Multi-Resource Interleaving for Deep Learning Training

Yihao Zhao,Yuanqiang Liu,Yanghua Peng,Yibo Zhu,Xuanzhe Liu,Xin Jin

摘要

训练深度学习 (DL) 模型需要多种资源类型，包括 cpu、gpu、存储 IO 和网络 IO。深度学习的进步已经产生了广泛的模型，这些模型在不同的资源类型上具有不同的使用模式。现有的 DL 调度器只关注 GPU 分配，而错过了将作业打包到多个资源类型的机会。

我们提出了一个多资源集群调度器 Muri，用于 DL 工作负载。Muri 利用 DL 培训作业的多资源交叉，实现了高资源利用率和减少作业完成时间 (JCT)。DL 作业具有独特的分阶段迭代计算模式。

与大数据工作负载的多资源调度器 (在空间维度上打包作业) 相比，Muri 利用这种独特的模式在时间维度上在同一组资源上交错作业。

Muri 采用 Blossom 算法对两种资源类型的单 gpu 作业找到最优分组方案，并将算法推广到处理两种以上资源类型的多 gpu 作业。我们构建了 Muri 的原型，并将其与 PyTorch 集成。在 64 个 gpu 集群上的实验表明，Muri 比现有的 DL 调度器平均提高了 3.6 倍的 JCT 和 1.6 倍的 makespan

关键词：资源共享；深度学习

1 引言

深度学习 (DL) 已经越来越多地融入到以数据为中心的互联网应用和服务中[1]。DL 模型的训练已经成为数据中心的一项重要工作，虽然深度学习调度方面已经取得了重大进展[2-5]，但一个主要的限制是大多数深度学习调度程序只关注 GPU 分配。这些调度器中一个关键的潜在假设是 GPU 是深度学习训练作业的瓶颈，因此在调度作业时只需要考虑 GPU，而不需要综合考虑其他资源类型。

然而，深度学习训练作业的过程是一个迭代分阶段的过程，每一个阶段使用一种特定的资源，在实践中，DL 培训需要多种类型的资源，包括来自本地或远程存储的存储 IO，用于将培训数据读入 worker，用于预处理和模拟的 cpu(例如，在强化学习中)，用于向前和向后传播的 GPU，网络 IO 用于分布式培训中工人之间的梯度同步，任何一种资源都可能成为瓶颈资源，这篇论文提出了一种针对深度学习工作负载的多集群资源调度器 Muri，与现有的 DL 调度器不同，Muri 在调度 DL 训练作业时利用多种资源类型来提高整体资源利用率。

2 相关工作

2.1 传统的深度学习作业调度方式

许多 DL 调度器已被提出用于 DL 训练工作负载[2-5]。已知在运行时间已知[3]时，最短作业优先 (SJF) 和最短剩余时间优先 (SRTF) 可以使平均 JCT 最小，而在运行时间未知[2]时，最小获得服务 (LAS) 和 Gittins 指数有效。一些 DL 调度器[2,6]扩展了这些算法，通过考虑每个作业使用的 GPU 数量和对分布式训练吞吐量很重要的位置偏好来调度 DL 训练作业。例如，在 DL 作业调度中，Tiresias[2]将 SRTF、LAS 和 Gittins 索引分别扩展为 SRSF (Shortest Remaining Service First)、2D-LAS 和 2D-Gittins 索引。这些解决方案在作业运行时专门为作业分配 GPU。

2.2 使用多资源交织的方式进行作业调度

多资源交织是本文的主体思想，DL 训练作业使用多种资源类型，使用模式是分阶段的。我们利用 DL 训练作业的独特的分阶段迭代计算模式，使细粒度多资源及时交织。图 1 说明了多资源交叉的好处。有四个作业，分别是 A、B、C 和 d。该图描绘了每个作业的一次迭代的资源使用情况。这四个作业在不同的资源类型上遇到了瓶颈。如果资源专门分配给它们，那么每次只能运行一个作业。当至少一种资源类型的峰值使用量较高时，多资源打包不能打包作业。它必须分别运行这四个作业，如图 1(a) 所示。另一方面，如图 1(b) 所示，通过交叉使用不同的资源类型，这四个作业可以重叠并并发运行，与单独运行相比，提高了资源利用率，吞吐量提高了 4 倍。

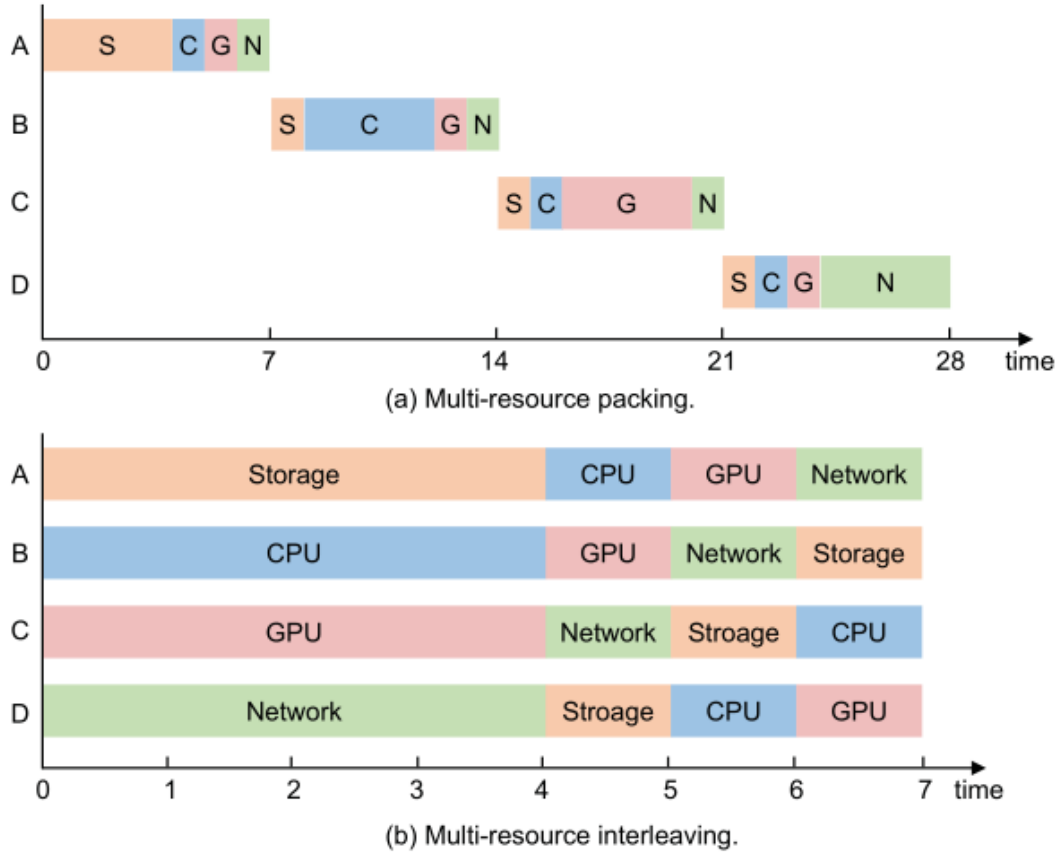

图 1: 当每个作业已经应用了作业内流水线时，多资源交织的好处。为了简单起见，该示例使用了两种资源。

3 本文方法

3.1 Muri

Muri 是本文提出的一个作业调度器，它的体系结构如图 2 所示。

Muri 调度器（Muri Scheduler）包括三个组件，分别是资源探查器（Resource Profiler）、作业调度器（Job Scheduler）、工作监视器（Worker Monitorer）资源探查器：分析每个作业的每种资源的使用情况，并估计不同作业组的交织效率，使用资源配置文件来估计交织效率，这将被用作调度算法的输入。作业调度器：将作业从作业队列调度到集群中的计算机。采用多轮作业分组算法决定哪些作业归为一组以共享资源，从而最大化资源利用率。工作监视器：收集每台机器的资源信息，并跟踪每台机器上资源的可用容量。当计划的作业完成时，监视器通知调度程序，释放的资源可以用于队列中的作业。

Muri 执行器（Muri Executor）从作业调度器接收作业和分组策略，并根据机器上的分组策略执行

作业。

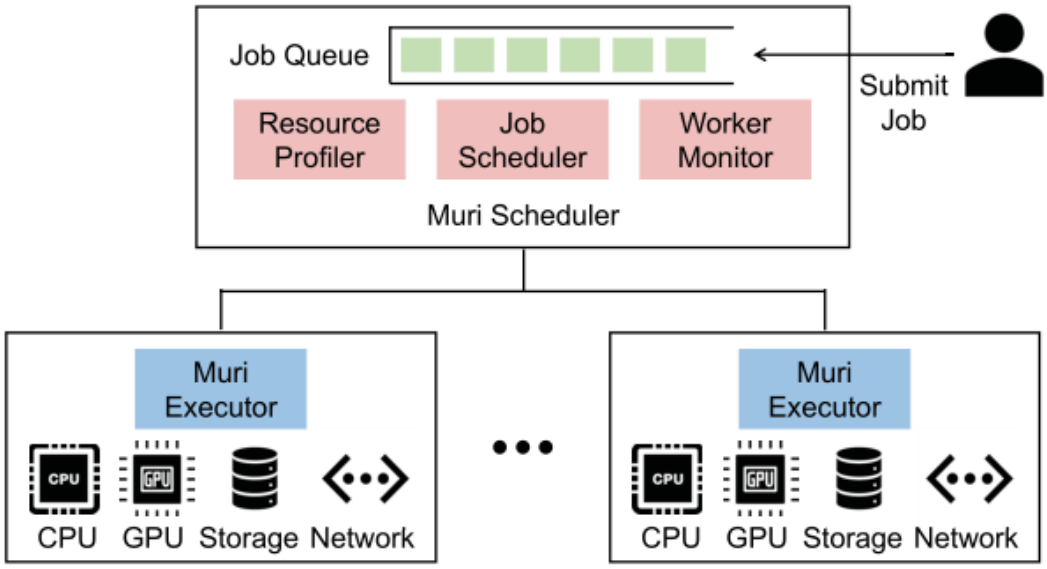

图 2: Muri 体系结构图

3.2 两种资源类型的单 GPU 作业设计

3.2.1 交织作业

在图 3 中使用了一个包含四个作业 (A、B、C 和 D) 和两种资源类型 (GPU 和 CPU) 的示例，以展示如何交织 DL 作业。该图表示每个作业的一次训练迭代的资源使用情况。为了便于说明，我们假设一个作业每次使用一种资源类型。矩形的长度表示作业使用特定资源类型的时间。我们改变不同工作的阶段，使工作相互交错图 2(A) 第一组中，任务 A 的 CPU 阶段与任务 B 的 GPU 阶段重叠，任务 A 的 GPU 阶段与任务 B 的 CPU 阶段依次重叠。我们在不同作业的重叠阶段之后添加同步障碍。以图 2(b) 中的第 1 组为例。C 的 CPU 阶段要等到 A 的 CPU 阶段结束，而不是在自身的 GPU 阶段结束后直接执行。避免两个作业同时使用一个资源的原因是由于干扰[7]可能会严重影响处理速度。

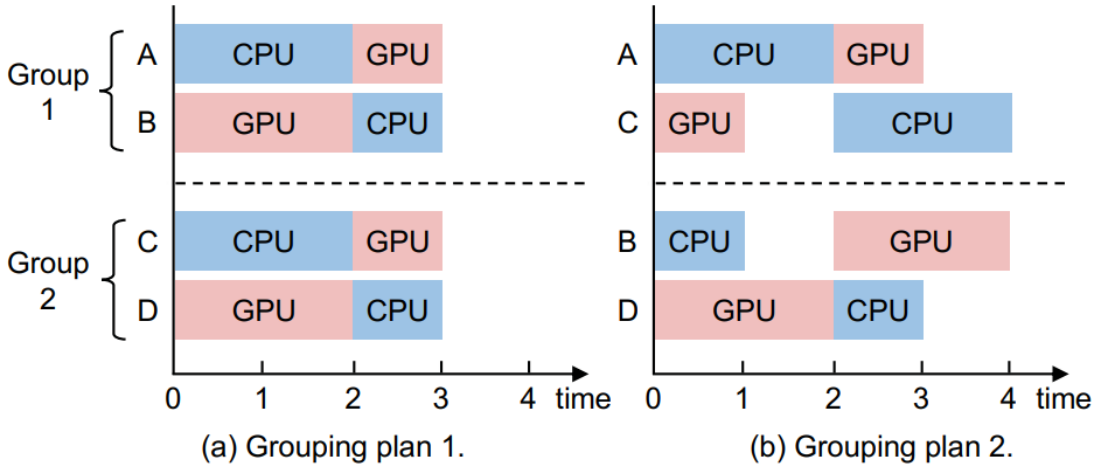

图 3: 分组计划对交织效率的影响。分组计划 1 可以在两个 GPU 上完美地重叠四个作业，而分组计划 2 增加了每次迭代时间

3.2.2 计算作业之间资源使用的重叠率

在分组两个作业时，需要一个度量来捕获资源交织效率。根据图 3 中的示例，可以完全重叠作业的资源使用的分组计划比不能完全重叠的分组计划要好。

3.2.3 计算最大加权匹配

将这个问题转化为图最大加权匹配问题，图中的每个节点表示作业，每条边表示边上的作业可以分为一组，每条边上的权重表示作业间的交织效率。分组计划对应于图形的匹配。因此，找到最佳分组计划可以转换为找到图的最大加权匹配。应用 Blossom 算法来找到图的最大加权匹配。如图 4（b）所示。图 4（b）中的方案 1 对应于图 3（a）中的匹配，图 4（c）中的计划 2 对应于图 3（b）的匹配。计划 1 的权重总和高于计划 2，表明资源效率更高。

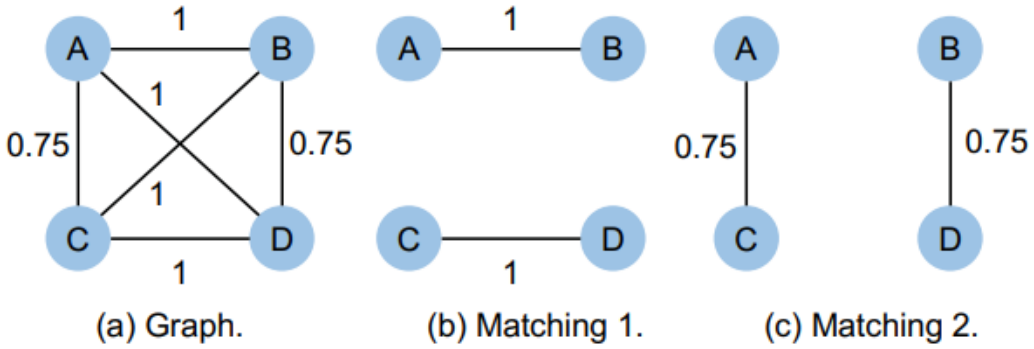

图 4: 分组计划采用最大加权匹配进行计算。（a）中的边缘权重表示每对作业的交织效率。图 4 中的分组计划 1 和 2 分别对应于（b）和（c）。（b）中匹配 1 的总权重高于（c）中匹配 2 的总权重。

3.2.4 连接多个作业的同一阶段

可能会连接多个作业的同一阶段，并将这些作业合并为一个作业。例如，我们可以融合图 4 中的作业 A 和作业 C，以获得一个作业 E。

4 复现细节

4.1 与已有开源代码对比

该论文过于庞大，目前只实现了作者提供的代码的一部分。

4.2 实验环境搭建

Vmvare Station、centos7、anaconda4.3、pytorch

5 实验结果分析

从图 5 中可以看出，在工作持续时间未知的情况下，与 Tiresias 和 Themis 相比，Muri-S 的平均 JCT 提高了 2.5 倍以上，完工时间提高了 1.4 倍以上，尾部 JCT 提高 2.5 倍以上。

	Tiresias	Themis	Muri-L
Normalized JCT	2.59	3.56	1
Normalized Makespan	1.48	1.47	1
Normalized 99^{th}%-ile JCT	2.54	2.60	1

图 5: 实验结果示意

6 总结与展望

根据深度学习训练任务分阶段、迭代的的运行模式，论文提出通过资源交织模型实现不同深度学习任务的并行，提高了集群和深度学习任务的效率。为了找到最佳的资源交织模型，创新性地设计出了基于带花树算法的任务分组策略，将其建模成无向图中的最大权匹配问题。经过实验，发现 Muri 可以降低作业运行平均 JCT 以及 Makespan。

参考文献

[1] DARGAN S, KUMAR M, AYYAGARI M R, et al. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning[J]. Archives of Computational Methods in Engineering, 2020: 1-22.

[2] GU J, CHOWDHURY M, SHIN K G, et al. Tiresias: A GPU Cluster Manager for Distributed Deep Learning[C]//Symposium on Networked Systems Design and Implementation. 2019.

[3] PENG Y, BAO Y, CHEN Y, et al. Optimus: an efficient dynamic resource scheduler for deep learning clusters[J]. Proceedings of the Thirteenth EuroSys Conference, 2018.

[4] XIAO W, BHARDWAJ R, RAMJEE R, et al. Gandiva: Introspective Cluster Scheduling for Deep Learning[C]//USENIX Symposium on Operating Systems Design and Implementation. 2018.

[5] XIAO W, REN S, LI Y, et al. AntMan: Dynamic Scaling on GPU Clusters for Deep Learning[C/OL]//14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). USENIX Association, 2020: 533-548. https://www.usenix.org/conference/osdi20/presentation/xiao.

[6] HWANG C, KIM T, KIM S, et al. Elastic Resource Sharing for Distributed Deep Learning[C]// Symposium on Networked Systems Design and Implementation. 2021.

[7] BAO Y, PENG Y, WU C. Deep Learning-based Job Placement in Distributed Machine Learning Clusters [J]. IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019: 505-513.