智能系统学报
CAAI TRANSACTIONS ON INTELLIGENT SYSTEMS

基于二维张量并行策略的大模型加速训练方法

朱仕通, 董琦

引用本文:
朱仕通, 董琦. 基于二维张量并行策略的大模型加速训练方法[J]. 智能系统学报, 2025, 20(5): 1256-1265.
ZHU Shitong, DONG Qi. Accelerated method for training large models based on a 2D tensor parallel strategy[J]. *CAAI Transactions on Intelligent Systems*, 2025, 20(5): 1256-1265.

在线阅读 View online: https://dx.doi.org/10.11992/tis.202411023

您可能感兴趣的其他文章

基于分类差异与信息熵对抗的无监督域适应算法
Unsupervised domain adaptation algorithm based on classification discrepancy and information entropy
智能系统学报. 2021, 16(6): 999-1006 https://dx.doi.org/10.11992/tis.202010020

一致性协议匹配的跨模态图像文本检索方法
Matching with agreement for cross-modal image-text retrieval
智能系统学报. 2021, 16(6): 1143-1150 https://dx.doi.org/10.11992/tis.202108013

面向观测融合和吸引因子的多机器人主动SLAM
Multi-robot active SLAM for observation fusion and attractor
智能系统学报. 2021, 16(2): 371-377 https://dx.doi.org/10.11992/tis.202006019

仿生机器人运动步态控制：强化学习方法综述
Locomotion gait control for bionic robots: a review of reinforcement learning methods
智能系统学报. 2020, 15(1): 152-159 https://dx.doi.org/10.11992/tis.201907052

弹性网络核极限学习机的多标记学习算法
Multi-label learning algorithm of an elastic net kernel extreme learning machine
智能系统学报. 2019, 14(4): 831-842 https://dx.doi.org/10.11992/tis.201806005

事件驱动的强化学习多智能体编队控制
Event-triggered reinforcement learning formation control for multi-agent
智能系统学报. 2019, 14(1): 93-98 https://dx.doi.org/10.11992/tis.201807010

DOI: 10.11992/tis.202411023
网络出版地址: https://link.cnki.net/urlid/23.1538.TP.20250811.1741.002

基于二维张量并行策略的大模型加速训练方法

朱仕通, 董琦
(中国电子科学研究院, 北京 100043)

摘　要: 近期语言模型领域的进展显示, 采用 Transformer 架构的大型预训练模型在自然语言处理应用中表现出优异的技术能力。然而, 受限于 GPU 内存, 训练大语言模型 (large language models, LLMs) 成为了一项挑战。张量并行方法要求单个 GPU 存储所有激活值, 难以突破内存瓶颈。为解决 GPU 内存对大语言模型训练的制约并提升训练效率, 本文提出一种二维张量并行方法 (2D tensor parallelism, TP2D)。2D 张量并行将输入数据和参数矩阵分割并分配至 4 个 GPU; 采用分布式通信, 进行 GPU 间数据的高速交互, 实现真正的分布式并行训练。以 GPT-2 模型作为基准模型, 测试了两种训练方法的软扩展 (soft scaling) 效率和训练效率。实验表明, 当使用 4 块 GPU 时, 2D 张量并行的训练速度是张量并行的 1.84 倍, 软扩展效率达到 86%, 并降低了内存占用。

关键词: Transformer; 张量并行; 注意力机制; 自然语言处理; 人工智能; 预训练; 分布式训练; 分布式通信
中图分类号: TP339　　**文献标志码:** A　　**文章编号:** 1673−4785(2025)05−1256−10

中文引用格式: 朱仕通, 董琦. 基于二维张量并行策略的大模型加速训练方法 [J]. 智能系统学报, 2025, 20(5): 1256–1265.
英文引用格式: ZHU Shitong, DONG Qi. Accelerated method for training large models based on a 2D tensor parallel strategy[J]. CAAI transactions on intelligent systems, 2025, 20(5): 1256–1265.

Accelerated method for training large models based on a 2D tensor parallel strategy

ZHU Shitong, DONG Qi
(China Academy of Electronics and Information Technology, Beijing 100043, China)

Abstract: Recent advancements in language modeling have shown that large pretrained models based on the Transformer architecture exhibit exceptional performance in natural language processing applications. However, training large language models (LLMs) poses a considerable challenge due to the limited memory capacity of GPUs. Traditional tensor parallelism methods require a single GPU to store all activation values, making it difficult to address memory bottlenecks. Aiming to solve the GPU memory constraint on LLM training and improve training efficiency, this paper proposes a two-dimensional tensor parallelism method (TP2D). TP2D partitions the input data and parameter matrices across multiple GPUs, leveraging distributed communication to facilitate high-speed data exchange between GPUs. This approach enables true distributed parallel training and alleviates memory constraints. GPT-2 was used as a benchmark model to evaluate the soft scaling efficiency and training efficiency of the two training methods. Experimental results show that, when using a 4-block GPU, the training speed of 2D tensor parallelism is 1.84 times that of tensor parallelism, with a soft scaling efficiency of 86% and reduced memory consumption.

Keywords: Transformer; tensor parallel; attention mechanism; natural language processing; artificial intelligence; pretraining; distributed training; distributed communication

语言模型 (language modeling, LM) 是自然语言处理 (natural language processing, NLP) 领域实现人工智能的基本方法[1], 自从 2017 年提出 Transformer[2] 架构后, 便取代了过去的循环神经网络 (recurrent neural network, RNN[3]) 和长短期记忆网络 (long short-term memory network, LSTM[4])。Transformer 架构, 采用多头注意力机制关注序列中的每个词之间的相关性, 以及输入序列和输出序列之间的关系。通过这些机制, Transformer 能够更好地理解语言的上下文信息, 并生成更加连贯和有意义的文本[5-6]。不断扩大预训练语言模型 (pre-trained language modeling, PLM) 的规模, 推动模型性能的边界提升, 会出现一种被称为“涌现能力”[7-8] 的显著现象。大语言模型的性能表现与大语言模型的参数量、训练数据集规模、训练的算力这 3 个因素呈现指数扩展律, 即扩展定律 (scal-

收稿日期: 2024−11−21.　**网络出版日期:** 2025−08−12.
通信作者: 董琦. E-mail: dongqiouc@123.com.

©《智能系统学报》编辑部版权所有

ing law)[9-10]。之后, 大语言模型的规模迅速提升至数十亿, 甚至万亿参数[11-12], 尤其是以 BERT(bi-directional encoder representation from Transformers)[13]、GPT3(generative pre-trained Transformer 3)[14]、GPT4[15]、LLaMA(large language model Meta AI)[16] 等为代表的大语言模型表现出惊人的能力, 引领着人工智能的发展[17-18]。随着大语言模型的参数急剧增加, 预训练模型对计算资源的需求同样急速上升。传统的并行训练方法如数据并行需要占用大量内存资源, 模型并行需要复杂的模型划分, 都无法解决现在大语言模型训练的加速问题。ZeRO[19] 基于数据并行提出了零冗余优化器, ZeRO++[20] 通过一些量化方法在数据迁移前后进行量化和反量化, 以降低不同切片在 GPU 间的通信开销。在 ZeRO 的基础上, Ren 等[21] 进一步提出了 ZeRO-Offload, 通过支持 CPU 卸载 (CPU-Offload), 突破 GPU 显存限制。Megatron[22-23] 改进了模型并行, 提出了张量并行。目前的大型语言模型预训练方法已经转变为张量并行、流水线并行 (pipeline)[24-26] 等多种方法相结合的多维并行方法。其中, 张量并行因为其适用性、易用性已经成为最流行的一种训练方法, 但是传统的张量并行在分配数据计算后, 采用全归约 (All_Reduce) 全局通信, 这个操作会收集所有 GPU 上的计算结果, 并保存到一个 GPU 上。这导致张量并行方法中每个 GPU 必须存储全部的中间激活值, 成为了张量并行方法的限制[27-28]。

本文提出的 2D 张量并行, 通过引入分布式通信与并行计算的方法[29-30], 解决困难, 节省了内存占用并提高了训练效率。此方法将数据与模型参数进行二维分割, 并分配给不同的 GPU。当数据需要交换时, 同一个通信组内的 GPU 通过局部通信 (环通信 ring 和广播通信 broadcast) 交换数据, 这样每个 GPU 只执行部分运算, 并保存部分中间激活值, 在训练过程中规避了每层 Transformer 中的 All_Reduce 全局通信。当训练完一个批次后, 再通过 All_Reduce 全局通信后收集数据。通过分布式通信进行数据传输, 保证了训练中的数据一致性。实验结果表明, 2D 张量并行在相同情况下节省内存占用, 并且提高了训练速度。此外, 由于 2D 张量并行对内存更为友好, 在可扩展性上同样表现优异。

1　相关工作

1.1　多头自注意机制

首先介绍 Transformer 中的自注意力机制。对于一个输入句子 $\boldsymbol{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$ 有 N 个词嵌入 (token), 将每个 token 编码为 3 个注意力嵌入 (即查询 $\boldsymbol{q}$、键 $\boldsymbol{k}$ 和值 $\boldsymbol{v}$)。自注意力机制通过将 $\boldsymbol{q}_i$ 与所有 token 的 $\boldsymbol{k}$ 相乘, 计算每个 token $\boldsymbol{x}_i$ 与 $\boldsymbol{X}$ 中所有其他 token 的注意力分数。为了方便并行计算, 将所有 token 的 $\boldsymbol{q}$、$\boldsymbol{k}$ 和 $\boldsymbol{v}$ 组合成 3 个矩阵: 查询 $\boldsymbol{Q}$, 键 $\boldsymbol{K}$ 和值 $\boldsymbol{V}$。输入的自注意力公式为

$$\boldsymbol{H}=\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=\text{Softmax}(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}/\sqrt{d_{\boldsymbol{K}}})\boldsymbol{V}$$

式中 $d_{\boldsymbol{K}}$ 表示矩阵 $\boldsymbol{K}$ 的维度。

1.2　张量并行

张量并行与传统的模型并行不同, 张量并行沿着特定维度将张量划分为几个部分, 将模型的各个层划分到多个设备上。张量并行对 Transformer 结构进行分解, 分别是自注意块 (self-sttention) 和多层感知机 (multilayer perceptron, MLP), 然后分别并行化, 采用 All-Reduce 通信来进行同步通信。张量并行的自注意模块如图 1 所示, 多层感知机如图 2 所示。

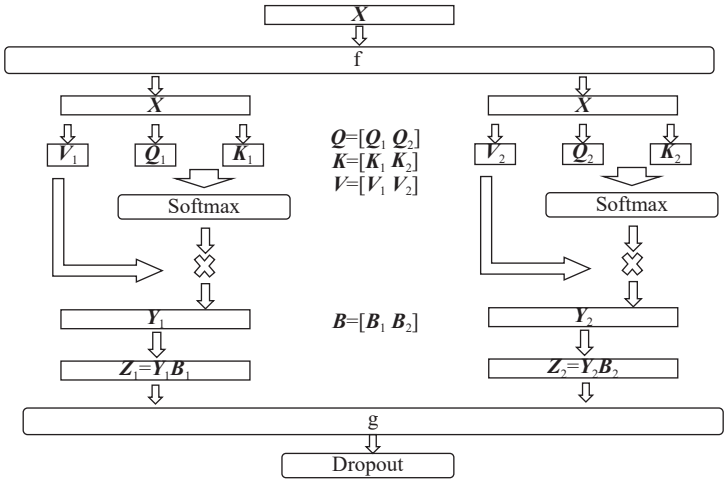

图 1　张量并行自注意模块

Fig. 1　Tensor parallel self-attention module

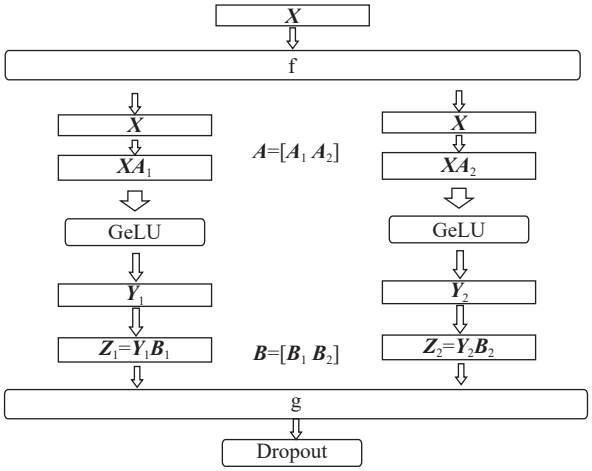

图 2　张量并行 MLP 模块

Fig. 2　Tensor parallel MLP module

在张量并行中，将参数矩阵按照行和列分别进行划分然后并行执行分布式计算。计算完成后需要经历一次 All-Reduce 通信进行同步。以 MLP 块为例，计算过程可以写为

$$\boldsymbol{A}=[\boldsymbol{A}_1 \quad \boldsymbol{A}_2]$$

$$\boldsymbol{Y}=[\boldsymbol{Y}_1 \quad \boldsymbol{Y}_2]=\left[\mathrm{GeLU}(\boldsymbol{X}\boldsymbol{A}_1) \quad \mathrm{GeLU}(\boldsymbol{X}\boldsymbol{A}_2)\right]$$

$$\boldsymbol{Z}=[\boldsymbol{Z}_1 \quad \boldsymbol{Z}_2]=[\boldsymbol{Y}_1 \quad \boldsymbol{Y}_2]\cdot\begin{bmatrix}\boldsymbol{B}_1\\ \boldsymbol{B}_2\end{bmatrix}$$

式中：$\boldsymbol{A}$、$\boldsymbol{B}$ 表示两层线性层的模型参数，分别进行列划分和行划分；$\boldsymbol{Y}$ 表示中间计算结果。张量并行同样在自注意层中分割张量。对于多头注意力，注意力头是按列划分的，并平均分配给设备。自注意力计算后的线性层的参数进行行划分。最后通过 All-Reduce 通信来聚合来自所有设备的自注意力输出。

1.3　分布式通信

张量并行方法引入了分布式通信，以支持大语言模型进行分布式训练。张量并行引入了 All-Reduce 全局通信，2D 张量并行在此基础上引用广播通信和环通信进行局部数据通信，用来支持分布式并行计算。

广播通信具有一个发送者和多个接收者，这些共同组成了一个通信组。在执行通信时，在通信组内由发送者将一部分数据发送给所有接收者，接收者接收数据并保存。如图 3 所示，4 台服务器组成一个广播通信组。

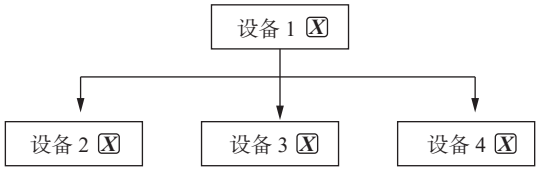

图 3　广播通信

Fig. 3　Broadcast communication

环通信具有多个发送者和多个接收者，组成了一个通信组。每个发送者向指定的接收者发送数据，接收者接收来自对应发送者的数据，进行多对多通信并发执行。如图 4 所示，4 台服务器组成一个环通信组。

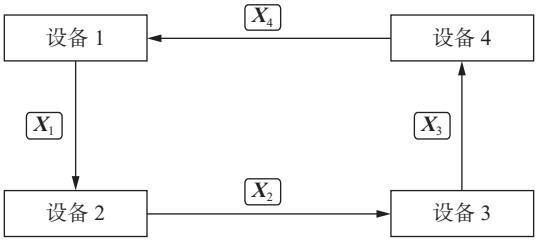

图 4　环通信

Fig. 4　Ring communication

All-Reduce 通信是一种在分布式计算中用于进行全局通信的方法。它的基本思想是将所有服务器上的数据进行规约操作 (如求和、求最大值等)。本文采用的分布式通信 (Broadcast 和 Ring) 与其的差异在于两点。其一，本文的分布式通信运用在局部通信，通过构建通信组实现，而 All-Reduce 通信属于全局通信。其二，All-Reduce 需要对数据进行规约操作，分布式通信仅进行数据传递。

1.4　困惑度

在自然语言处理中，困惑度 (perplexity, PPL) 是一种用来评估语言概率模型性能的方法。语言概率模型可以看作是整个句子或文本段落的概率分布。研究人员认为，给定一个高质量的句子，一个好的模型更有可能以更高的概率产生它。因此，研究者沿着这一思路，提出了困惑度的概念。本文采用困惑度来对比预训练模型的性能，采用条件概率函数 P 进行计算，计算公式为

$$P_{\mathrm{perplexity}}=N\sqrt{\prod_{i=1}^{N}\frac{1}{P(\boldsymbol{x}_i|\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_{i-1})}}$$

式中 $\boldsymbol{x}$ 表示句子中的 token。

2　2D 张量并行

2D 张量并行方法为了进一步提升训练大语言模型的速度，一方面在张量并行的基础上结合了数据并行的思想，另一方面采用分布式并行训练的思想，通过分布式的通信调度提高大语言模型训练速度。这种方法通过张量 2D 划分，将大语言模型扩展到分布式计算集群上，引入广播通信和环通信进行分布式计算。本节将详细讲述 2D 张量并行的原理，符号含义如下：

Z 表示自注意层的多头注意力的数量；

B 表示模型批次；

L 表示输入序列尺寸;
H 表示隐藏层大小;
A 表示注意力头尺寸;
N 表示 GPU 数量;
T 表示 Transformer 层数。

2.1　张量与数据划分

2D 张量划分将输入序列和模型参数都划分为相同的 $\sqrt{N}\times\sqrt{N}$规模, 并分发到多个 GPU 设备上进行计算, 如图 5 所示。对于输入序列, 按照行列进行划分, 对于无法完全划分的通过补零 (zero-padding) 保证规模一致。对于多头注意力机制, 对注意力头的参数矩阵 (查询 $\boldsymbol{Q}$、键 $\boldsymbol{K}$ 和值 $\boldsymbol{V}$) 进行划分。

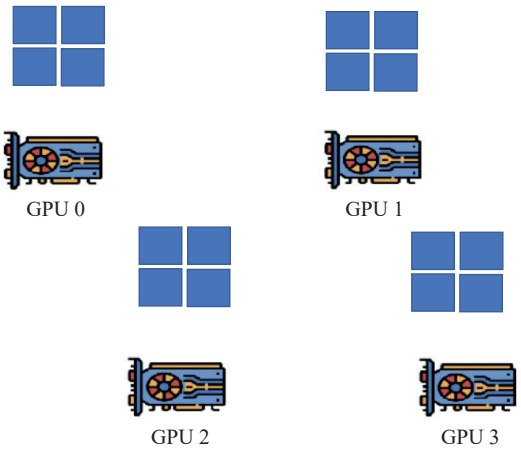

图 5　张量与数据划分

Fig. 5　Tensor and data partition

完成数据与模型参数划分之后, 主要的挑战是计算跨设备的注意力分数。为了方便表示, 按照逻辑顺序将 N 块 GPU 按照行和列进行编序, 其中, GPU_{ij} 表示第 i 行第 j 列的 GPU。$\boldsymbol{Q}_{ij}$、$\boldsymbol{K}_{ij}$ 和 $\boldsymbol{V}_{ij}$ 表示划分到 GPU_{ij} 上的参数矩阵。整个注意力分数计算可以分为两个阶段。

第一阶段, 计算查询$\boldsymbol{Q}$与键$\boldsymbol{K}$之间的注意力分数, 对于每一个查询 $\boldsymbol{Q}_{ij}$, 都需要与对应第 j 列上所有的键 $\boldsymbol{K}_{ij}$ 相乘并累加, 得到对应部分的注意力分数, 计算公式为

$$\boldsymbol{Q}\boldsymbol{K}_{ij}^{\mathrm{T}}=\sum_{t=0}^{\sqrt{N}-1}\boldsymbol{Q}_{it}\boldsymbol{K}_{tj}^{\mathrm{T}}$$

第二阶段, 将注意力分数经过 Softmax 归一化后, 与所有的值 $\boldsymbol{V}_{ij}$ 相乘累积, 计算自注意层的输出$\boldsymbol{O}_{ij}$:

$$\boldsymbol{S}_{ij}=\mathrm{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}_{ij}^{\mathrm{T}}}{\sqrt{d_{\boldsymbol{K}_{ij}}}}\right)$$

$$\boldsymbol{O}_{ij}=\sum_{t=0}^{\sqrt{N}-1}\boldsymbol{S}_{it}\boldsymbol{V}_{tj}$$

2.1.1　内存占用

为了比较 2D 张量并行与张量并行在内存占用方面的优劣, 对每块 GPU 上的内存分配进行建模分析。假设大语言模型训练一个批次过程中, 输入的数据量为 D, 模型参数占用内存为 P, 梯度占用内存为 G_{d}, 采用 Adam 优化器占用内存为 A_{d}, 中间激活值 M。使用 DP 代表数据并行, TP 代表张量并行, TP2D 代表 2D 张量并行, ZERO 表示零冗余优化器。对不同的训练方法, 分析单个 GPU 中的内存占用, 其中基准为 DP, 比较结果如表 1 所示。

表 1　不同训练方法对内存占用的比较

Table 1　Comparison of memory usage by different training methods

训练方法	数据量	参数量	梯度	优化器	中间激活值
DP	D	P	G_{d}	A_{d}	M
TP	D	P/N	G_{d}/N	A_{d}/N	M
TP2D	D/N	P/N	G_{d}/N	A_{d}/N	M/N
ZERO	D/N	$2P/N$	G_{d}/N	A_{d}/N	M/N

根据 Transformer 的体系结构, 张量并行将其分为 MLP 块和自注意块两部分。其中自注意块如图 1 所示, 张量并行需要通过 All-Reduce 收集所有 GPU 上的计算结果, 这直接导致了张量并行需要单个设备保留全部的中间激活值 M, 并成为张量并行扩展的瓶颈。2D 张量并行分割了输入数据和模型参数, 然后通过分布式通信将计算所需的数据进行调度, 这样每个 GPU 只需要保存一部分输入数据、模型参数和中间激活值。ZERO 方法通过将优化器状态、梯度、参数的内存占用和其他的冗余内存 (中间激活、临时缓冲区和碎片内存) 进行了内存管理与优化, 并按照 GPU 数量进行划分。在训练时 ZERO 需要复制当前计算的模型参数到所有的 GPU 上, 所以单个 GPU 需保存模型参数 $2P/N$。

与 ZERO 方法相比, 2D 张量并行对于内存优化最明显的地方在于模型参数量的划分与分配。与张量并行相比, 2D 张量并行降低了中间激活值的内存占用。2D 张量并行通过提高训练的并行度提高训练速度, 通过降低单个 GPU 的内存负载表现出更好的软扩展性。

2.2　分布式通信

2D 张量并行将所有的数据分发到多个设备上, 为了计算跨设备的注意力分数, 必须采用分布式通信方法调度参数, 保障逻辑运算结果一致。因此, 本方法采用了广播通信与环通信两种通信方法来传递数据, 保证计算一致性和同步性。分布式通信如图 6 所示。

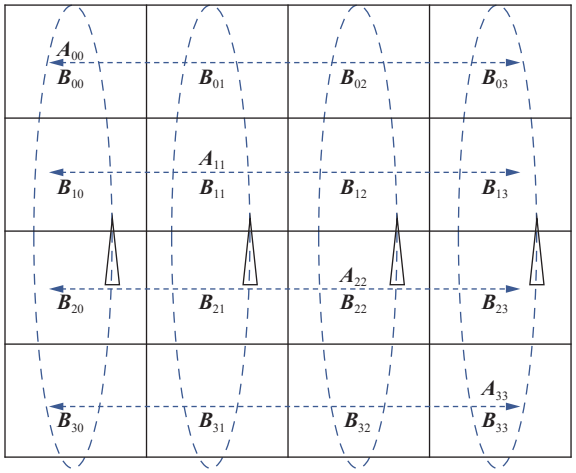

图 6　分布式通信

Fig. 6　Distributed communication

如图 6 所示，N个 GPU 按照行列进行逻辑编号，同一行内的 $\sqrt{N}$个 GPU 划分为一个广播通信组，共有 $\sqrt{N}$个行广播通信组。同一列内的 $\sqrt{N}$个 GPU 划分为一个环通信组，共有 $\sqrt{N}$个按列划分的环通信组。在进行分布式训练时，首先在每一行的广播通信组内进行广播通信。发送者是每行的第$k[i]$列 GPU，例如图 6 第 1 行的广播通信发送者为第 1 列，广播数据是 $\boldsymbol{A}_{00}$。接收者是广播通信组内其他 GPU，将收到的数据保存并作为计算数据。完成广播通信后，每个 GPU 并发执行矩阵乘法，并保存计算结果，计算完成后会清空之前接收的广播数据并更新序列$k[i]$。每块 GPU 上具有近乎相同的计算量，工作负载均衡，所以分布式计算后不需要大量的同步等待时间。

完成矩阵乘法后，在每一列的环通信组内按照逆时针进行环通信，更新参数矩阵 $\boldsymbol{B}$。经过环通信后，每块 GPU 都会更新自身的参数矩阵 $\boldsymbol{B}$，为下一次运算做准备。之后重复行内的广播通信，进行计算，环通信和数据更新，并将所有的输出结果累积，直到完成计算。上述分布式算法的逻辑原理如算法 1 所示。

算法 1　2D 张量并行

输入　$\boldsymbol{A}[\sqrt{N},\sqrt{N}]$

参数　$\boldsymbol{B}[\sqrt{N},\sqrt{N}]$

输出　$\boldsymbol{C}[\sqrt{N},\sqrt{N}]$

1) 初始化索引$k[\sqrt{N}]=[1,2,\cdots,\sqrt{N}]$

2) 按照行列构建行广播通信组和环通信组

3) while 训练 $\sqrt{N}$次 do

4) for　每一行　do

5) 在每一行通信组内广播$\boldsymbol{A}[i,k]$

6) end for

7) 分布式计算$\boldsymbol{C}=\sum \boldsymbol{C}[i,j]=\sum\sum \boldsymbol{A}[i,k]\cdot\boldsymbol{B}[k,j]$

8) for　每一列　do

9) 在每一列通信组内环通信并更新数据

10) end for

11) 更新索引$k[i]=[(k[i]+1)\mod \sqrt{N}]$

12) end while

13) 从每个 GPU 输出分布式矩阵 $\boldsymbol{C}$

2D 张量并行将矩阵按照块划分并进行分布式矩阵乘法：

$$\boldsymbol{C}=\boldsymbol{A}\cdot\boldsymbol{B}=[\boldsymbol{A}[1],\boldsymbol{A}[2],\cdots,\boldsymbol{A}[\sqrt{N}]]^{\mathrm{T}}*$$
$$[\boldsymbol{B}[1],\boldsymbol{B}[2],\cdots,\boldsymbol{B}[\sqrt{N}]]=$$
$$\sum \boldsymbol{A}[i]\cdot\boldsymbol{B}[i]=\sum[\boldsymbol{A}[i,1],\boldsymbol{A}[i,2],\cdots,$$
$$\boldsymbol{A}[i,\sqrt{N}]]*[\boldsymbol{B}[1,i],\boldsymbol{B}[2,i],\cdots,\boldsymbol{B}[\sqrt{N},i]]^{\mathrm{T}}=$$
$$\sum\sum \boldsymbol{A}[i,k]\cdot\boldsymbol{B}[k,j]=\sum \boldsymbol{C}[i,j]$$

通过矩阵运算证明，2D 张量并行方法不会改变计算结果，能够保证计算一致性和训练收敛性。

2.2.1　通信成本

张量并行，在前向传递和后向传递的过程中采用 All-Reduce 进行全局通信，将每块 GPU 上的输出结果合并到一个设备上，需要传输 N–1 块 GPU 上的数据，输出数据的规模为 $(B,Z,L/N,A)$。因此每次进行全局通信的数据量可以表示为

$$2(N-1)\times B\times Z\times(L\div N)\times A$$

由于在 MLP 层和自注意力层的前向和后向传递中有 4 次集体通信，因此总通信成本为

$$8(N-1)\times B\times Z\times(L\div N)\times A \tag{1}$$

在本方法中，主要的通信开销集中在自注意力层。除了引入的局部通信，在前向和后向传递中同样有 2 次集体通信 All-Reduce 来进行数据同步，因此全局的通信成本为

$$4(N-1)\times B\times Z\times(L\div N)\times A$$

除此之外，对于行内广播通信，每次通信数据量为 A/N，在 $\sqrt{N}$个层行内进行行广播，需要进行 $\sqrt{N}-1$次通信，因此在前向传递和后向传递的数据量为

$$2(\sqrt{N}-1)\times\sqrt{N}\times B\times Z\times(A\div N)\times A$$

对于列内环通信，每次通信的数据量为 A/N，在 $\sqrt{N}$个层列内进行环通信，需要进行 $\sqrt{N}-1$次通信，由于环通信仅将数据传输到相邻的计算节点，传递步数为 1，且为并行的同步传输，所以其通信成本与广播通信相等。总结 2D 并行的局部通信，在前向传递和后向传递的通信量为

$$4(\sqrt{N}-1)\times\sqrt{N}\times B\times Z\times(A\div N)\times A+ 4N\times B\times Z\times(L\div N)\times A \tag{2}$$

对比式 (1) 和 (2)，当 GPU 数量 N 足够大时，$(\sqrt{N}-1)\times\sqrt{N}\approx N$，2D 张量并行和张量并行的通

信成本的差异主要在于, 2D 张量并行降低了一半的全局通信开销, 同时将通信开销与注意力头尺寸 A 建立线性关系而非输入序列 L。易知, 数据的规模 L 远大于参数 A, 因此当使用更大规模的算力训练更大规模的模型时, 2D 张量并行的通信开销比张量并行更低。除全局通信 All-Reduce 外, 局部通信对通信带宽的需求仅为 L/N。

3 实验结果与分析

3.1 实验设置

实验在浪潮平台上进行, 最多使用了 8 个 NVIDIA RTX 4090 GPU。使用的数据集包括 Oscar 和 CLUECorpus2020, 进行预处理得到 10 GB 的数据。GPT-2 作为实验的基础模型, 具有可调节的超参数, 包括并行度、模型层数、注意头数等。此外, 实验使用正态分布初始化模型的权值; 随后, 在进入残差层之前对这些权值进行了缩放。使用 Adam 优化器进行训练, 权重衰减参数设置为 $\lambda = 0.01$, dropout 设置为 0.1。

实验包含两部分。一方面, 对不同规模的 GPT-2 使用数据并行、张量并行和 2D 张量并行进行训练, 主要对比不同训练方法的内存占用、训练速度和软扩展性; 另一方面, 使用 2D 张量并行方法应用于医疗图像分割领域, 证明与应用场景的适配性, 并与 ZERO 相结合, 展示了与其他优化策略的协同效果。

3.2 内存占用分析

本节为了评估 2D 张量并行对内存的优化, 使用了不同的训练方法来训练相同的模型, 然后对训练过程中内存占用进行比较。固定 GPT-2 模型的规模, 改变训练方法, 其中 TP 表示张量并行度, DP 表示数据并行度, TP2D 表示 2D 张量并行度。为了更好地进行比较, 将全局并行维数设置为 4 或 8, 输入序列长度为 512, 固定模型参数。使用 4 块 GPU, 训练 2.5 B 规模的模型 (B 表示模型参数规模是十亿级), 进行了 4 组实验, 将并行度分别设置为 DP=4、TP=2, DP=2、TP=4、TP2D=4; 此外, 使用了 8 块 GPU 进行了 2 组实验, 并行度分别为 TP2D=4, DP=2 和 TP=8。整个训练过程经过 500 次迭代, 其中前 200 次迭代训练用来预热, 表 2 给出的实验数据展示了剩余 300 次迭代训练的时间和训练中的内存占用。

从表 2 可以看出, 数据并行必须将模型参数存储在每个计算节点上, 这占用了大量的内存资源并导致无法训练。与数据并行性相比, 张量并行性节省了模型占用的内存, 但改进效果有限。与第二组实验 (TP=2,DP=2) 相比, 将张量并行度 (TP 或者 TP2D) 提升到 4, 张量并行仅减少了 0.8 GB 的内存, 而 2D 张量并行节省了 2.5 GB 内存。2D 张量并行能节省更多存储资源是因为进一步划分了输入数据, 并且每个 GPU 仅保存部分中间激活值, 在训练同样批次大型语言模型时, 数据占用的内存更少。此外, 二维张量并行的训练时间明显缩短, 训练速度较快。与 TP=2, DP=2 组相比, 训练时间从 168 h 减少到 64 h; 与张量并行 TP=4 对比, 训练时间也从 118 h 减少到 64 h, 训练速度为张量并行的 1.84 倍。

表 2　不同并行方法下内存占用和时间

Table 2　Memory allocated and time under different parallel methods

并行度	批次	显存占用/GB	时间/h	GPU数量
DP=4	32	OOB	OOB	4
TP=2, DP=2	32	22.3	168	4
TP=4	32	21.5	118	4
TP2D=4	32	19.8	64	4
TP=8	64	22.2	91	8
TP2D=4, DP=2	64	20.0	43	8

注: OOB表示无法训练。

当使用 8 块 GPU 时, 2D 张量并行比张量并行方法约节省了 10% 的存储资源, 每块 GPU 上的内存占用从 22.2 GB 下降到 20 GB。对比两种方法的训练速度, 2D 张量并行是张量并行的 2.1 倍。

3.3 软扩展分析

本节将展示在软扩展情况下, 张量并行和二维张量并行的模型参数对批处理大小的弱缩放。软扩展是指同时提升大模型训练规模和算力资源, 通常是通过缩放批次大小来实现的。但是, 通过缩放批次大小并没有解决在单一 GPU 上训练大模型的问题, 从而导致了大批次下训练的收敛性退化。本文通过更改模型的规模来训练更大的模型, 将能够训练的最大规模的模型参数量与理论上线性增长的模型参数量对比, 得到缩放系数。所有缩放因子的基线都是在单个 GPU 上训练的具有 0.54 B 参数的 GPT-2 基准模型, 模型规模设置见表 3。

表 3　用于缩放研究的参数

Table 3　Parameters used for scaling studies

模型参数/B	网络层数	隐藏层数	注意力头数	GPU个数
0.54	32	1 280	20	1
1.20	48	1 536	24	2
2.50	54	1 920	30	4
4.20	64	2 304	36	8

注: 每个注意力头的隐藏层大小保持为64个。

图 7 给出了张量并行度和二维张量并行度的软扩展系数。在使用不同规模的 GPU 算力资源训练不同规模的模型的情况下都观察到了极好的缩放比例。例如，在使用 4 块 GPU 训练 2.5 B 模型的情况下，2D 张量并行达到了 86% 的软扩展系数。张量并行需要全局同步通信，单个 GPU 需要保存中间激活值，因此缩放值略低，约为 83%。数据并行表现最差，这主要是由于数据并行必须将整个模型复制，在扩展时造成大量的内存被占用，所以缩放值最低，在 76% 左右。随着使用的计算资源增加，缩放系数普遍降低，当使用 8 块 GPU 时，张量并行的缩放系数降低到 74%，张量并行结合数据并行的缩放系数最低，降低到 64%，2D 张量并行的缩放系数降低到 79%。这样的结果表明 2D 张量并行软扩展性能更好。

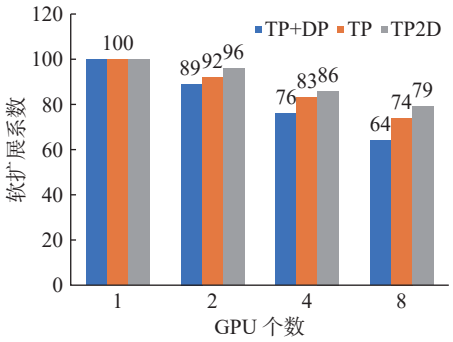

图 7　TP＋DP、TP 和 TP2D 的软扩展系数与 GPU 数量的关系

Fig. 7　TP + DP, TP and TP2D weak scaling efficiency as a function of the number of GPUs

3.4　训练效率

本文从大语言模型训练中的收敛速度和训练速度两个方面，比较了不同训练方法对大语言模型训练效率的影响。

本文通过训练损失和困惑度 (PPL) 的收敛速度来比较大型语言模型的收敛速度。图 8、9 给出了经过预热后，300 次迭代训练的损失值和困惑度的下降趋势。张量并行方法显示出了稳定的收敛结果，困惑度为 37.2，损失值仍然表现出相当大的波动。与张量并行法相比，2D 张量并行方法的困惑度为 29.1，损失值收敛速度更快，训练效果更稳定。现有最先进的大语言模型困惑度会降低到 19.8，本文的实验结果虽然未能达到最优，但是也表现出了困惑度稳定下降的趋势。

此外，本文从训练时间方面对训练的速度进行了评估，对比了不同的架构 (GPT、BERT 和 T5) 对不同训练方法的速度影响，分别使用张量并行和 2D 张量并行训练规模 2.5 B 的模型，并对比训练时间计算加速比。由表 4 可以得出，经过相同的 300 次迭代训练，2D 张量并行方法的训练速度明显更快。当使用 4 块 GPU 训练 2.5 B 规模的 GPT 模型时，张量并行方法消耗了 118 h，而 2D 张量并行仅使用了 64 h，可以得出，2D 张量并行的训练速度是张量并行的 1.84 倍。对于其他架构的大模型，如 BERT 和 T5，2D 张量并行方法同样有效。但是 T5 的加速度最小，GPT 的表现最有效，这可能是由 GPT 的模型结构单一导致的。

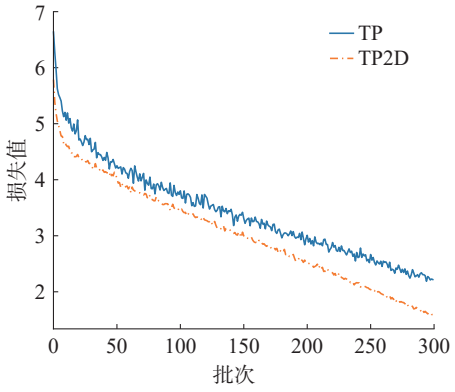

图 8　训练过程中损失值趋势

Fig. 8　Loss value trend throughout the training process

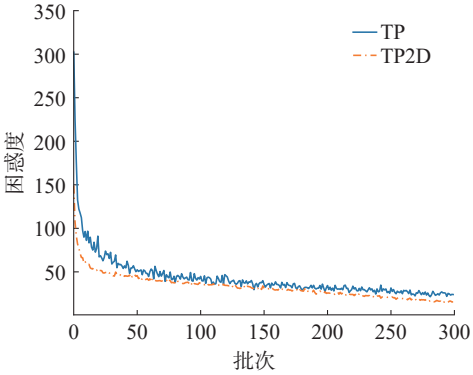

图 9　训练过程中困惑度趋势

Fig. 9　Perplexity value trend throughout the training process

表 4　2D 张量并行与张量并行的加速比

Table 4　Speedup for TP and TP2D

模型	训练时间/h		加速比
	TP	TP2D	
GPT	118	64	1.84
BERT	124	69	1.80
T5	166	101	1.64

本文进一步训练了 2.5 B 规模的大模型，包含 GPT、BERT 和 T5 这 3 种经典架构，图 10 给出了不同架构模型的损失收敛过程，3 种模型都能够稳定收敛，这证明了 2D 张量并行能够应用在不同架构的模型训练中，具有良好的适用性和优化效果。

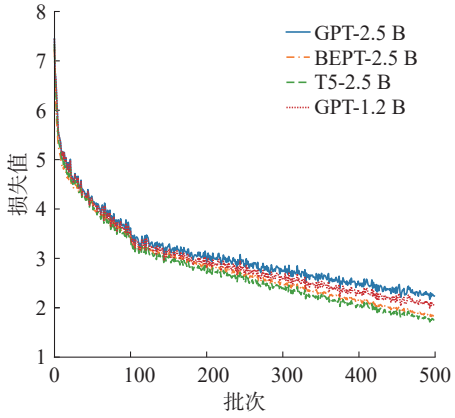

图 10　不同架构的模型训练损失值

Fig. 10　Model training loss was converged for different architectures

3.5　通信开销

本节统计了大模型训练过程中的计算和通信时间占比，比较了张量并行和 2D 张量并行的通信开销。以使用 4 块 GPU 训练 2.5 B 参数的 GPT 模型为例，使用 AIStation 软件统计了训练过程中的通信时间，使用 time.time(·) 函数记录训练中前向计算和后向传播的时间，分别得到了张量并行和 2D 张量并行的训练时间占比，如图 11、12 所示。

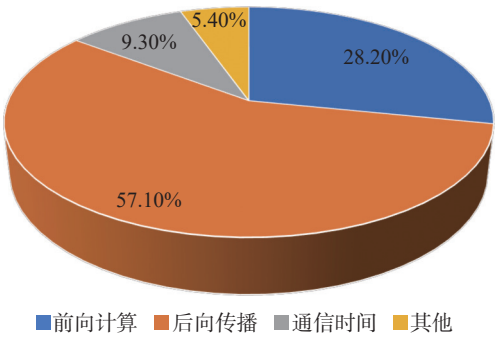

图 11　张量并行训练时间占比

Fig. 11　Time ratio of tensor parallel

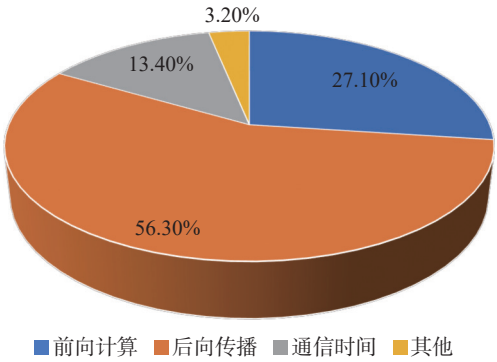

图 12　2D 张量并行训练时间占比

Fig. 12　Time ratio of 2D tensor parallel

2D 张量并行的通信时间占比为 12.4%，对比张量并行的 9.3% 有显著提高；而训练时间包括前向训练和后向传递占比略微下降，但是其他的时间占比明显降低，这意味着使用局部通信来代替全局通信提高了通信复杂度，使得通信更加频繁，但是降低了计算复杂度，使得 GPU 更加专注于计算，而减少了全局的同步和空等待。

3.6　方法应用

本节将 2D 张量并行方法与 ZERO 方法中的 CPU-Offload 相结合，并应用于心脏图像的分割领域，训练了图像分割模型 Swin-UNet，并和 UNet 等方法进行对比，分割结果如图 13 所示。图 13 中，本文方法和 Swin-UNet 使用相同的网络，但是差别在于训练方法，Swin-UNet 使用的是张量并行，本文方法使用 2D 张量并行。除此之外，所有模型的架构相同，均为 12 层网络，训练迭代次数也相同。对比原始标注，使用 2D 张量并行训练的模型在图像分割结果上表现最好。

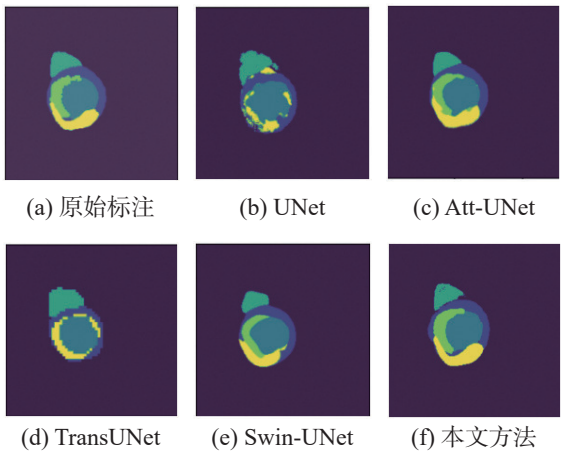

图 13　图像分割结果

Fig. 13　Results of image segmentation

本文还设计了消融实验，验证 2D 张量并行方法和 CPU-Offload 的作用，并与张量并行方法进行了对比。图 14、15 给出了 2D 张量并行方法结合 CPU-Offload 在训练过程中的 GPU 显存占用和内存资源利用率，并行度设置为 TP2D=4。表 5 给出了不同训练方法的 GPU 显存占用和内存资源利用率。2D 张量并行比张量并行方法占用更少的显存，当并行度设置为 4 时，2D 张量并行的显存占用为 3.5 GB，比张量并行的 4.8 GB 降低了 1.3 GB。表 5 的结果显示将 CPU-Offload 与 2D 张量并行相结合，降低了显存占用，并且内存资源利用率是 2D 张量并行的 3 倍，达到 1.8%。这是因为 CPU-Offload 将训练中间值保存到 CPU 内存中，在计算需要时再传递到 GPU 上，通过提高内存利用率降低 GPU 显存占用。

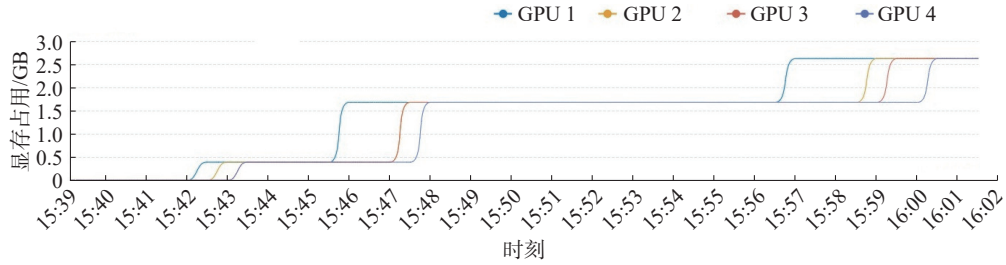

图 14　TP2D+CPU-Offload 的显存占用

Fig. 14　GPU memory usage for TP2D+CPU-Offload

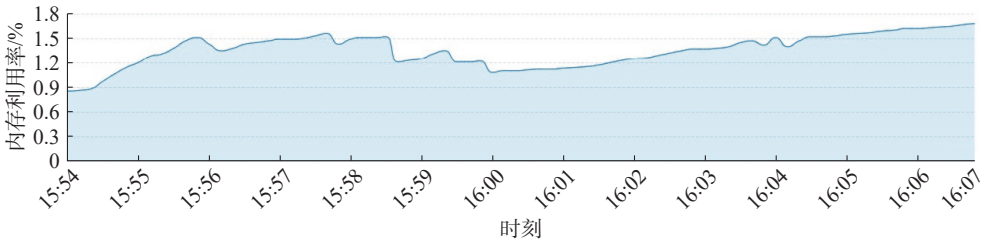

图 15　TP2D+CPU-Offload 的内存资源利用率

Fig. 15　Memory resource utilization for TP2D+CPU-Offload

表 5　消融实验

Table 5　Ablation experiment

训练方法	显存占用/GB	内存利用率/%
TP=2	8.0	0.2
TP=4	4.8	0.2
TP2D=4	3.5	0.6
TP2D=4+CPU-Offload	3.0	1.8

4　结束语

针对张量并行单个 GPU 内存限制，本文提出了 2D 张量并行，此方法融合了张量并行与数据并行的思想，并利用分布式训练进一步提升训练效率。实验证明当扩展到 4 个 GPU 时，2D 张量并行比传统张量并行具有更好的训练效率，训练速度达到了张量并行的 1.84 倍。此外，二维张量并行比张量并行表现出更好的软扩展性，说明随着 GPU 数量的增加，2D 张量并行能够训练更大规模的模型。本文通过实验证明 2D 张量并行是内存高效和训练高效的，能够保证训练稳定收敛。未来将进一步探究通信对大语言模型训练的影响，研究局部通信替代全局通信的优异性。

参考文献：

[1]　李蕾, 周延泉, 钟义信. 基于语用的自然语言处理研究与应用初探[J]. 智能系统学报, 2006, 1(2): 1−6.
LI Lei, ZHOU Yanquan, ZHONG Yixin. Pragmatic information based NLP research and application[J]. CAAI transactions on intelligent systems, 2006, 1(2): 1−6.

[2]　VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30: 5998−6008.

[3]　周飞燕, 金林鹏, 董军. 卷积神经网络研究综述[J]. 计算机学报, 2017, 40(6): 1229−1251.
ZHOU Feiyan, JIN Linpeng, DONG Jun. Review of convolutional neural network[J]. Chinese journal of computers, 2017, 40(6): 1229−1251.

[4]　HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735−1780.

[5]　LIU Yiheng, HE Hao, HAN Tianle, et al. Understanding LLMs: a comprehensive overview from training to inference[J]. Neurocomputing, 2025, 620: 129190.

[6]　MINAEE S, MIKOLOV T, NIKZAD N, et al. Large language models: a survey[EB/OL]. (2024−02−09)[2024−11−21]. https://arxiv.org/abs/2402.06196.

[7]　SCHAEFFER R, MIRANDA B, KOYEJO S. Are emergent abilities of large language models a mirage? [EB/OL]. (2023−05−22)[2024−11−21]. https://arxiv.org/abs/2304.15004v2.

[8]　WEI J, TAY Y, BOMMASANI R, et al. Emergent abilities of large language models[EB/OL]. (2022−06−15)[2024−11−21]. https://arxiv.org/abs/2206.07682.

[9]　KAPLAN J, MCCANDLISH S, HENIGHAN T, et al. Scaling laws for neural language models[EB/OL]. (2020−01−23)[2024−11−21]. https://arxiv.org/abs/2001.08361v1.

[10]　GORDON M A, DUH K, KAPLAN J. Data and parameter scaling laws for neural machine translation[C]//Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Punta Cana: Association for Computational Linguistics, 2021: 5915−5922.

[11]　SEVILLA J, HEIM L, HO A, et al. Compute trends

across three eras of machine learning[C]//2022 International Joint Conference on Neural Networks. Padua: IEEE, 2022: 1−8.

[12] LIU Qinghua, JIANG Yuxiang. Dive into big model training[EB/OL]. (2022−07−25)[2024−11−21]. https://arxiv.org/abs/2207.11912v1.

[13] 王楠禔. 基于 BERT 改进的文本表示模型研究[D]. 重庆: 西南大学, 2019.
WANG Nanzhi. Research on improved text representation model based on BERT[D]. Chongqing: Southwest University, 2019.

[14] BROWN T, MANN B, RYDER N, et al. Language models are few-shot learners[EB/OL]. (2020−05−28)[2024−11−21]. https://arxiv.org/abs/2005.14165.

[15] ACHIAM J, ADLER S, AGARWAL S, et al. GPT-4 technical report[EB/OL]. (2023−03−15)[2024−11−21]. https://arxiv.org/abs/2303.08774.

[16] TOUVRON H, LAVRIL T, IZACARD G, et al. LLaMA: open and efficient foundation language models. [EB/OL]. (2023−02−27)[2024−11−21]. https://arxiv.org/abs/2302.13971.

[17] 李德毅. 网络时代人工智能研究与发展[J]. 智能系统学报, 2009, 4(1): 1−6.
LI Deyi. AI research and development in the network age[J]. CAAI transactions on intelligent systems, 2009, 4(1): 1−6.

[18] 杨春生. 大模型不等于第三次 AI 浪潮[J]. 智能系统学报, 2023, 18(3): 409.
YANG Chunsheng. Large language models are not the third AI wave[J]. CAAI transactions on intelligent systems, 2023, 18(3): 409.

[19] RAJBHANDARI S, RASLEY J, RUWASE O, et al. ZeRO: memory optimizations toward training trillion parameter models[C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. Atlanta: IEEE, 2020: 1−16.

[20] WANG Guanhua, QIN Heyang, JACOBS S A, et al. ZeRO++: extremely efficient collective communication for giant model training[EB/OL]. (2023−06−16)[2024−11−21]. https://arxiv.org/abs/2306.10209v1.

[21] REN J, RAJBHANDARI S, AMINABADI R Y, et al. ZeRO-Offload: democratizing billion-scale model training[C]//2021 USENIX Annual Technical Conference. [S. l.]: USENIX Association, 2021: 551−564.

[22] SHOEYBI M, PATWARY M, PURI R, et al. Megatron-LM: training multi-billion parameter language models using model parallelism[EB/OL]. (2019−09−17)[2024−11−21]. https://arxiv.org/abs/1909.08053.

[23] NARAYANAN D, SHOEYBI M, CASPER J, et al. Efficient large-scale language model training on GPU clusters using megatron-LM[C]//SC21: International Conference for High Performance Computing, Networking, Storage and Analysis. Saint Louis: Association for Computing Machinery, 2021: 1−14.

[24] NARAYANAN D, PHANISHAYEE A, SHI Kaiyu, et al. Memory-efficient pipeline-parallel DNN training[C]//International Conference on Machine Learning. [S. l.]: PMLR, 2020: 7937−7947.

[25] HARLAP A, NARAYANAN D, PHANISHAYEE A, et al. PipeDream: fast and efficient pipeline parallel DNN training[EB/OL]. (2018−06−08)[2024−11−21]. https://arxiv.org/abs/1806.03377.

[26] NARAYANAN D, HARLAP A, PHANISHAYEE A, et al. PipeDream: generalized pipeline parallelism for DNN training[C]//Proceedings of the 27th ACM Symposium on Operating Systems Principles. Huntsville: ACM, 2019: 1−15.

[27] LI Shenggui, XUE Fuzhao, BARANWAL C, et al. Sequence parallelism: long sequence training from system perspective[EB/OL]. (2022−05−21)[2024−11−21]. https://arxiv.org/abs/2105.13120v3.

[28] 李铭. 基于 GPU 的张量分解及重构方法研究及应用[D]. 成都: 电子科技大学, 2018.
LI Ming. Research and application of tensor decomposition and reconstruction method based on GPU[D]. Chengdu: University of Electronic Science and Technology of China, 2018.

[29] 曹嵘晖, 唐卓, 左知微, 等. 面向机器学习的分布式并行计算关键技术及应用[J]. 智能系统学报, 2021, 16(5): 919−930.
CAO Ronghui, TANG Zhuo, ZUO Zhiwei, et al. Key technologies and applications of distributed parallel computing for machine learning[J]. CAAI transactions on intelligent systems, 2021, 16(5): 919−930.

[30] 舒娜, 刘波, 林伟伟, 等. 分布式机器学习平台与算法综述[J]. 计算机科学, 2019, 46(3): 9−18.
SHU Na, LIU Bo, LIN Weiwei, et al. Survey of distributed machine learning platforms and algorithms[J]. Computer science, 2019, 46(3): 9−18.

作者简介：

朱仕通，硕士研究生，主要研究方向为神经网络、自然语言处理和大语言模型。E-mail：zst555157@163.com。

董琦，高级工程师，博士，主要研究方向为智能博弈、自主规划和多智能体系统控制。2017 年获中国电子学会优秀博士学位论文奖，2019 年获吴文俊人工智能优秀青年奖。E-mail：dongqiouc@123.com。