基于超参数调优服务优化 GPU 利用率的综述

肖妙[1)]

1) (武汉数字工程研究所, 武汉市 中国 430200)

摘 要 深度学习(DL)在视觉和语音等许多领域的惊人表现和迅速普及，超参数调优是深度学习模型开发中必不可少的一步，它以大量资源为代价提供更好的模型性能。本文以超参数调优为切入点，先介绍了现有超参数调优的一般工作流程，调优算法的分类，阐述现有 GPU 利用率不足的背景，现有的应用于优化超参数调优系统的两种先进技术，接着重点介绍不同于这两种技术的超参数调优服务 Hydro，建立了一个整体系统，自动应用新颖的超参数转移理论和多种系统技术生成代理模型，通过实验间融合来提高 GPU 的利用率；利用交错和异构感知试验分配挤压泡沫资源，充分利用集群资源。最后总结了近三年内关于提高 GPU 利用率的四篇研究，概括了他们提高 GPU 利用率的重点方法。

关键词 GPU 利用率，代理模型，超参数优化，数据中心技术
中图法分类号 **** DOI 号 *投稿时不提供 DOI 号* 分类号

A review on optimizing GPU utilization based on hyperparameter tuning services

Xiao miao[1)]

[1)](Wuhan Digital Engn Inst, Wuhan, China)

Abstract With the amazing performance and rapid popularity of Deep Learning (DL) in many domains such as vision and speech, hyperparameter tuning is an essential step in the development of deep learning models, which provides better model performance at the cost of significant resources. In this paper, we take hyperparameter tuning as an entry point, and first introduce the general workflow of existing hyperparameter tuning, the classification of tuning algorithms, elaborate the background of the existing underutilization of GPUs, the two existing state-of-the-art techniques applied to optimize hyperparameter tuning systems, and then focus on Hydro, a hyperparameter tuning service which is different from the two techniques, and build a holistic system that automatically applies the novel Hyperparameter Transfer theory and multiple system techniques to generate agent models, improve GPU utilization through inter-experiment fusion; and make full use of cluster resources by utilizing interleaved and heterogeneous perceptual experimental allocations to squeeze foam resources. Finally, current research on improving GPU utilization in the last three years is summarized.

Key words GPU utilization, agent models, hyperparameter optimization, data center technologies

---

收稿日期：年-月-日；最终修改稿收到日期：年-月-日 *投稿时不填写此项*. 本课题得到… …基金中文完整名称(No.项目号)、… …基金中文完整名称(No.项目号)、… … 基金中文完整名称(No.项目号)资助.作者名1(**通信作者**)，性别，xxxx年生，学位(或目前学历)，职称，是/否计算机学会(CCF)会员（提供会员号）,主要研究领域为*****、****.E-mail: **********.作者名2（**通信作者**），性别，xxxx年生，学位(或目前学历)，职称，是/否计算机学会(CCF)会员（提供会员号）,主要研究领域为*****、****.E-mail: **********. 作者名3（**通信作者**），性别，xxxx年生，学位(或目前学历)，职称，是/否计算机学会(CCF)会员（提供会员号）,主要研究领域为*****、****.E-mail: **********.(**给出的电子邮件地址应不会因出国、毕业、更换工作单位等原因而变动。请给出所有作者的电子邮件**)
第1作者手机号码(投稿时必须提供，以便紧急联系，发表时会删除): … …, E-mail: … …*此部分6号宋体*

1 引言

在 DL 集群中，GPU 通常未得到充分利用，Hydro 开发了 Hydro Tuner 组件，用于作业级优化，它的 Model Shrinker 组件是通过对目标模型的自动跟踪、缩放和参数化来获得代理模型，由于缩放代代理模型容易造成 GPU 利用率不足，故通过 Trial Binder 组件通过利用不同试验的模型架构一致性，实现了一种试验间融合机制，可自动将多个模型合并为一个模型，从而实现更高的 GPU 利用率和训练吞吐量。

除 HPO 作业外，GPU 数据中心中还并存着多种工作负载，如推理、调试和大规模分布式训练作业。近年来，随着基础模型（如 GPT3）的迅速普及，我国数据中心长期存在一些大型模型预训练工作负载。正如许多用户抱怨的那样，大部分机器都被大型模型训练工作占用，这些工作通常会持续数天到数周，导致其他工作处于饥饿状态。此外，管道并行通常被用来支持大型模型，方法是将模型分成几个阶段，并将它们放置在多个工作站上。然而，同步流水线并行本身就存在气泡，例如常用的 1F1B 策略。此外，不同流水线阶段之间的不平衡峰值内存问题进一步加剧了资源的低效率。Hydro 设计了气泡挤压器（Bubble Squeezer），利用气泡以交错执行的方式大大扩展了调整作业资源，几乎不会损害大型模型的整个训练过程。

2 背景

2.1 超参数调优

多年来，深度学习(DL)在许多领域得到了广泛的应用。然而，获取一个合格的 DL 模型并非易事，因为 DL 模型性能对超参数高度敏感，这些超参数控制着训练过程，需要在训练前设置。较差的超参数导致训练不稳定和模型质量较差。相反，调优的超参数可以显著提高模型性能。因此，超参数调优成为深度学习模型开发过程中的一种常见做法。

超参数调整（即超参数优化，HPO）旨在通过大量的配置探索来确定最佳超参数。HPO 作业的一般工作流程是：（1）用户指定一个要探索的超参数搜索空间；（2）调整算法创建一组训练试验，每个试验包含从搜索空间采样的一个唯一超参数配置；（3）HPO 系统协调试验的执行，直到找到最佳超参数配置。现有研究通常从调整算法或系统方面优化 HPO 效率：

2.2 HPO 调优算法

调优算法根据是否启用早期停止可分为两类：（1）单保真度算法（如随机算法、贝叶斯算法）要求每次试验都经过充分训练，这种算法准确但效率低；（2）多保真度算法（如 ASHA 算法、BOHB 算法）通过连续减半或曲线拟合策略停止不乐观的试验。这些算法效率很高，但由于使用了 "低保真 " 评估，可能会错过最佳超参数配置。Hydro 同时支持单保真和多保真算法。

2.3 超参数转移技术

Scaling Laws 根据经验研究了不同模型规模下批量大小和学习率的幂律函数，对于有限的配置，当处理超过 10 亿个参数的模型时，经验法则公式就会失效。故 DL 理论家还提出了一些新颖的超参数转移策略：即当模型在宽度和深度上增长/收缩时，如何相应调整超参数的规则。下面对基础理论做了简明扼要的背景概述，系统地建立了一个参数化的连贯理论框架：MLP 模型的特征学习效果 γ 与下列参数成正比。

$$\gamma \equiv \frac{L}{W^{1-p}}, p \in [0,1]$$

其中，w、L 分别表示神经网络的宽度和深度。为简单起见，我们假设隐层神经元的数量都是相似的，即 w1、w2、...wL-1 ~ w。p 是一个元参数，它将不同的参数化策略插值到一个统一的框架中，这是由固有策略决定的。我们的目标有两个：第一，保持一个固定的 γ，使超参数可以在不同大小的模型中转移；第二，努力争取一个更大的γ，以促进

更好的特征学习。为此，超参数转移有两个方向：

(1) 神经切线（NT）超参数转移策略（p = 0）。它是在研究无限宽神经网络时自然产生的神经切线核（NTK），它可以通过将深度与宽度一起缩放为 L ~ w 来保持 γ 不变。NTK 是一种解释神经网络在训练过程中演化的核方法，它是通过对线性化模型应用一阶泰勒展开而得到的。它属于懒惰训练机制，权重移动很小，因此线性化近似保持在初始参数附近，而不会学习特征，这是 NTK 理论在实践中的致命弱点。此外，NT 超参数转移策略也没有意义，因为在这种情况下，更宽的模型并不总是表现更好，这与常见的观察结果相冲突。

(2) 最大更新（MU）超参数转移策略（p = 1）。它概括了 1 隐藏层情况下的均场极限，应该是唯一一种保留了大规模神经网络的表征学习能力（非严格意义上的主动训练，与 NT 超参数转移策略的懒散训练相反）的超参数转移策略，这意味着训练不会在大宽度极限下变得琐碎或卡死在初始化阶段。通俗地说，它旨在解决输入层的更新速度远慢于输出层的问题，并使所有隐激活的更新速度在宽度上保持一致。

本文介绍的 Hydro 采用了 MU 超参数转移策略， 通过应用新颖的超参数转移技术，可以用小得多的代理模型来调整模型。为了更清楚地说明基于代理的调整效果，我们在两个小模型上使用了 Hydro 超参数转移策略，并绘制了它们的收敛训练损失与一系列学习率的对比图，如图 1 所示。具体来说，目标 MLP 模型包含两个隐藏层（宽度=4096），我们在 CIFAR-10 上使用 SGD 对其进行训练。同样，目标 Transformer 模型包含两个 TransformerEncoder 层（宽度=4096，即 dmodel），我们在 WikiText-2 上用 Adam 对其进行训练。此外，我们用不同的缩放比例 S 生成代用模型，较小的模型用较浅的蓝线表示。例如，S=2 表示宽度=2048 的模型。显然，传统的训练范式（图 1 (a,c)）无法在不同大小的模型中共享最佳超参数，而且最佳学习率也会有数量级的变化。然而，Hydro 超参数转移策略（图 1 (b,d)）使代用模型与目标模型保持大致相同的最佳学习率，这意味着基于代用模型的调整是可行的。此外，Hydro 超参数转移策略还能提供更好的性能，因为经过调整的 MLP 和 Transformer 都能获得比同类模型更低的训练损失。一个直观的解释是，传统范式的学习率必须抑制对数的激增，但前面的层并没有明显的学习效果。

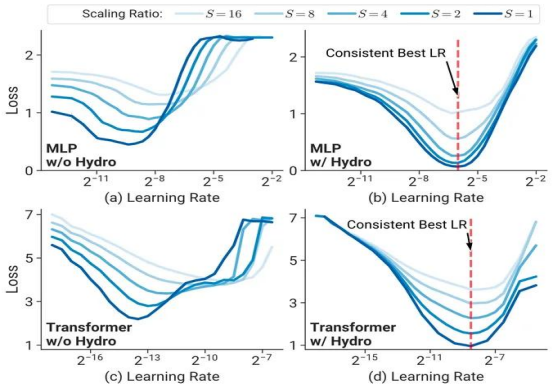

图 1 不同宽度的 MLP（a、b）和 Transformer（c、d）的训练损失 loss 与学习率 learning rate 的关系 S 表示模型缩放比

2.4 流水线并行和交错执行

除 HPO 作业外，GPU 数据中心中还并存着多种工作负载，如推理、调试和大规模分布式训练作业。近年来，随着基础模型（如 GPT3）的迅速普及，我国数据中心长期存在一些大型模型预训练工作负载。正如许多用户抱怨的那样，大部分机器都被大型模型训练工作占用，这些工作通常会持续数天到数周，导致其他工作处于饥饿状态。此外，管道并行通常被用来支持大型模型，方法是将模型分成几个阶段，并将它们放置在多个工作站上。然而，同步流水线并行本身就存在气泡，例如常用的 1F1B 策略。此外，不同流水线阶段之间的不平衡峰值内存问题进一步加剧了资源的低效率。

最近的研究从多个角度利用了流水线并行性引起的气泡。Bamboo 将冗余计算填入气泡，为可抢占式云实例提供弹性和快速恢复。EnvPipe 有选择性地降低气泡期的 SM 频率，以节省能耗。与

它们不同的是，Hydro 利用气泡通过交错执行来训练 HPO 试验，这受到了之前一些工作的启发。

2.5 GPU 利用率低

最近的调度工作报告可知，由于涉及中型或小型模型的大量训练工作，gpu 在深度学习集群中通常未得到充分利用。此外，尽管基础模型在集群中使用的趋势越来越多，但由于巨大的通信开销和管道并行性中存在气泡，大规模模型往往不能充分利用硬件资源。

在 GPU 集群中，超参数调优工作非常普遍，并且占用大量资源。根据微软的报告，“大约 90%的模型需要超参数调优，每个调优作业中位数包含 75 次试验。”但是，现有的调优系统只管理请求资源上的试验，并且缺乏与集群调度器的交互。同时，DL 调度器也忽略了超参数调优作业中固有的逐渐减少的硬件需求的明显特征。因此，集群会遇到资源不平衡的问题:活动调优作业始终占用静态资源，使其中一些资源空置，而排队作业无法从调度器请求这些空闲资源。这导致了严重的排队延迟，当长时间的大规模模型训练任务共存并且占用了大部分集群资源时，排队延迟会加剧。

2.6 超参数转移技术

虽然 HPO 作业在 GPU 数据中心非常普遍，但集群调度人员通常将其视为一般的训练工作负载，而没有任何特定的设计。另一方面，HPO 系统 与集群资源无关。这导致集群级效率低下，如作业排队延迟长、GPU 利用率低等。然而，HPO 作业的独特性为更有效的调整带来了机会。

(1) 对试验吞吐量不敏感。与一般的 DL 作业不同，HPO 作业对部分试验吞吐量减慢的容忍度更高。因此，我们可以利用数据中心长期存在的大型语言模型训练作业的短暂泡沫资源，运行更多的试验。⑵ 资源需求减少。多保真度 HPO 作业通常在开始阶段探索大量试验，然后逐渐降低搜索并发度。在最后阶段，只需利用少量试验。因此，我们不仅可以逐步减少资源总量，还可以适当利用异构 GPU 资源。如图 2 (b)所示，随着 GPU 计算能力的快速发展，对于大多数小规模试验而言，GPU 越来越容易被利用不足。将试验分配给适当的 GPU 可以显著提高整个集群的效率，而不会影响单个 HPO 作业的时间跨度。

3 基于代理的超参数调优服务

Hydro 是基于代理的超参数调优服务，可以在作业级和集群级粒度上优化调优工作负载。具体来说，它由两个关键部分组成，每个关键部分又分为三个组件:

(1)Hydro Tuner 通过缩放、参数化和融合自动生成和优化代理模型;

(2) Hydro Coordinator 通过自适应地利用短暂和异构资源，提高了调优效率和集群范围内的资源利用率。

3.1 Hydro Tuner

Hydro Tuner 是 Hydro 服务的核心组件，用于作业级优化，它的 Model Shrinker 组件是通过对目标模型的自动跟踪、缩放和参数化来获得代理模型，由于缩放代代理模型容易造成 GPU 利用率不足，故通过 Trial Binder 组件通过利用不同试验的模型架构一致性，实现了一种试验间融合机制，可自动将多个模型合并为一个模型，从而实现更高的 GPU 利用率和训练吞吐量。

3.1.1 Model Shrinker 导致 GPU 利用率不足

在 DL 集群中，GPU 通常未得到充分利用。图 2 (a) 绘制了我们集群中一个分区一周 GPU 利用率的累积分布函数 (CDF)，以及阿里巴巴的跟踪数据，以供参考。可以发现，在阿里巴巴和上海人工智能实验室中，分别只有 16% 和 35% 的 GPU 利用率超过 50%。如果采用基于 Hydro 代理的调整技术，这一问题将更加严重。例如，图 3 显示了在 ImageNet 上训练 WideResNet-50 的模型缩放效果，其中 GFLOPs 大约呈反平方（c1/s2）趋势下降，内存占用大约呈 c1/s+c2 趋势下降（c_i 表示常数）。这意味着模型缩放可以显著降低计算开销，但资源更容易未得到充分利用。

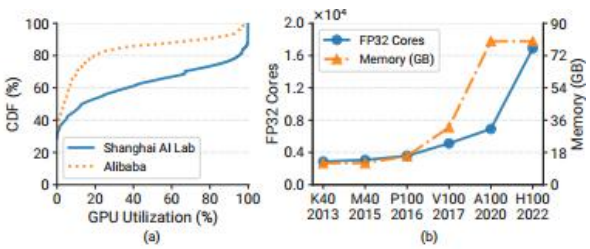

图 2（a）我们的集群和阿里巴巴 GPU 生产集群中一个分区的 GPU 利用率分布[115]。(b) 英伟达数据中心 GPU 能力的指数增长。x 轴： GPU 型号名称和发布年份

Model Shrinker 模块大大减少了每个试验的计算量（图 3），但它不可避免地会产生资源利用不足的问题，使中小型目标模型（如部署在边缘设备上的模型）的性能下降。

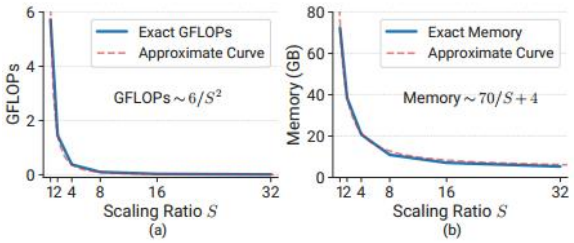

图 3 WideResNet-50 的模型缩放效应(a) 模型 GFLOPs（千兆浮点运算）(b) GPU 内存

3.1.2 Trial Binder 通过试验间融合机制提高 GPU 利用率

Model Shrinker 模块缩放代理模型会导致 GPU 利用率不足，为了解决这个问题，受 JAX vmap 函数的启发，Hydro 实现了一种试验间融合机制，可自动将多个模型合并为一个模型。由于 HPO 任务的特性：本质上是一组完全相同的模型（或略有突变），因此可以融合多个试验的运算器。与传统的 GPU 共享机制（如 MPS）相比，Hydro 可以实现更高的训练吞吐量、GPU 利用率和更低的内存占用。

Trial Binder 模块通过绑定多个试验和融合内部运算符来进一步优化代理模型，从而更好地利用加速器。我们在图 4 的底部对其机制进行了说明。它将一组可融合的试验合并为带有分组算子和优化器（③）的 Hydro 试验。为了进一步加快训练速度，我们自动对融合（内部）代用模型进行即时编译（JIT），以获得快速灵活的融合（内部）内核（④）（最后一个层距较近的模型代表了通过试验内融合减少的内存限制操作）。

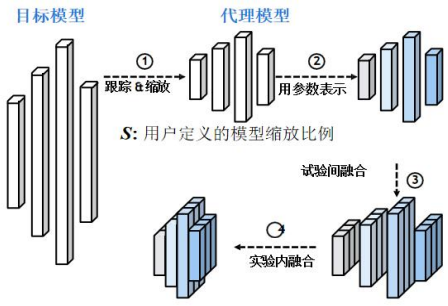

图 4 Model Shrinker（①、②）和 Trial Binder（③、④）的说明。每条的长度代表层宽。

（1）实验间融合

在 HPO 作业中，有大量具有相同或相似模型结构的试验。JAX vmap 通过将每个输入沿指定的轴进行矢量化来返回目标函数的批量版本，受此启发，可以通过融合多个试验的操作符来将多个试验批量合并为一个试验。Hydro 实现了一种试验间融合机制，可自动绑定代理模型。具体来说，试验绑定器会遍历跟踪的代理模型，并根据预定义的融合规则和试验规划器确定的融合计数 F，用分组的 hydro.nn 运算符替换 torch.nn 运算符。这些功能都是自动执行的，开发人员通常无需了解其原理和修改代码。

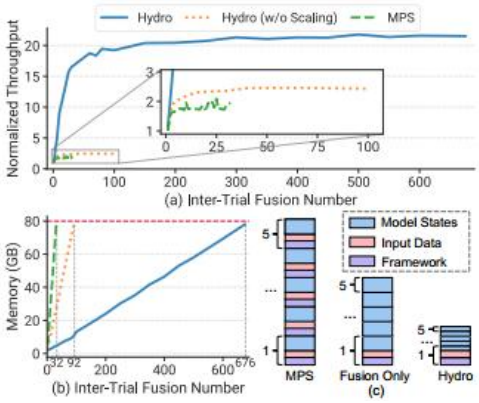

图 5 ResNet-18 的试验间融合效果。(a) 整个融合代用模型相对于目标模型的累积。(b) 不同融合次数对 GPU 内存的影响。红色横线表示 A100 内存限制。(c) GPU 与 MPS、Hydro 和 Fusion（无缩放）共享的 5 个模型的内存占用细节示意图

图 5 显示了在 ResNet-18 (S = 8) 上将模型扩展与试验间融合相结合所产生的非凡效果，在 CIFAR-10 上进行了测试，批量大小=256。很明显，Hydro 能够在单个 A100 GPU 上同时训练令人印象深刻的 676 个模型。与传统的 GPU 共享机制 MPS（MIG 具有类似的性能）相比，Hydro 的训练吞吐量提高了 10 倍以上，GPU 内存节省了 20 倍以上。如果我们直接将试验间融合应用于目标模型（不进行缩放），吞吐量的提高则相对有限。此外，我们在图 5（c）中提供了内存占用减少的直观解释。模型状态（蓝色块）包括与模型训练相关的所有方面，如模型权重、梯度、激活和优化器状

态。此外，除了更好地利用 GPU 和更高的吞吐量外，试验间融合还减轻了伴随数据加载融合而产生的 I/O 压力。

（2）实验内融合

Hydro 支持自动模型融合，以进一步加快基于 nvFuser 编译器后端的训练速度。尽管之前的大量工作证明，算子融合可以通过更好的内存定位提高训练吞吐量，但由于其编译开销较高，并不总能为 HPO 工作负载带来好处。为此，Hydro 临时采用了试验内融合。为简单起见，Hydro 目前仅适用于具有确定性训练步骤的试验，例如应用单保真度调整算法时的所有 Hydro 试验，以及使用转移的超参数训练目标模型的试验。

3.2 Hydro Coordinator 提高 GPU 和集群资源利用率

Hydro Coordinator 用于集群级优化，它的 Bubble Squeezer 组件利用大型模型的管道气泡，通过与流水线支持的大型模型训练任务交错训练来扩展调谐资源，有效地利用了每个节点上被称为气泡的空闲时间间隔，不会牺牲大型模型的吞吐量，从而优化 GPU 和集群资源利用率。

Hydro 设计了气泡挤压器（Bubble Squeezer），利用气泡以交错执行的方式大大扩展了调整作业资源，几乎不会损害大型模型的整个训练过程。Hydro 试验具有以下独特功能，非常适合气泡交错执行： (1) 对吞吐量不敏感。与一般的 DL 训练任务不同，调整任务更能容忍部分试验的减速。这启发我们挤压气泡的空余资源，以暂停和恢复的方式执行试验。(2) 确定性资源模式。一般的小规模工作负载（如调试）对资源的需求是未知和不可预测的。然而，Hydro 会对 HydroTrials 的资源消耗情况进行剖析和记录，从而在 HydroTrials 与大型模型共享时减轻潜在的内存不足（OOM）问题。(3) 弹性试验规模。基于 Model Shrinker，缩放模型的内存占用比原始模型小得多（图 3），这意味着我们通常不需要在主机托管期间交换其 GPU 内存。此外，我们还可以通过 Trial Binder 根据剩余 GPU 内存动态调整试验融合次数。

为了清楚地说明 Bubble Squeezer 的工作原理，我们首先在图 6（a）中介绍了 1F1B 流水线并行。它通过点对点通信在不同工人之间传输部分模型在前向和后向传递过程中的中间激活，因此每个工人都不能连续使用 GPU。对于工人 1 来说，在最后一个微批次（蓝色块 4）的前向传递之后，它必须等待第一个微批次（绿色块 1）的后向传递，这使得 GPU 长期处于闲置状态。其他工人也呈现出类似的气泡模式，但由于需要存储的微批次激活次数较少，因此占用的 GPU 内存也较少。

在图 6 (b) 中，Hydro 将四个不同大小的 HydroTrials 与大型模型训练工作量交织在一起。每个试验都以暂停和恢复的模式执行，以挤压气泡。由于 Hydro Tuner 已使用 hydro.nn 对每一层进行了跟踪和标注，我们进一步在试验的每个模块上注册了钩子，以支持在每一层的前向和后向传递中按需暂停和恢复。当存在大型模型训练任务时，Hydro 会与数据中心调度员协调，从该模型中获取更多 GPU，并将其标记为短暂资源。对于大型模型，我们还在其训练框架（即 DeepSpeed ）内实施了一个相应的钩子，以报告其训练进度和资源消耗情况。在 DeepSpeed 的流水线并行性下，每个 Worker 都会执行相应的流水线。因此，我们实施了一种细粒度同步机制，通过拦截 NCCL 内核的 CUDA 流状态，确保 HydroTrials 只能在气泡内执行。Hydro 还能进一步调整融合次数，以适应剩余内存并提高 GPU 利用率。在大型模型训练气泡的开始和结束时，我们通过 Linux 信号控制试验模型训练的恢复和暂停。细粒度的暂停-恢复控制消除了同时运行的 CUDA 内核对性能的干扰。

Bubble Squeezer 效果最好的情况可分为：

（1）在 HPO 作业方面，使用多保真度调整算法时，Hydro 更为有效，因为这些算法允许使用短暂资源在几个历元内结束大多数试验，并在独占资源上立即执行顶层试验，以避免交错减速可能造成的阻塞。

（2）层数较少的模型是首选，因为它们容易在冒泡时间内完成整个迭代，并且需要相对较少的内存来支持较高的融合数。

（3）至于流水线大型模型方面，如果预训练任务在更多服务器上有更多的流水线阶段，这意味着更高的冒泡率和更多的短暂资源，Hydro 可以获得更好的性能。大型模型预训练作业通常可以同时支持多个不同的 HPO 作业交织，并在预训练过程中加速数十个甚至数百个 HPO 作业（取决于其资源和持续时间规模）。

此外，在某些情况下，一些缩放模型仍然过大，

无法在预训练模型的任何 GPU 上分配。由于在我们的方案中内存交换开销很大，Hydro 不支持像 Bamboo 这样的卸载技术。因此，Bubble Squeezer 无法支持此类模型。

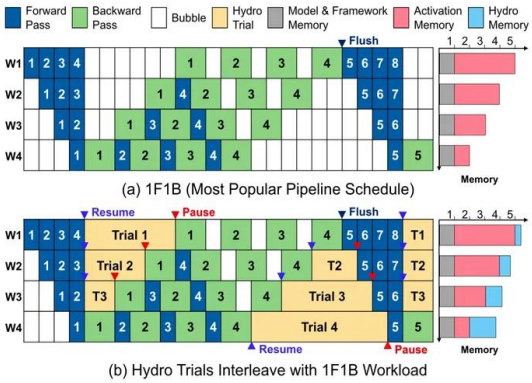

图 6 (a) 1F1B 管线和 (b) Hydro Bubble Squeezer 的示意图，包含四个管线阶段和四个微批次。请注意，右侧内存图只能反映相同色块在不同工人之间的相对关系。

4 现有研究成果

（1）在生成推理过程中提高 GPU 利用率，以获得更高的吞吐量。

目前的一个 LLM 服务框架为 KV 缓存保留了最大序列长度的内存，以保证生成完整的序列。这就限制了我们使用较小的批处理量，导致 GPU 利用率降低，尤其是吞吐量降低。Jin Y, Wu C F, Brooks D 等人认为，设计一个预先知道输出序列的系统可以缓解这一问题。为此，提出了 S3，这是一个旨在为基于 Transformer 的生成模型提供高吞吐量的框架。S3 利用预测器来估算生成序列的输出长度，并对其进行相应的调度，以最大限度地提高吞吐量。此外，S3 还能处理潜在的预测错误，以保证可靠性。通过为不同的输入分配不同大小的内存，S3 认识到并非所有序列都应得到同等对待。这种方法扩展了延迟和吞吐量边界之间的传统权衡，为优化延迟-吞吐量权衡铺平了道路。

（2）弹性训练，具有一致的准确性和更高的 GPU 利用率。

分布式同步 GPU 训练通常用于深度学习。使用固定数量 GPU 的资源限制使得大规模训练工作遭受资源分配排队时间过长之苦，并降低了集群利用率。适应资源弹性可以缓解这一问题，但由于缺乏将模型训练过程与资源分配解耦的能力，往往会带来不一致的模型精度。Mingzhen Li 等人提出的 EasyScale 是一种弹性训练系统，可在同构和异构 GPU 的资源弹性条件下实现一致的模型精度。EasyScale 严格保留了数据并行训练行为，仔细追踪一致性相关因素，利用深度学习特性实现了 EasyScaleThread 抽象和快速上下文切换。为了利用异构集群，EasyScale 根据任务内/任务间调度器动态分配工作者，最大限度地减少负载失衡，最大限度地提高聚合作业吞吐量。在一个在线服务集群中部署后，EasyScale 能让训练作业伺机利用闲置的 GPU，从而将集群的整体利用率提高了 62.1%。

通过 EasyScale，Mingzhen Li 等人成功证明了将 DL 训练过程与底层资源分配解耦，从而在弹性条件下实现精度一致的模型训练。具体来说，EasyScale 提出了几项创新来解决弹性训练过程中的非确定性问题，具体包括：1）引入 EST 抽象以在弹性和异构情况下保留训练行为；2）将分散在 DLT 软件栈中的非确定性行为进行寻源并加以解决；3）利用异构 GPU 集群开发任务内和任务间调度器。

（3）QoS 感知动态资源分配，提高 GPU 的利用率和能效。

尽管 GPU 在数据中心中是不可或缺的，但在 GPU 上进行任务整合时，要满足服务质量 （QoS）要求是极具挑战性的。以前的工作大多依赖于静态任务或资源调度，无法处理运行时的 QoS 违规。此外，现有工作未能利用批处理任务的计算特性，从而浪费了在提高 GPU 利用率的同时降低功耗的机会。

针对上述问题，Q Sun 等人提出了一种新的运行时机制 SMQoS，以满足 GPU 上任务整合的 QoS 要求，同时提高利用率和能效。在运行时，SMQoS 监控 LS 任务的性能并动态调整 SM 分配以满足 QoS 目标。同时，基于协同运行任务的混合，SMQoS 可以为批处理任务分配更多的 SM 以提高吞吐量，也可以为空闲 SM 供电以降低功耗。此外，Q Sun 等人基于 SMQoS 的思想，在真实的 GPU 硬件上实现了 RH-SMQoS，在提高 GPU 利用率的同时，为 LS 任务提供 QoS。实验结果表明，SMQoS 在提高 GPU 利用率和降低功耗方面比最先进的方法更有效，同时满足 LS 任务的 QoS 要求。此外，RH-SMQoS 可以满足 LS 任务的 QoS，批处理任务的吞吐量高于 Volta-MPS 和 SlateQoS。对于未来的工作，我们希望扩展 SMQoS 以支持包含具有依赖关系的不相同内核的复杂 GPU 任务的 QoS。

（4）实现 QoS 感知和提高空间多任务 GPU 的利用率

数据中心使用 GPU 提供面向用户的新兴服务所需的大量计算吞吐量。面向用户的服务的昼夜用户访问模式有力地推动了共用 GPU 的应用，从而提高了 GPU 的利用率，之前的工作主要集中在多核处理器和传统的非抢占式 GPU 上实现共用。然而，当前的 GPU 正朝着空间多任务的方向发展，这为消除违反服务质量的情况带来了一系列新的挑战。针对这些原因，Wei Zhang 等人提出了 C-Laius，这是一种运行时系统，它能为共址应用精心分配计算资源，以最大限度地提高批处理应用的吞吐量，同时保证面向用户的服务所需的 QoS。

C-Laius 提高了空间多任务 GPU 的硬件利用率，同时保证了面向用户的应用程序的服务质量要求。为实现这一目标，C-Laius 支持精确的任务持续时间预测、竞争感知资源分配和进度感知滞后补偿器。通过使用面向用户的新兴服务对 C-Laius 进行评估，我们证明了 C-Laius 在消除因计算资源不足、全局内存带宽争用和其他未管理共享资源争用而导致的 QoS 侵犯方面的有效性。与最先进的技术相比，C-Laius 将共址批处理应用的吞吐量平均提高了 20.8%，而不会违反面向用户服务 99%ile 延迟的 QoS。至于多个面向用户的应用程序，C-Laius 可确保不违反 QoS，同时将总体吞吐量平均提高 35.9%。

5 总结

在本文中，我以超参数调优为切入点，先介绍了现有超参数调优的一般工作流程，调优算法的分类，阐述现有 GPU 利用率不足的背景，现有的应用于优化超参数调优系统的两种先进技术，接着重点介绍不同于这两种技术的基于代理的超参数调优服务 Hydro，建立了一个整体系统，自动应用新颖的超参数转移理论和多种系统技术生成代理模型，通过实验间融合来提高 GPU 的利用率；利用交错和异构感知试验分配挤压泡沫资源，充分利用集群资源。最后总结了近三年内关于提高 GPU 利用率的四篇研究，概括了他们提高 GPU 利用率的重点方法。

参 考 文 献

[1] Jin, Yunho et al. “S3: Increasing GPU Utilization during Generative Inference for Higher Throughput.” ArXiv abs/2306.06000 (2023): n. pag.

[2] Li, Mingzhen et al. (2023).“EasyScale: Elastic Training with Consistent Accuracy and Improved Utilization on GPUs.” ACM 2023 1-14.

[3] Q Sun, L Yi, et al. “QoS-aware dynamic resource allocation with improved utilization and energy efficiency on GPU,”Parallel Comput., vol. 113, 2022, Art. no. 102958

[4] Wei Zhang, et al. “Towards QoS-Awareness and Improved Utilization of Spatial Multitasking GPUs.”,IEEE866-879

[5] Hu, Q., et al. (2023).“Hydro: Surrogate-Based Hyperparameter Tuning Service in Datacenters.” USENIX Symposium on Operating Systems Design and Implementation.

[6] X. Liu,et al."Efficient Hyperparameters optimization Through Model-based Reinforcement Learning and Meta-Learning," IEEE , 2020, pp. 1036-1041.

附录 论文汇报记录

1. GPU 利用不足的原因

1.1 由于巨大的通信开销和管道并行性中存在气泡，大规模模型不能充分利用硬件资源。

1.2 虽然 HPO 作业在 GPU 数据中心非常普遍，但集群调度人员通常将其视为一般的训练工作负载，而没有任何特定的设计。另一方面，HPO 系统 与集群资源无关。这导致集群级效率低下，如作业排队延迟长、GPU 利用率低等。

1.3 本文使用 Model Shrinker 组件缩放代用模型，大大减少了每个试验的计算量，但它不可避免地会造成 GPU 利用率不足的问题，使中小型目标模型（如部署在边缘设备上的模型）的性能下降。

2. 本文使用 HPO 作业的独特性来提高 GPU 利用率：

2.1 跨模型融合。可用于提高资源利用率，利用不同试验的模型架构一致性，将它们融合为一个模型，从而实现更高的 GPU 利用率和训练吞吐量。Hydro 还能进一步调整融合次数，以适应剩余内存并提高 GPU 利用率。

2.2 Hydro Coordinator 用于集群级优化，它的 Bubble Squeezer 组件利用大型模型的管道气泡，通过与流水线支持的大型模型训练任务交错训练来扩展调谐资源，有效地利用了每个节点上被称为气泡的空闲时间间隔，不会牺牲大型模型的吞吐量，从而优化 GPU 和集群资源利用率。