人工智能工作负载调度解决方案与实践指南

作者：Sabrina Di Leva、Kevin Hannon、Rajas Kakodkar、Ekin Karabulut、Victor Lu、Abhishek Malvankar、Alexander Scammon、Yuan Tang、Andrey Velichkevich、Marlow Warnicke

贡献者：Shravan Achar，Andrew Shunichi Aikawa，Ricardo Aravena，Malini Bhandaru，Sachi Desai，Johnu George，Josh Halley，Naadir Jeewa，Boris Kurktchiev，Niki Manoledaki，Claudia Misale，Mike SH Ng，Ronald Petty，Sudhanshu Prajapati，Brian Redmond，Joel Roberts，Vijay Rodrigues，Kay Yan，Michael Yao，Adel Zaalouk，Cathy Zhang

CLOUD NATIVE COMPUTING FOUNDATION

目录

系列介绍 03

执行摘要 03

先前 04

解决方案图景 04

Kubernetes原生功能 04

调度器扩展与插件 05

批处理调度器和队列管理器 05

机器学习平台工具 06

数据存储工具 06

工作流编排 07

非 Kubernetes 调度器 07

实用指南 10

选择调度堆栈 10 个常见模式与反模式 11 入门 11

结论12

CLOUD NATIVE COMPUTING FOUNDATION

系列介绍

本文是一系列关于云原生人工智能工作负载调度的第五篇也是最后一篇：

1. 理解用于 Kubernetes 调度的 AI 工作负载：什么是 AI 工作负载以及它们与传统云原生应用程序有何不同

2. **AI工作负载的调度基础：Kubernetes调度机制及其为何需要额外层**
3. **AI工作负载的作业编排挑战：组调度、公平性、队列、抢占、优先级和预留**

5. **人工智能工作负载调度解决方案与实践指南（本文）：工具、参考架构和实际应用案例**

每一篇论文都可以独立阅读，但合在一起，它们为在 Kubernetes 上运行 AI 工作负载提供了全面的指南。

执行摘要

理解 AI 工作负载、调度基础以及挑战是必要的，但并非充分——组织需要了解有哪些工具可用以及如何应用它们。 本文列出了可用于 AI 工作负载调度的解决方案：Kubernetes 原生功能、调度器扩展、批处理调度器、机器学习平台工具和工作流编排器。它提供了一个参考表格，将挑战映射到解决方案，提供了关于选择调度堆栈的实用指南、常见的模式与反模式，以及具体的用例，展示了组织如何为不同场景配置调度。

先前

本系列的前四篇论文确立了：

• **论文1：AI工作负载跨越多个生命周期阶段，具有不同的资源特征。** 训练是GPU密集型的，并且紧密耦合；推理对延迟敏感且具有状态性。

• **论文4：资源和基础设施挑战包括拓扑感知、GPU共享、可扩展性、I/O瓶颈、容错性和成本限制。** GPU直接通信和网络拓扑等基础设施考量也会影响调度决策。

本文列举了相关解决方案，并提供了应用这些方案的实际指导。

解决方案图景

本节列出了用于解决论文3和4中讨论的调度挑战的工具和Kubernetes功能。

Kubernetes原生功能

Kubernetes 包含几种与 AI 工作负载调度相关的功能：

• **容器优先级与抢占。** 优先级类为容器分配优先级等级。当资源稀缺时，高优先级容器可以抢占低优先级容器。限制：仅在容器级别生效，不作用于作业级别。 • 拓扑扩展约束。将容器分配到故障域（区域、区、节点）。有助于提升容错能力，但不适用于优化GPU通信拓扑。 • 可变调度指令。允许在作业挂起时修改其调度约束。自定义队列控制器使用此功能实现准入控制。 • 非抢占式优先级类。作业可以拥有高优先级（以便优先排队），而无需抢占正在运行的工作。适用于可以等待的紧急作业。

让资源自然释放。

• **动态资源分配（DRA）。** 一项较新的功能（GA，Kubernetes 1.34+），能够灵活分配专用资源。DRA 支持按需 GPU 划分，允许工作负载请求特定的 GPU 配置，而非整块 GPU。

调度器扩展与插件

Kubernetes调度器可以通过插件进行扩展：

调度器插件（kubernetes-sigs/scheduler-plugins）是一个存放树外调度器插件的仓库。 值得注意的插件包括：

• 联合调度：通过 PodGroups 实现组调度 • 节点资源：增强的资源分配算法
• 能力调度：弹性配额管理，支持超额使用

多个插件可以同时运行，但插件之间的交互需要仔细配置。

批处理调度器和队列管理器

这些工具在 Kubernetes 上增加了作业队列、公平共享和组调度功能：

• **火山（Volcano）是一个CNCF项目，提供全面的批次调度功能。** 其特性包括组调度、公平份额策略、队列管理和拓扑感知调度。广泛应用于训练工作负载。

Apache YuniKorn 提供了基于分层队列和公平共享的统一调度，用于批处理和交互式工作负载。

• Slinky（由NVIDIA维护的开源项目）是一套强大的集成工具，旨在将Slurm的功能引入Kubernetes，既允许Slurm作业在Kubernetes中运行，也将整套Slurm调度功能引入Kubernetes。

机器学习平台工具

这些工具为人工智能工作流程提供了更高层次的表达方式：

- **Kubeflow**

  • Kubeflow 训练器是一个原生的 Kubernetes 分布式 AI 平台，用于可扩展的大型语言模型微调和训练，支持跨多种框架的 AI 模型，包括 PyTorch、MLX、HuggingFace、DeepSpeed、JAX、XGBoost 等。它提供作业抽象层，用于处理工作节点协调，包括群组调度需求和高性能计算工作负载编排，如 MPI 和 Flux。

  • **Kubeflow MPI Operator 支持分布式训练中的基于 MPI 的集体通信。** 对于使用全归约（all-reduce）同步的工作负载至关重要。

  • **Kubeflow Katib 管理自动化机器学习工作负载——超参数优化和神经架构搜索。** 协调具有不同配置的多个训练作业。

• KServe 提供了一个标准化的分布式生成式和预测式 AI 推理平台，用于在 Kubernetes 上进行可扩展的多框架部署。

• **MLflow 提供实验跟踪、模型注册和部署工具。** 它不是调度器，但可与调度系统集成以进行工作流管理。

• **Ray/KubeRay 提供了一个拥有自己调度器的分布式计算框架。** KubeRay 是用于运行 Ray 集群的 Kubernetes 运算符。适用于需要 Ray 编程模型的工作负载。

数据存储工具

• **流体（CNCF Sbx）通过数据集缓存和数据移动优化，加速 AI 工作负载的数据访问。** 减少导致 GPU 空闲的 I/O 瓶颈。

- HAMi（异构人工智能计算虚拟化中间件）支持跨不同GPU类型进行灵活的GPU共享和分区。

- **Kubeflow（CNCF孵化项目）：**
  - Kubeflow 数据缓存为分布式训练工作负载提供高效的数据流。
  - Kubeflow Spark Operator 旨在让指定和运行 Apache Spark 应用程序像在 Kubernetes 上运行其他工作负载一样简单和自然。
  - Kubeflow Hub为机器学习模型开发者提供了统一的管理界面，用于索引和管理模型、版本以及机器学习工件元数据。

工作流编排

这些工具管理多步骤的机器学习流水线：

• Apache Airflow 是一种通用的工作流编排器，常用于机器学习（ML）工作流。 它定义任务的有向无环图（DAG），并处理调度、监控和重试。• Argo Workflows 是一种原生 Kubernetes 的工作流引擎。工作流被定义为 Kubernetes 资源，使其与其他 Kubernetes 工具的集成变得简单。• Flyte 是一个为机器学习设计的 workflow 编排平台。为工作流的输入和输出提供版本控制、缓存和强类型。• Kubeflow Pipelines 是一个平台，用于在基于 Kubernetes 的系统上使用容器构建和部署可移植且可扩展的机器学习（ML）工作流。

非 Kubernetes 调度器

• Slurm 是 HPC 环境中的主流调度器。 许多组织在拥有现有 HPC 基础设施时，会选择在 Kubernetes 旁边运行 Slurm 或用 Slurm 替代 Kubernetes 来处理 AI 工作负载（尤其是训练和多节点推理）。目前存在集成方法，可用于连接 Slurm 和 Kubernetes 环境。

CLOUD NATIVE COMPUTING FOUNDATION

将挑战映射到解决方案

一个参考表，将每个挑战映射到解决该挑战的方案。所有方案按字母顺序排列。

挑战	Kubernetes原生	调度员	Platforms/ 工具	适用范围	注释
Gang 日程安排	工作负载 API + 通用工作负载功能 gate (beta, K8s 1.37, off 默认情况下)	舰队，协同调度 插件，KAI，Kueue 弹簧，火山，尤尼玉米	Kubeflow 的中文翻译是：Kubeflow 训练师	Both	必不可少的 分布式 培训
资源 公平	资源配额（有限）	舰队，KAI，Kueue 弹簧，火山，尤尼玉米	-	Both	层级 配额要求 外部工具
排队 管理	-	舰队，KAI，Kueue 弹簧，火山，尤尼玉米	气流，Argo Workflows, 飞特	Both	Core 能力 批次的 调度程序
优先购买权	优先级（Pod级别）	凯，奎尤，斯林基 火山	-	Both	Job-level 抢占 需求 外部工具
优先 日程安排	优先舱	所有批次调度程序	-	Both	Job-level 优先考虑 批量 调度程序
预订 & 填补	-	弹簧，火山，尤尼玉米	-	培训	先进 特性 在某些情况下 调度程序
拓扑学 意识 (Node)	拓扑管理器 （NUMA），DRA CPU驱动程序 (CPU拓扑结构)	HAMi，KAI，Kueue，Slinky 火山	-	Both	GPU 互联 意识 变化多端
拓扑学 意识 (Cluster)	拓扑传播 约束，DRANET （网络DRA驱动程序） (limited)	凯，奎尤，斯林基 火山	-	Both	网络 拓扑学 意识 浮现的；出现的；新兴的
资源 异质性	节点选择器，标签	所有批次调度程序	-	Both	标准 Kubernetes (K8s) 特性 通常 充足

GPU共享	DRA (GA, K8s 1.34+)	KAI	HAMi, KubeRay, 火山	Both	MIG需要 DRA或 供应商工具
可扩展性	集群自动缩放器 卡普特纳	舰队，KAI，KEDA Kueue：库尤Slinky：弹簧蛇Volcano：火山	互链	Both	Large-scale 安排 挑战性的
I/O瓶颈	持久卷	-	流体	Both	存储 并缓存 解决方案
容错性	-	凯伊，弹簧蛇，火山 (checkpoint support)	Kubeflow 的中文翻译是：Kubeflow (elastic training)	培训	Framework- 依赖的
弹性	HPA, VPA	凯伊，弹簧蛇，火山	Kubeflow 的中文翻译是：Kubeflow 训练师	Both	PyTorch 弹性等
Budget/Cost	-	有限支持	飞腾 （现货/） 可中断的 tasks)	Both	浮现的 area
推断 请求 排程	机架级指标	动力路由器，llm-d- 推理调度器	Kubernetes （K8s）	推断	浮现的 area
推断 请求 自动扩展	机架级指标，事件 驱动	动力规划师，llm-d- WVA-autoscaler	KServe	推断	浮现的 area

- **关键观察：**

  • 历史上，资源调度一直是默认 Kubernetes 的最关键能力短板，现在正通过原生方式解决：v1.35 版本已进入 alpha 阶段，v1.37 版本已进入 beta 阶段（默认禁用），计划在 v1.38 版本正式发布（GA）。• 公平性和队列管理对于超出基本 ResourceQuota 范围的功能仍需依赖外部工具。• 资源拓扑感知已部分解决，但仍是积极开发中的领域。• GPU 共享随着 DRA 的应用正变得更容易，但内存限制限制了其在大模型上的适用性。• 当前工具在成本管理方面未能充分满足需求。

CLOUD NATIVE
COMPUTING FOUNDATION

实用指南

选择调度堆栈

合适的调度堆栈取决于你的工作负载、规模和现有基础设施。

如果要从基础开始：

- 您主要进行单节点训练或推理
- 您的工作负载不需要集群调度
- 您有一个小团队，且公平性要求简单

在这种情况下，标准的 Kubernetes 功能（PriorityClasses、ResourceQuota、节点选择器）可能就足够了。

若需使用批处理调度器，则：

- 您运行的工作负载范围从单个 Pod 到多 Pod 配置（例如分布式训练或推理），这些工作负载需要集群调度。 • 您有多个层级团队在竞争 GPU 资源。
- 您需要复杂的份额分配策略。

考虑多集群调度，如果：

- 您的资源需求已超出单个集群的容量
- 您的基础设施分布在不同区域或云平台
- 您希望跨多个集群实现统一的作业管理

若需使用与 Kubernetes 集成的传统 HPC 调度器，则：

- 您已拥有现有的HPC基础设施和工作流程
- 您正在运行训练工作负载
- 您有需要传统HPC调度器的用户
- 您需要Kubernetes调度器尚未提供的能力
- 您希望具备混合能力，能够在同一节点上运行传统HPC工作负载和Kubernetes负载

常见模式与反模式

行之有效的模式：

• 不同工作负载类型使用独立的队列。 生产推理、训练和实验具有不同的优先级和资源特征。使用独立队列可以使策略更清晰。

反模式：应避免的

• 在没有集群调度器的情况下运行集群调度的工作负载。 部分分配会浪费资源并可能导致死锁。

入门

针对平台工程师构建新基础设施：

1. 首先使用单个批次调度器来处理基本的排队和配额管理。

2. 定义映射到您组织结构（团队、项目、环境）的队列。
3. 根据预期的资源分配设置配额——您可以稍后调整。

对于使用现有基础设施的机器学习工程师：

1. 了解您集群中可用的调度工具。
2. 使用合适的作业抽象（如MPIJob、PyTorchJob、TrainJob等），而不是原始的Pod。

对于正在评估其选项的组织：

1. 盘点你的工作负载：有多少比例需要组调度？GPU共享？低延迟？
2. 评估你的规模：有多少GPU？有多少并发用户/团队？

结论

Kubernetes最初并非为AI工作负载而设计；然而，它已成为管理这些工作负载的准事实标准。其默认调度器擅长放置无状态微服务，但在历史上缺乏机器学习作业所需的功能：集群调度、跨团队公平共享、GPU互连的拓扑感知以及作业级（而不仅仅是Pod级）管理。生态系统已做出响应，Kubernetes本身也在不断发展以弥补这些差距。本系列的关键要点：

1. 了解您的工作负载。机器学习生命周期的不同阶段有不同的要求。数据准备、训练和推理每个都需要不同的调度策略。

2. 对抗调度对于人工智能工作负载至关重要。没有它，部分分配会浪费资源并导致死锁。如果你运行分布式训练，你需要一个支持对抗调度的调度器。

3. 公平性需要明确的政策。ResourceQuota 提供了基本的限制，但真正的多租户集群需要层级配额、借/贷机制和公平份额算法。

云原生AI领域持续发展。 新的硬件（不同的GPU架构、加速器）、新的工作负载模式（代理系统、分布式推理）以及新的Kubernetes功能将带来挑战与机遇。本系列提供了基础；保持更新需要持续参与社区。



CLOUD NATIVE
COMPUTING FOUNDATION