KubeDL 加入 CNCF Sandbox，加速 AI 产业云原生化

原创

阿里云开发者 于 2021-08-19 12:02:54 发布 116 收藏 1

文章标签： 机器学习/深度学习 存储 人工智能 Kubernetes Cloud Native 算法 调度 算法框架/工具 异构计算 容器 阿里云开发者

简介：2021 年 6 月 23 日，云原生计算基金会（CNCF）宣布通过全球 TOC 投票接纳 KubeDL 成为 CNCF Sandbox 项目。KubeDL 是阿里开源的基于 Kubernetes 的 AI 工作负载管理框架，取自"Kubernetes-Deep-Learning"的缩写，希望能够依托阿里巴巴的场景，将大规模机器学习作业调度与管理的经验反哺社区。

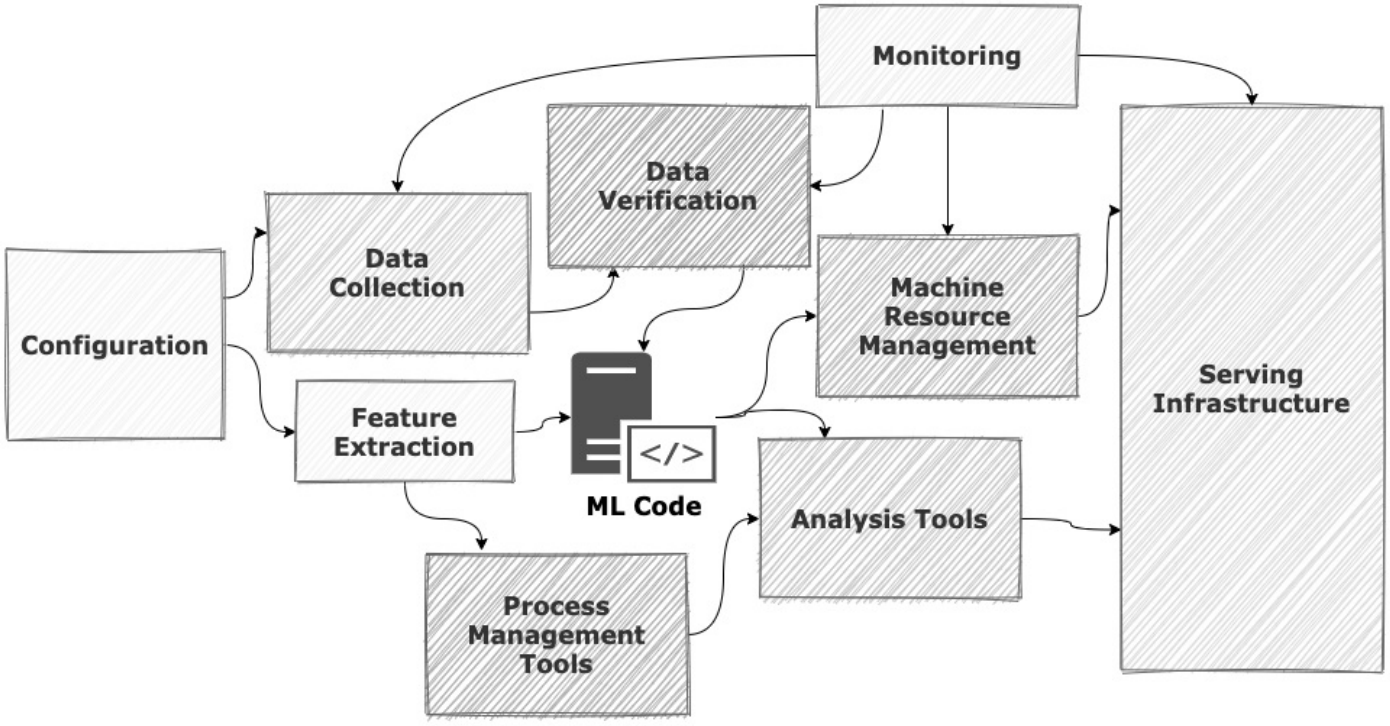
作者 | KubeDL 团队

2021 年 6 月 23 日，云原生计算基金会（CNCF）宣布通过全球 TOC 投票接纳 KubeDL 成为 CNCF Sandbox 项目。KubeDL 是阿里开源的基于 Kubernetes 的 AI 工作负载管理框架，取自"**Kube**rnetes-**D**eep-**L**earning"的缩写，希望能够依托阿里巴巴的场景，将大规模机器学习作业调度与管理的经验反哺社区。

项目地址：http://kubedl.io

项目介绍

随着 TensorFlow, PyTorch，XGBoost 等主流 AI 框架的不断成熟，和以 GPU/TPU 为代表的多种AI异构计算芯片的井喷式涌现，人工智能正快速进入“大规模工业化”落地的阶段。从算法工程师着手设计第一层神经网络结构，到最终上线服务于真实的应用场景，除 AI 算法的研发外还需要大量基础架构层面的系统支持，包括数据收集和清理、分布式训练引擎、资源调度与编排、模型管理，推理服务调优，可观测等。如以下经典图例所展示，众多系统组件的协同组成了完整的机器学习流水线。



与此同时，以 Kubernetes 为代表的云原生技术蓬勃发展，通过优秀的抽象和强大的可扩展性，将应用层与 IaaS（Infrastructure as a Service）层的基础设施完美解耦：应用能够以“云”的范式按需使用资源，无需关注底层基础设施的复杂性，从而解放生产力并专注于自身领域的创新。

Kubernetes 的出现解决了云资源如何高效交付的问题，但对于 AI 这类本身具备高度复杂性的工作负载还无法做到很好地原生支持，如何整合各类框架的差异并保留其通用性，同时围绕 AI 工作负载的运行时去建设一系列完善的周边生态及工具，业界还在不断探索与尝试。在实践中，我们发现了 AI 负载运行在 Kubernetes 生态中面临着如下挑战：

- 机器学习框架百花齐放，各自有不同的优化方向和适用场景，但在分布式训练作业的生命周期管理上又存在着诸多共性，同时针对一些高级特性也有相同的诉求（如网络模式，镜像代码分离，元数据持久化，缓存加速等）。为每类框架的负载单独实现 operater，各自独立进程无法共享 state，缺乏全局视角，使得全局 Job 层面的调度以及队列机制难以实现。此外，不利于功能的抽象和复用，在代码层面存在重复劳动。

- 原生 Kubernetes 无法满足离线任务多样的调度需求。Kubernetes 面向 Pod 调度的模型天然适用于微服务等 Long Running 的工作负载，但针对离线任务的高吞吐，Gang Scheduling 调度（All-Or-Nothing），Elastic Capacity 等多种调度诉求，社区演进出了多种调度方案。以机器学习分布式训练作业调度场景中极为常见的Gang Scheduling为例，社区目前就有 YuniKorn，Volcano，Coscheduling 等调度器实现，提供不同的交互协议，我们需要有插件化的手段来启用对应的调度协议。同时，像 PS/worker 这类根据业务特有属性，不同 role 之间有启动依赖的 DAG 编排诉求，需要在控制器中实现；

- 分布式训练的结果往往以模型作为 output，并存储在分布式文件系统中如(阿里云 OSS/NAS)，但如何从训练作业的视角去管理模型，像容器镜像那样成为AI服务的“不可变基础设施”并实现简单且清晰的版本管理与追溯，业界还缺乏最佳实践。同时，“训练”与“推理”两个阶段相对独立，算法科学家视角中的“训练->模型->推理”机器学习流水线缺乏断层，而“模型”作为两者的中间产物正好能够充当那个“承前启后”的角色；

- 分布式训练尚能大力出奇迹，但推理服务的规格配置却是一个精细活。显存量、 CPU 核数、BatchSize、线程数等变量都可能影响推理服务的质量。纯粹基于资源水位的容量预估无法反映业务的真实资源需求，因为某些引擎如 TensorFlow 会对显存进行预占。理论上存在一个服务质量与资源效能的最优平衡点，但它就像黑暗中的幽灵，明知道它的存在却难以琢磨。随着 GPU 虚拟化技术的成熟，这个平衡点的价值越来越凸显，更优的规格能显著提供单 GPU 卡的部署密度，节约大量的成本。

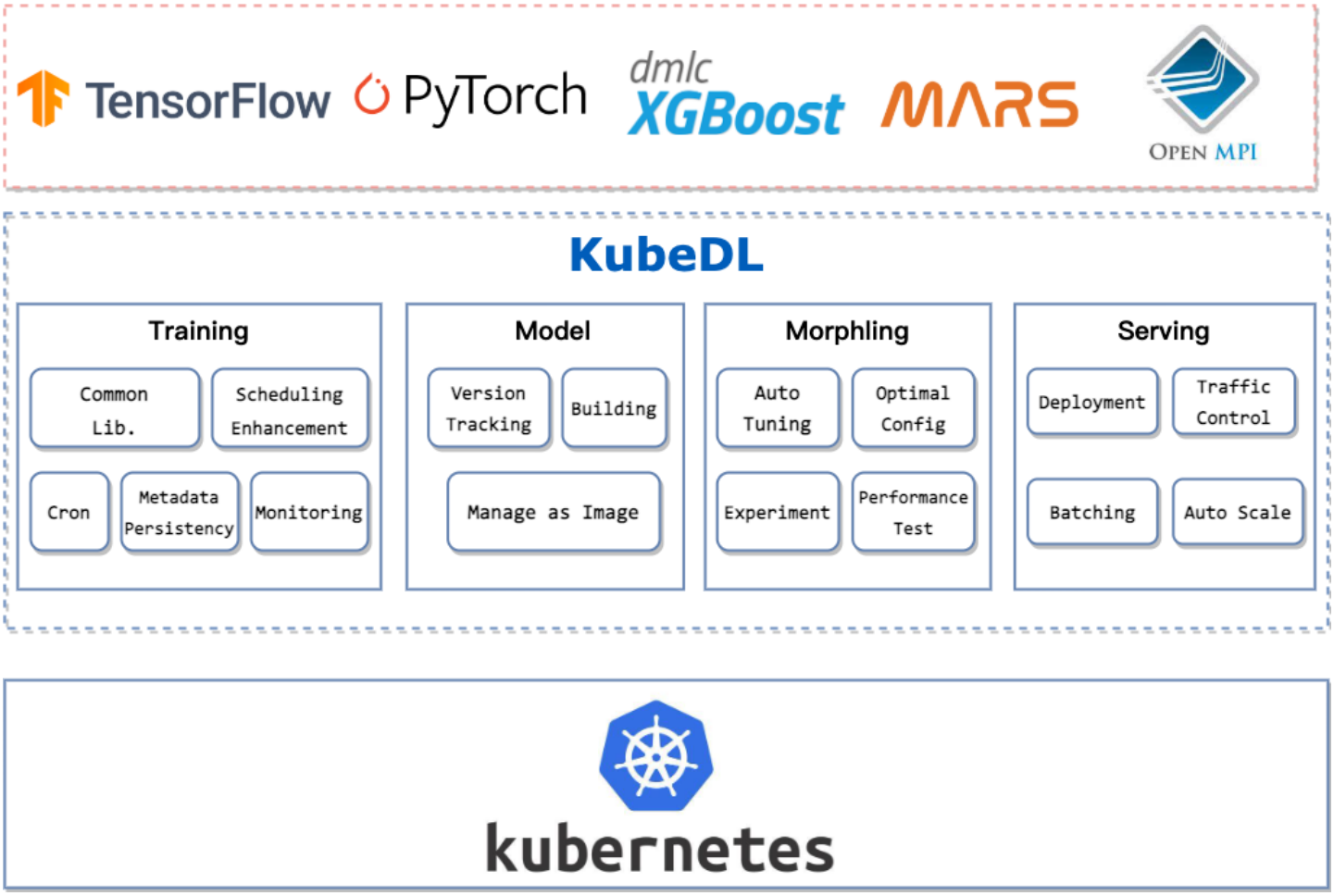
- 推理服务本身是一种特殊的 long running 微服务形态，除了基础的 deployment 外，针对不同的推理场景还欠缺一些实例与流量的管理策略，如：

  1) 算法科学家通常会同时部署两个甚至多个不同版本的模型实例进行 A/B Test 以验证最佳的服务效果，需要基于权重的精细化流量控制；

  2) 能够根据流量请求水平和当前推理服务的 metrics 来自动触发实例的扩缩，在充分保障服务可用性的前提下最小化资源成本等等。

KubeDL

针对上述难题，阿里巴巴云原生，集群管理和 PAI 团队将管理大规模机器学习工作负载的经验沉淀为通用的运行时管理框架——KubeDL，涵盖分布式训练，模型管理，推理服务等机器学习流水线的各阶段，使工作负载能够高效地运行在 Kubernetes 之上。



1、分布式训练

KubeDL 支持了主流的机器学习分布式训练框架（TensorFlow / PyTorch / MPI / XGBoost / Mars 等），其中 Mars 是阿里巴巴计算平台开源的基于张量的大规模数据计算框架，能够分布式地加速 numpy，pandas 等数据处理框架的效率，帮助 Mars 作业以更 native 的方式集成进云原生大数据生态中。

我们将各类训练作业生命周期管理中的共同部分进行抽象，成为一层通用的运行时库，被各分布式训练作业控制器复用，同时用户也可以在此基础上快速扩展出自定义的 workload 控制器并复用现有的能力。借助声明式 API 与 Kubernetes 网络/存储模型，KubeDL 能够进行计算资源的申请/回收，各 Job Role 之间的服务发现与通信，运行时的 Fail-over 等，算法模型的开发者只需声明好此次训练依赖的 Job Role 及各自的副本数，计算资源/异构资源数量等，然后提交任务。另外，我们针对训练领域的痛点也做了诸多的特性设计来提升训练的效率与体验：

- 不同的训练框架往往包含不同的 Job Role，如 TensorFlow 中的 PS/Chief/Worker 和 PyTorch 中的 Master/Worker，Role 与 Role 之间往往隐含着依赖关系，如 Worker 依赖 Master 启动之后才能正常开始计算，错乱的启动顺序不仅容易造成资源长时间空转，甚至可能引发 Job 直接失败。KubeDL 设计了基于 DAG（Direct Acyclic Graph）的调度编排控制流，很好地解决了 Role 之间的启动依赖顺序，并能够灵活扩展。

- 大模型的训练时长往往受制于计算节点间的通信效率，RDMA 等高性能网络技术的应用将极大地提升了数据的传输速度，但这些定制网络往往需要计算节点使用 Hostnetwork 进行互相通信，同时有些时候由于环境限制无法提供基于 Service 模式的服务发现机制。这就需要作业管理引擎能够支持 Host 网络模式下的服务发现机制，处理好各计算节点的网络端口分配，并与各训练框架的特性结合来处理节点 Fail-over 后的网络连通性，KubeDL 支持了 Host 网络模式下的高性能分布式训练。

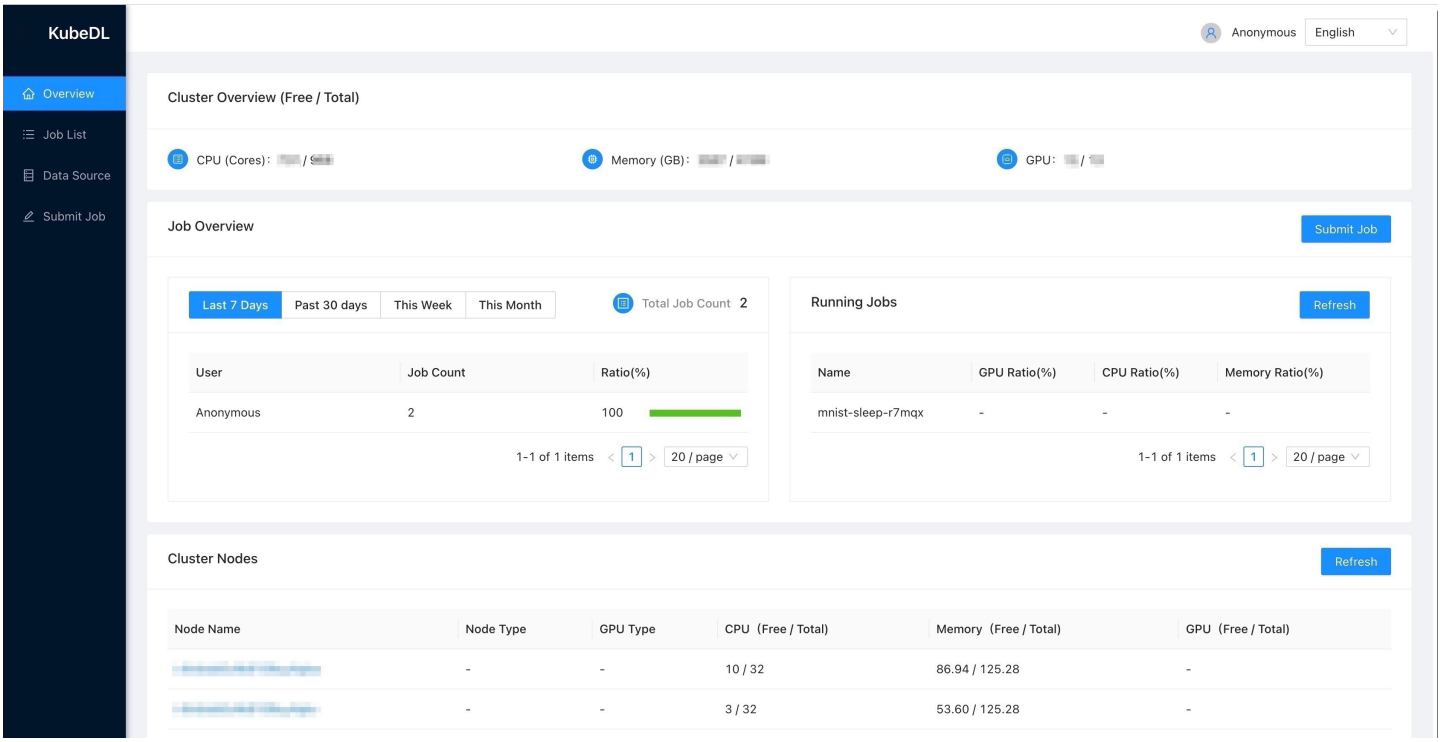
- Gang Scheduling 是分布式训练作业调度场景中的常见需求，组成单个训练作业的一簇 Pod 往往要求同时被调度，避免在集群容量紧张时因作业间的资源竞争出现活锁，但 Kubernetes 强大的可扩展性也使得不同的调度器实现了不同的 Gang Scheduling 协议，如 YuniKorn, KubeBatch 等。为了避免与具体调度器实现的耦合，适应不同用户环境的差异，KubeDL 将 Gang Scheduling 的协议实现插件化，按需启用对应的插件即可与调度器相互协作，实现作业的批量调度。

- Job 是一次性的，但在实际的生产应用中我们经常会遇到反复训练/定时训练的场景，如每日拉取某一时间区间的离线表并进行数据清洗以及模型的 Re-train，KubeDL 提供了一类单独的工作负载—Cron 来处理定时的训练请求，并支持任意类型的训练作业（如 TFJob，PyTorchJob 等），用户可以提交 cron tab 风格的定时命令及作业模板，并在 Cron 资源的状态中追踪训练作业的历史及当前进行中的作业。

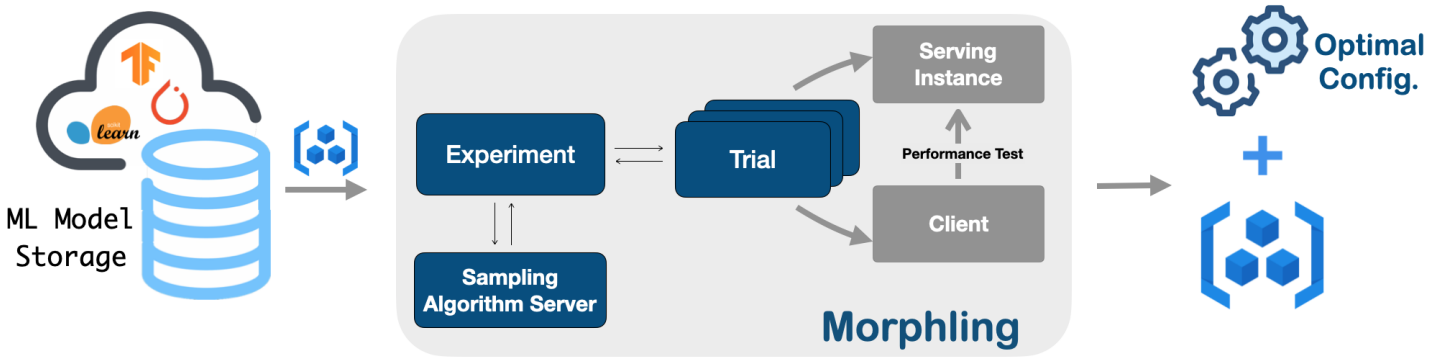
针对海量离线作业元数据需要长时间保存（Job CRD 被删除后元数据即从 etcd 销毁）的诉求，KubeDL 还内置了元数据的持久化，实时监听 Job/Pod/Events 等资源对象的变化，转化成对应的 Databse Schema Object 并持久化到存储后端中。存储后端的设计也是插件化的，用户可以根据自己的线上环境来实现存储插件并在部署时 enable。在 KubeDL 中 Job/Pod 默认支持了 Mysql 的存储协议，以及将 Events 收集到阿里云 SLS 服务中。

同时我们还提供了管控套件：KubeDL-Dashboard，用户不需要去理解 Kubernetes 的众多 API 并在各种 kubectl 命令中挣扎，即可界面化地上手简单易用的机器学习作业。持久化的元数据也可以直接被 Dashboard 消费使用。Dashboard 提供了简单的作业提交、作业管理、事件/日志查看、集群资源视图等功能，以极低的学习门槛帮助机器学习用户上手实验。


图片违规！

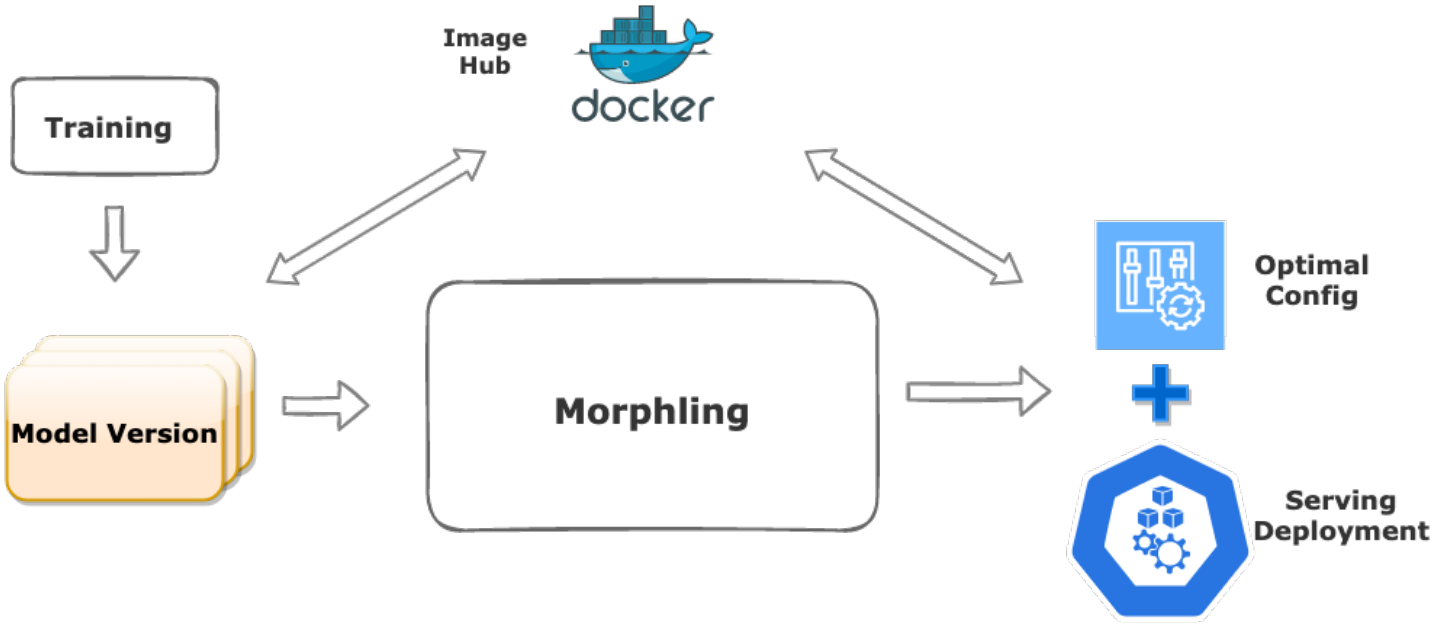

KubeDL
Overview
Job List
Data Source
Submit Job
Anonymous
English
Cluster Overview (Free / Total)
CPU (Cores):
Memory (GB):
GPU:
Job Overview
Submit Job
Last 7 Days
Past 30 days
This Week
This Month
Total Job Count 2
User
Job Count
Ratio(%)
Anonymous
2
100
1-1 of 1 items
1
20 / page
Running Jobs
Refresh
Name
GPU Ratio(%)
CPU Ratio(%)
Memory Ratio(%)
mnist-sleep-r7mqx
1-1 of 1 items
1
20 / page
Cluster Nodes
Refresh
Node Name
Node Type
GPU Type
CPU（Free / Total)
Memory（Free / Total)
GPU（Free / Total)
10 / 32
86.94 / 125.28
3 / 32
53.60 / 125.28

2、推理服务规格调优

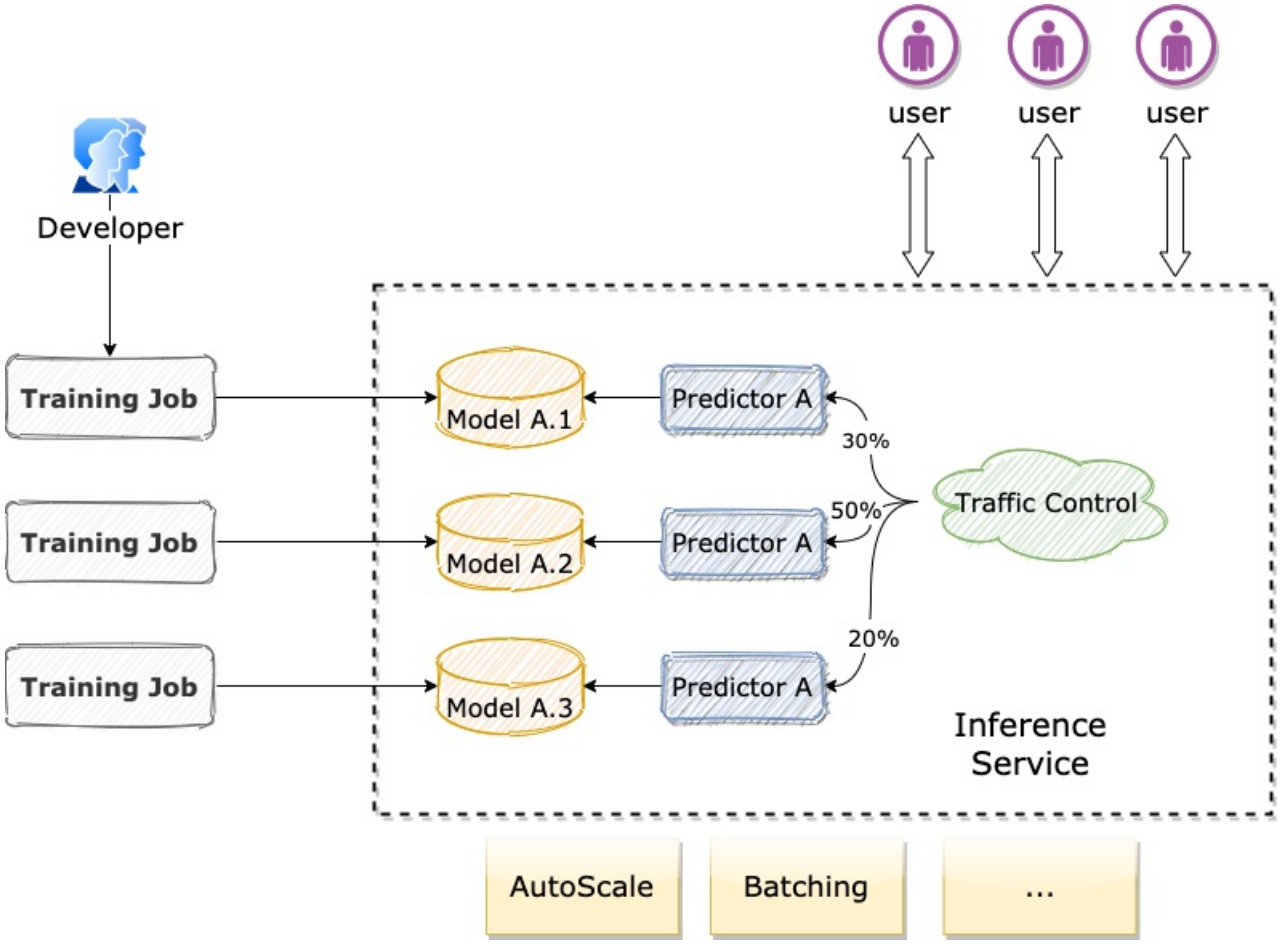


GPU 虚拟化与分时复用技术的发展和成熟，让我们有机会在一块 GPU 上同时运行多个推理服务，显著降低成本。然而如何为推理服务选择合适的 GPU 资源规格，尤其是不可压缩的显存资源，成为一个关键难题。一方面，频繁的模型迭代让算法工程师无暇去精确估计每个模型的资源需求，流量的动态变化也让资源评估变得不准确，因此他们倾向于配置较多的 GPU 资源冗余，在稳定性和效率之间选择牺牲后者，造成大量浪费；另一方面，由于 Tensorflow 等机器学习框架倾向于占满所有空闲的显存，站在集群管理者的角度，根据显存的历史用量来估计推理业务的资源需求也非常不准确。在 KubeDL-Morphling 这个组件中我们实现了推理服务的自动规格调优，通过主动压测的方式，对服务在不同资源配置下进行性能画像，最终给出最合适的容器规格推荐。画像过程高度智能化：为了避免穷举方式的规格点采样，我们采用贝叶斯优化作为画像采样算法的内部核心驱动，通过不断细化拟合函数，以低采样率（<20%）的压测开销，给出接近最优的容器规格推荐结果。

3、模型管理与推理服务



模型是训练的产物，是计算与算法结合后的浓缩精华，通常收集与维护模型的方式是托管在云存储上，通过组织文件系统的方式来实现统一管理。这样的管理方式依赖于严格的流程规范与权限控制，没有从系统层面实现模型管理的不可变，而容器镜像的诞生解决的就是 RootFS 的构建-分发-不可变等问题，KubeDL 将两者进行结合，实现了基于镜像的模型管理。训练成功结束后，通过 Job Spec 中指定的 ModelVersion 会自动触发模型镜像的构建。用户可以在 ModelVersion.Spec 中约定模型的存储路径，目标的镜像 Registry 等基本信息，将每次的训练输出 Push 到对应的镜像仓库。


user
user
user
Developer
Training Job
Training Job
Training Job
Model A.1
Model A.2
Model A.3
Predictor A
Predictor A
Predictor A
30%
50%
20%
Traffic Control
Inference Service
AutoScale
Batching
...

同时镜像作为训练的输出，以及推理服务的输入，很好地串联起了两个阶段，也借此实现了分布式训练->模型构建与管理->推理服务部署的完整机器学习流水线。KubeDL 提供了 Inference 资源对象提供推理服务的部署与运行时控制，一个完整的 Inference 服务可以由单个或多个 Predictor 组成，每个 Predictor 对应前序训练输出的模型，模型会被自动拉取并挂载到主容器 Volume 中。当多个不同模型版本的 Predictor 并存时，可以根据分配的权重进行流量的分发与控制，达到 A/B Test 的对照实验效果，后续我们还会在 Batching 批量推理和 AutoScale 上针对推理服务场景做更多的探索。

KubeDL 分布式训练在公有云上的实践

- **PAI-DLC**

随着云计算的深入人心以及越来越多的业务都用云原生的方式进行，阿里云计算平台 PAI 机器学习团队推出了 DLC（Deep Learning Cloud）这一深度学习平台产品。DLC 采用全新的云原生架构，底层采用 Kubernetes 作为资源底座支持，而训练部分全面采用 KubeDL 进行管理，是 KubeDL 在深度学习云计算场景中的大规模实践。

DLC 在阿里集团内部内广泛支撑了众多的业务，包括淘系安全部达摩院的图像视频、自然语言、语音、多模态理解、自动驾驶等众多业务部门的深度学习计算需求。在服务于深度学习驱动的前沿业务生产中，PAI 团队在框架和平台建设方面积累了许多的经验，沉淀了兼容社区（eg，TensorFlow/PyTorch）并且具有鲜明特色的大规模工业界实践过的框架平台能力，如万亿规模参数的M6模型的训练、工业级图神经网络系统 Graph-Learn、极致资源管理和复用能力等等。

如今，PAI-DLC 的能力也在全面拥抱公有云，为开发者和企业提供的云原生一站式的深度学习训练平台，一个灵活、稳定、易用和高性能的机器学习训练环境，以及全面支持支持多种社区和 PAI 深度优化的算法框架，高性能且稳定的运行超大规模分布式深度学习任务，为开发者和企业降本增效。

公有云的 DLC 作为阿里巴巴集团机器学习平台最佳实践的透出，在产品细节、框架优化、平台服务等方面都吸取了工程实践中的宝贵的经验。除此之外，DLC 产品在设计之初就充分考量了公有云场景中的独特属性，提供了竞价实例、自动 Fail-Over、弹性扩缩等功能，为客户努力降低 AI 算力成本。

进一步的，DLC 也与 PAI 的其他公有云产品相结合，比如说服务于算法工程师建模的 DSW、服务于企业级 AI 全流程的、自动化的 AutoML、在线推理服务 EAS 等，打造全流程的 AI 标杆性产品。

原文链接：https://developer.aliyun.com/article/787300?