Microsoft

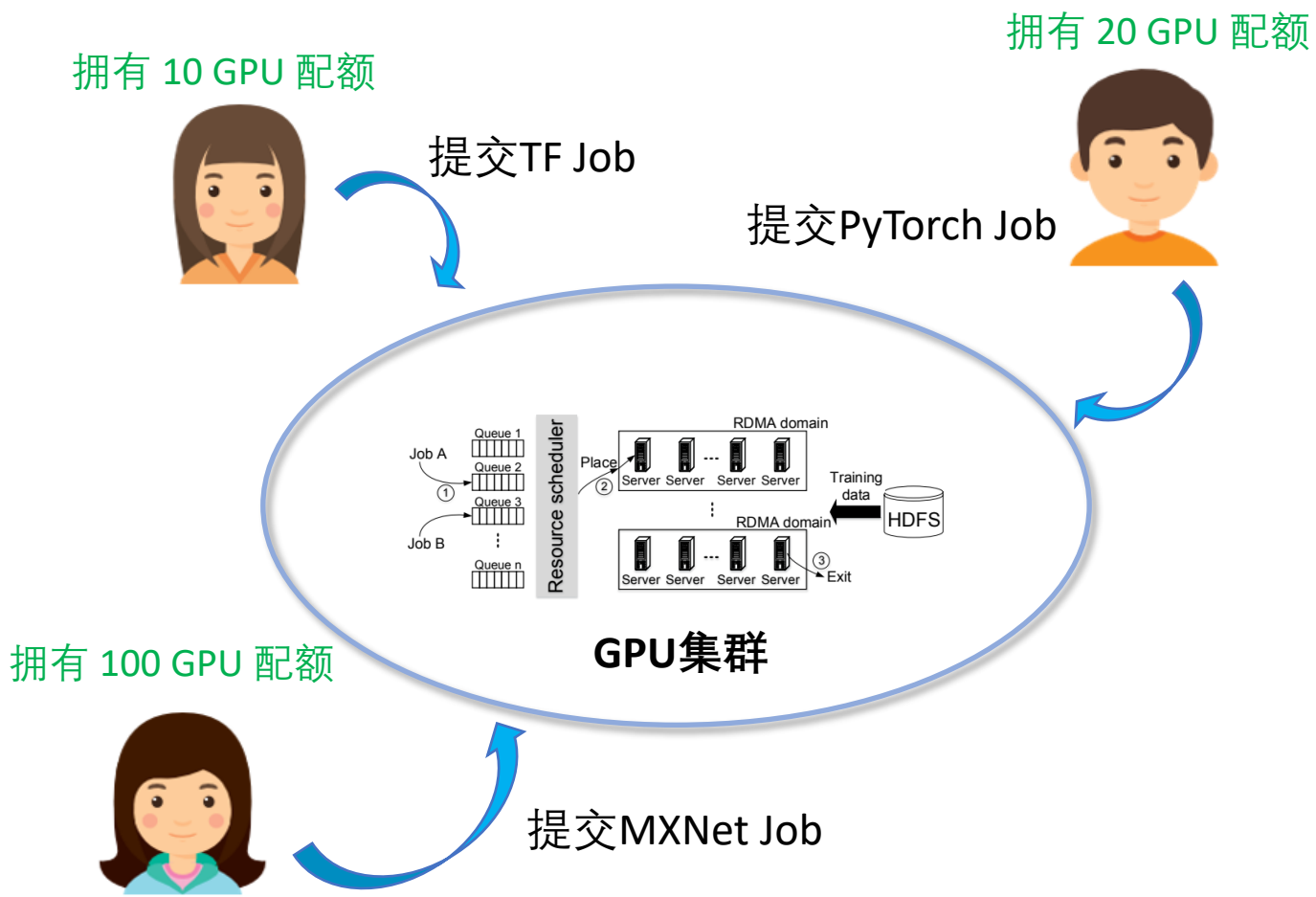
人工智能系统 System for AI

异构计算集群调度与资源管理系统
Scheduling and resource management system

课程概览

- 异构计算集群管理系统简介
- 作业，镜像与容器
- 调度
- 代表性异构计算集群管理系统

在多租(Multi-Tenant) GPU 集群运行作业(Job)



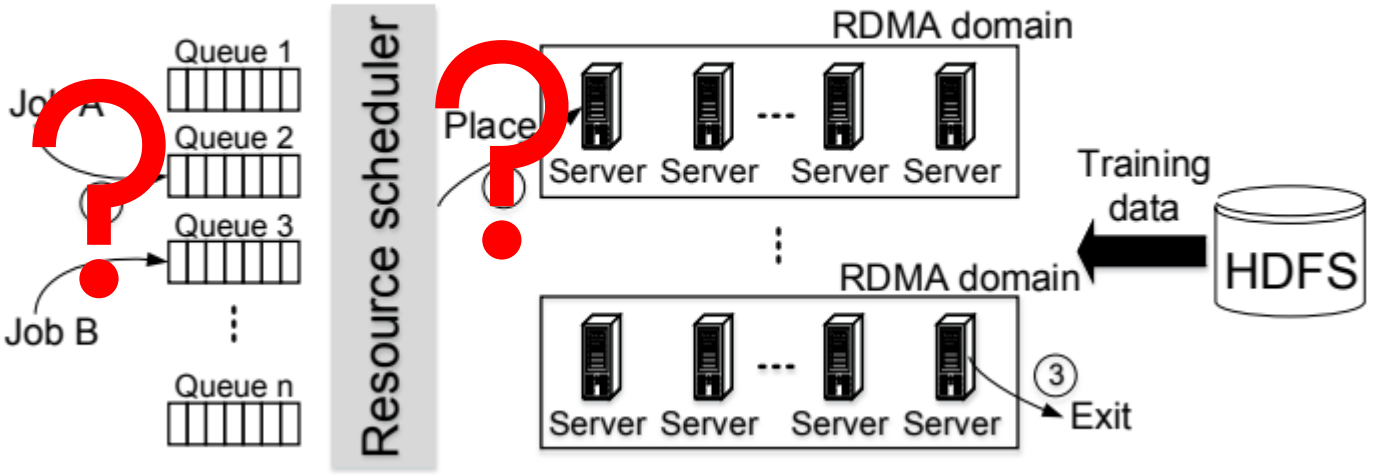
多用户**共享**多GPU服务器

- 多作业(Job)，多用户
- 作业环境需求多样
- 作业资源需求多样
- 服务器软件环境单一
- 服务器空闲资源多样

调度与资源管理系统重要性

- 提供人工智能基础架构支持
  - 深度学习作业(Job)调度(Scheduling)与管理
  - 异构硬件管理
- 提升生产力
  - 用户专注于模型创新，无需关注系统部署，管理
  - 模型，代码和数据共享，加速研究与创新

深度学习作业(Job)的生命周期



GPU集群

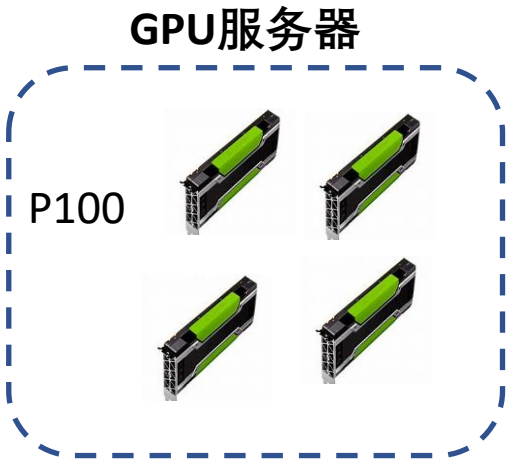
1. **作业**提交与排队
2. 作业资源**分配**与**调度**
3. 作业执行完成与释放

- 如何提交作业与解决环境依赖问题?
- 如何高效调度作业并分配资源?
- 如何将启动的作业运行时资源与命名空间隔离?

作业，镜像与容器

- 深度学习作业
- 镜像
- 容器

独占服务器执行深度学习作业



- 独占环境，无需考虑环境，资源隔离问题
- 环境依赖路径：本地/anaconda3
- GPU环境依赖：本地/usr/local/cuda
- 数据路径：本地/data
- 直接执行启动脚本：

```
python train.py --batch_size=256  --model_name=resnet50
```

作业提交到平台

```
{
    "jobName": "restnet",
    "image": "example.tensorflow:stable",
    "dataDir": "/tmp/data",
    "outputDir": "/tmp/output",
    ...
    "taskRoles": [
        {
            ...
            "taskNumber": 1,
            "cpuNumber": 8,
            "memoryMB": 32768,
            "gpuNumber": 1,
            "command": "python train.py --batch_size=256 \
                --model_name=resnet50"
        }
    ]
}
```

环境依赖

数据与代码

任务数量

作业启动命令

资源占用

提交作业

GPU集群

作业环境依赖问题

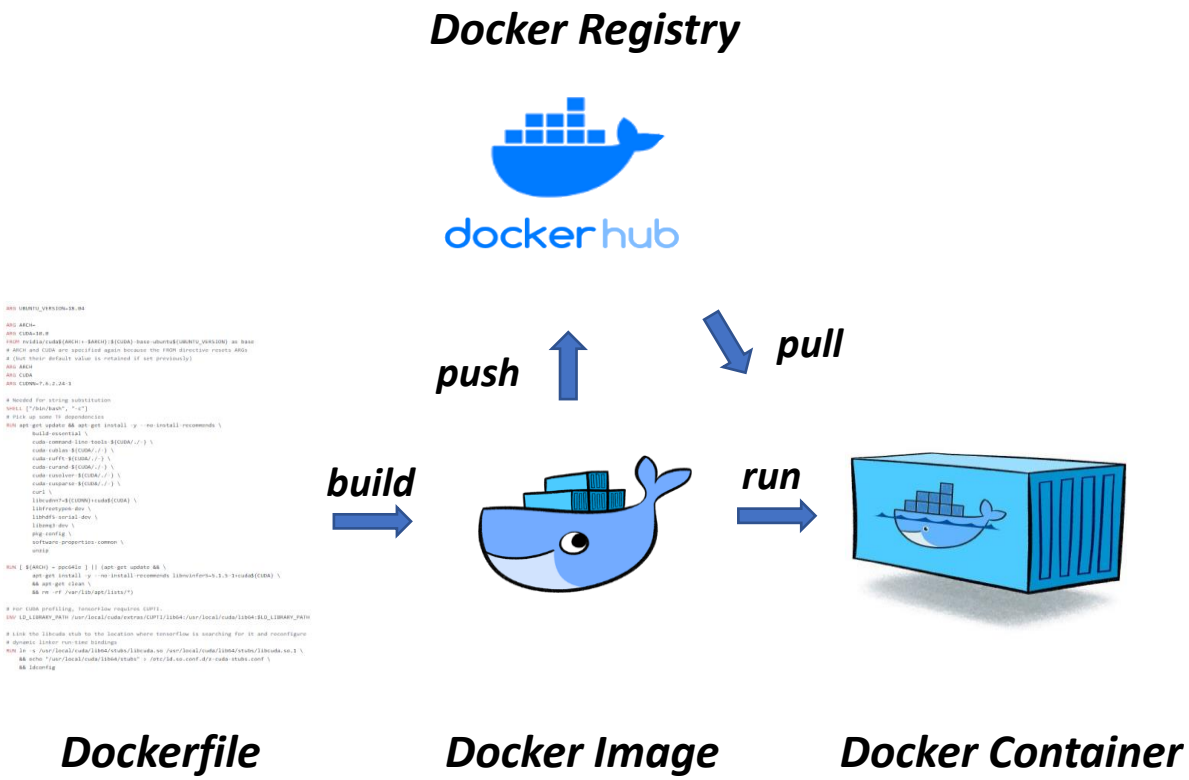
- 问题：
  - 服务器上没有预装好所需要的个性化环境?
  - 不同作业需要的框架，依赖和版本不同，安装繁琐且重复?
  - 部署服务器上可能会有大量重复安装的库，占用空间?
- 深度学习特有问题：
  - 深度学习作业需要安装CUDA依赖和深度学习框架等
- 目标：
  - 复用整体安装环境并创建新环境
  - 层级构建依赖，复用每一层级的依赖

作业运行时资源隔离问题

- 问题：
  - 集群资源被共享，如何保证作业互相之间不干扰和多占用资源？
  - 如何能够让不同作业可以运行不同的操作系统和命名空间？
  - 如何保证隔离的同时，作业启动的越快越好？
- 深度学习特有问题：
  - GPU和GPU memory如何隔离
- 目标：
  - 资源隔离
  - 轻量级启动

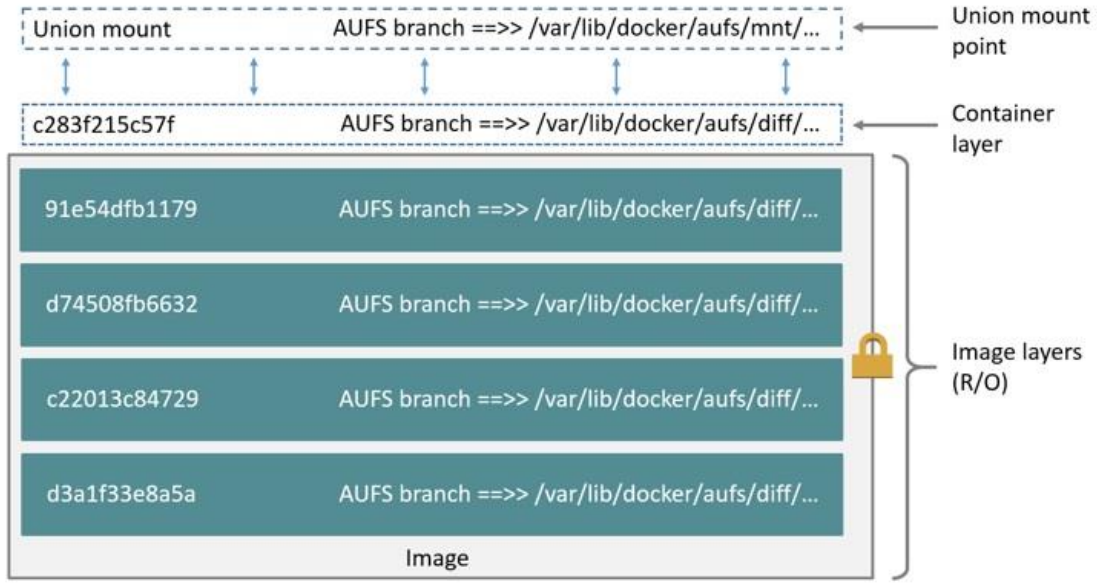
镜像与容器 - Docker

- Image
  - 打包作业环境依赖
- Registry
  - 共享镜像
- Container
  - 运行时作业资源管理与隔离



镜像 (Image)

- 含有可读和读写层
- union mount
- 支持多种文件系统
  - AUFS
  - OverlayFS
  - …



容器镜像层级关系

Dockerfile实例

```
FROM nvidia/cuda:10.0-cudnn7-devel-ubuntu16.04
ARG PYTHON_VERSION=3.6
…
RUN apt-get update && apt-get install -y --no-install-recommends \
…
RUN curl -o ~/miniconda.sh
…
 /opt/conda/bin/conda install -y -c pytorch magma-cuda100 && \
…
WORKDIR /opt/pytorch
COPY . .
…
WORKDIR /workspace
RUN chmod -R a+w .
```

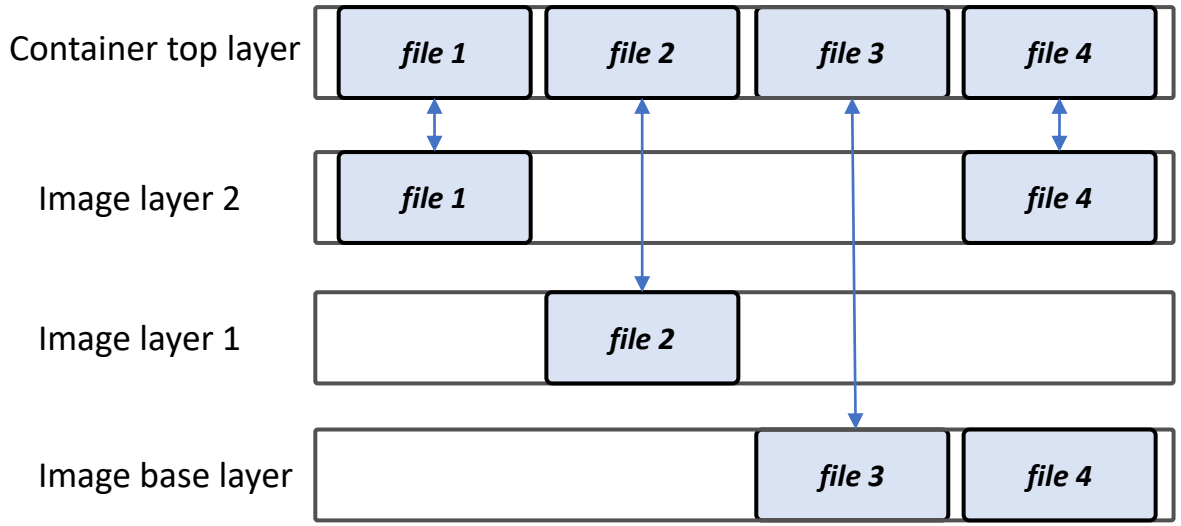
…

https://github.com/pytorch/pytorch/blob/build/1.3.0/docker/pytorch/Dockerfile

PyTorch Docker Image 实例

```
IMAGE          CREATED             CREATED BY                                      SIZE
ba2da111b833   7 weeks ago         /bin/sh -c chmod -R a+w /workspace              0B
<missing>      7 weeks ago         /bin/sh -c #(nop) WORKDIR /workspace            0B
<missing>      7 weeks ago         /bin/sh -c #(nop)  ENV LD_LIBRARY_PATH=/usr/…   0B
<missing>      7 weeks ago         /bin/sh -c #(nop)  ENV NVIDIA_DRIVER_CAPABIL…   0B
<missing>      7 weeks ago         /bin/sh -c #(nop)  ENV NVIDIA_VISIBLE_DEVICE…   0B
<missing>      7 weeks ago         /bin/sh -c conda install pytorch torchvision…   2.85GB
<missing>      7 weeks ago         /bin/sh -c #(nop)  ENV PATH=/opt/conda/bin:/…   0B
<missing>      7 weeks ago         /bin/sh -c curl -v -o ~/miniconda.sh -O  htt…   1.13GB
<missing>      7 weeks ago         /bin/sh -c #(nop)  ENV PYTHON_VERSION=3.6       0B
<missing>      7 weeks ago         /bin/sh -c apt-get update && apt-get install…   215MB
<missing>      7 weeks ago         /bin/sh -c #(nop)  LABEL com.nvidia.volumes.…   0B
<missing>      2 months ago        /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      2 months ago        /bin/sh -c mkdir -p /run/systemd && echo 'do…   7B
<missing>      2 months ago        /bin/sh -c set -xe   && echo '#!/bin/sh' > /…   745B
<missing>      2 months ago        /bin/sh -c rm -rf /var/lib/apt/lists/*          0B
<missing>      2 months ago        /bin/sh -c #(nop) ADD file:a5b5bea2fa5358461…   121MB
```

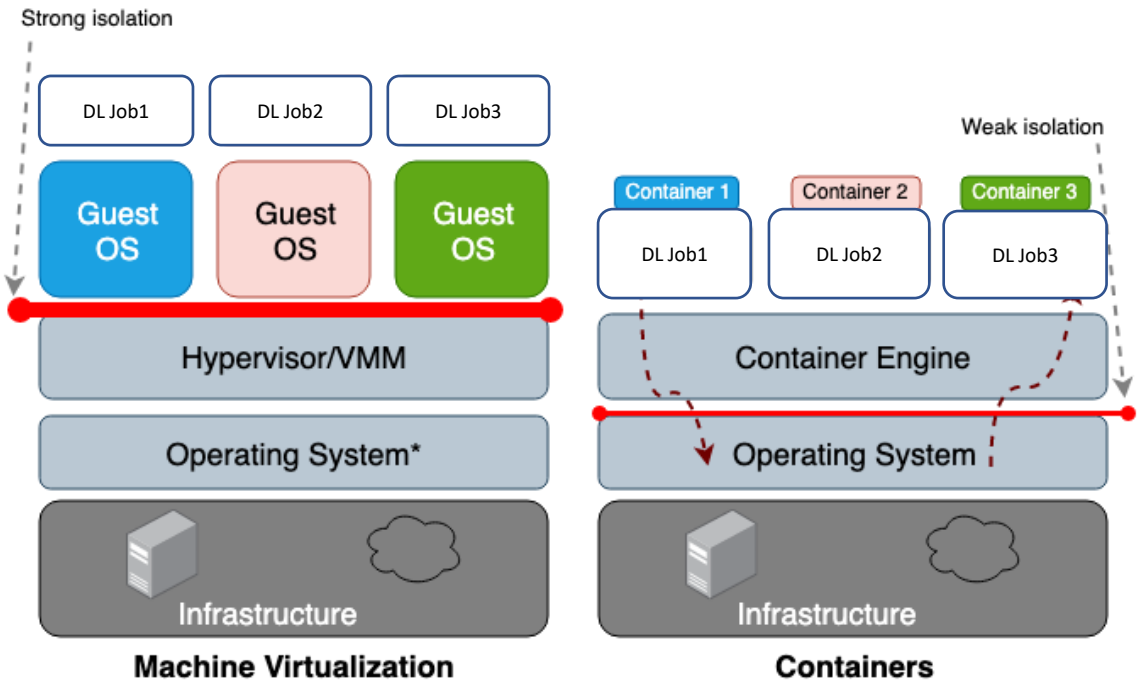
镜像文件系统实例



AUFS 存储驱动实例

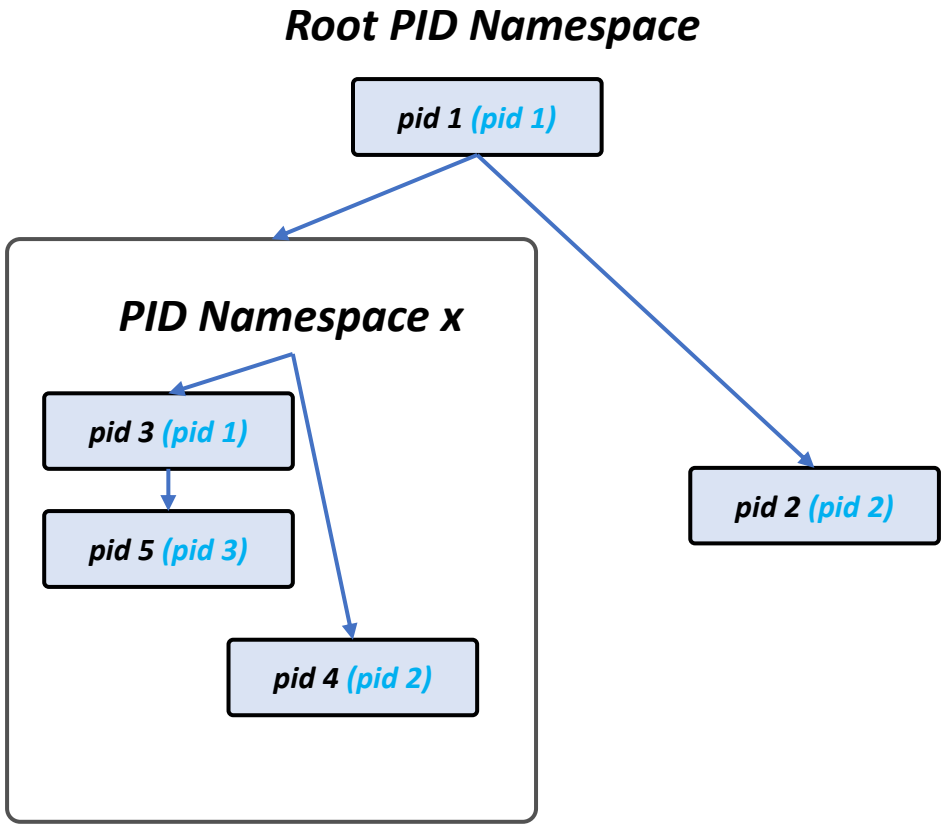
容器 (Container)

- Wiki定义:
  - Linux containers are implementations of operating system-level virtualization for the Linux operating system.
- 支撑技术:
  - cgroup, namespace



命名空间(Namespace)

- 命名空间隔离
- 类型
  - pid
  - net
  - mnt
  - ipc
  - user
  - …

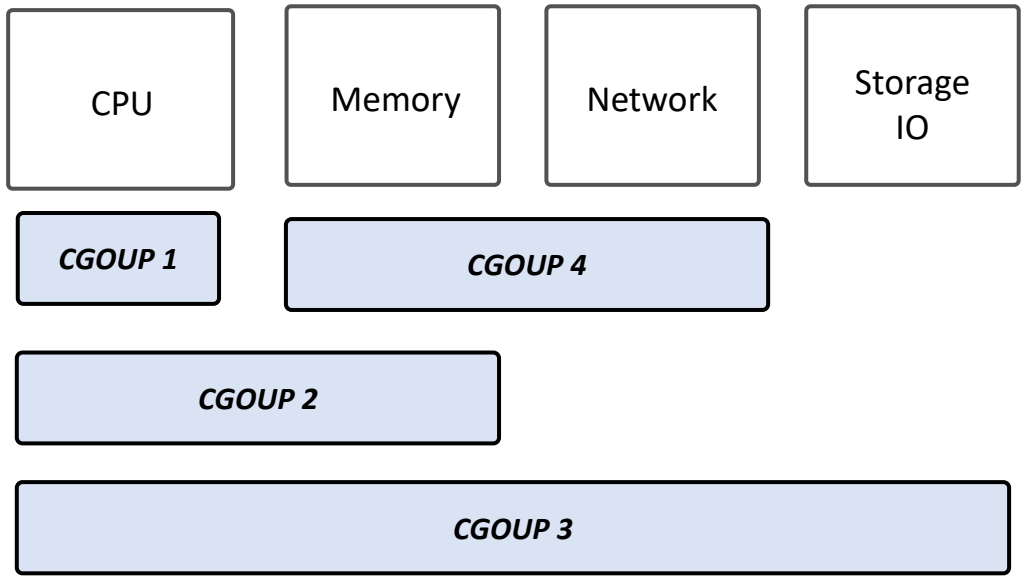


black: Real PID

blue: getid() to get this PID

控制组(Control Group)

- Linux通过cgroup对进程组的资源：
  - 控制 (control)
  - 计数 (accounting)
  - 隔离 (isolation)
- 类型：
  - cpu
  - memory
  - block i/o
  - network
  - …

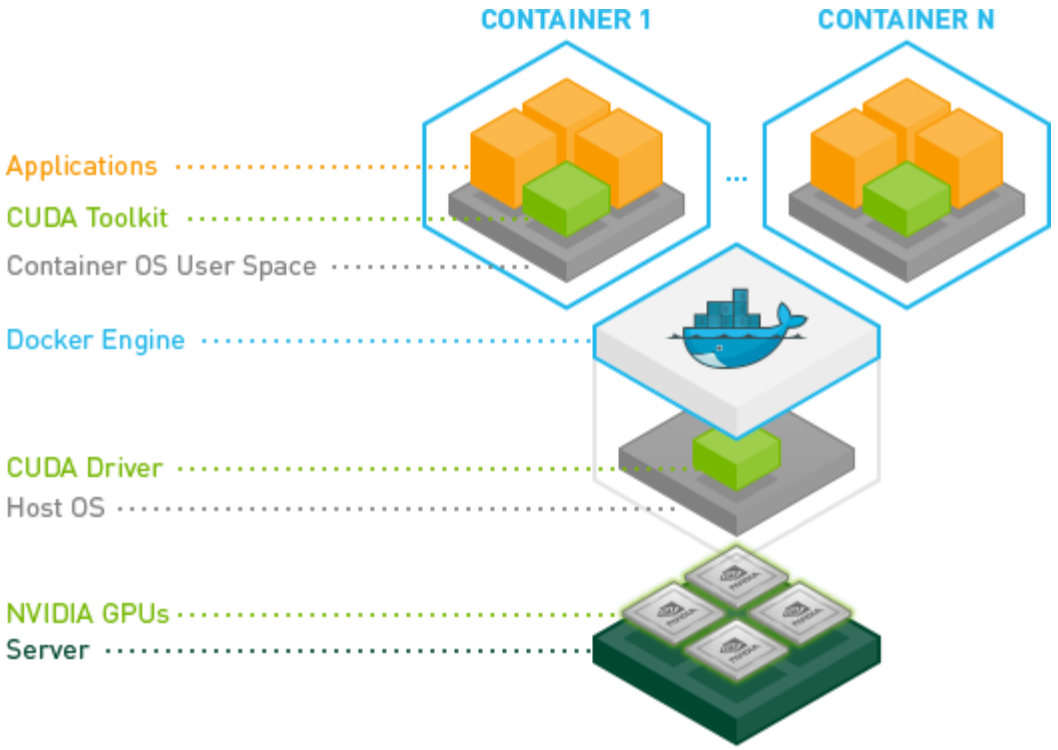


容器资源分配与隔离子系统实例

Subsystem	Type	Description
cpuset	isolation	Confine processes to processor and memory node subsets
ns	isolation	For showing private view (namespace) of system to processes in cgroup
cpu	control	Share CPU bandwidth between groups
cpuacct	accounting	The CPU Accounting (cpuacct) subsystem generates automatic reports on CPU resources
memory	control	The memory controller supports reporting and limiting of process memory, kernel memory, and swap used by cgroups.
devices	isolation	This supports controlling which processes may create (mknod) devices as well as open them for reading or writing.
rdma	control	The RDMA controller permits limiting the use of RDMA/IB-spe-cific resources per cgroup.
blk_io	control	The blkio cgroup controls and limits access to specified block devices by applying IO control

GPU资源分配与隔离 - Nvidia Docker

- 功能：GPU粒度的隔离
- 问题：
  - 无法像传统OS对CPU时分复用隔离
  - GPU内存无法隔离
- 潜在解决方法：
  - Nvidia MPS， 当前还未集成进行nvidia docker



Nvidia Docker实例

```
# Test nvidia-smi with the latest official CUDA image
$ docker run --gpus all nvidia/cuda:9.0-base nvidia-smi

# Start a GPU enabled container on two GPUs
$ docker run --gpus 2 nvidia/cuda:9.0-base nvidia-smi

# Starting a GPU enabled container on specific GPUs
$ docker run --gpus '"device=1,2"' nvidia/cuda:9.0-base nvidia-smi
```

小结与讨论

- 容器与镜像解决了环境依赖，资源隔离进而奠定未来调度系统多租的基石
- 相比传统OS，在GPU技术栈还不完善的功能是？

调度

- 群调度 (Gang Scheduling)
- DRF调度 (Dominant Resource Fairness Scheduling)
- 容量调度 (Capacity Scheduling)
- 抢占 (Preemption)
- 前沿调度算法

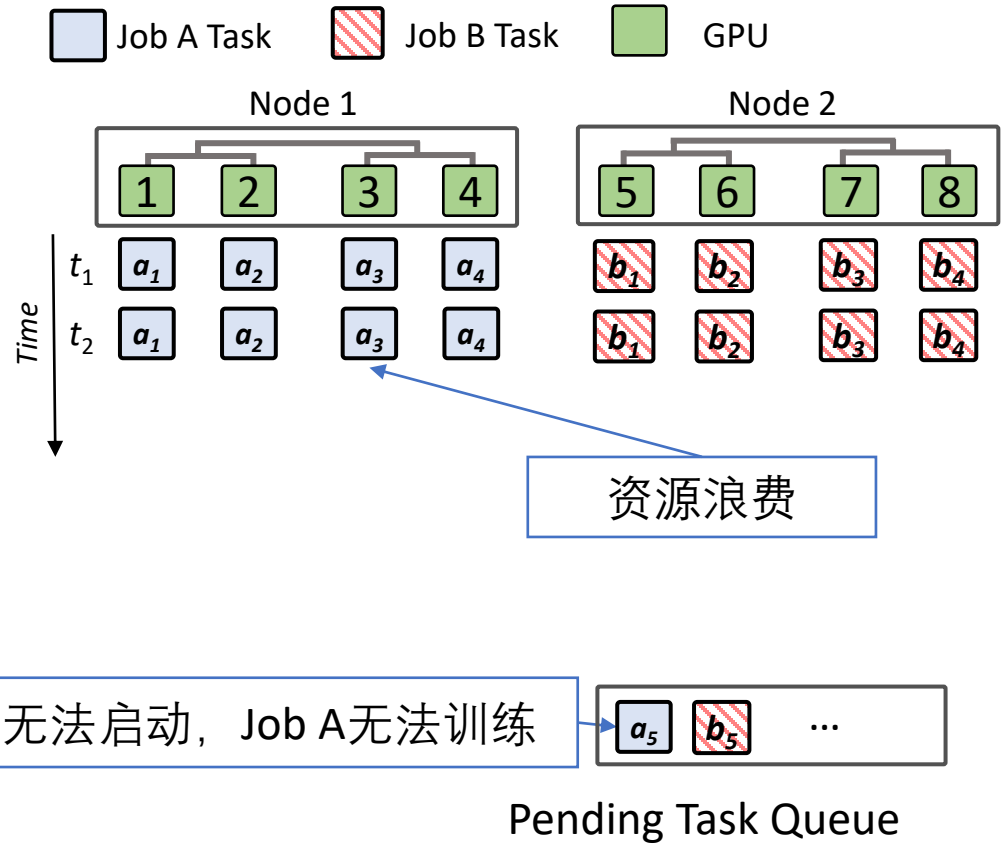
调度性能指标(metrics)

针对一批作业调度，常常考虑以下指标：

- 吞吐(Throughput)
- 完工时间(Makespan) / 平均响应时间(Average Response Time)
- 公平(Fairness)
- 利用率(Utilization)
- 服务等级协议(SLA)

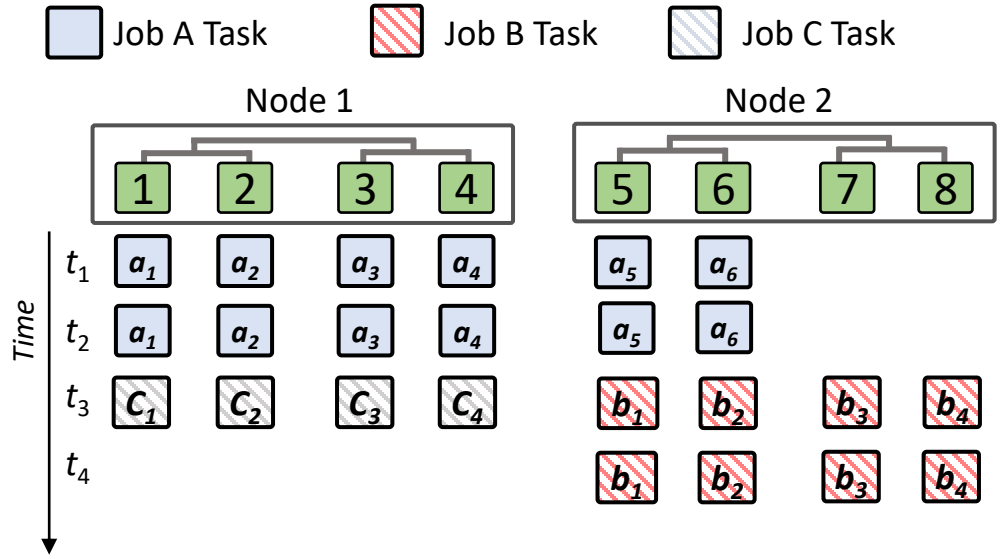
调度并行或分布式作业会有哪些问题?

- 问题：并行作业可以同时执行多个任务，如果有依赖任务没启动，已启动任务会在同步点忙于等待或者频繁上下文切换 (如右图)：
  - 无法训练：等待不能启动的任务
  - 资源浪费：已启动的任务造成资源浪费
- 深度学习训练特点：
  - 深度学习作业任务需要同步梯度
  - GPU软件栈无法支持任务上下文切换
- 目标：High Throughput, High Utilization and Short Response Times



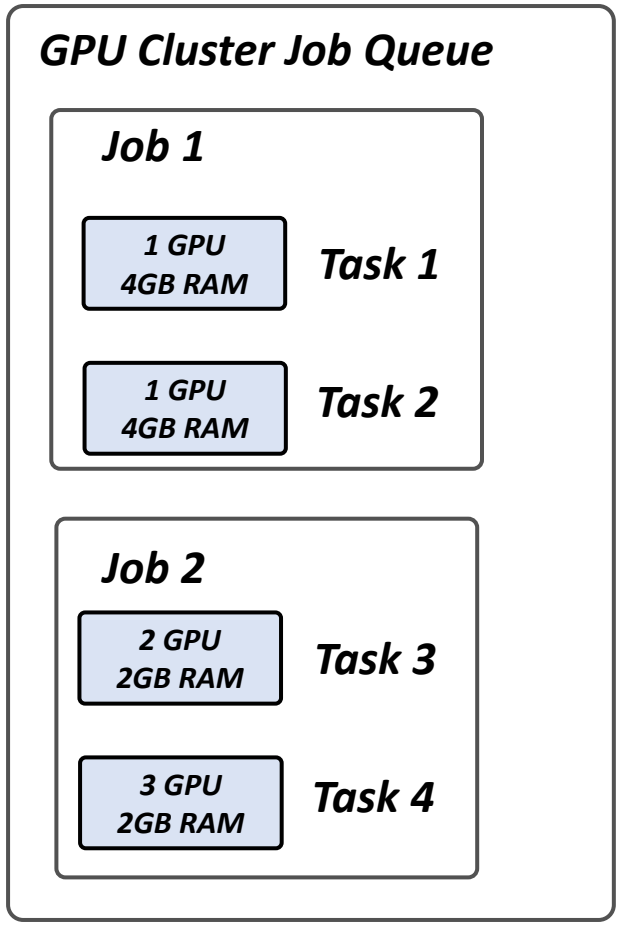
群调度 (Gang Scheduling)

- Wiki定义:
  - A scheduling algorithm for parallel systems that schedules related threads or processes to run simultaneously on different processors.
- 策略:
  - 同时启动深度学习任务进程



异构资源如何公平(Fairness)调度?

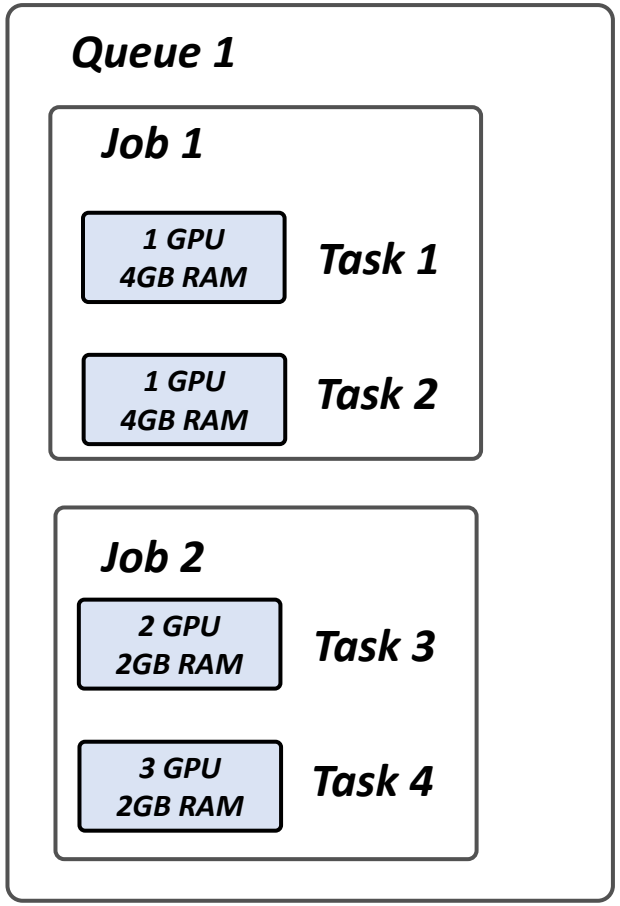
- 问题：包含**异构资源类型**的系统中如何进行**多作业公平**(Fairness)的资源调度?
- 挑战：相比传统单资源公平调度，深度学习作业也需要使用多种异构资源 (CPU, Host memory, etc.)，并且需要调度GPU及GPU memory
- 设计目标：吞吐(Throughput)与公平(Fairness)



Cluster Resources: [10 GPU, 20GB RAM …]

Dominant Resource Fairness (DRF)

- 优化目标:
  - DRF 尝试最大化系统中的最小主导份额(**smallest dominant share**)
- 策略:
  - 通过同类型资源在集群整体资源中的份额确定**主导资源 (dominant resource)**
  - 基于最大最小公平（**max-min fairness**）的针对**多资源类型（e.g. GPU, CPU）**的调度算法



Cluster Resources: [10 GPU, 20GB RAM]

Job 1:
Total GPU 1+1 = 2 GPU
GPU Share = 2/10 = 0.2
Total Memory 4+4 = 8GB
Memory Share = 8/20 = 0.4

SHARE = 0.4 [Dominant resource is Memory]

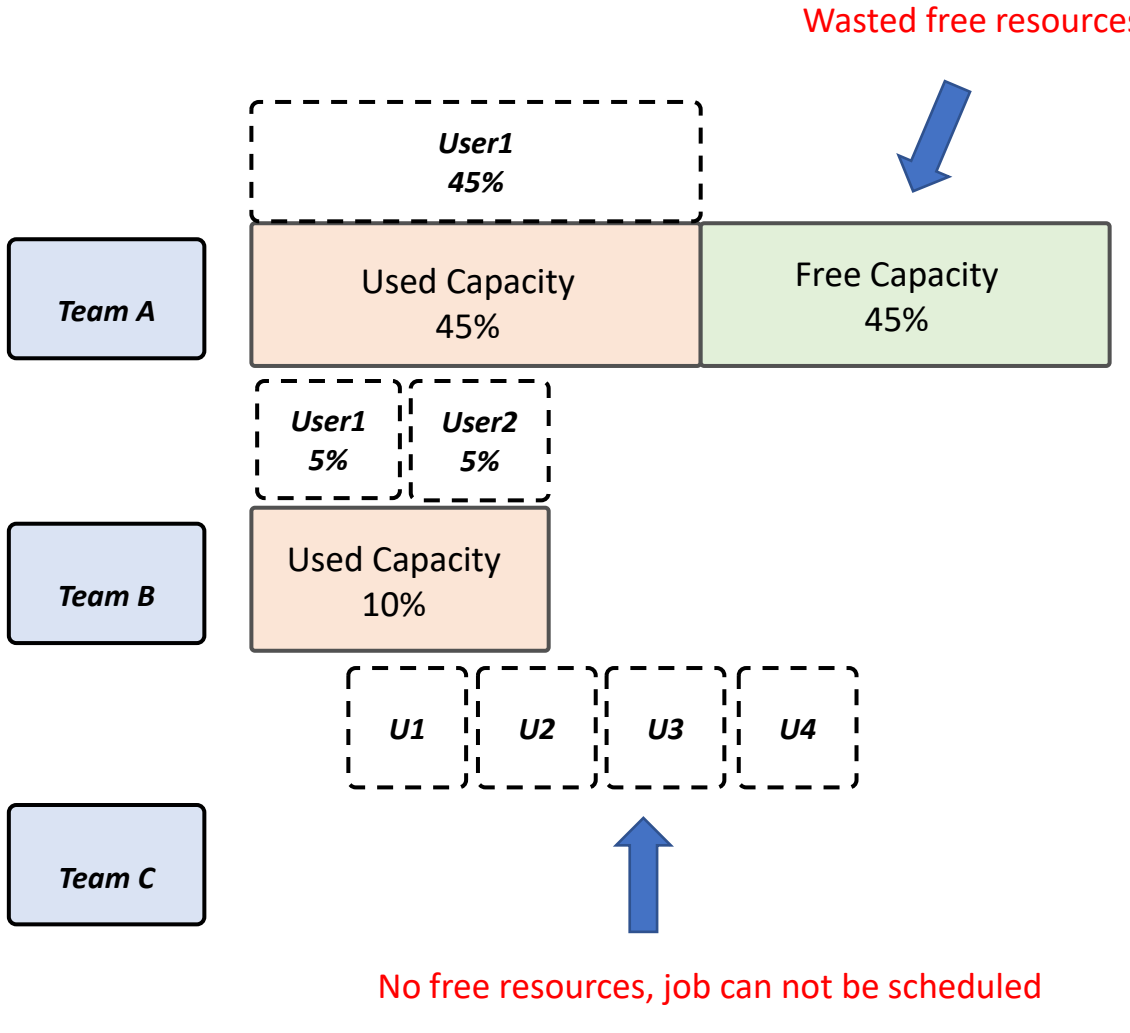
Job 2:
Total GPU 2+3 = 5 GPU
GPU Share = 5/10 = 0.5
Total Memory 2+2 = 4GB
Memory Share = 4/20 = 0.2

SHARE = 0.5 [Dominant resource is GPU]

Job 1 has higher priority than Job 2 as Job 1 share 0.4 is less than Job2 share 0.5

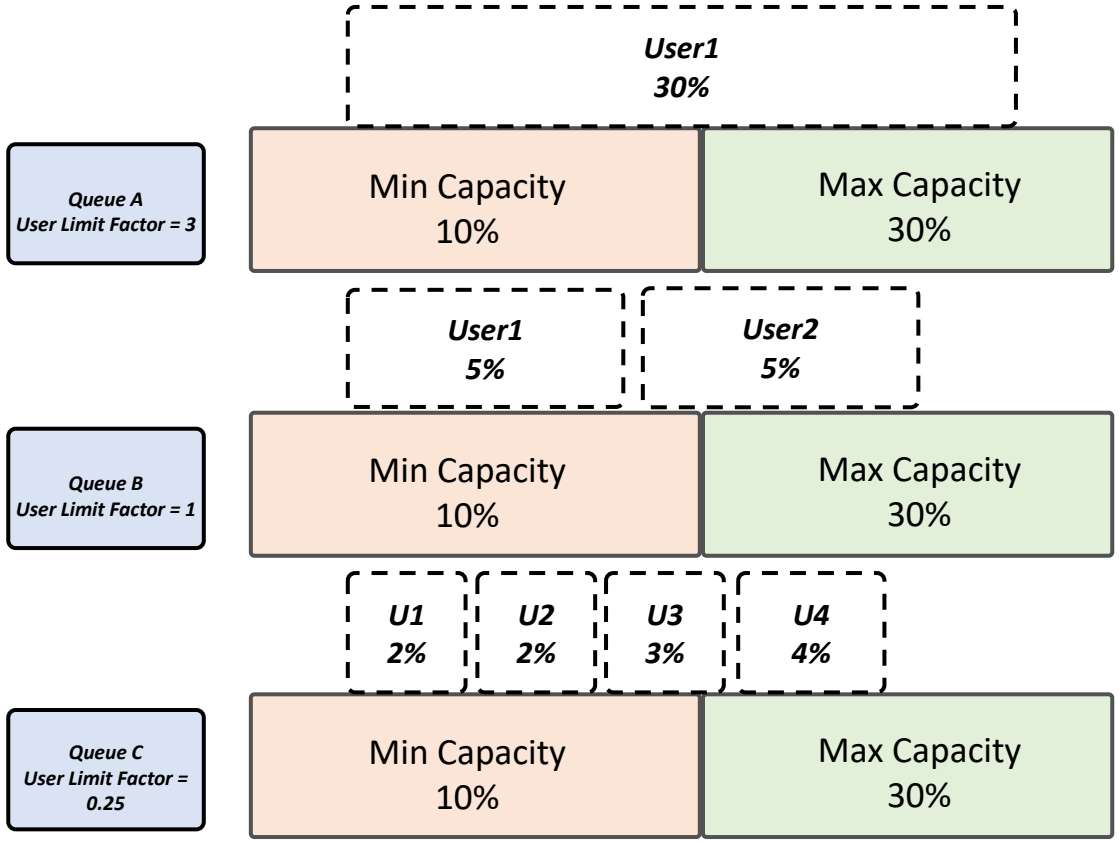
如何让多个小组共享集群?

- 问题:
  - 能否为多个组织共享集群资源?
  - 共享集群资源的同时，如何同时为每个组织提供最小容量保证?
  - 空闲资源能否弹性为其他组织利用?
- 挑战：相比传统容量调度调度，深度学习作业也需要考虑调度GPU及GPU memory
- 设计目标：Utilization, Fairness and SLA



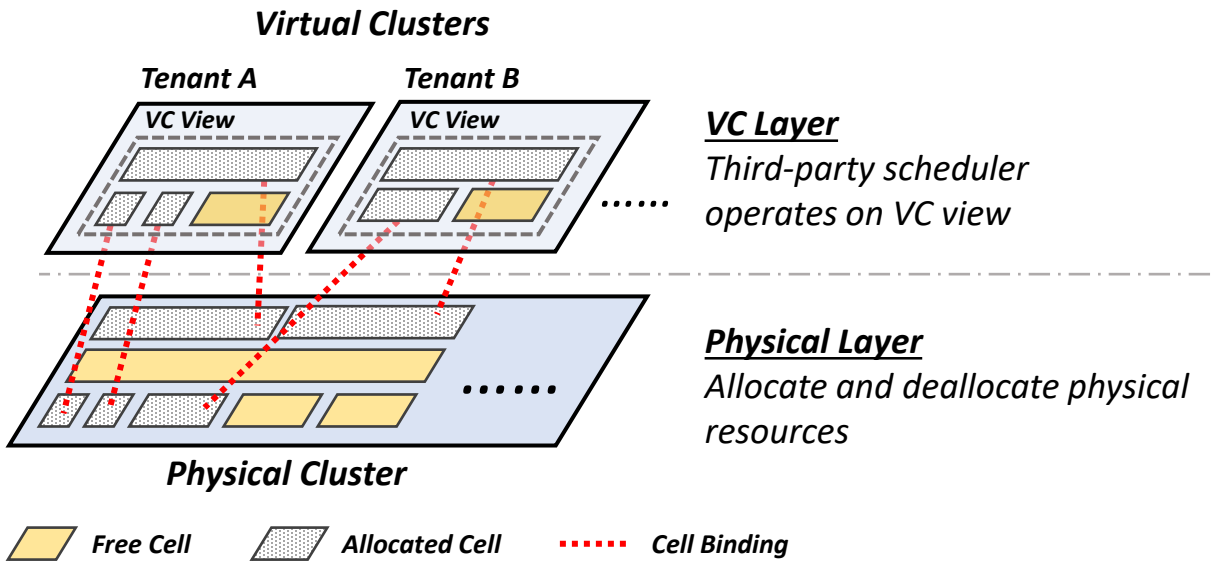
容量调度 (Capacity Scheduling)

- 目的:
  - 支持多租(Multi-tenant）资源共享
- 策略:
  - Utilization:
    - 虚拟集群 (Virtual Cluster）
    - Bonus Resource
  - Fairness:
    - Dominant Resource Fairness (DRF)
    - User Limit Factor：控制单用户的可以消耗的最大资源
  - SLA
    - Preemption



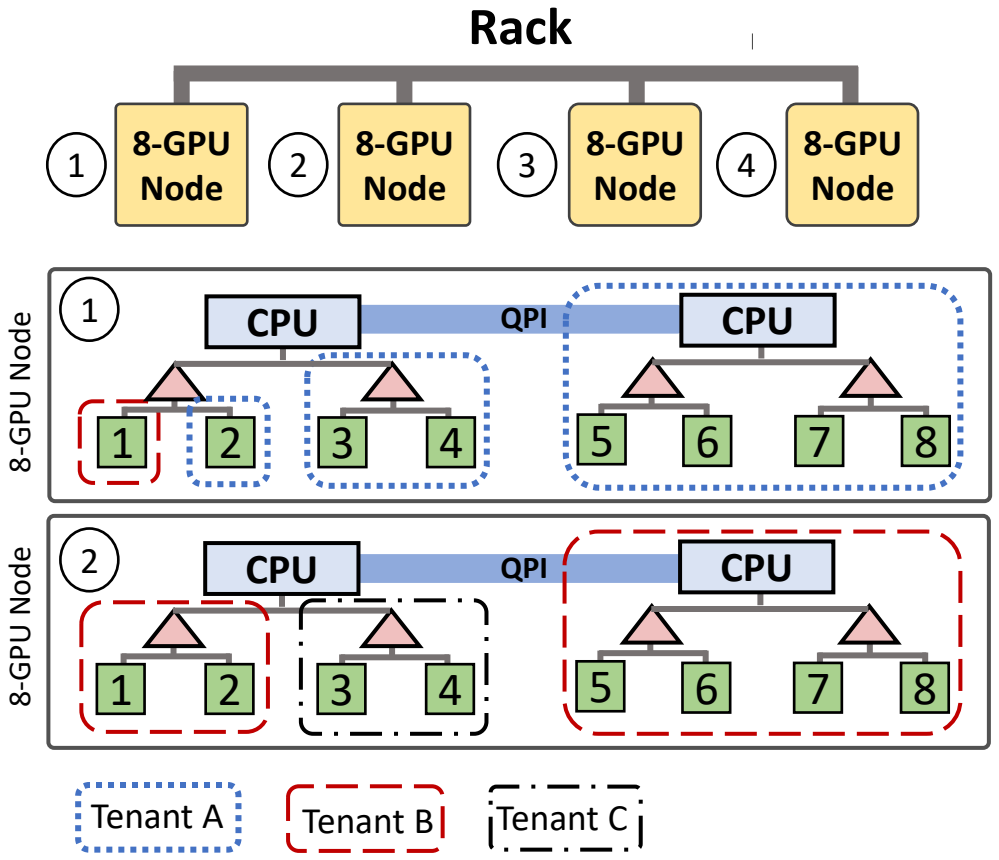
虚拟集群 (Virtual Cluster) 映射

- 虚拟集群根据小组的配额进行定义
- 将虚拟集群映射到物理集群



虚拟集群实例

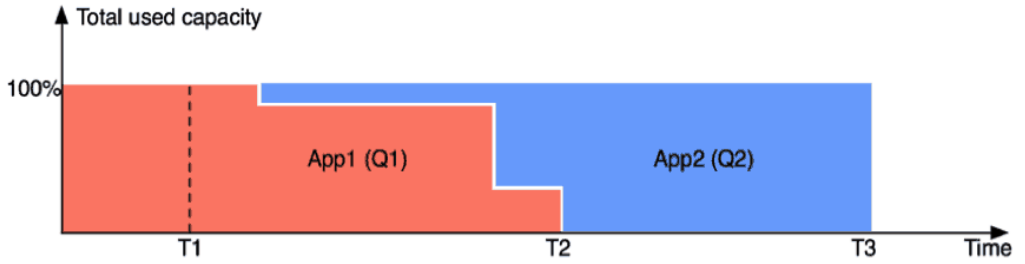
- 资源被分配给租户(Tenants)
- 每个Tenant构成了一个虚拟集群(VC)



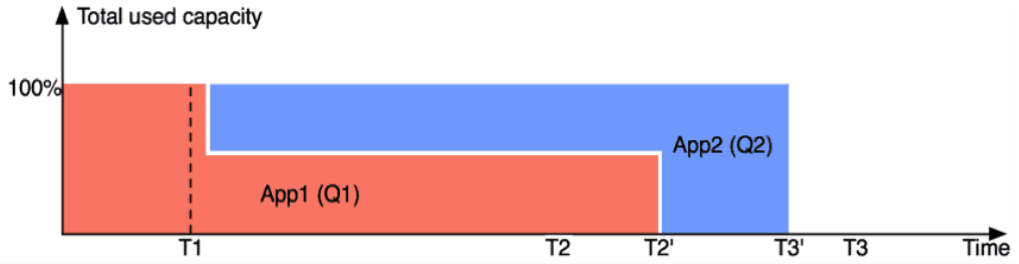
如何让调度兼顾SLA?

- 问题：
  - 共享集群资源被其他组织占用提升了资源利用率
    但是无法保证SLA?
- 挑战：
  - 在深度学习作业下，被抢占的作业当前只能失败，
    无法像传统OS上下文切换
- 设计目标：
  - 兼顾有限资源利用和服务等级协议（SLA）

不使用抢占调度



使用抢占调度

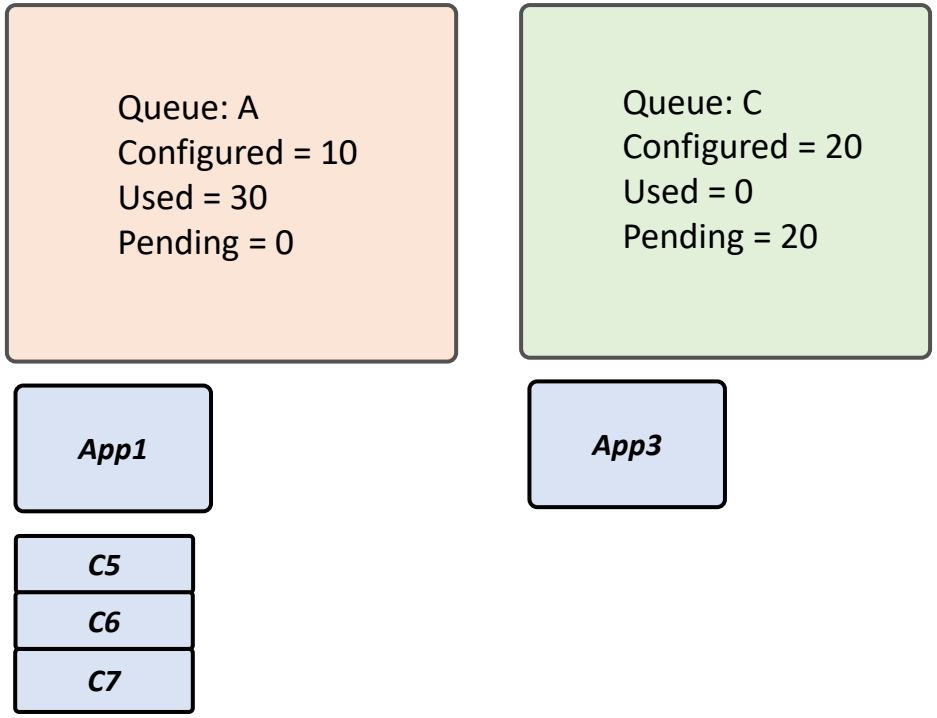


- App2更快结束
- App1的变慢可接受，因初始已经获得更多资源

YARN抢占 (Preemption) 实例

策略:

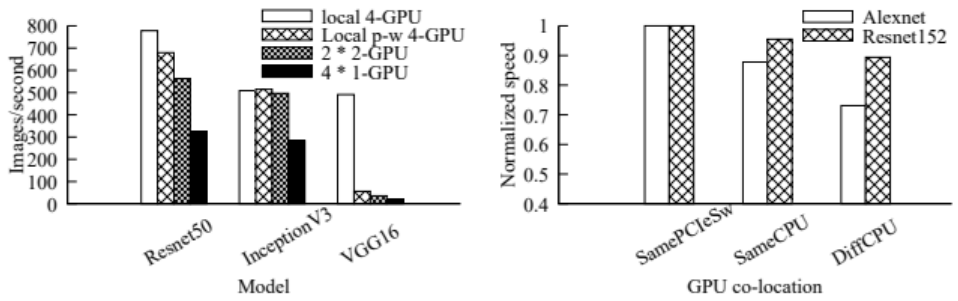
- Step 1: 从过度使用的队列中获取需要被抢占的容器
- Step 2: 通知ApplicationMasters即将触发抢占
- Step 3: 等待直到被抢占终止运行



App1的容器C6, C7 被标记为可以被抢占

GPU的拓扑结构影响多GPU作业性能?

- 多GPU深度学习作业问题与挑战：
  - 受到GPU拓扑结构影响
  - 受到服务器上同时运行作业的干扰
- 目标：Makespan, Response Time

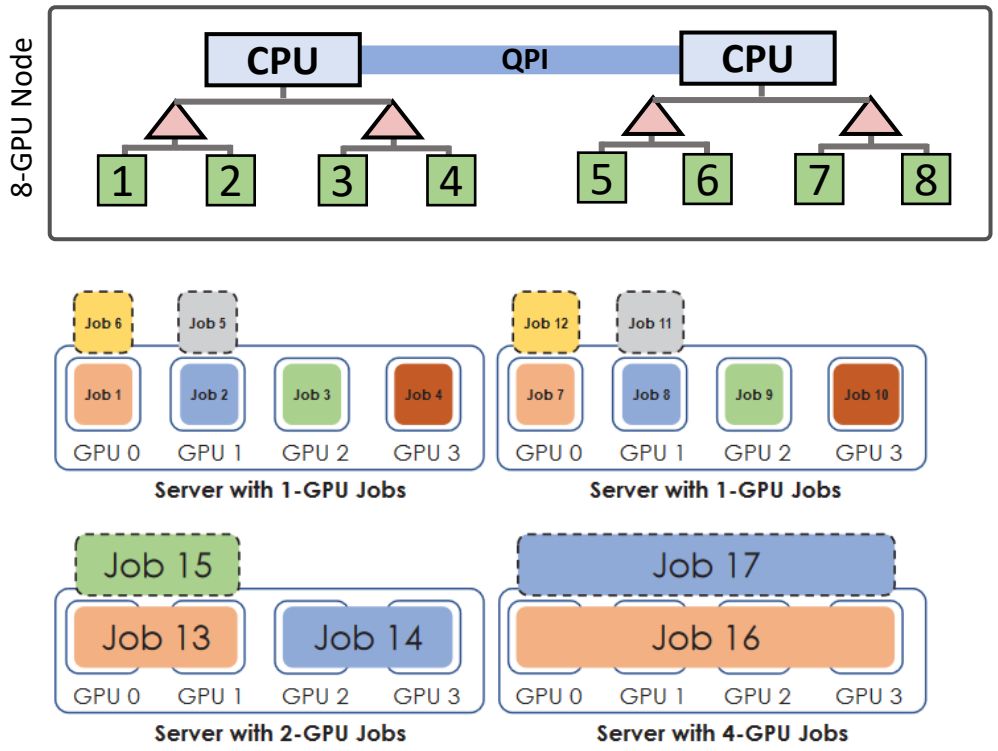


(a) Inter-server locality. (b) Intra-server locality.

Figure 1: Mutli-GPU performance with locality setting.

Gandiva 拓扑感知调度

Reactive Mode: 考虑GPU 拓扑（topology）和亲和性（affinity）的调度



Algorithm 1 getNodes(in job, out nodes)

```
1:  nodes0 ← findNodes(job.gpu, affinity ← job.gpu)
2:  nodes1 ← minLoadNodes(node0)
3:  nodes2 ← findNodes(jog.gpu, affinity ← 0)
4:  nodes3 ← findNodes(job.gpu)
5:  if nodes1 and height(nodes1) < 1:
6:      return nodes1              // Same affinity with free GPUs
7:  if nodes2 and numGPUs(nodes2) ≥ job.gpu:
8:      return nodes2              // Unallocated GPU servers
9:  if nodes3:
10:     return nodes3              // Relax affinity constraint
11: elif nodes1:
12:     return nodes1              // Allow over-subscription
13: else:
14:     enqueue(job)               // Job queued
```

深度学习还有其他问题影响调度?

- 深度学习作业特点：
  - 超参调优等多作业同时运行，希望能及早获得反馈
  - 作业资源占用多样，造成资源分配后容易形成低利用率和资源碎片
- 挑战：GPU软件栈对GPU状态备份，资源隔离做的不够完善，无法很好的支持时分复用(time slicing)，装箱(packing)等传统OS功能，难以支持早反馈和高利用率
- 目标：Response Time, Utilization

Gandiva Introspective Mode目标与策略

- 早反馈(Early feedback):
  - time slicing: micro-tasks (mini-batch), suspend-resume
  - over-subscription: allocating GPUs to a new job immediately
- 高效集群(Cluster efficiency):
  - packing: profiling
  - migration: locality
  - grow-shrink

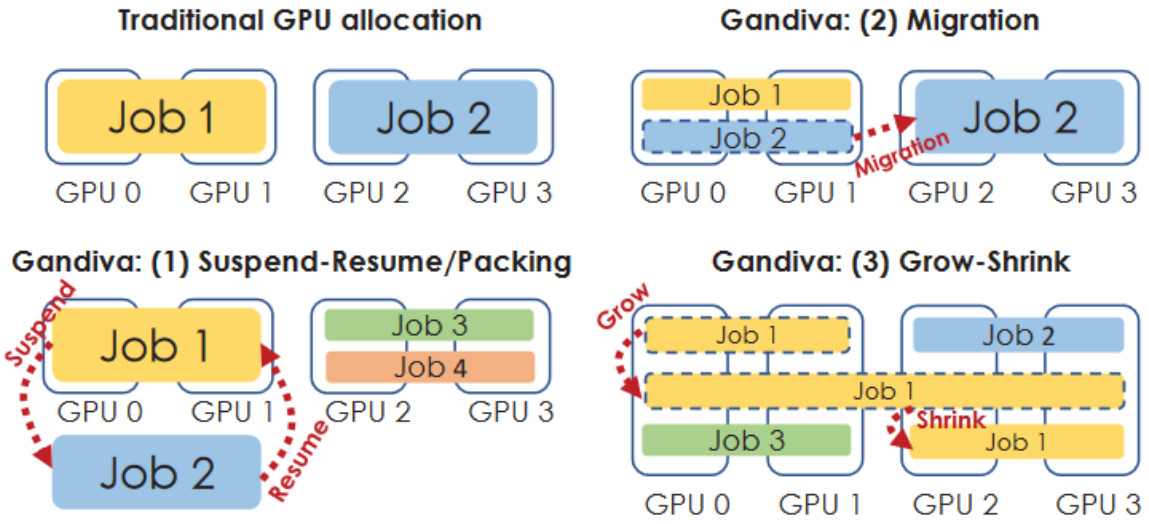


Figure 6: GPU usage options in Gandiva.

小结与讨论

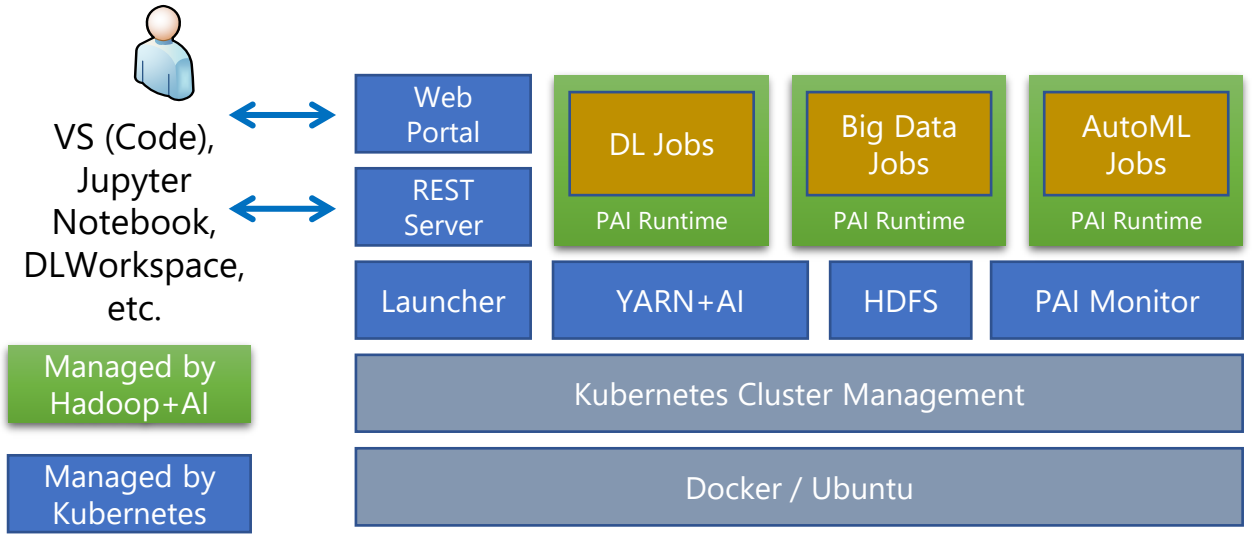
- 不同调度算法为解决不同的问题和目标而优化，对比不同算法的侧重点
- 如果在异构集群环境下，管理员倾向于组间的资源使用的公平性，会倾向于选择哪种调度算法?

代表性异构计算集群管理系统简介

- Kubeflow
- OpenPAI
- …

OpenPAI

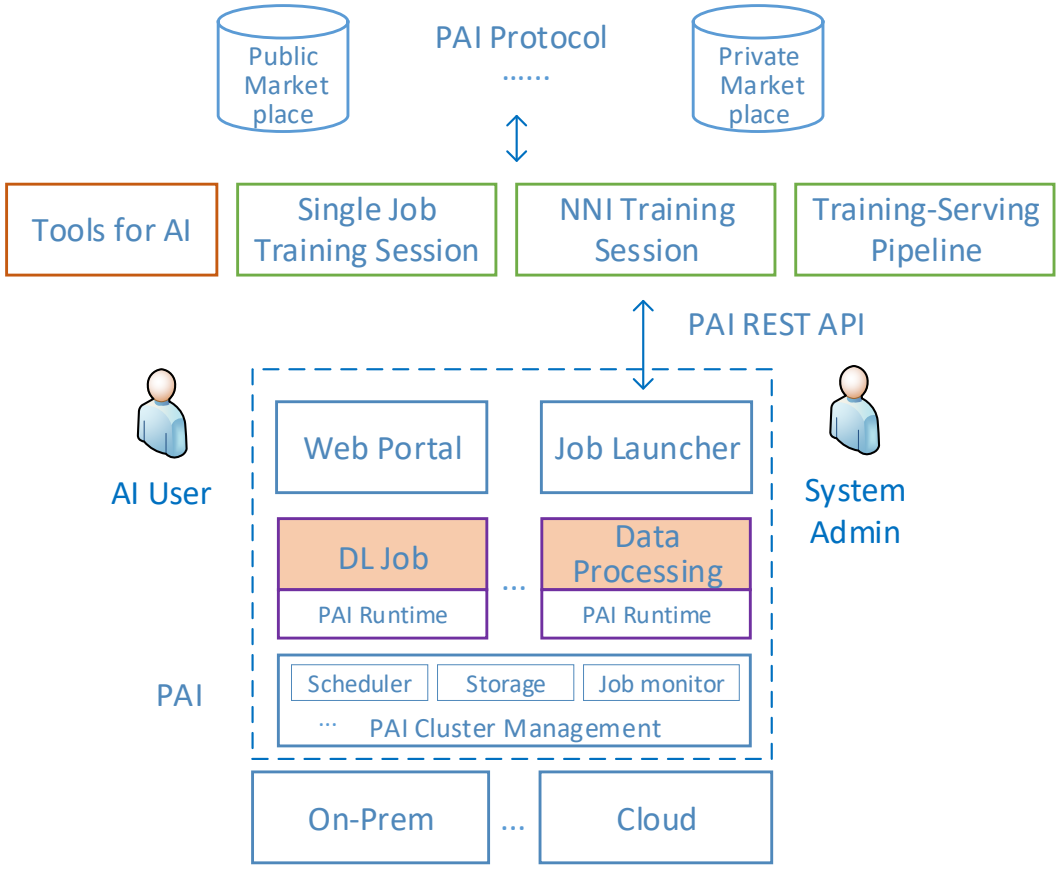
- **开源**
  - 利用YARN可扩展性强，成熟度高
  - 利用K8s 当前流行的服务部署模式
- **可扩展**: 支持所有 Deep Learning frameworks, GPU/FPGA/ASIC
- **模块化**: 微服务，组件模块化 (Storage, Scheduler)
- **高效**: 细粒度GPU 调度, 支持 IB/RDMA
- **可管理**：作业和平台的监控，部署和升级
- **鲁棒性**: 容错
- **云原生**: 支持本地和云部署



https://github.com/Microsoft/pai

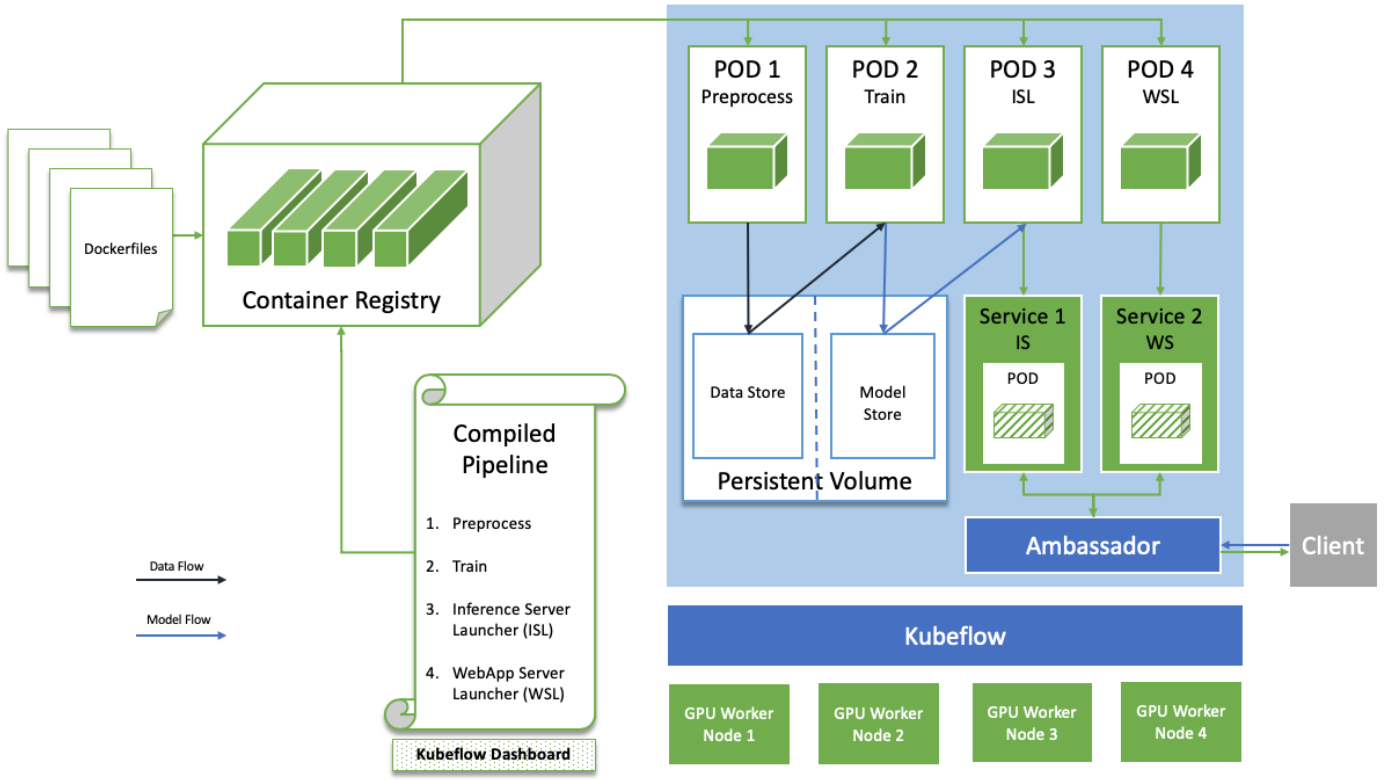
OpenPAI 生态系统

- PAI 市场(Marketplace)
  - 共享代码，模型等
- PAI 协议(Protocol)
  - Data, Code, Docker Image
  - 硬件需求
- 作业启动器
  - 理解PAI协议并执行作业
  - 可以支持更多类型的负载
- 可以部署到多种环境
  - Single-box, Cloud, On-Prem, Hybrid



Kubeflow

- k8s**原生支持**的AI平台
  - 通过k8s管理作业
- 支持多种机器学习框架:
  - TensorFlow, PyTorch等
- 支持多种硬件:
  - GPU, TPU等



https://github.com/kubeflow/kubeflow

小结

- 异构资源管理系统构建了分布式训练环境下的操作系统
- 异构硬件资源支持，灵活多样的作业，功能不完善的软件栈支持，造成系统设计挑战较大

参考文献

- Multi-tenant GPU Clusters for Deep LearningWorkloads: Analysis and Implications
- Gandiva: Introspective Cluster Scheduling for Deep Learning
- Dominant Resource Fairness: Fair Allocation of Multiple Resource Types
- All You Need to Know about Scheduling Deep Learning Jobs
- https://github.com/microsoft/pai
- https://www.kubeflow.org/
- Kubeflow Pipelines With GPUs
- YARN – The Capacity Scheduler
- Better SLAs via Resource-preemption in YARN’s Capacity Scheduler
- Kube-Batch: Dominant Resource Fairness (DRF)
- Linux Programmer's Manual CGROUPS(7)
- https://github.com/NVIDIA/nvidia-docker
- Use the AUFS storage driver
- Making Containers More Isolated: An Overview of Sandboxed Container Technologies

谢谢！

Microsoft