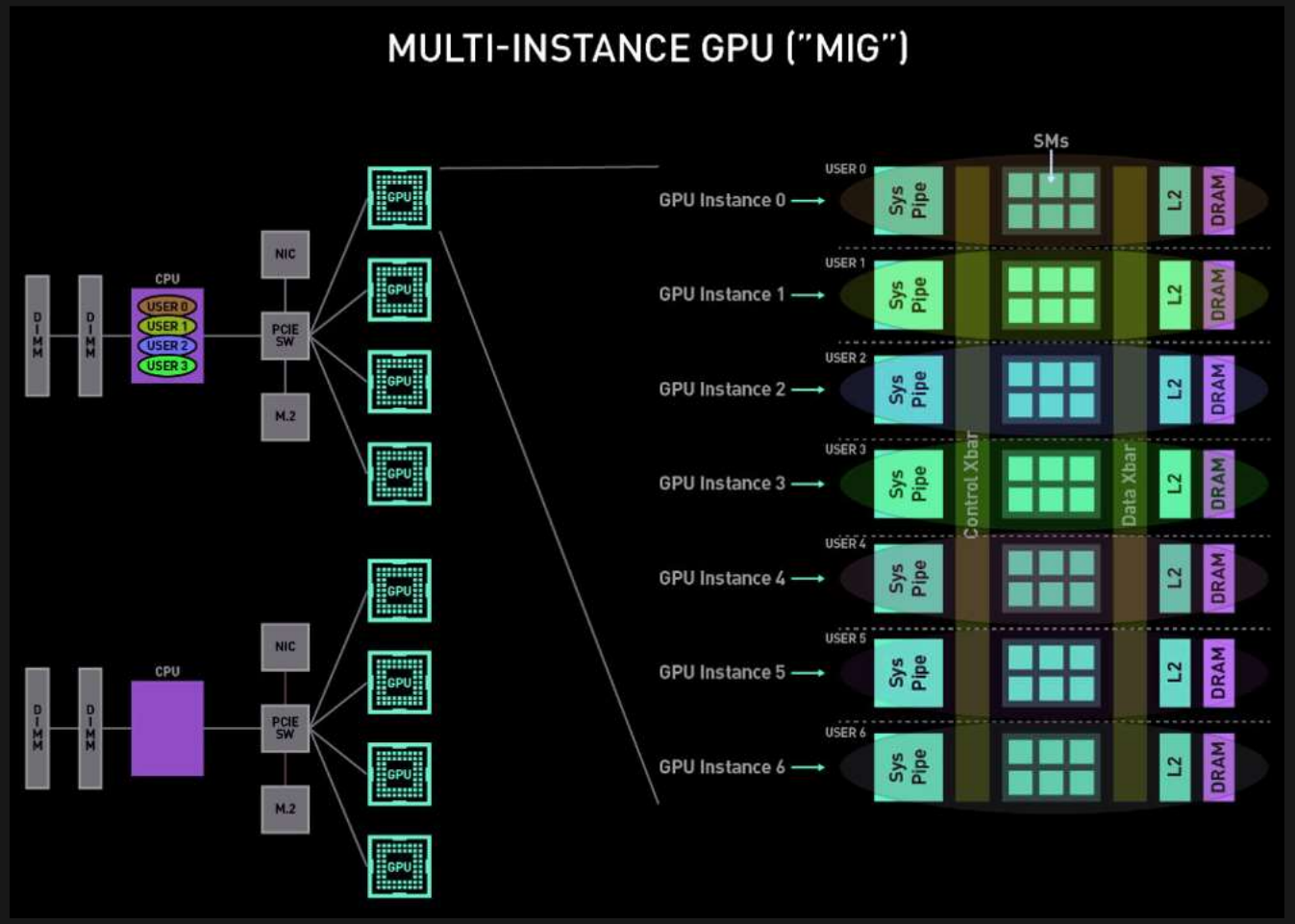
更细粒度的 GPU 资源分配和管理，可以根据需求动态调整实例的数量和大小。

资源利用率高

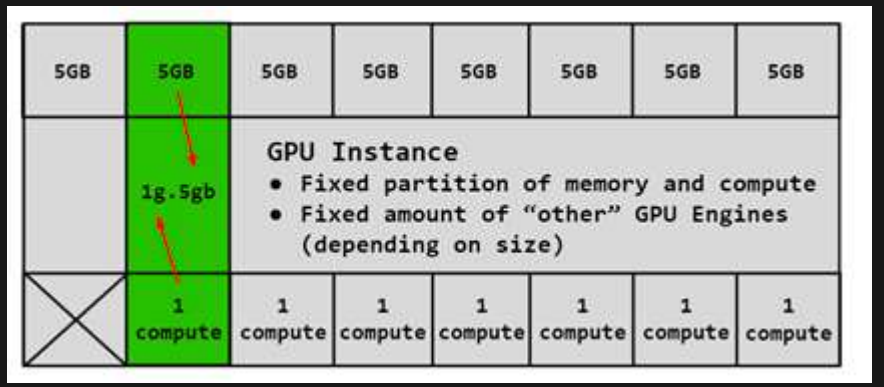
MIG 模式资源利用率高，可以同时运行多个任务，但划分实例的过程可能比较复杂。

MIG 模式的 GPU 切分

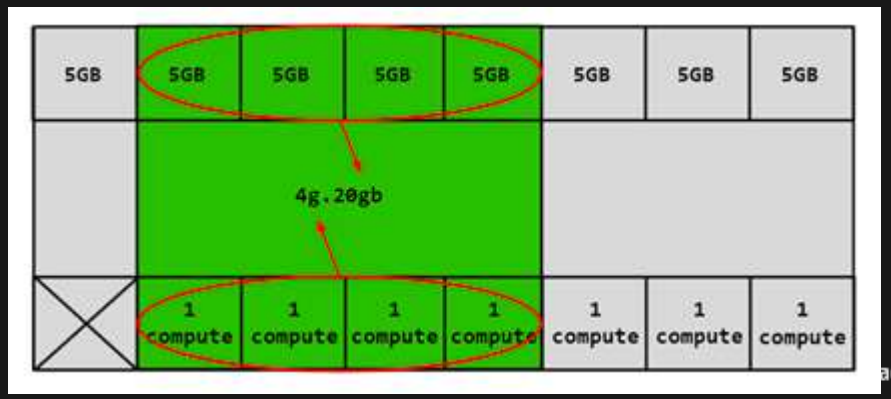
GPU 的分区是使用内存切片进行的，因此可以认为 A100-40GB GPU 具有 8x5GB 内存切片和 7 个 GPU SM 切片。同时，创建 GPU 实例需要将一定数量的内存切片与一定数量的计算切片相结合。



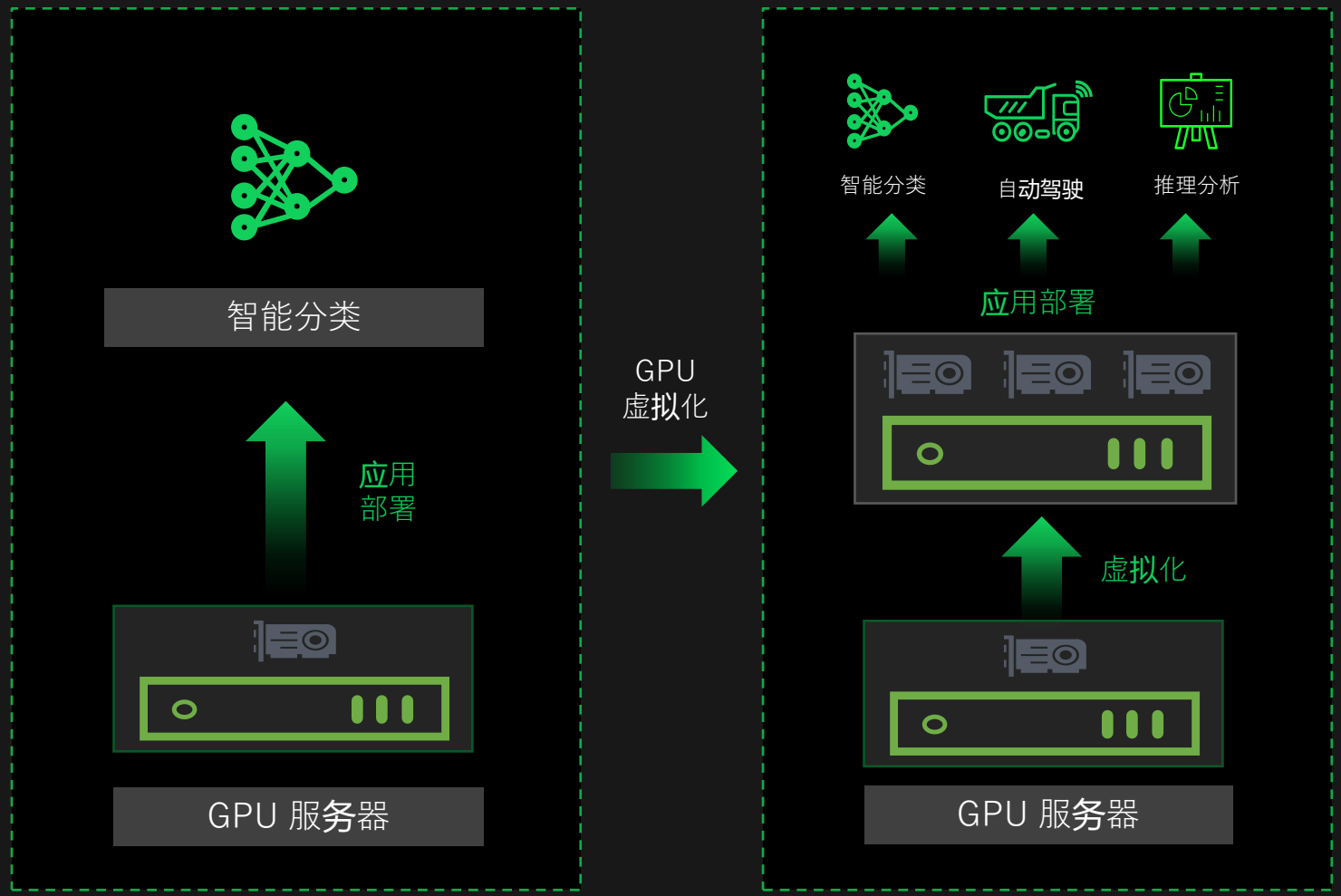
一个 5GB 内存切片与 1 个计算切片相结合，以创建 1g.5gb GI 配置文件



与 4x1 计算切片结合使用以创建 4g.20gb 的 GPU。



资源调度-vGPU 模式



虚拟出多个 GPU 卡

在虚拟化模式下，GPU的资源被划分为多个虚拟GPU（vGPU），每个vGPU可以独立分配给不同的应用。

自定义 GPU 虚拟卡数

支持自定义GPU卡的分割数，若分割数配置为 N 的话，每个 GPU 上最多可以同时存在 N 个任务。

显存/算力超分

自定义超分参数 S，若算力超分，则这张 GPU 卡分出的 vGPU 将总共包含 S * 100% 算力

GPU 资源调度

基于优先级的调度

当高优 Pod 开始使用 GPU 算力，所有普通 Pod会立刻被暂停使用 GPU 算力，直到高优 Pod 计算任务结束，普通任务会重新继续使用 GPU 算力。

基于GPU卡型号的调度

在 vGPU 模式下，当存在多种型号的 GPU 卡时，支持将 Pod 调度到指定型号的 GPU 卡上。

基于 GPU 卡调度

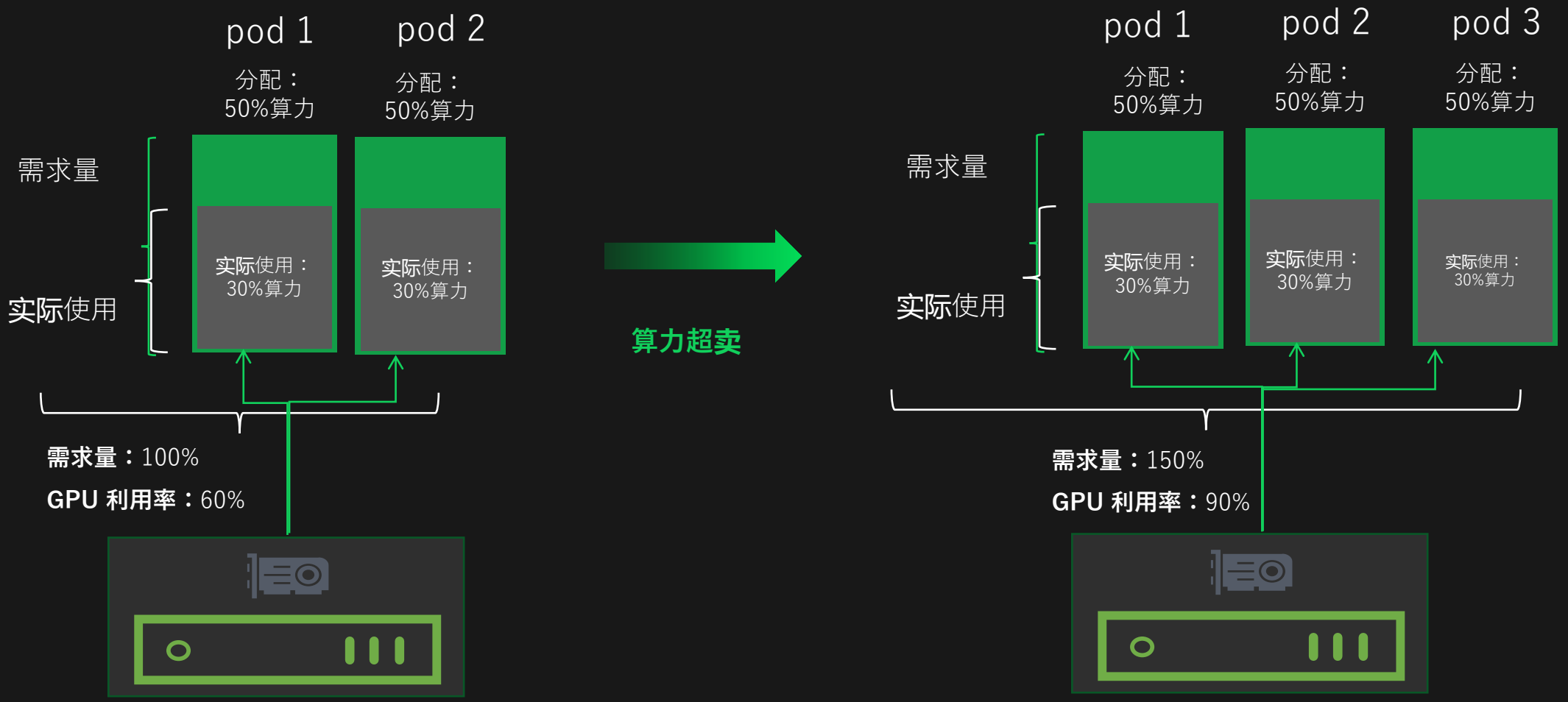
- 在选定节点后进行卡维度的分配决策，为 Pod 中每个容器选择和分配节点上的 GPU 卡：
- Binpack：优先选择资源利用率高的 GPU 卡，容器集中到同一块卡上
- Spread：优先选择资源利用率低的 GPU 卡，容器分散到不同的卡上

基于节点调度

根据节点 GPU 算力和显存进行加权平均打分提供 2 种策略：

- Binpack：GPU 分配率越高，打分越高，Pod 集中调度到同一个节点
- Spread：GPU 分配率越高，打分越低，Pod 分散调度到各个节点

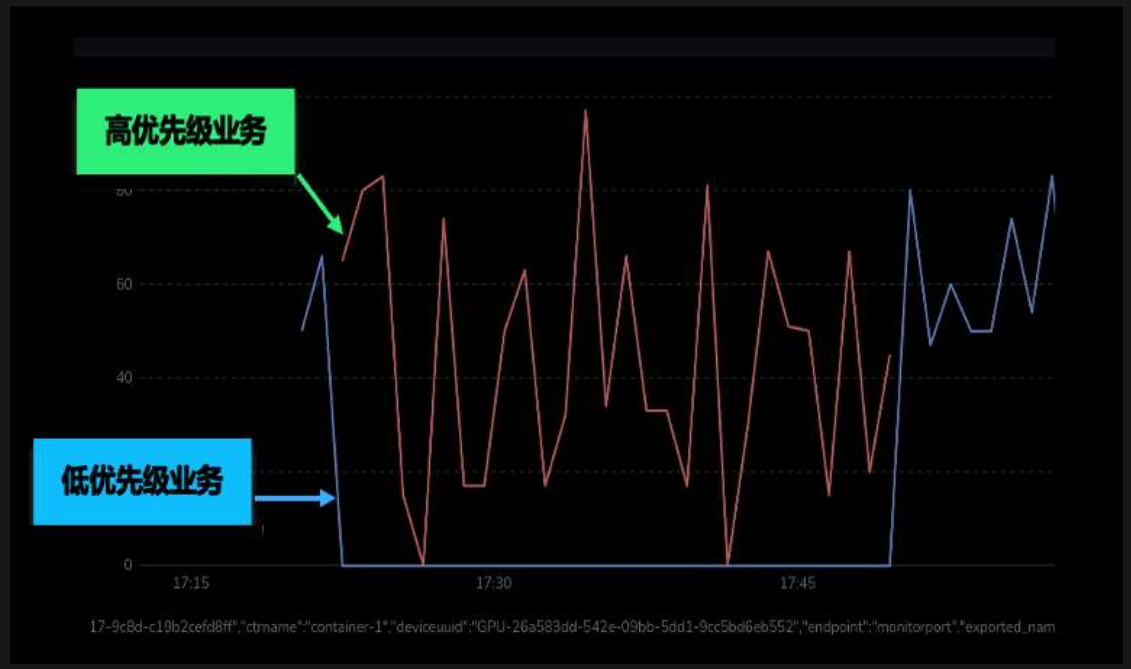
GPU 算力/显存超卖



算力超卖：支持给 Pod 分配超过 100% 的 GPU 算力。

显存超卖：支持给 Pod 分配超过 GPU 卡实际的显存。

优先级抢占



场景：

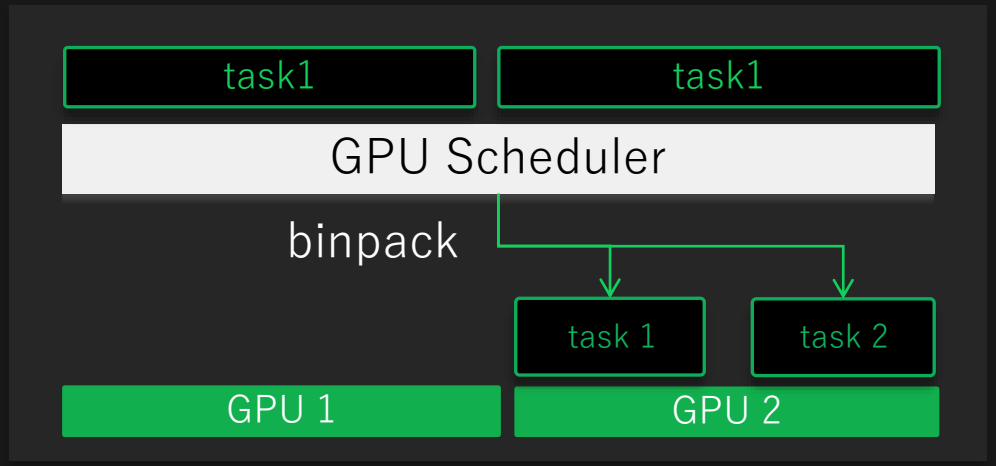
1. 推理任务实时性要求高，训练任务实时性要求低
2. 高低优先级任务混部
3. 资源抢占

效果：

1. 当低优先级的任务来了，如果此时 GPU 的算力没有被高优先级的 Pod 占用，那么低优 Pod 可以完全使用这块GPU 的全部算力。
2. 当高优先级的 Pod 有计算需求时，系统立即暂停正在占用 GPU 的低优先级 Pod，将 GPU 算力完全释放给高优先级 Pod 使用。

基于节点/GPU 卡的调度

优先调度到同一张 GPU 卡



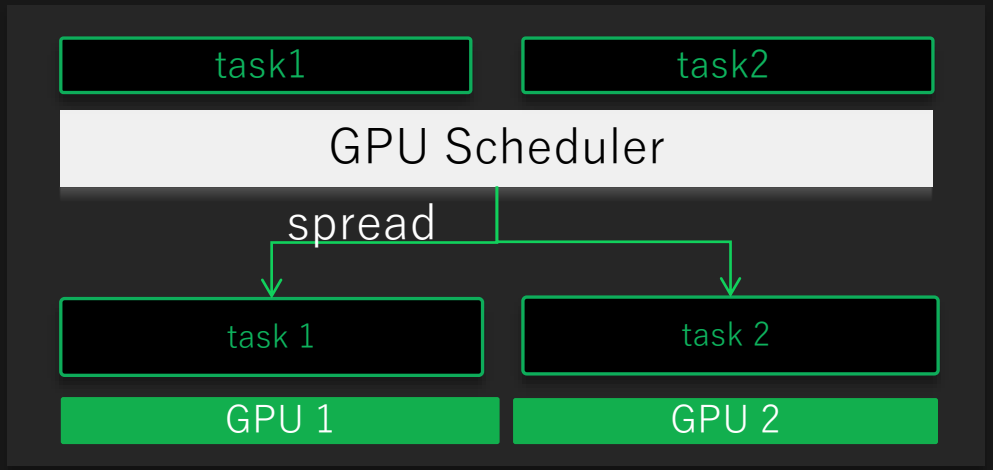
场景：

1.小任务分散在不同的 GPU 卡上，导致大任务调度失败

效果：

1. 优先调度到节点的同一张GPU卡上
2.避免共享业务造成 GPU 资源碎片

优先调度到不同的GPU卡



场景：

对 GPU 卡故障隔离要求高

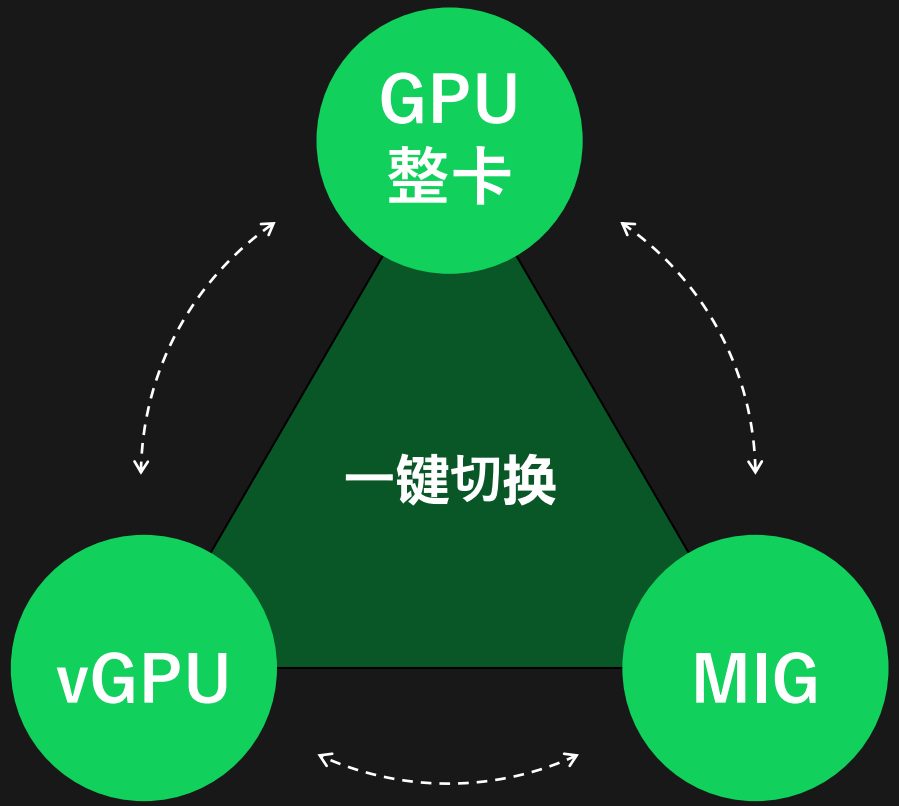
效果：

1.共享业务分散到节点的不同 GPU 卡，降低故障影响
2.实现负载均衡，避免某些卡过载而其他卡空闲的情况

NVIDIA 各种使用方式对比

	场景对比	功能对比	使用对比
NVIDIA GPU	整卡使用，用在资源消耗较多的场景，如 AI 训练。	资源被独占，没有额外的开销。 可以使用 NVLink 连接多块 GPU 卡。	使用起来最简单和理解。
NVIDIA Mig	最多可以虚拟化为 7 个实例，因为是物理隔离，服务 QOS 较高，可用于 AI 推理场景。	进行 MIG 切割会存在显存被浪费的情况。	在 single 模式下和使用原生GPU 一样，对上层应用影响较小，可以无缝切换。
vGPU	可以虚拟化的数量由用户自己指定，可用在 AI 开发或者AI推理场景。	虚拟化的数量越多，用户多后会出现资源消耗较多，大部分时间在用户切换。 可用在卡少用户多的场景去测试和使用。 可支持超分，让使用用户数增加。	不能兼容之前已经在 K8s 上使用的 GPU 应用。

GPU 模式切换



支持**节点维度**的 GPU 模式切**换**

场景：

1.**业务**改**变**，不契合当前的**业务场**景

2.**资**源利用率低，需要切分 GPU 卡

3.不同的**节**点需要根据特定的**负载**情况和任**务**来**调**整 GPU 的使用模式。

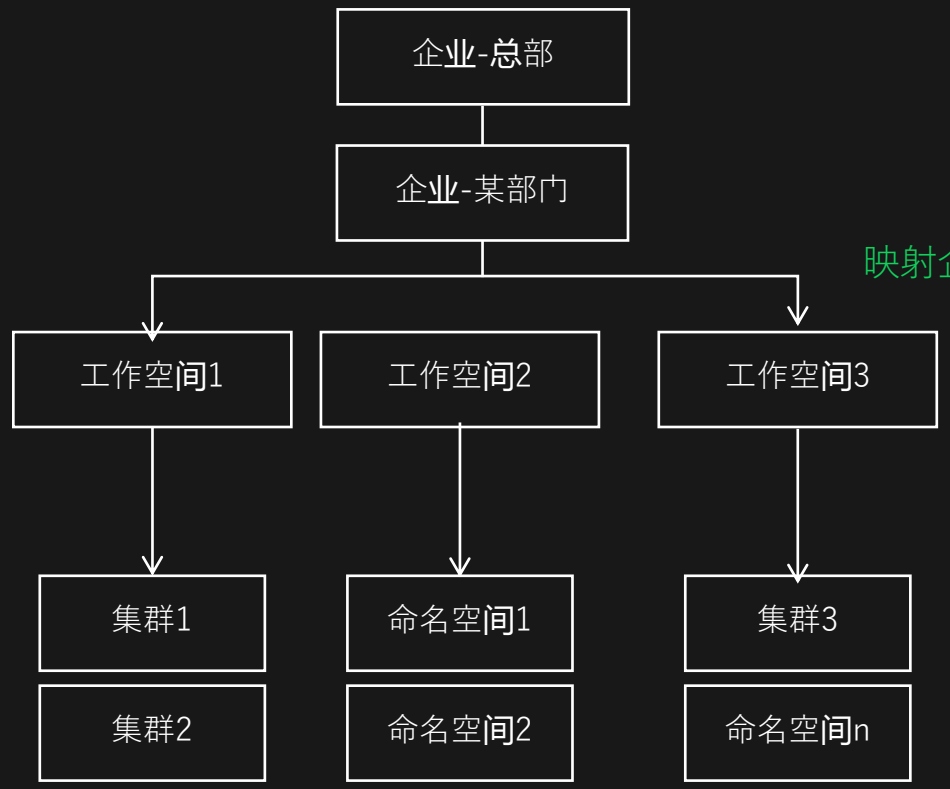
效果：

1. 一**键**切**换**，无需重装相关**驱动**、工具等**组**件。

2.自**动发现** GPU 上的 pod 的运行状**态**。

GPU 工作空间/命名空间资源限额

跨集群、跨命名空间资源隔离



映射企业项目

资源

工作空间

命名空间

容器（Pod）

细粒度的租户配额管理

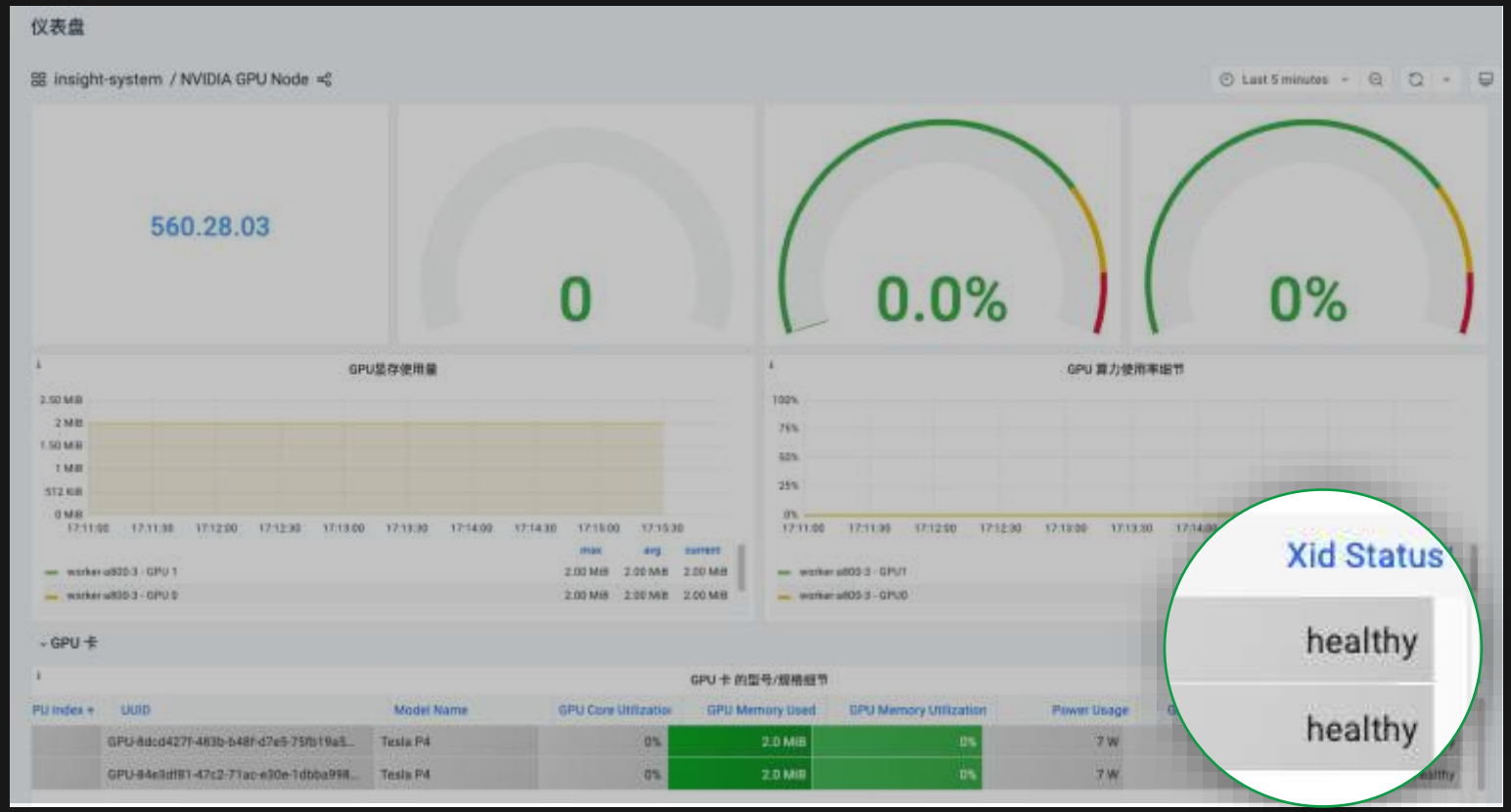
集群 GPU(算力/显存)、CPU(核)、内存(Gi)、存储(Gi)…

- 支持将集群资源共享给多个工作空间并进行资源限额
- 支持一个工作空间同时使用多个集群资源

命名空间 GPU(算力/显存)、CPU(核)、内存(Gi)、存储(Gi)…

GPU(算力/显存)、CPU(核)、内存(Gi)

监控告警体系构建



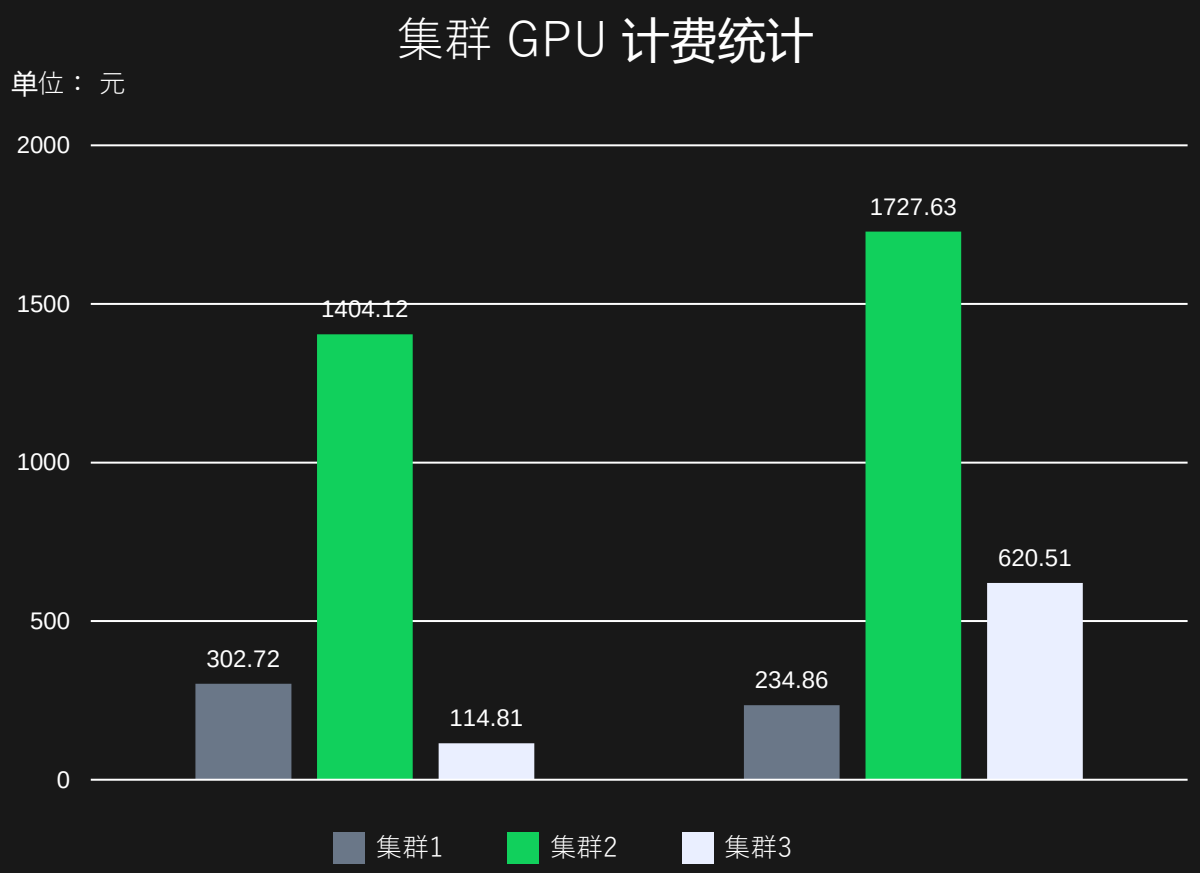
GPU 卡维度的监控告警

- GPU 利用率
- 显存利用率
- 显存剩余量
- GPU 的温度度数
- XID 错误码告警

应用维度的监控告警

- Pod 对 GPU 的使用率
- Pod 对 GPU 显存的使用率
- Pod 对 GPU 显存的使用量

计量计费



注：当前仅支持 GPU 整卡模式计费

报表+计费

报表：GPU 算力/显存利用率
计费：GPU 算力/显存花费金额

自定义单价

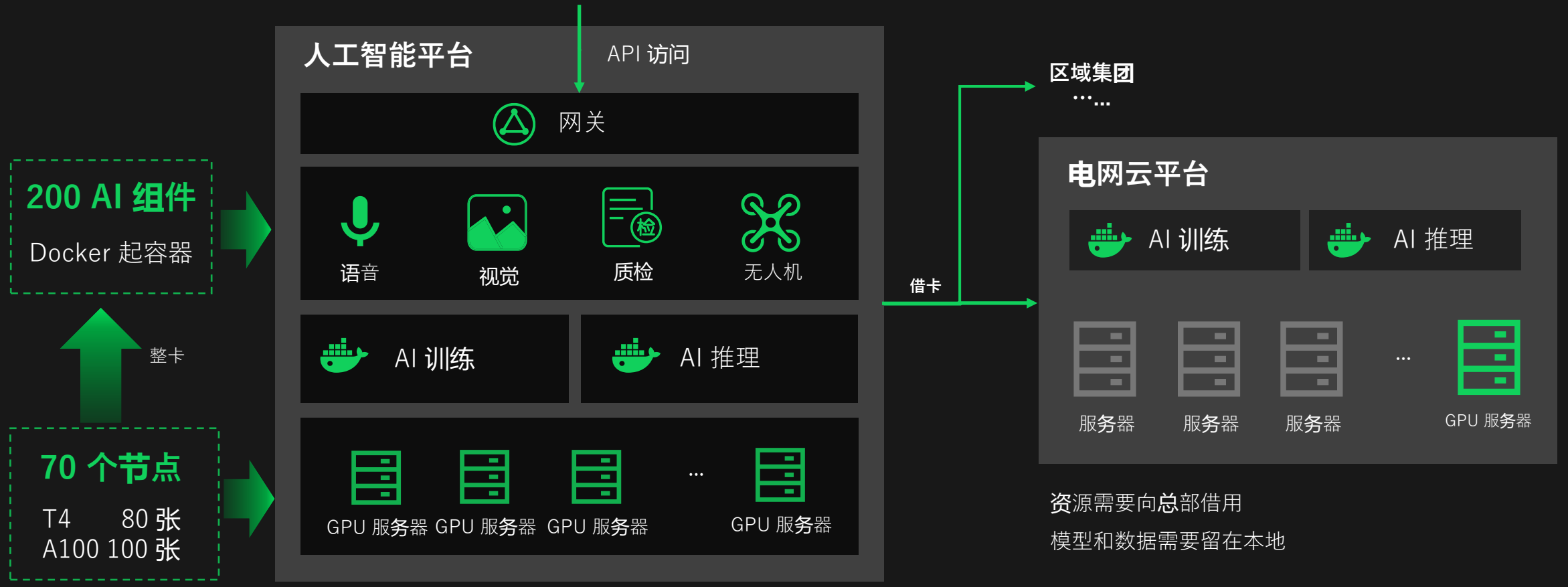
支持按照 GPU 卡型号设置单价，支持对集群、节点、容器组、工作空间、命名空间分别计费。

Part 03

应用场景

应用场景 - 某电网集团

1. 人工智能平台由某电网总部承建，GPU 资源充足
2. AI 组件由下属地州区域 供电局 提供



应用场景：某证券公司

用户痛点：

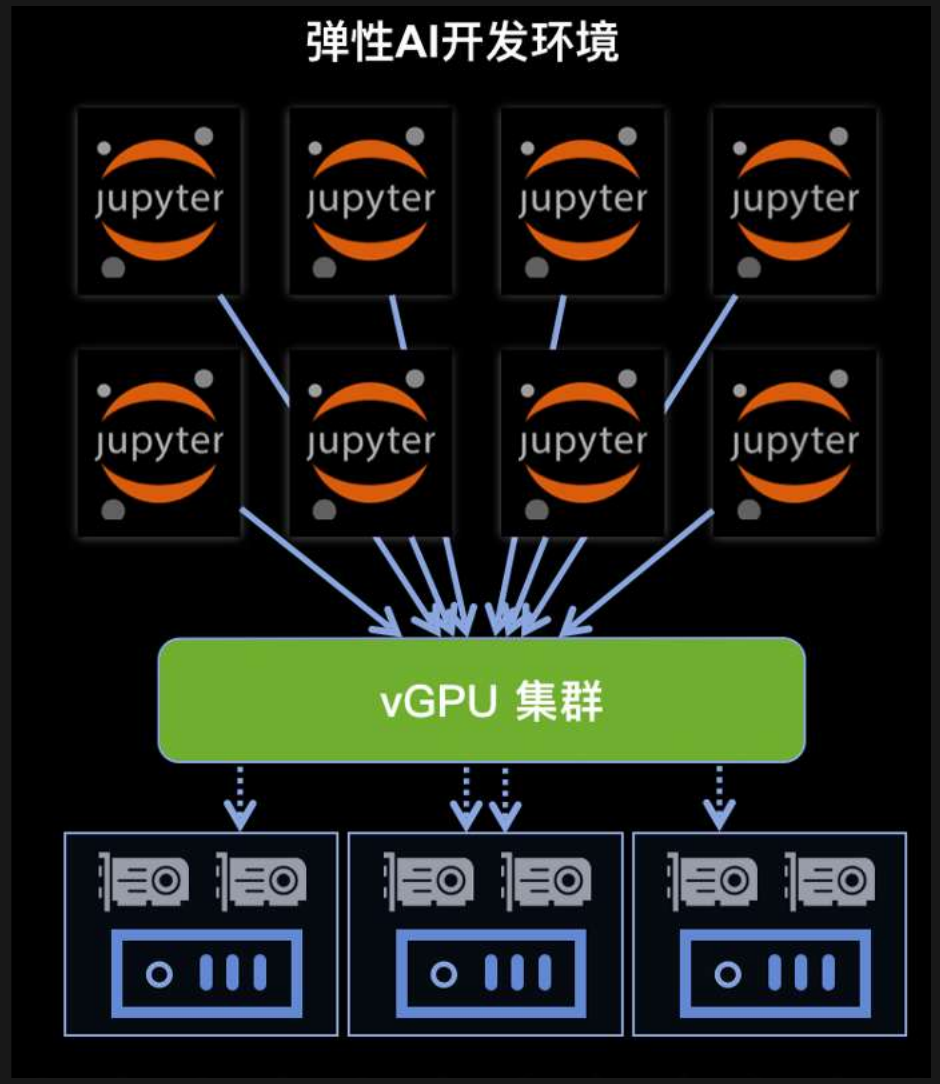
目前公司搭建了一套 kubeflow 平台，供算法工程师使用。但是默认情况下一张显卡只能分配给一个 notebook，并且如果 notebook 不停止/销毁的话，会一直独占这张显卡。（增加使用场景和痛点，找指标）

解决方案：

采用 vGPU Scheduler 技术，可以使一张物理显卡同时让多个算法工程师使用，提升工作效率。

客户收益：

- GPU 变为动态分配，使用效率获得提升；
- 支持 GPU 虚拟化，使用策略上更加灵活；
- 打通开发环境和训练任务过程，算法工程师不必困扰于资源分配的过程，可以更专注于业务的研发。



Part 04

Demo 演示

群聊：AI 进阶指南课程课后群 2

该二维码 7 天内 (11月4日前) 有效，重新进入将更新

https://10.20.100.201/kpanda/clusters

docker.samzong.me/chrstnhntschl/gpu_burn

Thanks.