arXiv:2304.03946v1 [cs.DC] 8 Apr 2023

FlexMoE：通过动态设备放置扩展大规模稀疏预训练模型训练

XIAONAN NIE，中国北京大学XUPENG MIAO，美国卡内基梅隆大学ZILONG WANG，中国微软
ZICHAO YANG，美国卡内基梅隆大学JILONG XUE，中国微软研究院LINGXIAO MA，中国微软研究院
GANG CAO，中国北京智源人工智能研究院BIN CUI*，中国北京大学

随着数据量不断增加，使用大规模预训练模型将知识存储在海量模型参数中的趋势日益明显。这些模型的训练由大量稠密代数运算组成，需要消耗巨量硬件资源。近年来，稀疏门控混合专家模型（混合专家模型）日益流行，并在各种下游任务中展现出出色的预训练可扩展性。然而，由于路由不平衡和波动问题，这种稀疏条件计算在实际系统中的效果可能不如预期。总体而言，混合专家模型正成为数据生命周期中的一种新型数据分析范式，并面临着前所未有的规模、复杂度和粒度方面的独特挑战。

在本文中，我们提出了一种新颖的深度神经网络训练框架 FlexMoE，能够系统且透明地解决动态数据流导致的低效率问题。我们首先对混合专家模型训练中的问题与机遇进行了实证分析，这促使我们通过动态专家管理和设备放置机制来克服路由不平衡和波动问题。随后，我们在现有 DNN 运行时之上引入了一个新颖的调度模块，用于监控数据流、制定调度计划，并根据实时数据流量动态调整模型到硬件的映射。在训练过程中，我们采用了一种简单但高效的启发式算法，动态优化设备放置。我们在自然语言处理模型（例如 BERT 和 GPT）及视觉模型（例如 Swin）上进行了实验。结果表明，与现有系统相比，FlexMoE 在真实世界工作负载上能够实现更优的性能——FlexMoE 平均性能优于 DeepSpeed 1.70×，最高可达 2.10×；平均性能优于 FasterMoE 1.30×，最高可达 1.45×。

CCS 概念：• **计算方法** → **自组织**；大规模并行算法 *s.*
A附加关键词和短语：深度学习系统、分布式计算、稀疏模 el

*Bin Cui 是通讯作者。

作者地址：Xiaonan Nie，xiaonan.nie@pku.edu.cn，北京大学计算机科学学院与高可信软件技术重点实验室（教育部）；Xupeng Miao，xupeng@cmu.edu，美国卡内基梅隆大学；Zilong Wang，zilongwang@microsoft.com，中国微软；Zichao Yang，yangtze2301@gmail.com，美国卡内基梅隆大学；Jilong Xue，jxue@microsoft.com，中国微软研究院；Lingxiao Ma，lingxiao.ma@microsoft.com，中国微软研究院；Gang Cao，caogang@baai.ac.cn，中国北京智源人工智能研究院；Bin Cui，bin.cui@pku.edu.cn，北京大学（青岛）计算机科学学院与高可信软件技术重点实验室（教育部）、计算社会科学研究所，北京大学，中国北京。

允许为个人或课堂使用而免费制作本作品的全部或部分数字或纸质副本，前提是不得为营利或商业利益而制作或分发副本，并且副本中包含此声明和第一页的完整引用。必须尊重本作品中由非作者拥有的组件的版权。经注明出处后进行摘要编写是允许的。若要以其他方式复制、再版、发布到服务器或重新分发到列表，则需要事先特别许可和/或支付费用。请通过permissions@acm.org 申请许可。© 2023年 版权所有归所有者/作者所有。出版权授权予ACM。2836-6573/2023/5-ART110 $15.00 https://doi.org/10.1145/3588964

ACM 引用格式：
Xiaonan Nie、Xupeng Miao、Zilong Wang、Zichao Yang、Jilong Xue、Lingxiao Ma、Gang Cao 和 Bin Cui。2023年。FlexMoE：通过动态设备放置扩展大规模稀疏预训练模型训练。*Proc. ACM Manag. Data* 1，1，文章 110（2023年5月），共19页。https://doi.org/10.1145/3588964

1 引言

大规模预训练模型（PTMs），例如 Transformer模型，推动了深度学习（DL）在各种复杂任务上的发展，包括自然语言处理，例如 BERT [9], GPT [6], T5 [41], 计算机视觉，例如 ViT [10], Swin [25], 广告推荐，例如 M6 [24], 等。这些模型也被称为基础模型，因为它们在数百 GB 的数据上进行训练，并且可以适应广泛的下游任务，例如特定任务微调。然而，近期研究 [6, 20]表明，模型质量会随着数据规模、参数规模和计算预算呈幂律增长。当前最先进的基础模型已过度扩展至数万亿个参数，因此训练成本极其高昂且耗时。

为避免触及可训练性瓶颈，一项知名的研究路线采用稀疏门控混合专家（MoE） [45]结构，使预训练模型具备成本效益。这种稀疏混合专家架构通过扩展专家网络来扩大模型参数规模，同时保持计算预算稳定。具体而言，每个MoE层由一个门控网络和一系列专家（例如数十个）组成，其中门控网络仅将输入令牌（即该层的输入）路由至少量专家，而不是全部专家，从而提高计算效率。MoE 模型已成功用于提升各种预训练模型的性能，例如语言模型（如 GLaM [11] 在 SuperGLUE自然语言处理基准测试 [47]中实现了最先进的性能）和视觉模型（如 V-MoE [43] 仅用一半的计算时间就达到了最强视觉Transformer的水平）。

由于表现优异，近年来，大规模混合专家模型在工业企业中引起了广泛关注，企业在构建真实世界的大数据应用时纷纷采用这类模型，包括谷歌 [11],Meta [44],微软 [42] 和阿里巴巴 [24]。这类数据密集型工作负载正成为数据科学研究领域的一种新型数据分析范式。随着混合专家模型的迅速扩展，数据生命周期（尤其是模型开发、部署和管理环节）迎来了前所未有的机遇，同时也面临着挑战。例如，谷歌创建了 ST-MoE [50] 模型，以帮助提升其搜索引擎的问答质量；该模型在 AIRC 排行榜 [5], 上取得了最先进的成绩，例如，使用混合专家模型后，成绩从 81.40% 提升至 86.52%。然而，这样的大型混合专家模型不仅需要大量数据进行训练，例如 ST-MoE 需要 1.5T 个令牌，还会在训练过程中消耗大量昂贵的计算资源。因此，有必要解决现有系统中隐藏的数据管理问题，并通过优化来克服这些挑战。

由于单个混合专家层就可能轻易超出有限的图形处理器内存，先前的分布式训练技术（例如，数据并行 [23] 和模型并行 [30, 32, 33, 36]）已逐渐无法满足需求。为了训练大型 MoE 模型，Lepikhin 等人 [21] 提出了专家并行 技术，将专家分布到多个图形处理器上，以适应有限的图形处理器内存。对于每个训练令牌，在门控网络确定路由后，该令牌会被发送到其目标专家所在的图形处理器。由于适应了 MoE 结构的特殊特性，专家并行已成为多个知名框架中训练基于 MoE 的模型最广泛使用的技术之一，包括 DeepSpeed-MoE [42],Tutel [19] 和 HetuMoE [37]。

工作负载不平衡的挑战。 尽管专家并行已得到广泛应用 [21], 但它仍受到混合专家层稀疏条件计算特性所导致的严重工作负载不平衡问题的困扰。具体而言，输入令牌以细粒度方式组织。

并由门控网络路由至各个专家。我们发现，输入令牌对专家的偏好高度不均衡，且路由结果在训练过程中持续变化 [34]。正如 [17, 22, 26] 所报告并经我们的实证研究验证的那样，这种工作负载不平衡可能导致效率显著下降和严重的资源利用不足。

当前状况。 为了解决工作负载不平衡问题，大多数现有研究都会在每个混合专家层上使用辅助均衡损失，以强制实现均衡路由。为控制系统效率（工作负载均衡）与统计效率（模型质量）之间的权衡，负载均衡损失还会配以可调节的惩罚权重。此外，Fedus 等人 [12]引入了专家容量，以避免向专家分配过多输入词元。换言之，超出专家容量的词元将被专家丢弃。尽管如此，这些方法只能缓解工作负载不平衡问题，而无法从根本上保证负载均衡。此外，这些方法让用户陷入两难境地：我们是否应该为了系统效率而牺牲模型质量，以及牺牲多少？例如，为负载均衡损失设置较大的惩罚权重（或为专家设置较低的容量），有助于实现专家间的负载均衡，却可能损害模型质量，因为大量词元会被路由到某些不太理想的专家（或被丢弃）；反之亦然。

直观来说，上述方法试图尽可能均匀地调整令牌的数据流，即强制所有专家处理几乎相同数量的令牌。这些方法易于适配现有的专家并行系统，因为用户只需更改模型的定义（例如负载均衡损失和容量），无需对底层深度学习系统的实现进行任何修改。换句话说，这类方法适合不熟悉深度学习系统的用户，但他们不得不牺牲模型质量来换取训练效率。不同于可能导致模型质量下降的以往模型修改方法，我们尝试从系统角度优化工作负载不平衡：

> 如何设计一个在不影响模型质量的同时实现最大系统效率的分布式混合专家训练系统？

方法概要。 在本工作中，我们提出了 FlexMoE，这是一种面向大规模稀疏混合专家模型的新型分布式训练系统。我们从专家放置这一全新视角解决**路由不平衡** 问题：不同于传统专家并行中将每个专家放置在单个 GPU 设备上，也不同于传统数据并行中将所有专家复制到所有 GPU 设备上，我们引入了**细粒度复制专家并行**，选择特定的高负载专家，将其复制到多个设备上，并将输入令牌分散到各个副本。对于稀疏混合专家模型而言，这并非易事，因为这些专家的设备放置会改变原始输入数据分配方式（即不同专家之间的全互联通信），并带来额外的同步开销（即复制专家之间的全归约通信）。FlexMoE 通过精确建模混合专家层的实际执行过程，估算出用于确定具体专家复制方案的开销。

我们进一步研究了各种混合专家训练工作负载中稀疏激活专家的不均匀分布。我们的关键发现是，混合专家模型在迭代训练过程中表现出**路由波动**。我们观察到，不平衡的专家偏好分布在训练过程中持续且平稳地变化（将在第 2.4节中进行深入描述）。门控网络的训练高度不稳定 [7]，因为基于梯度的优化算法会强化之前的路由决策，直到达到某个临界点并摆脱错误的学习方向。为应对这一障碍，我们在 FlexMoE 中设计了一种**动态专家管理**机制，以在训练过程中自适应地调整专家到设备的映射和令牌分配。具体而言，我们采用数据驱动方法，通过监控流量趋势，在训练过程中自适应地改变专家放置。例如，FlexMoE 逐步

为工作负载不断增加的专家扩展资源，以加快计算；并收缩工作负载不断减少的专家的资源，以降低副本同步开销。

FlexMoE 被设计为构建于现有深度神经网络框架之上的新型调度模块，用于监控数据流量、制定调度计划，并动态调整数据流图与分布式设备之间的映射关系。具体而言，我们提供了三种专家放置调整原语（即扩展、收缩和迁移），以灵活管理专家放置；同时引入门控流量控制机制，以实现自主的全局流量优化。在此基础上，我们设计了一种简单而有效的算法，用于估算每次调整带来的收益或开销，从而确定最优调度计划。

最后但同样重要的是，我们在 PyTorch [38] 之上实现了 FlexMoE，并通过全面的实验验证了本工作的优越性。在一个包含 64 个图形处理器 A100 的集群上训练多种热门的基于混合专家的预训练模型（例如 BERT、GPT、Swin）并执行各种任务时，与现有最先进的混合专家训练系统相比，FlexMoE 的加速比最高可达 2.1× 。此外，在模型质量方面，FlexMoE 始终优于这些系统，因为我们不会以牺牲令牌路由为代价来调整专家放置。

论文组织。在第 2 节中，我们讨论采用专家并行训练稀疏混合专家模型的低效性，并探索通过系统级优化来优化此类问题的机会。在第 3 节中，我们介绍 FlexMoE的系统设计，该设计利用 vExpert 抽象来促进动态专家管理和设备放置。我们在第 4 节中详细介绍系统实现，并在第 5 节中评估 FlexMoE 的有效性。

2 背景与动机

我们首先介绍本文中使用的符号。

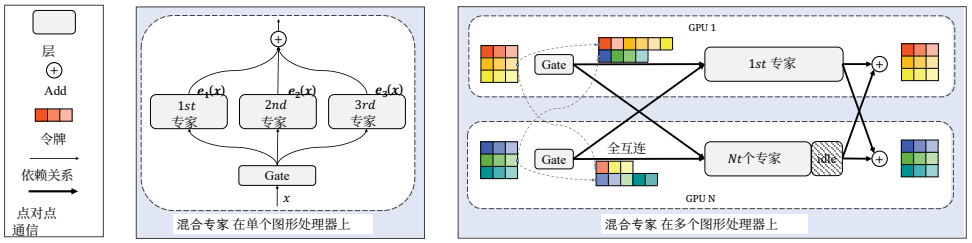
* E：一系列专家 $\{e_1, ..., e_N\}$。* G：一组 GPU，其中 $g \in$ G。
* I：令牌分配，其中$I_{e,g} \subset$ I。$I_{e,g}$ 表示专家 e 分配到 GPU g 的令牌数。* P：专家到设备的映射，其中 $(e, g) \in$ P。$(e, g) \in$ P 表示 GPU g 上存在专家 e。* TPS：专家的每秒令牌数。
* $Bw_{g,g'}$：GPU g 与 g'之间的带宽。* BPS(G′)：一组 GPU G′上全归约的每秒字节数。

2.1 Transformer与专家混合模型

Transformer 架构 [46] 已在各种任务 [6,9, 10, 25, 27, 41, 49], 中展现出优异的性能，其主要由注意力网络和前馈网络组成。每个注意力网络首先将输入令牌线性转换为相应的查询（Q）、键（K）和值（V），然后对它们执行缩放点积，如公式1所示，其中 d 是查询和键的维度。同时，该注意力网络能够捕获序列中令牌之间的依赖关系，因此在序列转换任务中十分有效。

$$\texttt{Attention}(Q, K, V) = \texttt{softmax}(\frac{QK^T}{\sqrt{d}})V \tag{1}$$

每个前馈网络（FFN）由两个全连接层和一个激活函数（即 ReLU）组成，如公式 2 所示。与注意力网络不同，FFN 通过将单个令牌扩展到更大的维度空间，来建模不同特征维度之间的关系。



(a) 混合专家层的工作流 (b) 使用 Top-1 门控的混合专家层分布式训练

图 1. 混合专家（MoE）示意图。图 1(a) 展示了典型混合专家层的工作流（详见第 2.1 节）。图 1(b) 展示了混合专家层的分布式训练，即专家并行：专家被分区，而非 MoE 层（如自注意力、门控）则在各设备上复制。例如，根据门控的结果，GPU 1 将 6 个输入词元路由至 $1st$ 专家，并将其余 3 个令牌路由至 Nth 专家（详见第 2.2 节）。

例如， W_1 的输出维度始终设为其输入维度的 4× 。现有的知名预训练模型通常由一系列 Transformer 层堆叠而成，以提升模型容量和模型质量。

$$\mathtt{FFN}(x) = W_2 \cdot \mathtt{ReLU}(W_1 \cdot x + b_1) + b_2 \tag{2}$$

近年来，通过增加数据量和模型参数量来扩展模型，显著提升了模型质量 [6, 10]，但同时也需要大量计算资源进行训练。为了高效扩展模型，研究人员采用了稀疏门控混合专家（MoE）范式 [12, 45]，以混合专家层取代前馈神经网络（FFN）；在该层中，每个输入词元仅激活一部分模型参数，从而引入模型稀疏性。

混合专家层的关键组成部分包括一个依赖于数据的稀疏门控网络 $g(x)$和一系列专家 E，如图 1(a) 所示。对于每个输入词元 x，门控网络首先计算其相对于所有专家的概率 x，然后将其路由 x到相应的专家。Top-K 门控 [45] 在公式 3 中定义，它会在 Softmax 函数之前仅保留最大的 k 个值。

$$g(x) = \mathtt{softmax}(\mathtt{TopK}(x \cdot W_g)) \tag{3}$$

每个专家 e_i一旦收到其输入词元 $x(e_i \in \mathrm{E})$，便会生成相应的输出$e_i(x)$，随后通过将门控网络 $g(x)_i$的输出作为权重，对各专家的输出 $e_i(x)$进行线性组合，得到最终输出 y，如下所示：

$$y = \sum_{i=1}^{N} g(x)_i \cdot e_i(x). \tag{4}$$

当门控网络的稀疏性固定时，例如 Top-1 [12] 或 Top-2 [21]，，给定输入的计算与通信开销几乎不会随专家数量的增加而变化，因此可以借助混合专家有效扩展模型规模。

为了帮助读者更好地理解混合专家层的运行机制，我们还在图 1(b) 中提供了示意图，其中每个 GPU 对应一个专家，并且 9 个令牌被馈送至门控网络，再路由至其目标专家。例如，GPU 1 将 6 个和 3 个令牌分别路由至 $1st$ 和 Nth 专家，而 GPU N 则将 4 个和 5 个令牌路由至这两个专家。在当前的专家放置方式下，这种分配会导致工作负载不平衡问题（即 GPU 1 处理 10 个令牌，而 GPU N 处理 8 个令牌）。由于专家计算之后还有一个反向路由步骤，所有 GPU 都必须相互等待，然后才能执行后续层。这种不平衡的工作负载不可避免地会导致 GPU 利用不足和较低的训练效率。

2.2 分布式训练

随着模型通过混合专家进行稀疏扩展，多个图形处理器将参与模型管理和训练加速 [33, 35, 42]。接下来我们将分析现有的主流并行策略：

数据并行。 数据并行（DP）通常用于扩展训练规模，此时模型可以容纳在设备可用的 GPU 内存中，因为数据并行的通信原语（例如全归约）具有良好的可扩展性性能。在数据并行中，训练样本会被分区，而模型参数会在每个设备上复制一份。给定一批训练样本，每个工作器执行前向和反向计算，在全局同步梯度（即进行平均）后，根据同步后的梯度更新本地参数。然而，每个设备都必须维护模型的完整副本，因此数据并行无法用于将规模扩展到大型模型。

模型并行。如果给定模型的内存需求超过图形处理器内存，就应将其分布到多个设备上。在张量并行（TP）中，每个张量都可以分布到多个设备上，以减少每个图形处理器的内存占用。例如，Megatron-LM [33] 提出通过利用多头注意力操作中固有的并行性来划分注意力网络，其中查询（Q）、键（K）和值（V）矩阵可以按列并行方式进行划分。流水线并行（PP）是另一种划分方法，其中不同的层组被放置在不同的设备上。例如，GPipe [18]将输入批次划分为若干微批次，并在微批次之间对每个设备的计算进行流水线处理，以提高设备的资源利用率。

专家并行。 GShard [21] 首次提出专家并行（EP），将其作为一种针对 MoE 模型的特定模型并行方法：混合专家层中的专家放置在不同设备上，而非 MoE 层则在设备上以 DP 方式复制。图 1(b) 展示了对采用 Top-1 门控的混合专家层进行分布式训练的工作流。为每个令牌确定目标专家后，需要通过一次全互联通信将令牌发送给目标专家进行处理。之后还需要另一次全互联通信，将数据发回，以执行数据并行的非 MoE 层。由于 MoE 模型通常包含大量专家，例如 Switch Transformer [12], 中有 1024 个专家，因此与模型并行相比，EP 能更好地随模型规模扩展。

2.3 问题形式化

为了对训练成本进行建模，我们考虑单个混合专家层，该层由一个门控网络和一组专家E($e \in$ E) 组成。给定当前的专家到设备的映射 P((e, g) $\in$ P 表示图形处理器 g拥有专家 e)、当前步骤的令牌分配 I ($I_{e,g}$ 表示专家 e 到图形处理器g的令牌数量)，训练成本可以表示为 T(I, P)。我们的目标是最小化该训练成本，可表示如下：

$$\min\ T(\mathcal{I}, \mathcal{P}) = \min\ (\max_{g \in \mathcal{G}} \sum_{e}^{(e,g) \in \mathcal{P}} \{T_C(I_{e,g}) + T_{A2A}(I_{e,g}) + T_{Sync}(\mathcal{P}, e)\}) \tag{5}$$

前两项分别表示专家计算和全互联通信的成本，这些成本由令牌分配决定，即$I_{e,\ g}$。第三项表示同步成本（T_{Sync}），因为同一个专家可能以数据并行的方式存在于多个 GPU 上，因此需要同步其梯度以保持一致。每个 GPU 将其所有本地专家在这三项上的成本相加，得到其执行时间；全局训练时间T(I，P)则是这些 GPU 中最大的执行时间。为了在不修改模型定义的前提下降低训练成本，我们设计了一种动态专家管理机制，在训练过程中自适应地调整专家到设备的映射 P 和令牌分配 I。

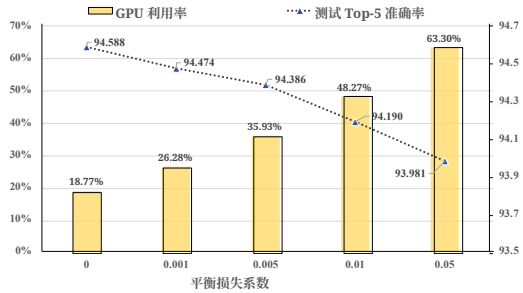


图 2. 不同负载均衡损失系数下 Swin-MoE 的 Top-5 准确率。此处不限制每个专家的容量，以确保没有令牌缺失。

现有的系统友好型优化。如第 1 节所述，现有工作通过修改模型定义（例如负载均衡损失和容量）来解决工作负载不平衡问题。这样无需用户修改底层深度学习系统，因此具有系统友好性。

• The 负载均衡损失 描述了 I 的不均衡程度，在混合专家训练中得到了广泛应用 [12, 21]。一旦门控网络产生不均衡的令牌分配，就会受到较大的负载均衡损失惩罚。实际训练过程中，目标是最小化负载均衡损失与训练损失的加权和。因此，这里存在一种权衡——强调负载均衡损失会促使分配 I 更均衡，但会损害模型质量，因为训练损失会更高，反之亦然。

• 容量 是一个阈值，用于限制每个专家可处理的输入令牌数量（即，$I_{e,\ g}$）。超出容量的令牌会被专家跳过，并通过残差连接直接传递到下一层。简而言之，容量限制了负载最重的专家的训练成本上限，以提高整体效率，但代价是模型质量下降，因为有一定数量的令牌无法得到充分训练。

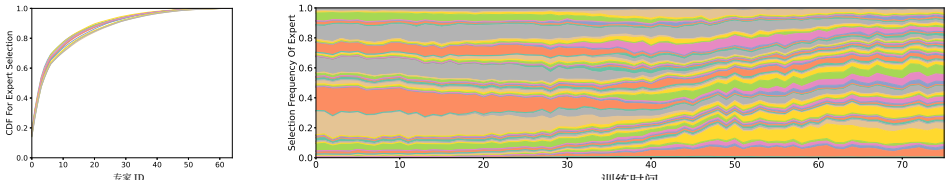
2.4 观察与机遇

观察 1：现有技术的局限性。我们首先通过实证研究了现有技术的效果，并以 Swin-MoE [25] 为例，调整负载均衡损失系数，同时不限制每个专家的容量，以确保没有令牌被丢弃。结果汇总于图 2。将负载均衡损失系数从 0 提高到 0.05 后，GPU 利用率从 18.77% 提高到 63.30%，而 Top-5 准确率则从 94.588% 降至 93.981%。这表明，通过调整令牌分配来强制实现工作负载均衡，不可避免地会导致系统效率与模型质量之间的权衡。

观察 2：动态不均衡的工作负载。我们还记录了训练GPT-MoE 模型（每个混合专家层包含64 个专家）时的轨迹，并研究了训练期间专家的负载。结果总结于图 3 中，我们确定了两个关键特征：

• **偏斜度**。如图 3(a) 所示，我们将专家的计算负载可视化为累积分布函数（CDF），观察到 Top-10 专家（64 个中的 10 个）接收了近 75% 的令牌，导致混合专家训练中的路由不平衡。如果像专家并行那样将专家均匀分配到各个 GPU 上 [21], ，这种不平衡的工作负载会造成严重的资源利用不足，因为大多数专家都需要等待最慢的专家。

• **平滑性和连续性**。在图3(b)中，我们展示了整个训练过程中的专家负载变化，其中不同的区间代表不同的专家。我们观察到，每个专家的负载在训练期间都在持续变化，例如，从较少到



(a) 专家负载的累积分布函数 (b) 整个训练过程中的专家负载变化

图 3. 专家负载示意图。对于单个步骤，我们根据计算负载对每个混合专家层中的专家进行排序，并在图 3(a) 中将相应的累积分布函数（CDF）可视化，不同颜色表示不同的混合专家层。对于整个训练过程，图 3(b) 展示了每个专家负载的动态变化，不同颜色区域代表不同的专家。

更多，从更多到更少，再从更多变为更少，随后又变得更多，等等，这在训练 MoE 模型时会造成路由波动。幸运的是，每个专家的负载不会在短时间内发生剧烈变化，这意味着变化是平滑且连续的。

挑战。 受这些观察结果的启发，我们的工作探索如何通过调整专家到设备的映射（即 P）来实现工作负载均衡，从而提高系统效率，同时保持模型质量。此外，系统应能在训练过程中动态适应不断变化的路由分布。然而，由于令牌路由和专家负载均由数据依赖的门控机制决定，我们无法在执行前确定最优映射。主要挑战在于，如何在 GPU 计算速度快且工作负载迅速变化的情况下，高效地设计和调整专家到设备的映射。另一个挑战是如何高效地实现这些不规则的 MoE 操作。

机遇。 幸运的是，如前所述，该分布变化平稳且连续。这意味着，最优的专家到设备的映射不会在短时间内发生显著变化。因此，无需预测接下来几个训练步的最优映射，而是可以根据临时的路由决策来优化映射。此外，计算和通信的成本可以在实际执行之前进行估算。基于成本模型，我们可以预测不同映射候选方案的收益和开销，从而找到最佳方案。

3 FlexMoE 设计

3.1 概述

我们的 FlexMoE 工作流如图 4 所示：(1) 输入令牌首先由门控网络处理，以确定其对应的目标专家；随后 (2) 路由器利用贪心算法，将令牌分发到各专家的不同副本。最后，(3) 每个副本对分配给自己的令牌执行计算，并返回输出结果。

为应对动态工作负载不均衡，我们设计了一种动态专家管理机制，可在训练期间自适应地调整专家到设备的映射 P 和令牌分配 I。除了常规工作流外，我们还引入了两个新组件：调度器和策略制定器。(4) 调度器监控专家的实时负载，并在当前不均衡指标（即方程 6 中的均衡比率）超过预设阈值时将其发送给策略制定器。策略制定器根据收到的负载和当前的专家放置情况生成修改指令，并将其发回调度器。随后，调度器与当前深度学习框架的执行器交互，并在运行时触发专家放置的修改。

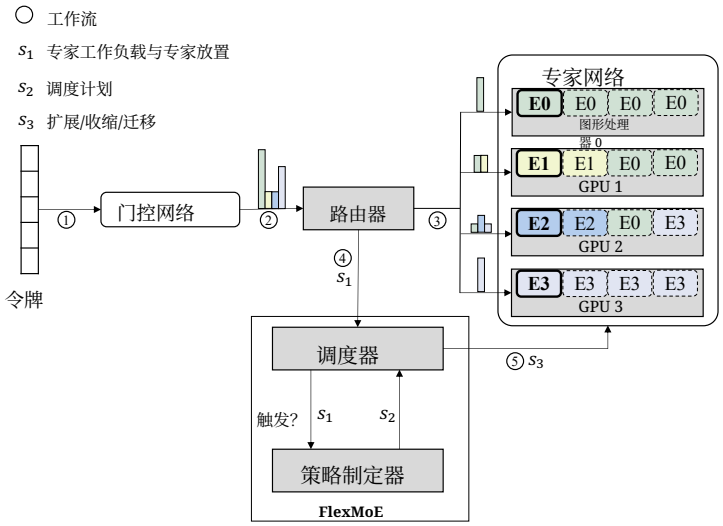


图 4. FlexMoE 系统架构。

为应对灵活的专家到设备映射及其动态修改所带来的庞大决策空间，FlexMoE 提出了 vExpert 抽象，用以表示调度的最小单位，详见第 3.2 节。此外，FlexMoE 使用三种放置修改原语将专家放置的修改解耦，包括扩展、收缩和迁移，详见第 3.3 节。最后，我们设计了一种贪心启发式算法，用于生成这些修改原语的顺序组合，从而制定专家到设备映射的调整计划，详见第 3.4 节。

3.2 动态专家管理与 vExpert

为了解决与专家到设备的映射及其对应的令牌路由相关的优化问题，我们引入了一种名为 vExpert 的新型抽象，它定义了为专家调度 GPU 计算的最小单元，并支持动态专家管理。vExpert 抽象有助于确定如何复制专家、何时增加或减少副本，以及如何在副本之间划分令牌。

vExpert 具有以下特征：

- 每个 vExpert 只能被分配为一个专家的副本，并处理主专家的部分令牌。
- 同一图形处理器上、属于同一专家的各个 vExpert 共享权重，这意味着要将该图形处理器上的相同 vExpert 打包在一起。
- 将工作负载（即令牌）均匀分配给同一专家对应的所有虚拟专家。

通过这种抽象，我们可以在 vExpert 层级做出决策，从而大幅缩小优化问题的庞大搜索空间。此外，我们设计了三种放置修改原语，以在 vExpert 层级进行动态专家管理，包括扩展、收缩和迁移（详情请见第 3.3 节）。通过以不同方式组合这些原语，可以任意修改专家放置。此外，由于每次迭代的数据量固定（$\sum B_i = B$），我们可以计算每个 vExpert 的理想容量（$B/(G*E)$），以此作为决策参考，其中 B 是输入令牌总数，G 是 GPU 数量，E 是每个设备上的 vExpert 数量。

算法1： 调度器：监控实时工作负载

输入： I：令牌分配；
P：专家到设备的映射；

```
当正在训练时 执行
    balance_ratio ←balance(I, P) // 方程 6 ;
    当 balance _ratio >阈值 执行
        plan ←制定调度计划(I, P) // 算法 2 ;
        如果 plan 为空 则
            break ;
        P ← 修改专家放置 P，使用 plan ;
        balance_ratio ←balance(I, P) ;
    P ← 修改专家放置 P with 迁移 ;
```

3.3 调度器

图 4 展示了 FlexMoE 中调度器连接运行时执行器与策略制定器的作用。我们的设计引入了一个指标——均衡比率，用于衡量实时工作负载，并确定何时触发策略制定器做出调整决策。此外，我们为运行时执行器提出了三种用于专家调度的原子操作原语。下一节将进一步介绍这些原语。

均衡比率。由于混合专家层采用同步执行模式，最慢的图形处理器将决定全互联通信以及全局训练的完成时间。因此，我们将均衡比率设计为方程6：累加单个图形处理器上所有虚拟专家的负载，并在所有图形处理器中找出最大值，以获得最终结果。虽然还可以使用其他指标来衡量负载，但我们将方程 6 中的均衡比率与方差这一指标进行比较，并在第 5.3 节中对我们的发现进行详细分析。

$$\texttt{balance}(\mathcal{I},\mathcal{P}) = \frac{1}{\sum_{(e,g)\in\mathcal{P}} I_{e,g}/|\mathcal{G}|} \max_{g\in\mathcal{G}} \sum_{(e,g)\in\mathcal{P}} I_{e,g} \tag{6}$$

扩展。 当某个专家接收到不断增加的工作负载时，先前分配的 vExpert 资源可能会比其他资源处理得更慢，从而产生拖尾任务（过度利用）。在这种情况下，FlexMoE 将调用扩展原语，一次为该专家分配一个额外的新 vExpert 资源。扩展原语通过参数共享复制源 vExpert 的专家参数及相应的优化器状态到目标 vExpert，以进行 GPU 内通信；对于 GPU 间通信，则采用点对点通信。

收缩。 相反，当工作负载减少导致先前分配的虚拟专家资源未得到充分利用时，FlexMoE 会调用收缩原语，每次释放一个虚拟专家资源。收缩原语通过标签执行，无需任何通信，因此不会产生额外开销。

迁移。 如果一个专家在不同图形处理器上保留了多个副本，训练速度会因同步所需的额外通信（即梯度全归约）而降低。这种通信的效率取决于所涉及的图形处理器数量及其局部性。为减少保存专家副本的图形处理器数量并降低同步成本，FlexMoE 将调用迁移原语，在两个虚拟专家之间交换模型状态。

我们在算法1中展示了调度器的工作流。它会监控实时工作负载，即令牌分配 I，并衡量当前部署 P 下的均衡比率（第1-2行）。如果当前均衡比率超过预先定义的 *threshold*，调度器会反复向策略制定器请求修改原语，直到不存在有益的修改（第3-8行）。接着，

算法 2：策略制定器：基于虚拟专家的调度

输入：I：令牌分配；
P：专家到设备的映射；

函数*MakeSchedulingPlan(*$\mathcal{I}$*,* $\mathcal{P}$*)*：
  根据公式5，使用 I，P 估算时间成本；
  对于 $e \in \mathcal{E}$ **执行**
    $n_e \leftarrow$ 为 e 分配的 vExpert 数量；
    $cap_e \leftarrow \mathrm{I}_e/n_e$ // e 的 vExpert 容量；
  e_0，$e_1 \leftarrow \arg\max_{e\in\mathcal{E}}\{cap_e\}$，$\arg\min_{e\in\mathcal{E}}\{cap_e\}$；
  在扩展 e_0 并收缩 e_1后估算时间成本 t_1；
  如果 $t_1 < t_0$ **那么**
    返回 {(扩展, e_0), (收缩, e_1)}；
  return {}；

调度器转而执行迁移操作，以降低同步成本，并在后端持续对其进行优化（第9 行）。直观地说，基于虚拟专家的扩展与收缩原语解决了动态变化的工作负载下复杂的专家设备映射问题，而迁移原语则持续优化每个专家的副本放置。

3.4 策略制定器

利用上述修改原语，FlexMoE 采用一种高效的基于虚拟专家的调度算法来生成一系列修改操作，如算法 2 所示。具体而言，我们采用一种成本模型驱动的搜索规划方法，根据模拟训练时间的反馈作出决策。

为了对混合专家层的训练成本进行建模，我们按照公式5将其分解为三部分，包括计算成本 T_C、全互通通信成本 T_{A2A}和专家同步成本 T_{Sync}。对于每一部分，我们构建成本模型，考虑输入变量（例如，I：令牌分配，以及P：专家到设备的映射）和环境变量（例如，TPS：某个专家的每秒令牌数）。通过采用基于性能分析的方法，我们首先分析函数在不同输入规模下的运行时间，然后估计相应的环境变量。此外，我们还考虑了专家调整的成本，这些调整可以与模型训练并发执行。

计算成本。计算成本是指训练期间专家执行前向与反向计算所需的时间，可表示为：

$$T_C(I_{e,g}) = \frac{I_{e,g}}{TPS} \tag{7}$$

其中，$I_{e,\ g}$表示专家 e在图形处理器 g上接收的输入令牌数量，TPS表示给定 GPU 计算一个专家时的吞吐量（每秒处理的令牌数），该值通过性能分析获得。计算成本与专家接收的输入令牌数量成正比，与 GPU 的吞吐量成反比。换言之，专家接收的输入令牌越多，前向与反向传播所需的计算时间就越长；GPU 越慢，处理每个令牌所需的时间也就越长。

全互连开销。全互通操作用于将令牌发送给目标专家，并在处理完成后将结果发送回来；每个训练步中总共会调用 4 次。我们使用拓扑感知模型（如公式 8 所示）来预测全互连开销，其中 GPU g 上的专家 e 已接收来自 GPU g' 的 $I_{e,g}.count(g')$ 个令牌，而$Bw_{g,g'}$是 GPU g 与 GPU g' 之间经分析得到的带宽。我们的成本模型分别考虑节点内带宽（例如 PCIe、NvLink）和

算法 3： 灵活令牌路由

输入： I：专家工作负载；P：专家放置； g：当前图形处理器；
输出： r: 路由方案

/*遍历所有专家*/
对于 $e \in \mathcal{E}$ **执行**
  $n_e \leftarrow$ 为 e 分配的 vExpert 数量；
  $cap_e \leftarrow \mathrm{I}_e / n_e$ // e 的 vExpert 容量；
  $r_{e,g} \leftarrow \min(cap_e \times n_{e、g}、\mathrm{I}_{e、g})$ // 优先考虑局部性；
  $s_e \leftarrow \mathrm{I}_{e、g} - r_{e、g}$ // 其他图形处理器的令牌；
  $c_{e、g} \leftarrow r_{e、g} - \mathrm{I}_{e、g}$ // 本地可用容量；
  对于 $g' \in \mathrm{G} - \{g\}$ **执行**
    $c_{e,\ g'} \leftarrow \min(cap_e \times n_{e,\ g'},\ \mathrm{I}_{e,\ g'}) - \mathrm{I}_{e,\ g'}$；
    $r_{e,\ g'} \leftarrow s_e \times c_{e,\ g'} / \sum c_e$ // 与可用性成正比；
return $r \leftarrow \{r_e \mid e \in \mathcal{E}\}$；

节点间带宽（例如 IB、NIC）。

$$T_{A2A}(I_{e,g}) = 4 * \sum_{g' \in \mathcal{G}} \frac{I_{e,g}.count(g')}{Bw_{g,g'}} \tag{8}$$

同步成本。 当专家 e 在不同 GPU 上持有多个副本时，其梯度必须通过AllReduce通信进行同步，这取决于消息大小、所涉及设备的数量及其连接的网络。我们枚举不同的设备组，并在训练前对其进行性能分析，以获取其 BPS（字节/秒）。我们通过查找包含 e（即，$\mathrm{P}.index(e)$）的设备组，然后从性能分析数据中获取其对应的 BPS，来预测专家 e的同步成本。 $size(e)$表示一个专家的梯度大小。

$$T_{Sync}(\mathcal{P}, e) = \frac{size(e.gradients)}{BPS(\mathcal{P}.index(e))} \tag{9}$$

专家调整开销。 FlexMoE 提出了三种修改原语，包括扩展、收缩和迁移。扩展和迁移原语涉及通过 NCCL 点对点通信将模型状态从源 GPU 传输到目标 GPU。另一方面，收缩原语通过标记标签即可执行，不产生任何开销。我们将模型状态传输开销预测为 $^{size}(e.model_states)\ ^{Bw_{g,g'}}$.

策略制定器的调度策略总结于算法2 中。首先，该算法获取当前放置的时间成本 t_0 作为基线（第 2 行），然后找出工作负载最大的专家ID 和工作负载最小的专家ID（第 3–6行）。之后，我们的策略制定器估算应用扩展和收缩原语后的时间成本 t_1 （第7行），并通过比较 t_0 和 t_1 来决定是否返回修改结果（第 8–10 行）。

4 实现

我们基于 PyTorch [38] ，通过添加新的自定义算子和 CUDA 内核，实现了所提出的机制和算法。FlexMoE 也是一种新型分布式深度学习系统 Hetu [28, 29, 31] 的组成部分。为了更高效地调度动态数据流，FlexMoE 纳入了以下系统级优化：

灵活令牌路由。路由器应根据复杂的专家到设备映射 P，高效地将来自多个设备（例如 GPU）的输入令牌传输到目标专家。为此，FlexMoE 设计了一种贪心策略，如算法 3 所示，该策略优先将

表1. 用于评估的模型。

模型	参数	#层	模型维度	d_{FFN}	#专家
BERT-MoE-S	0.988B	12	768	3072	32
BERT-MoE-L	6.69B	24	1024	4096	64
GPT-MoE-S	0.988B	12	768	3072	32
GPT-MoE-L	39B	24	2048	8192	64
Swin-MoE-S	946M	24	-	-	32
Swin-MoE-L	1.83B	24	-	-	64

令牌路由到本地图形处理器，然后按照其他 GPU 的可用容量比例，将剩余令牌分散到这些 GPU 上。FlexMoE 还实现了一种高效的逐专家布局转换，以便将输入排列在每个专家对应的连续空间中。

并行操作修改。 FlexMoE 使用队列依次插入由调度器触发的修改原语，例如扩展、收缩和迁移。为了降低调整和内核启动的时间成本，我们将多个连续且可并行执行的操作合并，使其并发运行。例如，如果两个操作具有相同的源和目标，就可以将它们合并，以增大消息大小并提高带宽利用率。与此同时，如果它们的源和目标均不相同，则可以并发执行，从而提高集群的利用率。

全归约协调。 当单个图形处理器上的虚拟专家被分配给不同的专家时，需要针对每个专家分别调用同步；由于不同图形处理器上的调用顺序不一致，可能导致死锁 [4]。为避免这一死锁问题，我们为每个专家分配一个逻辑标识符，并使每个副本（vExpert）的逻辑标识符与其主专家相同。然后，每个图形处理器都按专家逻辑标识符的升序调用同步。

尽力调整。 由于放置修改可能会阻塞训练过程，因此在动态工作负载下，执行当前修改是否有益尚不明确。为解决这一问题，FlexMoE 利用独立的 CUDA流，在可用网络带宽范围内并发进行调整，并采用尽力调整策略，避免妨碍训练过程。调度器与深度学习执行器交互，以确定是否调用候选队列中的第一个操作。

NCCL组管理。 FlexMoE 使用 NCCL [4] 库在多个 GPU 之间执行集体通信，并且需要多个 NCCL组来完成专家之间复杂的同步。然而，可同时存在的 NCCL组数量有上限，而且在使用完这些组后将其销毁并不高效。FlexMoE 使用最近最少使用（*LRU*）缓存来维护 NCCL组，从而降低创建和销毁组的开销。

5 评估

本节将介绍详细的评估结果，以展示 FlexMoE 的有效性和可扩展性。

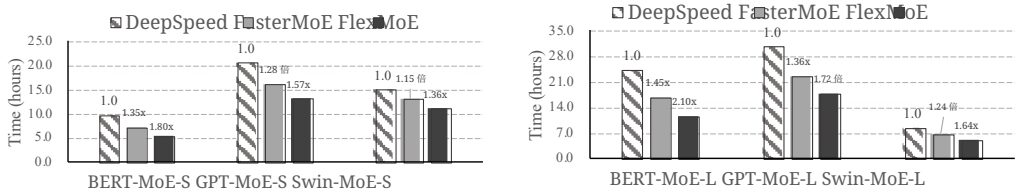
5.1 实验设置

机器环境。 我们在 Azure虚拟机 [1], 上进行实验，每台配备192核AMD中央处理器和8个 NVIDIA AmpereA100图形处理器。GPU通过NVLink3.0进行节点内连接，服务器则通过 8个 InfiniBand网卡连接（总计8*200 Gbps）。默认使用 RDMA，PyTorch版本为1.11。

基线方法。我们将 FlexMoE 与其他流行的框架进行比较，包括 DeepSpeed [42] 和 FasterMoE [17]。DeepSpeed 使用了由 GShard [21] 首次提出的专家并行。FasterMoE 提出了影子策略，将热门专家复制到所有 GPU 上。

表 2. 模型质量对比

	指标	掩码语言模型		语言建模		指标	图像分类	
		BERT-MoE-S	BERT-MoE-L	GPT-MoE-S	GPT-MoE-L		Swin-MoE-S	Swin-MoE-L
DeepSpeed	困惑度↓	3.53	3.31	12.2	10.71	准确率@1↑ 准确率@5↑	77.316 93.838	77.022 93.642
FlexMoE	困惑度↓	3.14	3.07	11.72	10.47	acc@1↑ acc@5↑	77.754 94.042	77.109 93.663



(a) 32 个图形处理器

(b) 64 GPU

图 5. 系统效率对比

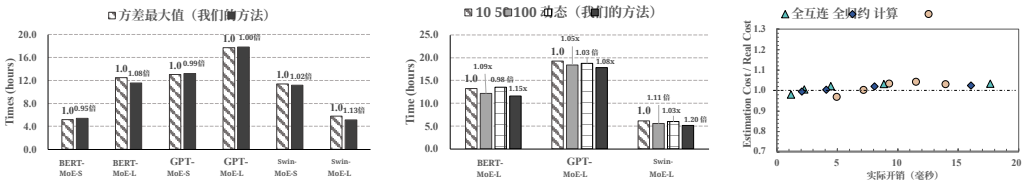
基准测试和数据集。如表1所示，我们通过混合专家架构扩展不同应用领域中的代表性 Transformer 模型，包括自然语言处理领域的 BERT [9] 和 GPT [40]，以及计算机视觉领域的 Swin [25]。我们为每个模型采用 Top-2 门控，该方法被广泛使用的稀疏混合专家模型（GShard [21], GLaM [11], V-MoE [43]）采用，并将每个专家的容量因子设为 1.0，负载均衡损失设为 0.001。我们使用掩码语言模型（MLM）和下一句预测（NSP） [9] 任务预训练 BERT-MoE，使用语言建模任务（LM） [40], 预训练 GPT-MoE，并使用图像分类任务 [19, 25]预训练 Swin-MoE。自然语言处理模型在维基百科 [3] 上训练，视觉模型在 ImageNet-1K [2]上训练。所有超参数（例如学习率）均与相同模型保持一致。

5.2 总体性能

为了评估 FlexMoE 的端到端效率，我们将其与 DeepSpeed 和 FasterMoE 进行比较。对于 DeepSpeed，我们采用传统的专家并行方法，并将每个混合专家层中每个图形处理器上的专家数量设为 1。我们在分别配备 32 个和 64 个图形处理器的设备上，评估两种不同宽度的模型：包含 32 个专家的 X-MoE-S 和包含 64 个专家的 X-MoE-L。

模型质量。为了说明不丢弃令牌的重要性，我们比较了 FlexMoE 与 DeepSpeed 在各种模型和任务上的模型质量。对于语言模型（例如 BERT-MoE 和 GPT-MoE），我们采用验证集困惑度进行评估，数值越低越好；对于视觉模型（例如 Swin-MoE），则采用 ImageNet-1K 的 Top-1/Top-5 准确率。如表 2 所示，FlexMoE 在几乎所有任务上的表现都优于 DeepSpeed，这表明现有 MoE 系统（例如 DeepSpeed）的容量有限，可能导致模型质量下降。考虑到训练成本，Swin-MoE 模型（基于 Swin-B 扩展）训练了 100 个训练轮次（少于标准配置中的 300 个训练轮次），因此准确率略低于基准：在 Top-5 准确率方面，FlexMoE 为 94.04%，而基准为 96.5%。我们认为，这足以展现不丢弃令牌的优势。此外，由于训练数据规模较小，可能导致过拟合，Swin-MoE-L 的表现略逊于 Swin-MoE-S。

系统效率。我们还评估了 FlexMoE 与其他最先进系统在效率方面的表现。为了衡量效率，我们记录了达到目标模型质量所需的训练时间，结果如图 5 所示。实验表明，FlexMoE 实现了最佳训练



(a) 不同指标 (b) 不同的调度策略 (c) 成本模型

图 6. 对不同指标、不同调度策略和成本模型的研究

在效率方面，平均比 DeepSpeed 高出 1.70×，最高可达 2.10×；平均比 FasterMoE 高出 1.30×，最高可达 1.45×。如上所述，尽管 DeepSpeed 凭借有限的容量实现了最短的迭代时间，但它会丢弃令牌以跳过专家网络，因此需要更多迭代才能收敛。FasterMoE 提出了用于负载均衡的动态影子策略，将热门专家复制到所有图形处理器上。然而，由于其专家管理粒度较粗（即部署在 1 个图形处理器上或所有图形处理器上），最终只能退而采用次优方案。随着图形处理器数量增加，FasterMoE 会受到专家副本全局同步的影响。

除了专家网络外，模型训练还包括其他部分，例如非专家计算、优化器更新和通信。由于 FlexMoE 仅优化专家网络的执行，我们将在以下章节中主要分析混合专家层。

5.3 消融研究

为验证 FlexMoE 的有效性，我们进行了若干消融研究，具体如下。

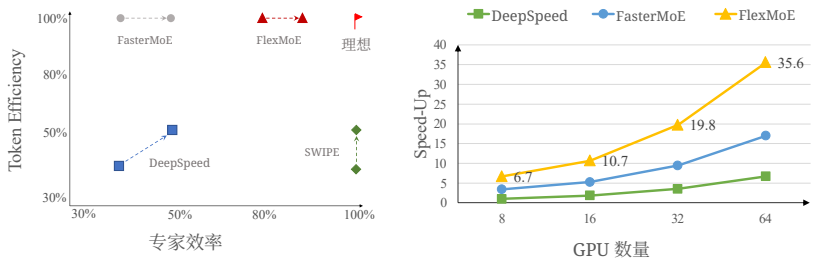
不同指标。 FlexMoE利用均衡比率作为触发调整的关键指标。为研究其他指标的影响，我们还研究了使用*方差* 作为比率的情况，其公式为$\sum_{g\in\mathcal{G}}(I_g - I)^2/|\mathrm{G}|$，此外还研究了由方程 6 给出的*最大值*（我们的） 比率。结果如图 6(a) 所示，表明*最大值*（我们的）的性能平均优于*方差* 1.03×，在 Swin-MoE-L上最高可达 1.13× 。与此同时，*方差* 在 BERT-MoE-S和 GPT-MoE-S 上也表现良好。由于训练时间由最慢的专家主导，*最大值*（我们的） 是一个简单但有效的指标。*方差* 会考虑全局工作负载分布，因此更频繁地触发调整，但由于它并不总是与实际训练相关，通常会从策略制定器获得空操作。

不同的调度策略。 FlexMoE 根据预定义的均衡比率动态触发调整。我们还针对静态调度策略进行了实验，该策略以固定的时间间隔触发调整，并在训练前将其全部执行完毕。如图6(b) 所示，由于工作负载动态变化，我们的动态调度优于大间隔和小间隔。具体而言，小间隔会频繁触发调整，从而引入更多调整开销；而大间隔无法很好地应对动态工作负载，因为它无法及时进行调整。我们的动态策略根据当前工作负载动态决定调整时机，因此适用于动态工作负载。

成本模型评估 图 6(c) 展示了我们成本模型的有效性，其中分别对计算、全互换和全归约，在不同输入规模下比较了估算开销与实际开销。可以观察到，对于所有实验模型，我们的估算结果都非常接近实际执行开销，平均预测误差低于 3%。

5.4 令牌效率和专家效率

在本节中，我们分析整个训练过程中不同混合专家训练方法的令牌效率和专家效率。令牌效率指输入令牌的比例



(a) 令牌效率和专家效率 (b) 可扩展性

图 7. 图 7(a) 展示了训练期间不同方法的令牌效率和专家效率变化趋势。图 7(b) 展示了不同方法的可扩展性。

由专家网络处理，而专家效率 指的是图形处理器的有效计算。对于混合专家层而言，这两个指标均达到 100% 是理想状态，如图 7(a)中的红色标记所示。传统的专家并行方法（例如 DeepSpeed）为了实现负载均衡，会丢弃超出容量的令牌，因此令牌效率和专家效率都较低。BaGuaLu 提出的 SWIPE [26],通过修改门控算法，将输入重新分配给其他专家，以实现严格的负载均衡，从而提高专家效率。然而，这种方法改变了令牌与专家之间的关系，因此导致令牌效率较低。FasterMoE [17] 在每个图形处理器上复制热门专家，并在实现中保证不丢弃令牌。然而，它并未将负载均衡作为设计目标，因此专家效率较低。我们的系统 FlexMoE 保证 100%的令牌效率，并优化计算资源的分配，以平衡各图形处理器之间的工作负载，是最接近理想状态的方案。随着训练的进行，由于负载均衡损失的惩罚，不均衡的工作负载逐渐得到改善，所有方法都在向更高的效率迈进。

5.5 可扩展性

我们还在 8、16、32 和 64 个 GPU 上评估了 FlexMoE 的可扩展性，实验均在包含 64 个专家的单个混合专家层上进行。结果如图 7(b) 所示，并以 DeepSpeed-8GPUs 的吞吐量为基准进行归一化。结果表明，FlexMoE 的表现显著优于 DeepSpeed 和 FasterMoE。由于实验是在高速互连集群上进行的（节点内 8*300Gbps，节点间 8*200 Gbps），GPU 之间的计算均衡至关重要，而 FlexMoE 正是针对这一点设计的。

6 结论

本文介绍了 FlexMoE，这是一种用于应对大规模 MoE 模型训练过程中动态不均衡挑战的新型解决方案。FlexMoE 在现有深度神经网络框架之上集成了调度模块，可监控数据流量、制定调度计划，并在训练过程中动态调整专家到设备的映射。我们在六种广受欢迎的 MoE 模型上进行的实证结果表明，FlexMoE 的表现平均优于 DeepSpeed 1.70×，最高可达 2.10×；同时，其表现平均优于 FasterMoE 1.30×，最高可达 1.45×。

7 致谢

本研究得到国家重点研发计划（编号：2020AAA0105200）、国家自然科学基金（编号：61832001 和 U22B2037）以及北大-腾讯联合研究实验室的资助。Bin Cui 为通讯作者。

参考文献

[1] 2014. Azure A100 虚拟机。https://docs.microsoft.com/en-us/azure/virtual-machines/nda100-v4-series. [2] 2014. ImageNet-1K 数据集。https://huggingface.co/datasets/imagenet-1k. [3] 2014. 维基百科数据集。https://huggingface.co/datasets/wikipedia. [4] 2017. NCCL 2.0。https://github.com/NVIDIA/nccl. [5] 2021. AI2 推理挑战赛（ARC）。https://leaderboard.allenai.org/arc/submissions/public.[6] Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell 等。2020年。语言模型是少样本学习器。《神经信息处理系统进展》33（2020年），1877–1901。[7] Damai Dai、Li Dong、Shuming Ma、Bo Zheng、Zhifang Sui、Baobao Chang 和 Furu Wei。2022年。StableMoE：混合专家模型的稳定路由策略。载于《计算语言学协会第 60 届年会论文集》（第 1 卷：长篇论文）。7085–7095。[8] Jacob Davoudian、Liu Chen、Hongwei Tu 和 Mengchi Liu。2021年。用于分布式图存储的工作负载自适应流式分区器。《数据科学与工程》6（2021年），163–179。[9] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019年。BERT：用于语言理解的深度双向变换器预训练。载于《2019年计算语言学协会北美分会年会：人类语言技术论文集》，第 1 卷（长篇与短篇论文）。4171–4186。[10] Alexey Dosovitskiy、Lucas Beyer、Alexander Kolesnikov、Dirk Weissenborn、Xiaohua Zhai、Thomas Unterthiner、Mostafa Dehghani、Matthias Minderer、Georg Heigold、Sylvain Gelly、Jakob Uszkoreit 和 Neil Houlsby。2021年。一张图像胜过 16×16 个词：面向大规模图像识别的变换器。载于《国际学习表征会议》。[11] Nan Du、Yanping Huang、Andrew M Dai、Simon Tong、Dmitry Lepikhin、Yuanzhong Xu、Maxim Krikun、Yanqi Zhou、Adams Wei Yu、Orhan Firat 等。2022年。Glam：通过混合专家模型高效扩展语言模型。载于《国际机器学习会议》。PMLR，5547–5569。[12] William Fedus、Barret Zoph 和 Noam Shazeer。2022年。Switch Transformers：通过简单高效的稀疏性扩展至万亿参数模型。《机器学习研究期刊》23，120（2022年），1–39。[13] Fangcheng Fu、Yuzheng Hu、Yihan He、Jiawei Jiang、Yingxia Shao、Ce Zhang 和 Bin Cui。2020年。别浪费你的比特！通过 tinyscript 压缩深度神经网络的激活值和梯度。载于《国际机器学习会议》。PMLR，3304–3314。[14] Fangcheng Fu、Jiawei Jiang、Yingxia Shao 和 Bin Cui。2019年。大规模 GBDT 系统的实验评估。《VLDB会刊》12，11（2019年），1357–1370。[15] Fangcheng Fu、Xupeng Miao、Jiawei Jiang、Huanran Xue 和 Bin Cui。2022年。通过缓存增强的本地更新实现高通信效率的纵向联邦学习训练。《VLDB会刊》15，10（2022年），2111–2120。[16] Jia-Ke Ge、Yan-Feng Chai 和 Yun-Peng Chai。2021年。WATuning：一种结合基于注意力的深度强化学习的工作负载感知调优系统。《计算机科学与技术学报》36，4（2021年），741–761。[17] Jiaao He、Jidong Zhai、Tiago Antunes、Haojie Wang、Fuwen Luo、Shangfeng Shi 和 Qin Li。2022年。FasterMoE：大规模动态预训练模型训练的建模与优化。载于《第 27 届 ACM SIGPLAN 并行编程原理与实践研讨会论文集》。120–134。[18] Yanping Huang、Youlong Cheng、Ankur Bapna、Orhan Firat、Dehao Chen、Mia Chen、HyoukJoong Lee、Jiquan Ngiam、Quoc V Le、Yonghui Wu 等。2019年。GPipe：使用流水线并行训练巨型神经网络的高效方法。《神经信息处理系统进展》32（2019年），103–112。[19] Changho Hwang、Wei Cui、Yifan Xiong、Ziyue Yang、Ze Liu、Han Hu、Zilong Wang、Rafael Salas、Jithin Jose、Prabhat Ram、Joe Chau、Peng Cheng、Fan Yang、Mao Yang 和 Yongqiang Xiong。2022年。Tutel：大规模自适应混合专家模型。CoRR abs/2206.03382（2022年）。[20] Jared Kaplan、Sam McCandlish、Tom Henighan、Tom B Brown、Benjamin Chess、Rewon Child、Scott Gray、Alec Radford、Jeffrey Wu 和 Dario Amodei。2020年。神经语言模型的扩展定律。CoRR abs/2001.08361（2020年）。[21] Dmitry Lepikhin、HyoukJoong Lee、Yuanzhong Xu、Dehao Chen、Orhan Firat、Yanping Huang、Maxim Krikun、Noam Shazeer 和 Zhifeng Chen。2021年。GShard：通过条件计算和自动分片扩展巨型模型。载于《国际学习表征会议》。[22] Mike Lewis、Shruti Bhosale、Tim Dettmers、Naman Goyal 和 Luke Zettlemoyer。2021年。基础层：简化大型稀疏模型的训练。载于《国际机器学习会议》。PMLR，6265–6274。[23] Shen Li、Yanli Zhao、Rohan Varma、Omkar Salpekar、Pieter Noordhuis、Teng Li、Adam Paszke、Jeff Smith、Brian Vaughan、Pritam Damania 等。2020年。PyTorch 分布式：加速数据并行训练的实践经验。《VLDB会刊》13，12（2020年），3005–3018。[24] Junyang Lin、Rui Men、An Yang、Chang Zhou、Ming Ding、Yichang Zhang、Peng Wang、Ang Wang、Le Jiang、Xianyan Jia 等。2021年。M6：一种中文多模态预训练模型。CoRR abs/2103.00823（2021年）。

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021年。Swin Transformer：使用移位窗口的分层视觉 Transformer。载于 IEEE/CVF国际计算机视觉会议论文集。10012–10022。[26] Zixuan Ma, Jiaao He, Jiezhong Qiu, Huanqi Cao, Yuanwei Wang, Zhenbo Sun, Liyan Zheng, Haojie Wang, Shizhi Tang, Tianyu Zheng, 等。2022年。BaGuaLu：以超过 3700 万个核心为目标的脑尺度预训练模型。载于 ACM SIGPLAN 并行编程原理与实践研讨会第 27 届会议论文集。192–204。 [27] Atsuki Maruta 和 Makoto P Kato。2022年。意图感知数据可视化推荐。数据科学与工程 7，4（2022年），301–315。[28] Xupeng Miao 、 Xiaonan Nie 、 Hailin Zhang 、 Tong Zhao 和 Bin Cui。2023 年。Hetu：高效的自动并行分布式深度学习系统。中国科学：信息科学 66，1（2023年），1–2。 [29] Xupeng Miao、Yining Shi、Hailin Zhang、Xin Zhang、Xiaonan Nie、Zhi Yang 和 Bin Cui。2022年。HET-GMP：一种用于扩展大型嵌入模型训练的基于图的系统方法。载于 2022年国际数据管理会议论文集。470–480。 [30] Xupeng Miao、Yujie Wang、Youhe Jiang、Chunan Shi、Xiaonan Nie、Hailin Zhang 和 Bin Cui。2022年。Galvatron：使用自动并行在多个图形处理器上进行高效 Transformer 训练。VLDB会刊 16，3（2022年），470–479。[31] Xupeng Miao、Hailin Zhang、Yining Shi、Xin Zhang、Xiaonan Nie、Zhi Yang、Yangyu Tao 和 Bin Cui。2021年。HET：通过缓存支持的分布式框架扩展超大规模嵌入模型训练。VLDB会刊 15，2（2021年），312–320。[32] Deepak Narayanan、Aaron Harlap、Amar Phanishayee、Vivek Seshadri、Nikhil R Devanur、Gregory R Ganger、Phillip B Gibbons 和 Matei Zaharia。2019年。PipeDream：用于深度神经网络训练的通用流水线并行。载于 ACM操作系统原理研讨会第 27 届会议论文集。1–15。[33] Deepak Narayanan、Mohammad Shoeybi、Jared Casper、Patrick LeGresley、Mostofa Patwary、Vijay Korthikanti、Dmitri Vainbrand、Prethvi Kashinkunti、Julie Bernauer、Bryan Catanzaro，等。2021年。使用 Megatron-LM 在 GPU集群上进行高效的大规模语言模型训练。载于高性能计算、网络、存储与分析国际会议论文集。1–15。[34] Xiaonan Nie、Shijie Cao、Xupeng Miao、Lingxiao Ma、Jilong Xue、Youshan Miao、Zichao Yang、Zhi Yang 和 Bin Cui。2021年。用于混合专家的稠密到稀疏门控。CoRR abs/2112.14397（2021年）。 [35] Xiaonan Nie、Yi Liu、Fangcheng Fu、Jinbao Xue、Dian Jiao、Xupeng Miao、Yangyu Tao 和 Bin Cui。2023年。Angel-PTM：腾讯可扩展且经济的大规模预训练系统。arXiv 预印本 arXiv:2303.02868（2023年）。[36] Xiaonan Nie、Pinxue Zhao、Xupeng Miao、Tong Zhao 和 Bin Cui。2022年。TSPLIT：通过张量切分实现高效深度神经网络训练的细粒度 GPU 内存管理。载于国际数据工程会议。IEEE，2615–2628。 [37] Xiaonan Nie、Xupeng Miao、Zhi Yang 和 Bin Cui。2022年。HetuMoE：高效的万亿规模混合专家分布式训练系统。CoRR abs/2203.14685（2022年）。[38] Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、Luca Antiga，等。2019年。PyTorch：命令式风格的高性能深度学习库。神经信息处理系统进展 32（2019年），8026–8037。[39] Yun Peng、Byron Choi 和 Jianliang Xu。2021年。组合优化的图学习：最先进技术综述。数据科学与工程 6，2（2021年），119–141。 [40] Alec Radford、Jeffrey Wu、Rewon Child、David Luan、Dario Amodei 和 Ilya Sutskever。2019年。语言模型是无监督多任务学习者。OpenAI 博客 1，8（2019年），9。[41] Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J Liu。2020年。使用统一文本到文本 Transformer 探索迁移学习的极限。机器学习研究期刊 21（2020年），1–67。[42] Samyam Rajbhandari、Conglong Li、Zhewei Yao、Minjia Zhang、Reza Yazdani Aminabadi、Ammar Ahmad Awan、Jeff Rasley 和 Yuxiong He。2022年。DeepSpeed-MoE：推进混合专家推理与训练，赋能下一代人工智能规模。载于国际机器学习会议。PMLR，18332–18346。[43] Carlos Riquelme、Joan Puigcerver、Basil Mustafa、Maxim Neumann、Rodolphe Jenatton、André Susano Pinto、Daniel Keysers 和 Neil Houlsby。2021年。使用稀疏混合专家扩展视觉。神经信息处理系统进展 34（2021年），8583–8595。 [44] Stephen Roller、Sainbayar Sukhbaatar、Jason Weston，等。2021年。用于大型稀疏模型的哈希层。神经信息处理系统进展 34（2021年），17555–17566。[45] Noam Shazeer、Azalia Mirhoseini、Krzysztof Maziarz、Andy Davis、Quoc V. Le、Geoffrey E. Hinton 和 Jeff Dean。2017年。超大规模神经网络：稀疏门控混合专家层。载于国际学习表征会议。[46] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017年。注意力就是你所需要的一切。神经信息处理系统进展 30（2017年），6000–6010。

[47] Alex Wang、Yada Pruksachatkun、Nikita Nangia、AmanpreetSingh、Julian Michael、Felix Hill、Omer Levy和Samuel Bowman。2019年。SuperGLUE：面向通用语言理解系统的更具黏性的基准。神经信息处理系统进展32（2019年），3266–3280。[48] Zhuping Wang、Jingcheng Liu、ChaoHuang、HaoZhang和 HuaichengYan。2023年。具有长不确定输入时延系统的全局输出反馈自适应稳定化。中国科学：信息科学66，1（2023年），119201。[49] Hua-Peng Wei、Ying-Ying Deng、Fan Tang、Xing-JiaPan 和 Wei-Ming Dong。2022年。基于卷积神经网络和Transformer的视觉风格迁移比较研究。计算机科学与技术学报 37，3（2022年），601–614。[50] Barret Zoph、Irwan Bello、Sameer Kumar、Nan Du、Yanping Huang、Jeff Dean、Noam Shazeer 和William Fedus。2022年。ST-MoE：设计稳定且可迁移的稀疏专家模型。*CoRR* abs/2202.08906（2022年）。

收稿日期：2022年7月；修订日期：2022年10月；录用日期：2022年11月