DaoCloud

AI 进阶指南：教你如何轻松驾驭异构算力

主讲人：冷荣富

为数字世界寻找全局最优解

Powered By DaoCloud

Content

目录

Part 01

异构算力介绍

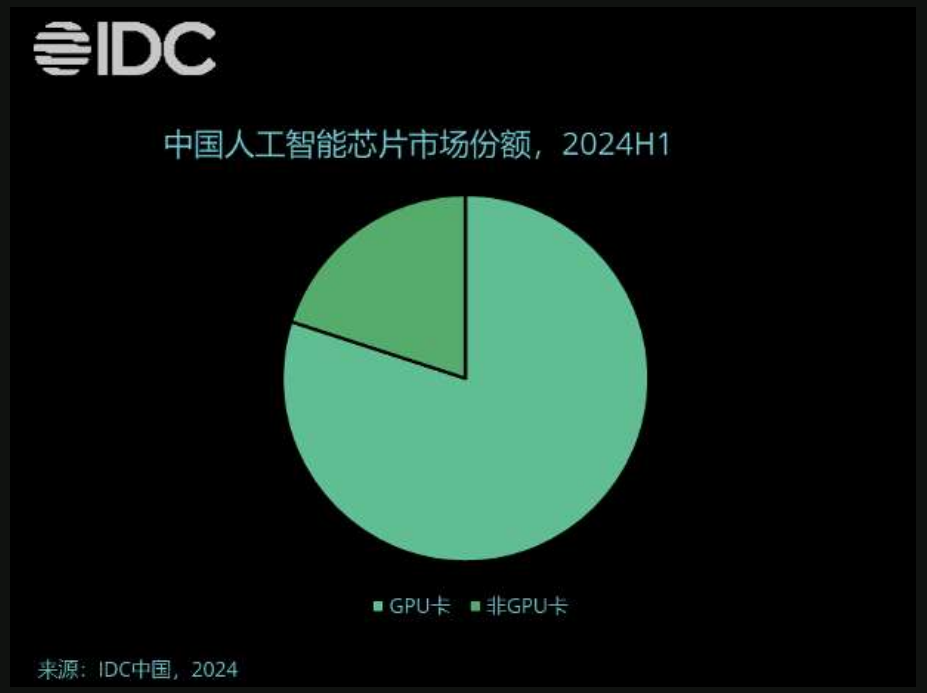
异构算力有哪些

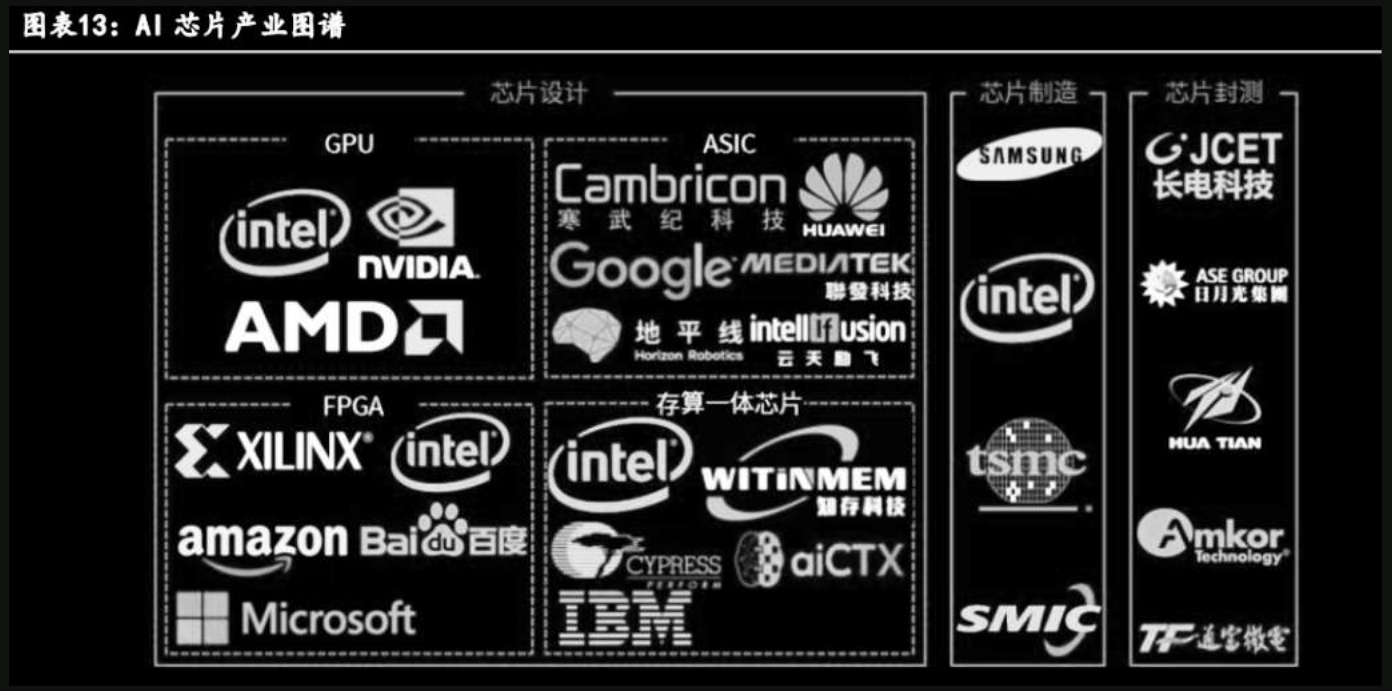
异构算力有哪些

异构算力通常是指 CPU、GPU、FPGA、ASIC 等多种不同的算力处理体系，能够满足不同场景中的应用需求，实现计算效力最大化

异构 GPU 算力有 Nvidia、AMD、Intel、昇腾、寒武纪、天数智芯、沐曦等。

异构算力市场





国外：80%
国内：20%

AI 芯片分类

异构算力编程

编程语言：CUDA
通信库： NCCL
外部通信: RDMA(IB)
内部通信：NVLink

编程语言：OpenCL 和 CUDA
通信库： RCCL
外部通信: RDMA(RoCE)
内部通信：Infinity Fabric

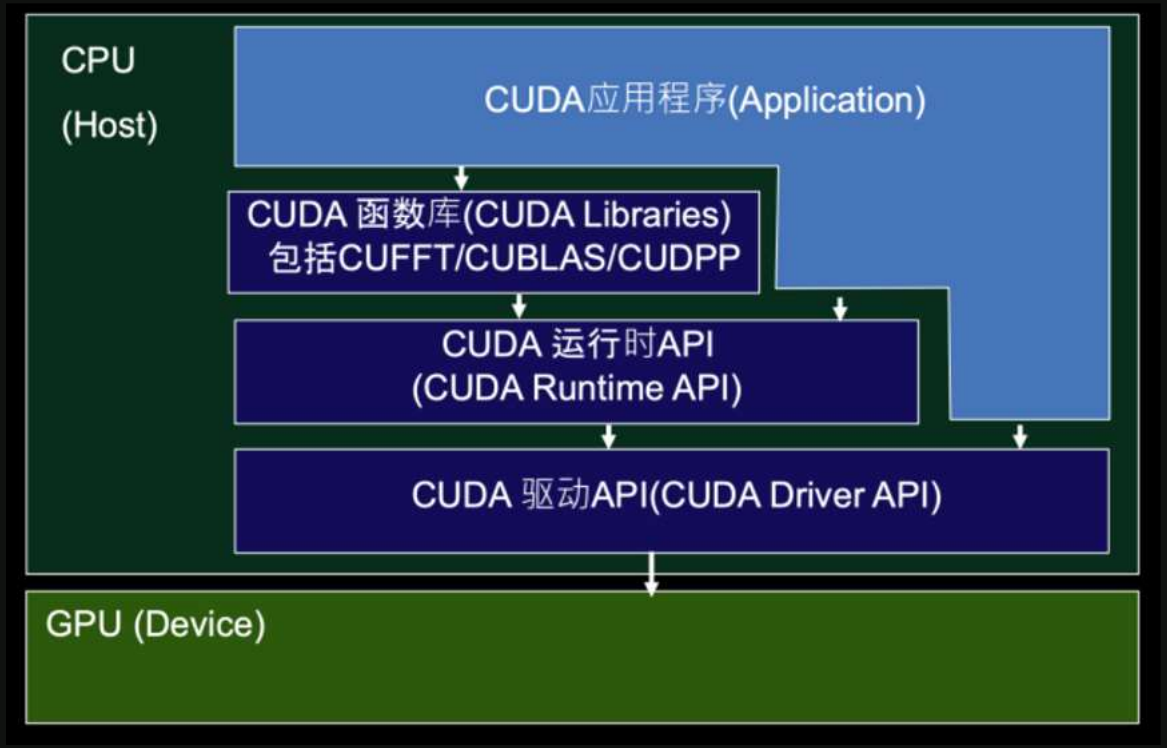
编程语言： oneAPI
通信库： OneCCL
外部通信: OPA(IB)
内部通信： EMIB

编程语言：CANN
通信库： HCCL
外部通信: HCCN
内部通信：HCCS

编程语言：CNML
通信库： CNCL
外部通信: RDMA(RoCE)

编程语言： DTK
通信库： RCCL
外部通信: RDMA(RoCE)
内部通信：无

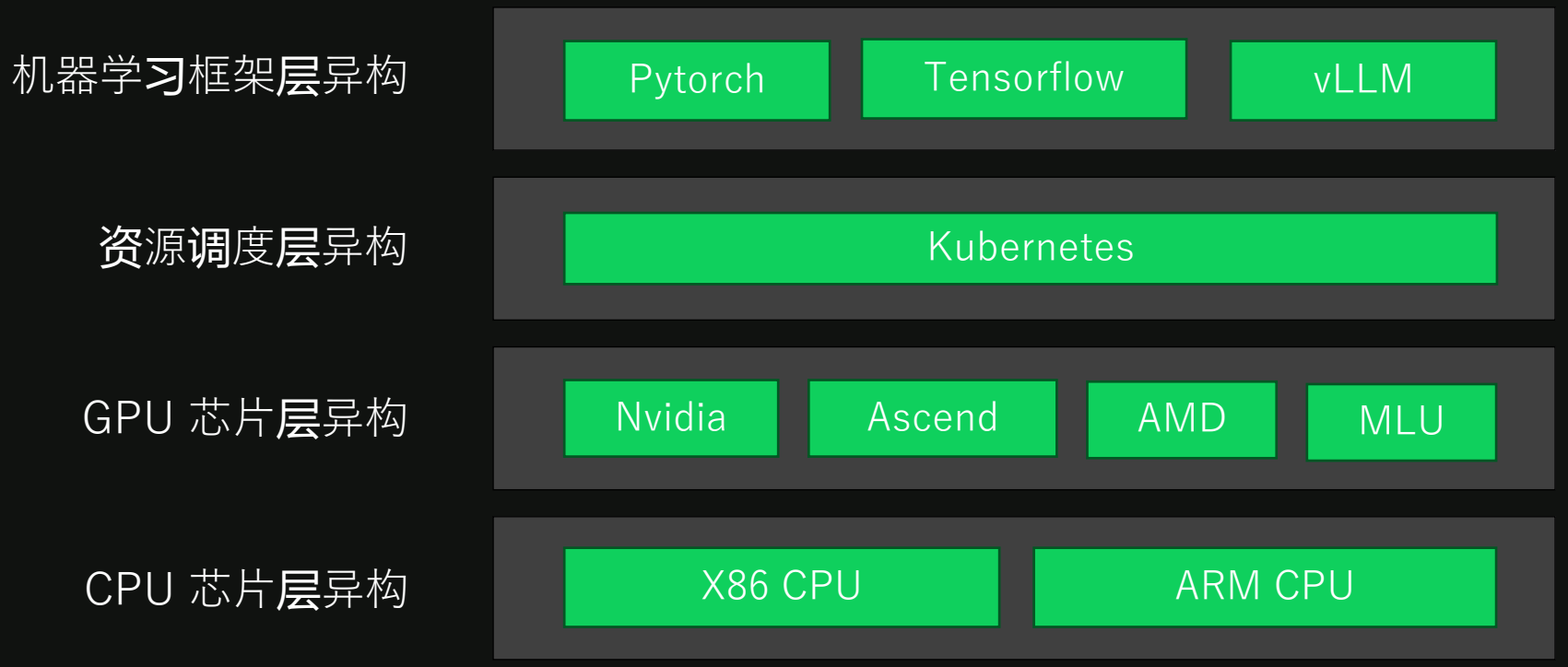
异构算力编程



NVIDIA

Ascend

异构算力使用


机器学习框架层异构
Pytorch
Tensorflow
vLLM
资源调度层异构
Kubernetes
GPU 芯片层异构
Nvidia
Ascend
AMD
MLU
CPU 芯片层异构
X86 CPU
ARM CPU

Part 02

基础设施如何管理异构算力

管理异构算力存在哪些挑战

01

每个厂商的软件生态和硬件都不兼容。

02

同一厂商的不同芯片算力和显存也不同。

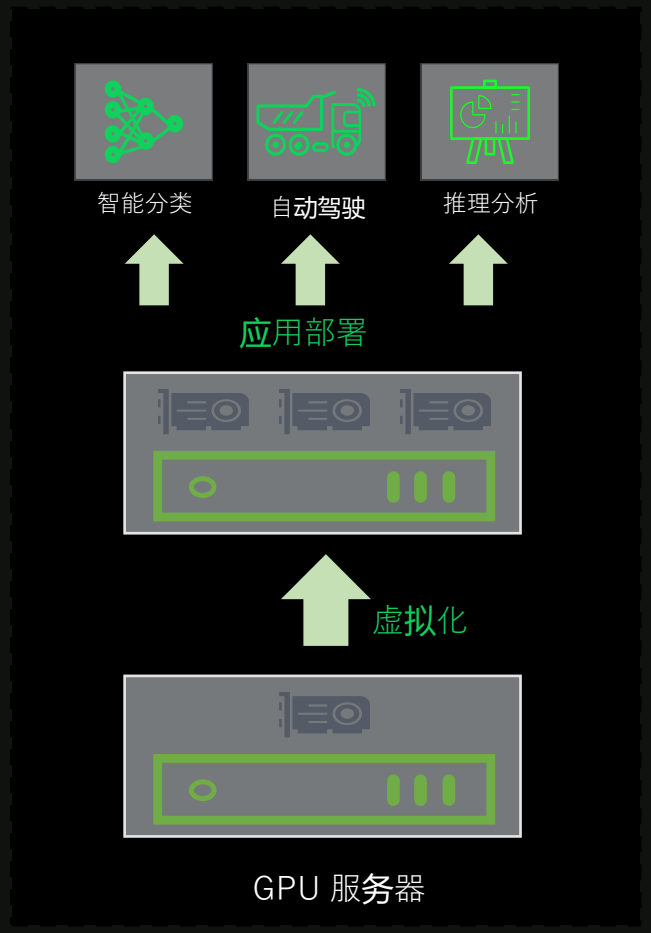
03

Kubernetess 上需要分别部署不同厂商的芯片管理组件。

如何管理异构算力


推理
训练
统一优化调度
统一算力管理
统一算力监控
统一算力共享
异构算力调度和管理平台（d.run）
Nvidia
AMD
Intel
国产

如何提升异构算力利用率

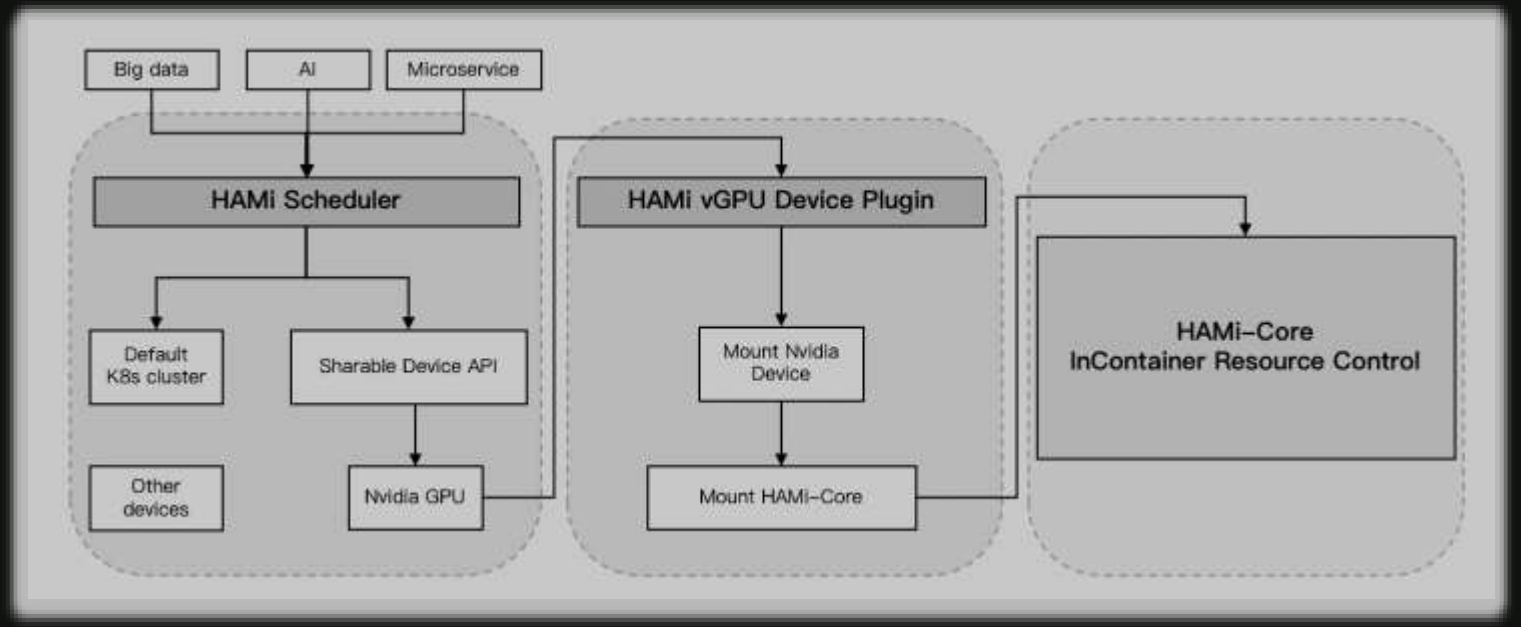


通过共享异构算力。

01. 基于优先级的调度
02. 基于 GPU 卡型号的调度
03. 基于 GPU 卡调度
04. 基于节点调度

通过统一调度、监控等来管理异构算力

Nvidia vGPU 工作方式



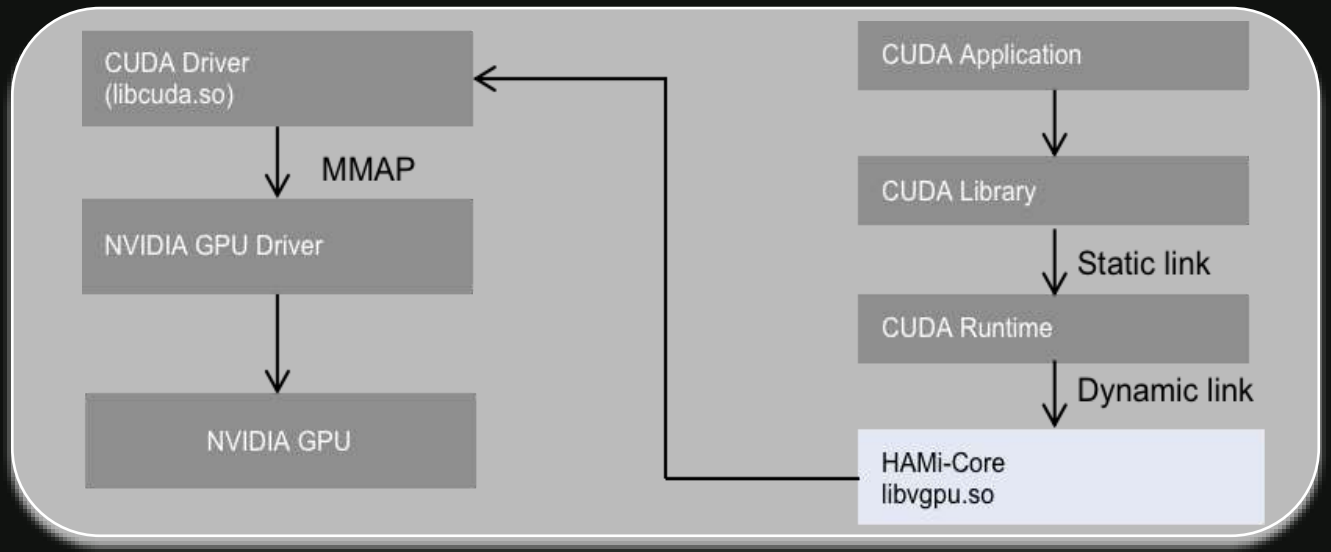
HAMi-Core uses symbolic hijacking to operate inside containers

Prerequisites:

- Nvidia driver version >= 440
- CUDA version >= 10.2

Features:

- Device Memory isolation
- Core utilization limitation
- Fault isolation
- Transparent to GPU tasks



GPU 资源调度

01. 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。

02. 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。

03. 基于GPU卡调度

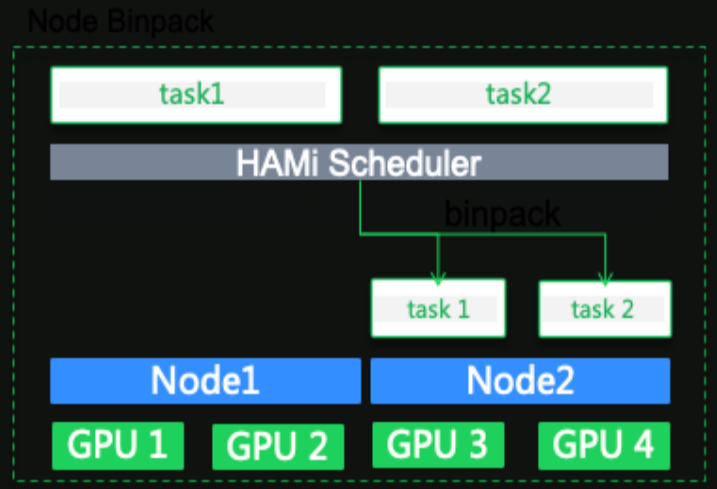
- 在选定节点后进行卡维度的分配决策，为 Pod 中每个容器选择和分配节点上的 GPU 卡：
- Binpack：优先选择资源利用率高的 GPU 卡，容器集中到同一块卡上
- Spread：优先选择资源利用率低的 GPU 卡，容器分散到不同的卡上

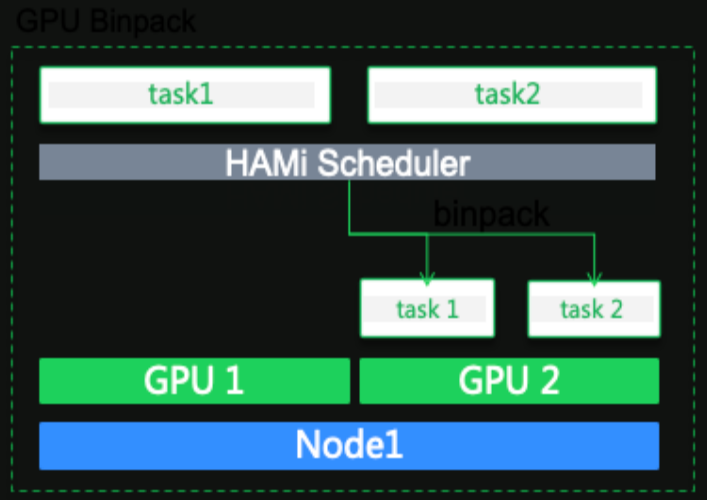
04. 基于节点调度

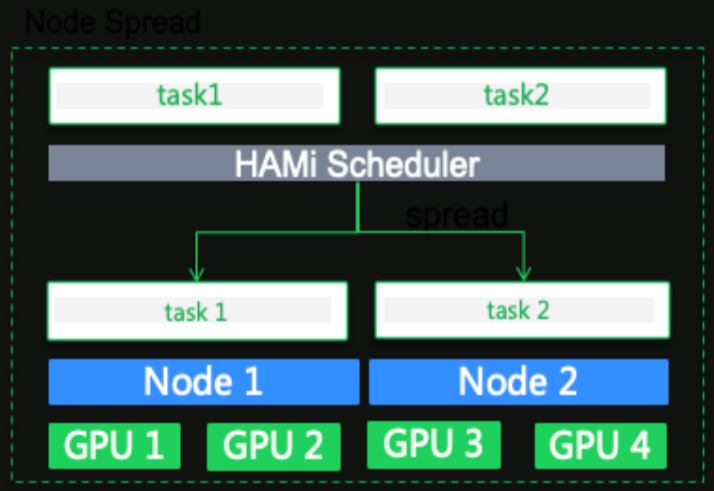
根据节点 GPU算力和显存进行加权平均打分提供 2 种策略：

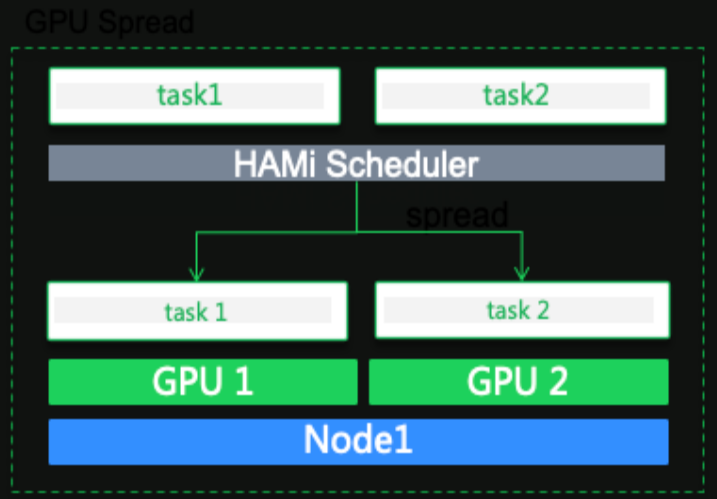
- Binpack：GPU 分配率越高，打分越高，Pod 集中调度到同一个节点
- Spread：GPU 分配率越高，打分越低，Pod 分散调度到各个节点

调度策略（ binpack & spread ）





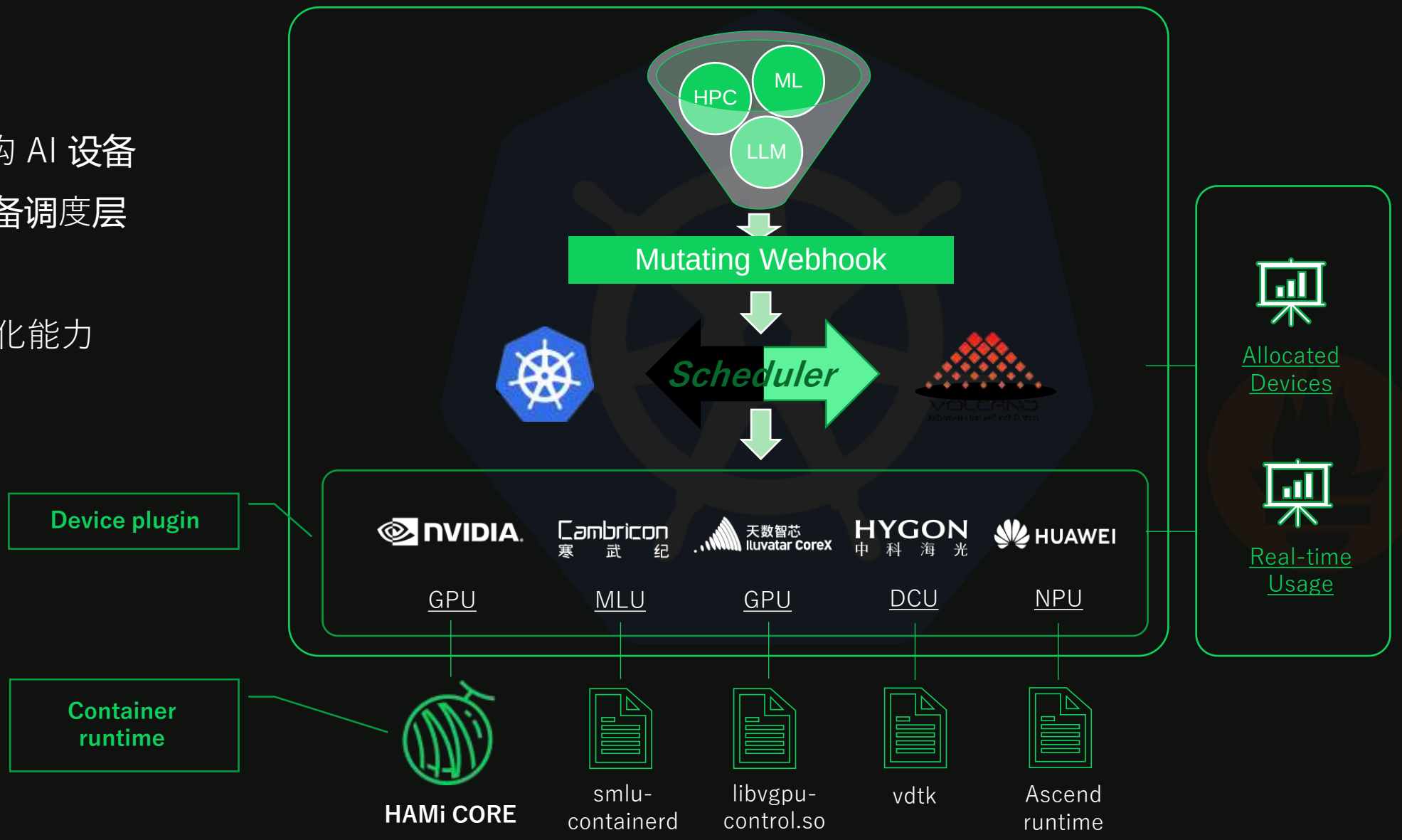




- 实际场景和需求
- 小任务饿死大任务
- 高优先级任务避免放在一个篮子中

HAMI简介

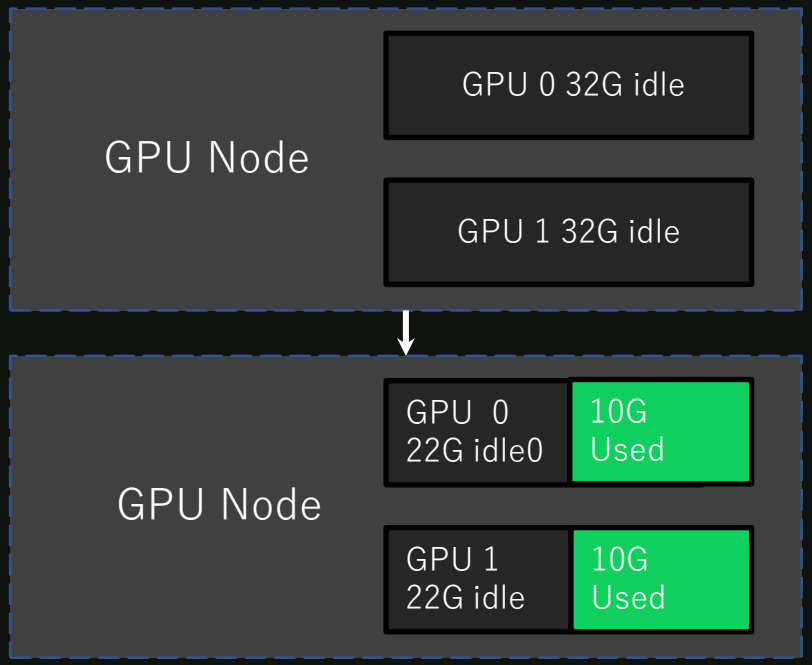
- 统一管理多种异构 AI 设备
- 统一的异构 AI 设备调度层
- 统一可观测性
- Nvidia GPU 虚拟化能力



异构设备 NVIDIA

参数描述:
nvidia.com/gpu: 表示容器中可见 GPU 的数量。
nvidia.com/gpumem: 指定每个 GPU 使用的内存大小。如果没有设置，默认是使用所有可用的 GPU 内存。
nvidia.com/gpucores : 指定每个 GPU 使用的百分比。

```
$ cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod12
spec:
  containers:
    - name: ubuntu-container
      image: ubuntu:18.04
      command: ["bash", "-c", "sleep 86400"]
      resources:
        limits:
          nvidia.com/gpu: 2 # requesting 1 vGPUs
          nvidia.com/gpumem: 10240
          nvidia.com/gpucores: 30
```




```
root@gpu-demo-6b6b88b75b-x9tjb:~# nvidia-smi
[HAMI-core Msg(35:140111864956736:libvgpu.c:836)]: Initializing.....
Thu Apr 18 06:22:54 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.12             Driver Version: 535.104.12   CUDA Version: 12.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A800 80GB PCIe          On  | 00000000:13:00.0 Off |                    0 |
| N/A   36C    P0              81W / 300W |      0MiB / 10240MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA A800 80GB PCIe          On  | 00000000:1C:00.0 Off |                    0 |
| N/A   39C    P0              82W / 300W |      0MiB / 10240MiB |      0%      Default |
|                                         |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
[HAMI-core Msg(35:140111864956736:multiprocess_memory_limit.c:434)]: Calling exit handler 35
root@gpu-demo-6b6b88b75b-x9tjb:~#
```

异构设备 Ascend

参数描述:

huawei.com/ascend910: 指定容器中可见的Ascend 910的数量。
huawei.com/ascend910memory: 指定每个Ascend 910s使用的内存大小。如果没有设置，默认是使用所有可用的设备内存。

```
+-------------------------------------------------------------------------------------------+
| npu-smi 24.1.rc1                 Version: 24.1.rc1                                        |
+---------------------------+---------------+---------------------------------------------------+
| NPU   Name                | Health        | Power(W)    Temp(C)           Hugepages-Usage(page)|
| Chip                      | Bus-Id        | AICore(%)   Memory-Usage(MB)  HBM-Usage(MB)        |
+===========================+===============+===================================================+
| 0     910B3               | OK            | 95.9        39                0    / 0          |
| 0                         | 0000:C1:00.0  | 0           0    / 0          24082/ 65536         |
+---------------------------+---------------+---------------------------------------------------+
```

container

```
$ cat <<EOF | kubectl apply -f -

spec:
  containers:
    - ...
      resources:
        limits:
          huawei.com/Ascend910: 1
          huawei.com/Ascend910-memory: 16384
```

```
+-------------------------------------------------------------------------------------------+
| npu-smi 24.1.rc1                 Version: 24.1.rc1                                        |
+---------------------------+---------------+---------------------------------------------------+
| NPU   Name                | Health        | Power(W)    Temp(C)           Hugepages-Usage(page)|
| Chip                      | Bus-Id        | AICore(%)   Memory-Usage(MB)  HBM-Usage(MB)        |
+===========================+===============+===================================================+
| 1     910B3vir05_1c_16g   | OK            | 89.9        25                0    / 0          |
| 0                         | 0000:C2:00.4  | 0           0    / 0          1235 / 16384         |
+===========================+===============+===================================================+

+---------------------------+---------------+---------------------------------------------------+
| NPU     Chip              | Process id    | Process name              | Process memory(MB)      |
+===========================+===============+===================================================+
| No running processes found in NPU 1                                                           |
+===========================+===============+===================================================+
```

其它异构设备

```
spec:
  containers:
    - …
      resources:
        limits:
          cambricon.com/vmlu: 1 # requesting 1
          cambricon.com/mlu.smlu.vmemory: 20
          cambricon.com/mlu.smlu.vcore: 10
```

寒武纪

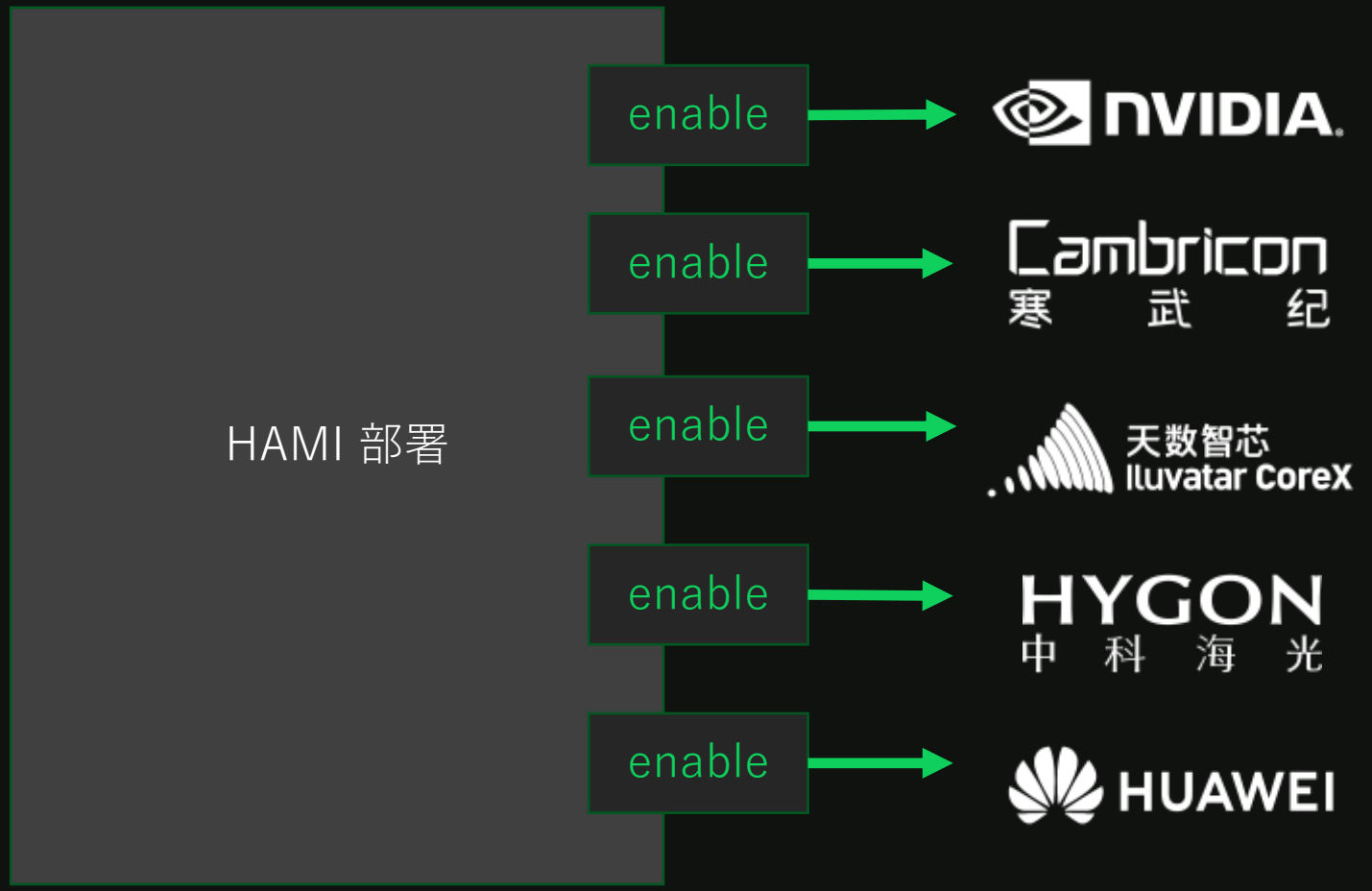
```
spec:
  containers:
    - …
      resources:
        limits:
                hygon.com/dcunum: 1
                hygon.com/dcumem: 2000
                hygon.com/dcucores: 60
```

海光

```
spec:
  containers:
    - …
      resources:
        limits:
          iluvatar.ai/gpu: 1
          iluvatar.ai/vcuda-core: 50
          iluvatar.ai/vcuda-memory: 64
```

天数智芯

HAMI一键部署异构算力管理组件



Part 03

大模型如何使用异构算力

LLM 如何使用异构算力

01 资源统一调度,机器学习框架、推理框架进行适配；（国外芯片适配的比较好）

02 通信库进行适配，用通信库来兼容异构算力

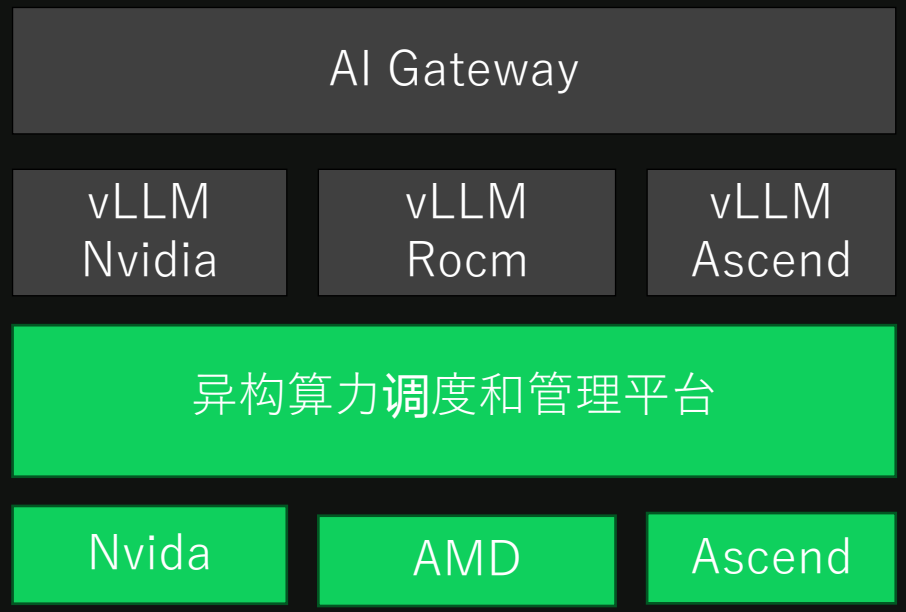
LLM框架适配异构算力的情况

Pytorch

vLLM

Pytorch 机器学习框架支持 Nvidia 和 AMD 以及 CPU 三种异构算力。
vLLM 推理框架支持 Nvidia、AMD、Intel、TPU、CPU、AWS 加速芯片六种异构算力芯片。

LLM框架如何调度使用异构算力



1. 不同的 AI 负载借助框架可以运行在不同的芯片上.
2. 针对不同的框架部署不同的 vLLM 实例.
3. 调度异构 vLLM 实例到对应的异构算力芯片上.
4. 通过 AI Gateway 来统一暴露服务.

LLM 异构芯片通信库挑战

1. 没办法互联互通：昇腾 910B 服务器内部通过 HCCS 连接，服务器之间通过华为自研的内置RDMA连接，卡卡之间使用 HCCL 通信库进行相互通信。Nvidia 服务器内部通过 NVLINK 连接，服务器之间通过 RDMA(IB、Roce) 连接，卡卡之间使用 NCCL 通信库进行相互通信。

2 算力不均衡：由于 GPU、昇腾等芯片在计算能力，显存大小，I/O 吞吐，通信库等均存在差异。

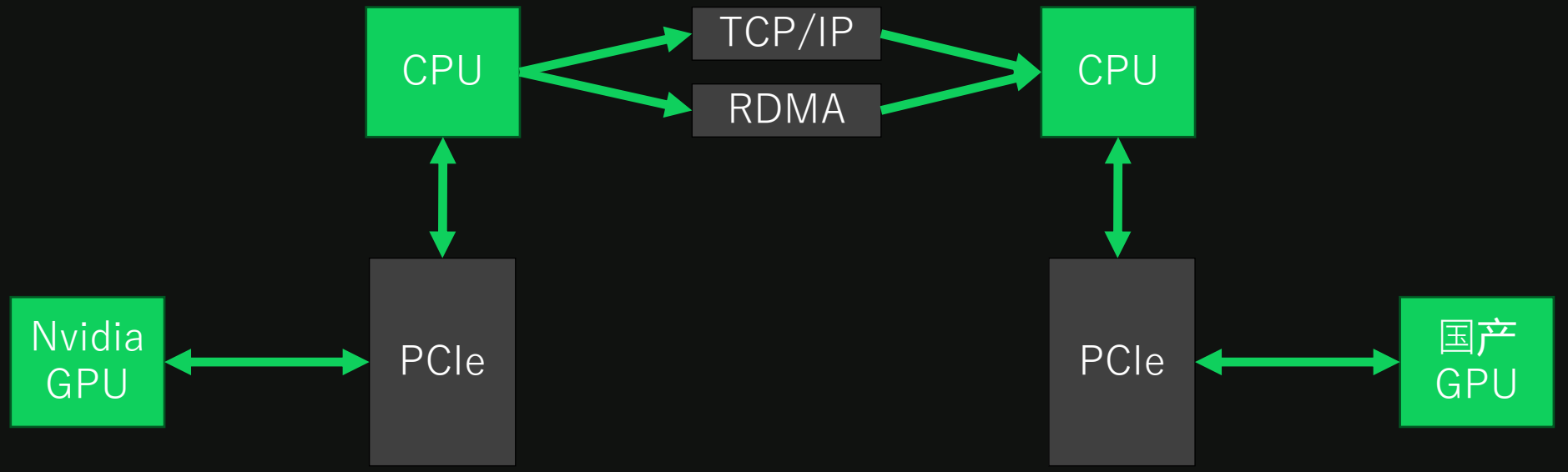
如何解决互联互通

挑战：
不同厂商的 GPU 或 AI 芯片为了优化通信，通常会构建自己的通信库，在内部架构和通信接口上存在差异，导致不同通信库之间无法通信。

解法：

1. 基于 CPU 中转的异构通信
2. 无需 CPU 中转的异构通信（直接芯片间RDMA）

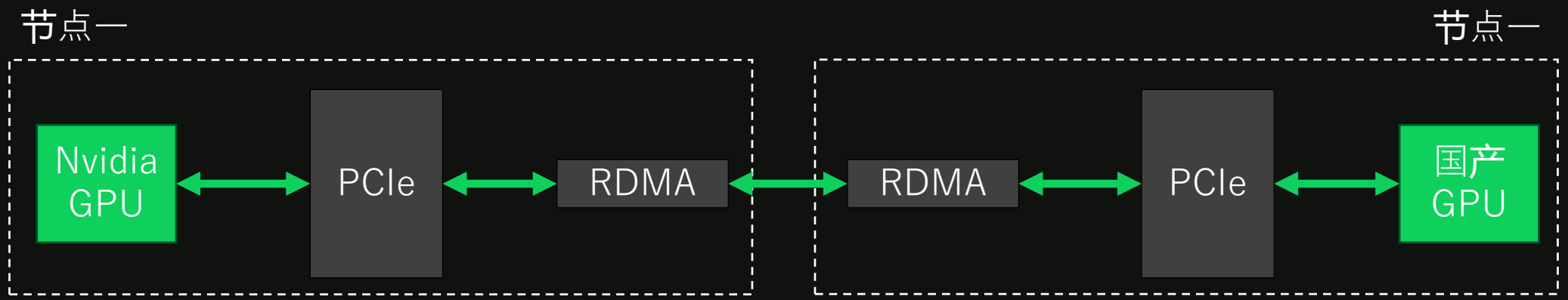
解法一：基于CPU中转的异构通信



通信流程：

1. 从源 GPU 移动数据到异构 GPU 时，首先会经由 PCIe 通道，将数据从 GPU 复制到源节点的 CPU。
2. 随后通过 RDMA 或 TCP 跨节点通信，抵达目标节点的 CPU。
3. 数据再次借助 PCIe 的高速能力，被复制至目标 GPU 上

解法二：无需 CPU 中转的异构通信（直接芯片间 RDMA）



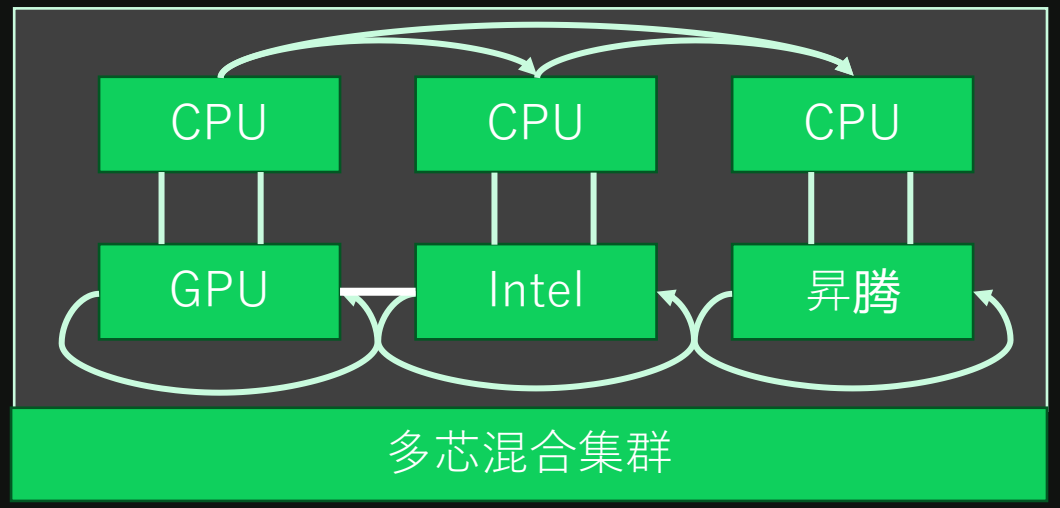
通信流程：如果目标芯片也支持标准的 RDMA 协议，那就可以直接基于 RDMA 通信。
1.从源 GPU 移动数据到异构 GPU 时，首先会经由 PCIe 通道，然后经过 RDMA 网卡直接发送数据.
2. 通过发送到目标 RDMA 网卡之后，在发送到目标 PCIe 通道上，在发送到国产 GPU 上.

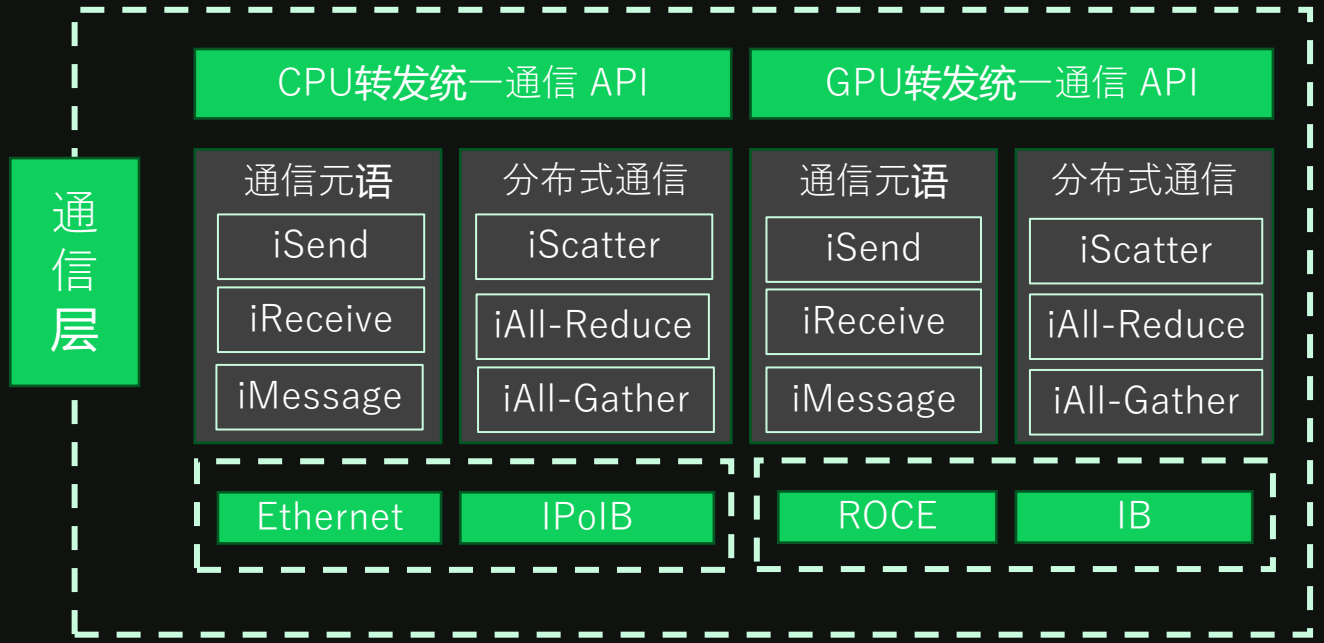
如何实现上述两种解法：统一通信库

上述两个解法只是解决了芯片之间通信的问题，还需要解决通信库的问题？

解法：

1. 通过统一抽象出一层通信 API，再把各种芯片的通信库适配上。
2. 或者是以 NCCL 的 API 为标准，实现异构算力芯片的通信。





挑战二：算力不均衡

挑战：

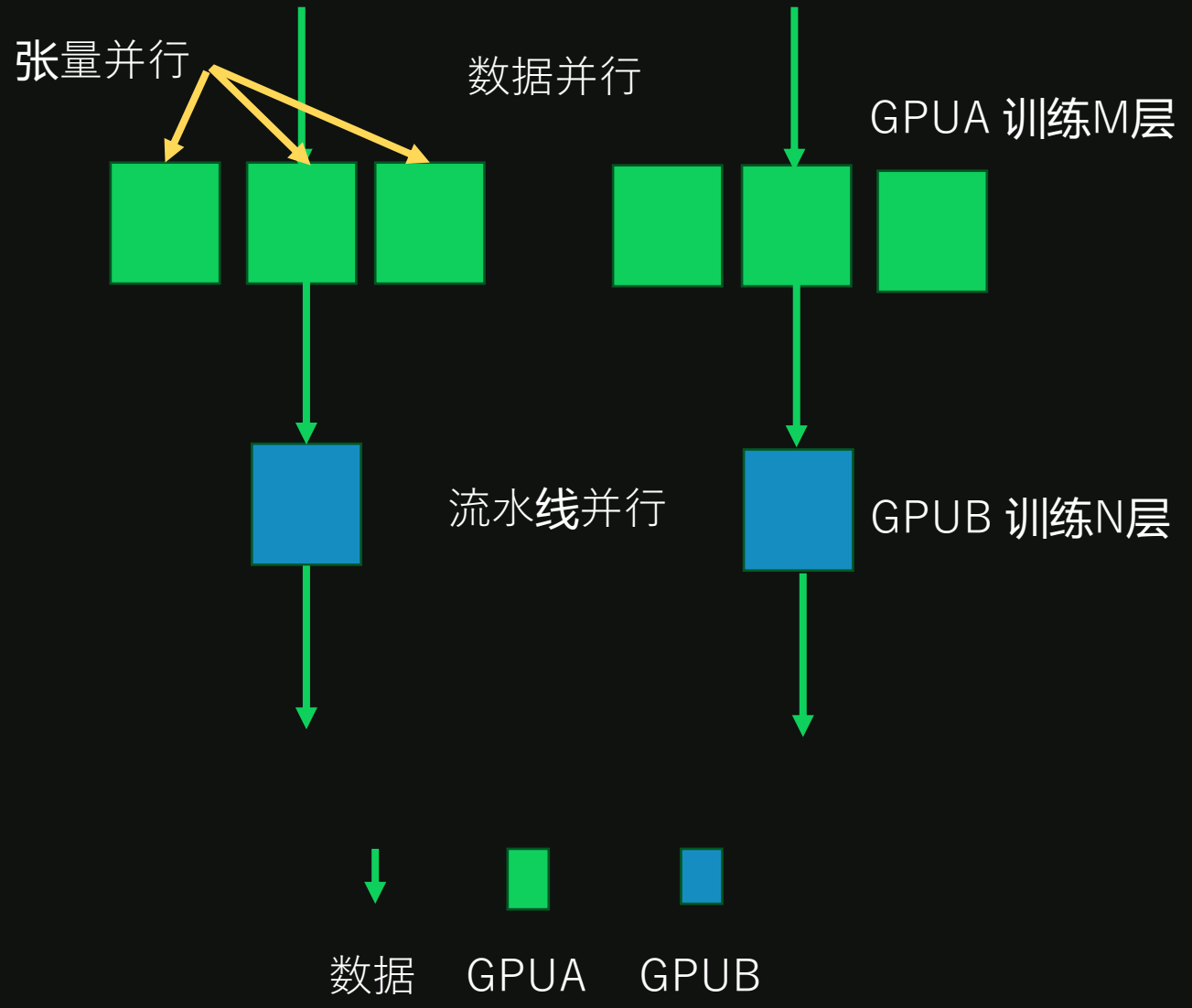
系统如何在算力分布不均匀的情况下，根据模型特征和系统的实时训练状态对任务进行均匀或非均匀的切分，保证算力的最大化利用。

解法：

1. 基于流水线并行的非均匀拆分策略
2. 基于数据并行的异构训练
3. 异构节点的流水线并行

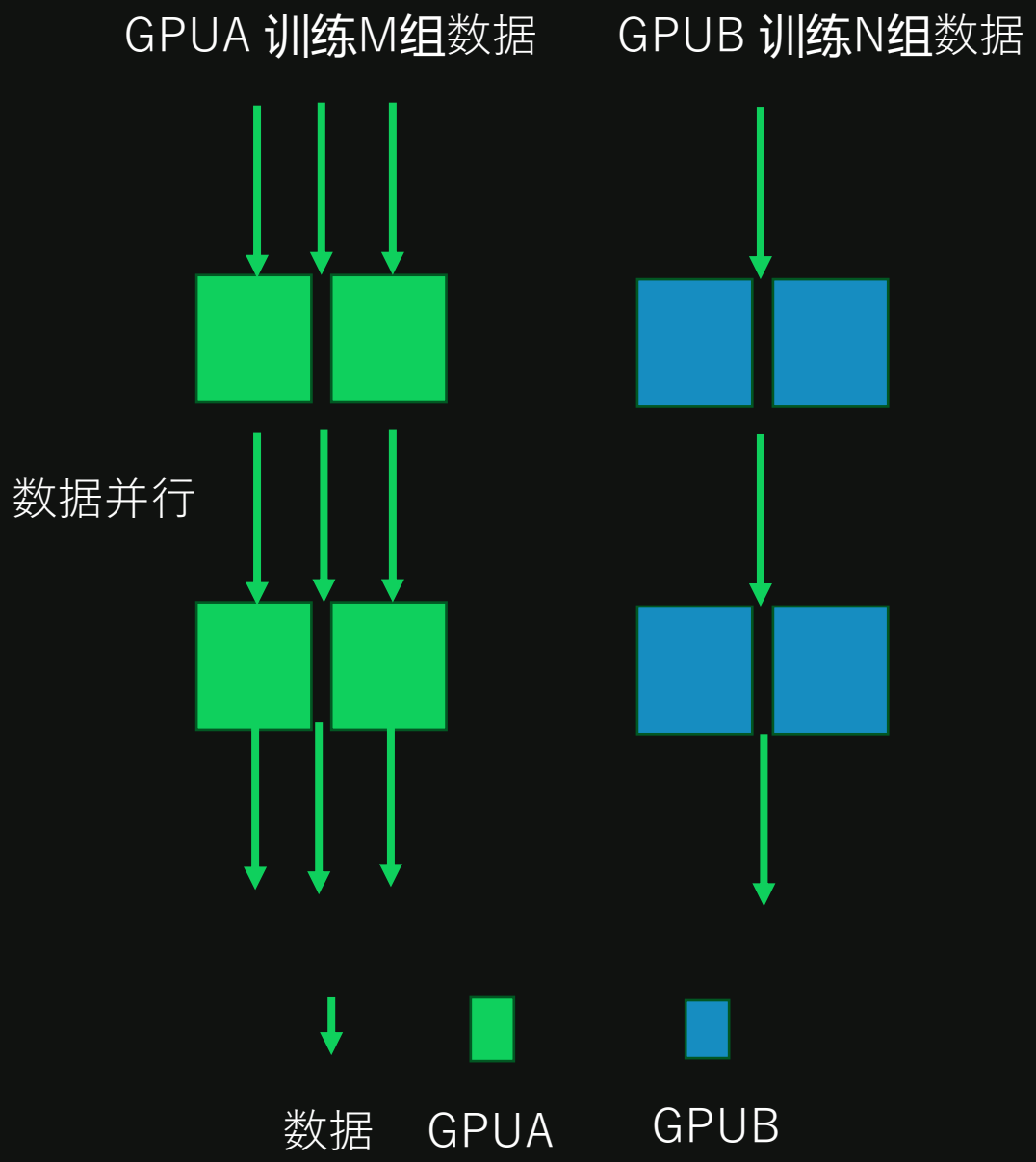
解法一：基于流水线并行的非均匀拆分策略

1. 考虑不同算力处理不同的层，比如 N 卡性能比国产卡高 3 倍. N 卡一次训练 3层，国产卡一次训练1层.
2. 需要实现模型的非均匀拆分.



解法二：异构数据并行

1. 考虑不同算力处理不同的数据批处理规模，比如 N 卡性能比国产卡高 3 倍. 那N 卡一次处理 3 张图片，国产卡一次处理 1 张图片.
2. 需要实现数据的非均匀拆分.



解法三：异构节点的流水线并行

考虑到异构 GPU 之间的通信性能低于同构 GPU 之间的通信性能，因此在做非均匀拆分是需要按照如下规则来考虑：

1. 同构节点上采用数据并行
2. 节点内张量并行
3. 跨异构节点的流水线并行。

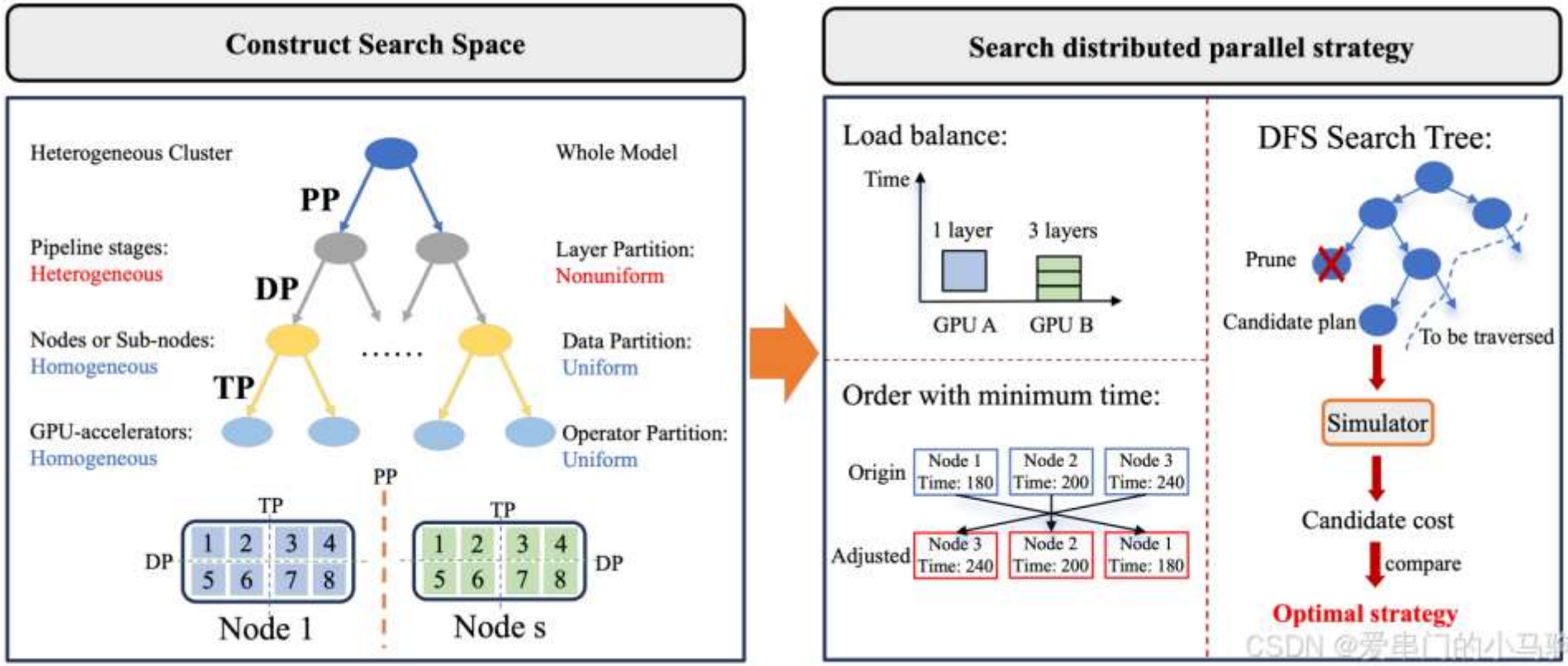
算力非均匀切分实现难点

挑战：上述讲述了通过流水线并行、数据并行的非均匀拆分策略来应对算力不均匀的问题，但是在实际切分过程中需要考虑很多因素：

1. 芯片在流水线并行中的位置顺序：不同位置的芯片可能需要前后节点的数据作为输入，进而影响其实际的计算效率。
2. 显存大小与重算策略：根据芯片显存容量，可能需要启用重算机制，以节省显存占用，但这可能会影响整体性能。
3. 单款芯片算子优化程度：不同芯片对特定算子的执行效率不尽相同，需针对性地进行优化，以发挥最大效能。

异构算力模型自动拆分过程

根据模型拆分的基本思路，构建检索空间，检索出模型最终拆分方式。



1. 构建一个三级搜索树来表示模型分布式训练策略的搜索空间，根节点表示整个模型，其他节点表示拆分后的子模型。叶节点表示在单个 GPU 加速器中执行的最终模型。
2. 在训练过程中，从搜索空间中按照负载平衡、最短的端到端训练时间来搜索分布式训练策略.

https://arxiv.org/html/2405.16256v2

Part 04

总结

异构算力管理

总结：

1. 在异构算力集群中使用异构资源的方式有多种方式.
2. 使用 d.run 可以应用在异构资源的推理场景.
3. 使用 HAMI 可以异构资源的管理和调度.

群聊：AI 进阶指南课程课后群 2

该二维码 7 天内 (11月12日前) 有效，重新进入将更新

Powered By DaoCloud

Thanks.