深度神经网络动态分层梯度稀疏化及梯度合并优化方法

巨涛，康贺廷，刘帅，火久元
（兰州交通大学电子与信息工程学院，730070，兰州）

摘要：针对数据并行方法加速大规模深度神经网络时易出现的通信开销大、训练耗时长、资源利用率不高的问题，提出了一种深度神经网络动态分层梯度稀疏化及梯度合并优化方法。首先，将梯度稀疏化压缩与流水线并行技术相结合，提出动态分层梯度稀疏优化方法，为每层神经网络匹配一个合适的阈值，通过在后续迭代时动态调整该阈值，实现对每层网络传输梯度的自适应压缩。然后，提出了层梯度合并方法，利用动态规划算法对层梯度合并时的通信开销、稀疏化及层梯度计算时间进行权衡优化，求解出最佳的层梯度合并组合，并将多层小尺度梯度张量合并为一层通信，以降低分层梯度决策时引入的过高通信延迟开销。最后，将求解出的最佳层梯度合并组合应用于具体的训练迭代过程。实验结果表明：与已有方法相比，所提方法可在保证模型训练精度的同时大大降低通信开销，提升模型的训练速度；与未压缩方法相比，训练速度最大可提升1.99倍。

关键词：深度神经网络；分布式训练；同步数据并行；梯度压缩；层梯度合并
中图分类号：TP311 **文献标志码**：A
DOI：10.7652/xjtuxb202409011 **文章编号**：0253-987X(2024)09-0105-12

A Dynamic Layer-Wise Gradient Sparsity and Gradient Merging Optimization Method for Deep Neural Networks

JU Tao, KANG Heting, LIU Shuai, HUO Jiuyuan
(School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract: A dynamic layer-wise gradient sparsity and gradient aggregation optimization strategy for deep neural networks is proposed to address the challenges posed by substantial communication overhead, prolonged training duration, and suboptimal resource utilization associated with the acceleration of large-scale deep neural networks through data parallelism. Initially, a dynamic layer-wise gradient sparsity optimization method is proposed by combining gradient sparsity compression with pipeline parallelism. Each neural network layer is assigned an appropriate threshold, which is adjusted dynamically in subsequent iterations to achieve adaptive compression of gradient transmission for each layer. Subsequently, a layer-wise gradient merging method is introduced. Leveraging dynamic programming, this method optimizes communication overhead, sparsity, and layer gradient computation time during layer-wise gradient merging, determining the optimal combination for merging multiple layers of small-scale gradient tensors into a single communication layer. This aims to reduce the high communication latency introduced during layer-wise gradient decision-making. Finally, the

收稿日期：2024-01-19。 作者简介：巨涛(1980—)，男，副教授，硕士生导师。 基金项目：国家自然科学基金资助项目(61862037，62262038)；甘肃省科技计划资助项目(23CXGA0028)。
网络出版时间：2024-05-28 网络出版地址：https://link.cnki.net/urlid/61.1069.T.20240528.1016.002

determined optimal layer-wise gradient merging combination is applied to the specific training iteration process. Experimental results demonstrate that the proposed method, compared to existing methods, significantly reduces communication overhead and enhances model training speed while ensuring model training accuracy. It achieves a maximum training speed up of 1.99 times compared to the uncompressed method.

Keywords: deep neural network; distributed training; synchronous data parallelism; gradient compression; layer gradient merging

近年来,随着深度学习在各应用领域的快速发展,加之复杂模型架构的不断更迭和数据量的不断增加,训练大规模深度神经网络需要更多的算力资源支撑。由于单个加速设备的计算能力和存储容量已不能满足训练大规模模型的需求,因此采用分布式思想对模型进行并行化训练已成为现行的一种有效方式[1-2]。较为常见的并行化训练是基于参数服务器架构的同步数据并行方法,即各个计算节点通过维护模型参数副本的一致性,在每次训练中使用不同的小批量样本计算梯度并进行模型参数的迭代更新。具体执行思想为:每个节点在完成对本地局部梯度的计算后,将局部梯度发送给服务器节点;随后,服务器节点将来自所有计算节点的局部梯度进行聚合平均得到全局梯度,并将全局梯度用于更新模型参数[3];之后,各个计算节点从服务器节点获取最新的模型参数进行下一轮迭代,直至收敛[4-5]。其中,同步随机梯度下降算法因具有良好的收敛性,现已成为分布式深度学习训练常用的一种优化方法[6]。

然而,由于计算节点在完成反向传播后,需要将本地局部梯度发送给服务器节点并从服务器节点获取最新的模型参数,因此节点之间会产生频繁的梯度传输操作,这将使得额外的通信开销超过计算开销,逐渐成为加速深度神经网络分布式训练的主要制约因素[7]。因此,如何在不影响模型训练精度的前提下实现计算节点间的高效通信,是基于参数服务器架构的同步数据并行训练中亟待解决的问题。

1 相关工作

1.1 梯度压缩

目前,在深度神经网络分布式训练通信优化方面已有大量的研究。梯度压缩是常采用的一种有损压缩优化技术,通过减少深度神经网络训练时传输的梯度量来优化通信,主要分为量化[8-9]和稀疏化[10]两类方法。量化采用低精度的数据表示方法,通过减少每个梯度值的比特数来压缩通信量。然而,由于量化后至少需要一个比特来维持原本的梯度信息,理论上只能将梯度张量最多压缩 32 倍,无法更进一步实现高效的通信效率提升[11]。稀疏化采用更激进的通信压缩方式,通过降低无更新操作梯度的传输频率,并选择重要梯度进行传输,可以在确保模型收敛的前提下显著减少通信量。文献[12]提出了一种基于固定阈值的梯度稀疏方法,当梯度值大于设定阈值时进行传输操作,但该方法对每层梯度使用统一的均匀压缩,忽略了模型不同层之间的异构性,且难以对各类深度神经网络设置合理的阈值,缺少良好的泛化性。针对该问题,文献[13-14]提出了局部选择和动态阈值调整方法,该方法通过局部梯度选择和动态阈值调整,选择一定比例的梯度值进行训练时的通信传输操作。然而,在高压缩率的情况下,容易造成模型训练精度的损失。稀疏化的一种变体是更新重要梯度(Top-K)方法[15-17],主要通过将所有梯度进行 Top-K 操作后,筛选对模型更新影响率较大的梯度值进行传输;但在进行大规模深度神经网络训练时,由于梯度张量非常庞大,Top-K 操作往往会造成很大的时间开销。为此,文献[18]通过随机选择一定比例的梯度值进行 Top-K 操作后得到稀疏阈值,最终在全部梯度上使用此阈值进行梯度筛选;但该方法对阈值计算不够准确,在原始梯度向量的子集上需要进行两次 Top-K 采样,会造成较高的额外计算开销。Top-K 方法虽然可以显著减少通信量,但忽视了深度神经网络的分层结构特性,并且稀疏化操作通常在所有层梯度计算结束后进行,使得计算和通信无法达到良好的并行效果。为此,文献[19]提出了分层自适应梯度稀疏方法(LAGS),即在每层中通过执行 Top-K 操作选择梯度,并证明了该方法的收敛性。该方法可在梯度计算和梯度通信尽可能并行执行的前提下,压缩传输过程中的通信量,但该方法对梯度的选择不够高效,导致稀疏化时间占比较大,不能有效减少模型训练时间。

1.2 梯度分布

为实现梯度张量的高效压缩,减少稀疏化时间

开销,许多研究通过探索深度神经网络训练过程中的梯度分布情况来估计压缩阈值。文献[20-21]通过理论分析与实验验证指出,训练过程中梯度大多位于 0 附近并遵循高斯分布,并提出了一种高效的梯度选择算法 Gassian K,大幅度降低了稀疏化的时间开销,但随着训练的进行,梯度值越来越接近于 0。该方法根据高斯分布预测的阈值较大,导致模型在相同训练时间内往往难以收敛。为更准确地模拟梯度分布特征,文献[22]将训练过程中梯度分布建模为双指数分布、双伽马分布和双广义帕累托分布 3 种分布,并提出了一种基于稀疏分布的压缩方法,通过多段式的估计方法计算阈值,但当压缩比变大时,对真实梯度分布的拟合会出现较大的偏差,影响模型的收敛性。文献[23]利用对数正态分布较高斯分布更准确的特点,结合梯度量化方法提出了一种基于对数正态分布的压缩方法,达到了 80%的稀疏度,大幅度减少了通信量,但该方法仍存在对训练过程中阈值预测不够准确的问题,影响了训练精度。

1.3　计算与通信重叠

在深度神经网络的反向传播过程中,梯度从后往前依次进行计算,即前一层网络的梯度计算独立于后一层的通信,后一层的参数更新也与前一层无关。因此,可以通过将通信任务与计算任务重叠来隐藏部分通信开销[24]。然而,计算任务与通信任务重叠本质上并没有真正减少通信时间,只是尽可能将计算和通信并行执行,从而提高模型的整体训练效率。文献[25]提出了一种无等待反向传播(WFBP)算法,实现了层间梯度通信和计算的流水线并行,但由于不同层之间计算和通信时间的差异性,在高延迟或低带宽的网络环境中难以隐藏额外的通信开销。文献[26]在 WFBP 算法的基础上进行扩展,提出了基于优先级传输的方法,通过将每一层梯度张量按照预设的阈值进行切分,并利用任务间的优先级实现调度传输,达到计算和通信重叠的效果;但由于处理不同运行环境时需要频繁的超参数调整操作,使得该方法难以适应软硬件资源动态变化的适行环境。由于一些大张量的通信可能会阻塞较高优先级张量的执行,因此文献[27]提出了优先级调度中的张量分区,将大张量切分成若干小张量执行,以实现更大程度的计算与通信重叠;但每个切分后的小张量需要插入推送操作,切分过多的小张量会带来更长的总启动时间,导致带宽利用率较差。

综上,在深度神经网络模型的分布式训练中,需要结合具体的计算资源进行动态分析,通过梯度分布规律设计梯度压缩方式,将计算与通信进行重叠调度执行,可进一步优化通信传输开销,提高训练效率。

针对以上分布式数据并行训练大规模深度神经网络过程中存在的问题,本文提出一种深度神经网络动态分层梯度稀疏化及分层梯度合并优化方法 DLGS。首先,将全局 Top-K 稀疏化与流水线并行技术相结合,提出动态分层梯度稀疏决策,为每层神经网络计算一个合适的阈值,并通过在后续迭代时动态调整该阈值,实现对每层传输梯度的自适应压缩。其次,为了进一步降低通信延迟,加速模型训练,提出了层梯度合并方法,利用动态规划算法对层梯度合并时的通信耗时、稀疏化计算及层梯度计算时间进行权衡优化,求解出最优的层合并组合,将多层小尺度梯度张量合并为一层进行通信。然后,将最优的层合并组合应用于具体的训练迭代过程。最后,通过实验测试,验证了本文方法的有效性。

2　动态分层梯度稀疏化

首先,通过对已有全局 Top-K 梯度稀疏方法的局限性分析,引入分层 Top-K 梯度稀疏化方法,为每层计算一个合适的压缩阈值,使得稀疏化不需要等待反向传播的完成,将部分梯度通信开销隐藏于梯度计算过程中;之后,为了减少梯度稀疏化的时间开销,设计了阈值重用,实现对梯度高效压缩,进一步提高神经网络训练速度。

2.1　全局 Top-K 梯度稀疏化局限性分析

基于 Top-K 稀疏化具有较大绝对值的梯度对模型收敛影响更大的特点,在梯度更新过程中,每个计算节点只传递小部分绝对值最大的梯度值,剩余未选择的梯度值累积为本地梯度残差,添加到下一次迭代计算的梯度中。全局 Top-K 方法依赖于所有层梯度,如图 1 所示 Top-K 稀疏化执行过程,当反向计算完所有层梯度后,才开始对所有层梯度执行稀疏化和梯度通信等操作,由于不会使得反向传播过程中梯度计算和通信操作重叠,因此会导致训练效率较低。

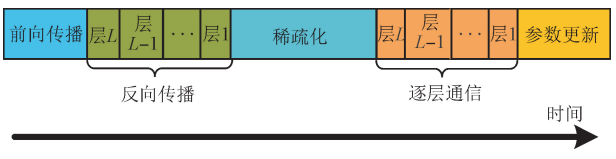

图 1　Top-K 稀疏化执行过程

Fig. 1　Top-K sparsification execution process

2.2　动态分层梯度稀疏化方法

为了使梯度计算和通信在尽可能并行执行的前

提下减少梯度传输的通信量，本文在每一层上应用 Top-K 梯度稀疏化和残差累积，具体执行时不是从所有层选择 K 个梯度值，而是从每层 l 中选择 K 个梯度值，当层 l 计算完成梯度后，立即对该层计算得到的梯度进行稀疏化操作，这样不仅利用稀疏化方法减少了传输过程中的通信量，而且还利用深度神经网络的分层结构使得梯度计算和通信重叠。具体层阈值计算与动态更新过程为：初始迭代时，对每层 l 计算得到的梯度应用 Top-K 方法，将每层中通过 Top-K 方法选择的第 K 个梯度值作为层 l 的阈值，表示为 H_l，用阈值列表 $\boldsymbol{V}=[H_1,H_2,\cdots,H_l]$ 存储每层的阈值。通过比较多次迭代后每层的阈值变化，发现阈值只有轻微的差异，因此不需要在每次迭代过程中对层 l 计算得到的梯度执行 Top-K 操作计算阈值，而是每隔 s 次迭代，使用 Top-K 操作计算每层的准确阈值，并使用所计算的准确阈值来动态更新层阈值列表 $\boldsymbol{V}$。然后，在接下来的 $(s-1)$ 次迭代过程中重用该阈值。通过阈值重用，可以减少在层 l 执行 Top-K 的时间，从而进一步降低稀疏化的时间开销。

整个动态分层梯度压缩执行过程如图 2 所示。首先层 l 计算出第 i 次训练的梯度 $g_i^{n,l}$（n 为计算节点），将层 l 计算的梯度 $g_i^{n,l}$ 与该层中的梯度残差 $e_{i-1}^{n,l}$ 相加，梯度残差是计算节点中层 l 局部累积的所有先前未传输梯度的总和；然后对该层梯度执行 Top-K 操作，得到该层梯度压缩阈值 H_l，通过该阈值对层 l 的梯度 $g_i^{n,l}$ 压缩；最后将压缩后的梯度 $C(g_i^{n,l})$ 发送给参数服务器，并累积本次迭代的梯度残差 $e_i^{n,l}$，用于之后迭代计算得到的梯度中。

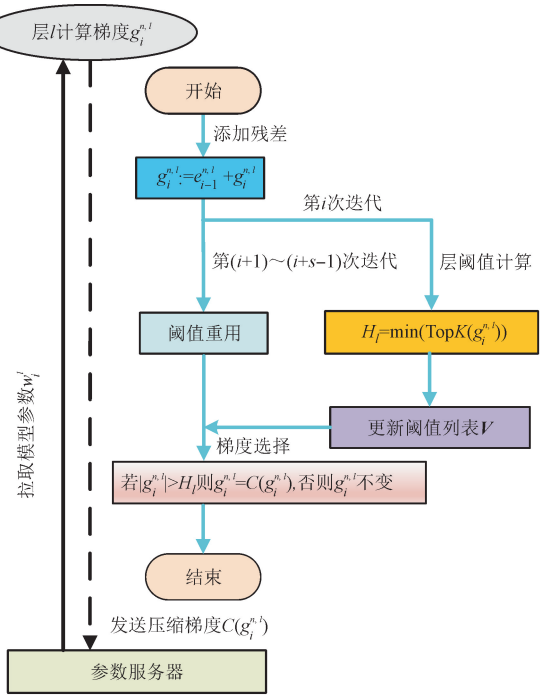

图 2 分层梯度压缩执行过程

Fig. 2 Execution process of layered gradient compression

2.3 动态分层梯度压缩算法

在对每层梯度计算合适的压缩阈值基础上，利用每层计算的阈值，实现分层梯度压缩。假设在计算节点 n 上给定一个具有 L 层的深度神经网络 M，在第 i 次训练时将压缩后的梯度 $C(g_i^{n,l})$ 发送给参数服务器，当参数服务器收到各个计算节点计算的梯度后，对所有梯度求平均值并更新模型参数。具体的参数更新公式为

$$w_{i+1}^l = w_i^l - \alpha_i \frac{1}{N}\sum_{n=1}^{N} C(g_i^{n,l}) \tag{1}$$

式中：w_i^l 表示第 i 次迭代层 l 模型参数；α_i 是学习率；N 为计算节点数；$g_i^{n,l}$ 表示计算节点 n 在第 i 次迭代中计算的无偏随机梯度与层 l 的本地梯度残差的累加和；$C(g_i^{n,l})$ 表示对层 l 计算的梯度 $g_i^{n,l}$ 进行压缩。动态分层梯度压缩算法迭代训练的具体过程见算法 1。

算法 1 动态分层梯度压缩算法

输入：数据集 D，每层初始权重 w^l，层压缩率 K，学习率 α，迭代次数 T，节点数量 N

输出：模型参数 w_T

初始化 $\boldsymbol{V}=[H_1,\cdots,H_L]=0$
初始化 $E=[e_i^{n,1},\cdots,e_i^{n,L}]=0$
for $i=1$ to T do
  从数据集 D 中采样小批量数据 D_i^n
  前向计算；
  for $l=L$ to 1 do
    $g_i^{n,l}=e_{i-1}^{n,l}+\nabla \mathrm{Loss}(w_t, D_i^{n,l})$；
    $\boldsymbol{M}=|g_i^{n,l}|>H_l$；
    $C(g_i^{n,l})=g^{n,l}\odot\boldsymbol{M}$；
    $e_i^{n,l}=g_i^{n,l}\odot\boldsymbol{M}$；
    $g_{\mathrm{golobal}}^{n,l}=\mathrm{AVG}(\mathrm{Reduce}(C(g_i^{n,l}))$；
  end for
  if i mod $s==0$ then
    update V
  end if
  $w_{i+1}^l=w_i^l-\alpha_i g_{\mathrm{global}}^{n,l}$
end for

算法主要执行步骤如下。

步骤 1 每个计算节点加载训练模型和部分训练数据集，初始化每层梯度阈值列表 V，初始化每层梯度

压缩残差矩阵 $\boldsymbol{E}$。

步骤 2　迭代训练,共迭代训练 T 次,每次迭代从数据集 D 中采样小批量数据 D_i^n,前向计算损失。

步骤 3　从最后一层 L 逐层往前计算梯度 $g_i^{n,l}$,将层 l 梯度值同该层梯度残差 $e_{i-1}^{n,l}$ 相加得到 $g_i^{n,l}=g_i^{n,l}+e_{i-1}^{n,l}$,之后根据该层中的阈值 H_l 对该层梯度进行压缩,将压缩后的梯度 $C(g_i^{n,l})$ 发送给参数服务器更新模型参数,并将小于阈值的梯度在本地累积为梯度残差,用于之后的迭代训练。

步骤 4　参数服务器收到所有计算节点计算的梯度后,对梯度求平均值,并用式(1)更新模型参数。

步骤 5　参数服务器将最新的模型参数广播给计算节点,更新计算节点的模型参数,并开始下一轮的迭代训练。

3　层梯度合并

分层梯度稀疏化通过减少通信量可以有效降低通信开销,加速模型训练过程,但在分层压缩后的梯度通信过程中,对于不同类型的层,要通信的梯度张量大小存在较大差异,传输过多层的小张量不仅会导致带宽利用不足,还会增加通信延迟[28]。为此,本文提出多层梯度合并方法,将多个层的梯度张量合并一起发送,以降低通信延迟开销,加速模型训练过程。

3.1　层梯度合并策略

层梯度合并的目标是在训练过程中尽可能多地将多个稀疏后的梯度张量合并到一起通信,在实现梯度计算和梯度通信最佳重叠的同时,最小化通信延迟开销。在合并多个层梯度张量过程中,如将整个模型的梯度张量合并成一个单一张量,虽然通信操作只调用一次,通信延迟最小,但通信过程必须等待反向传播结束,没有使梯度计算与通信重叠,从而影响训练效率。为了获得最佳的层梯度合并结果,应同时考虑梯度计算时间、梯度压缩时间和梯度通信时间对整个训练效率的影响。本文首先将层梯度合并问题转化为使得一次迭代最小的优化问题,优化目标是通过层梯度合并来降低通信延迟开销,从而最小化迭代时间;然后,对训练过程中梯度计算时间、梯度压缩时间和梯度通信时间进行分析,判断层梯度合并的前提条件;最后,利用动态规划算法求解最佳的层梯度合并组合,并用于之后的迭代训练过程。

3.1.1　训练过程分析

本文提出的动态分层梯度稀疏化方法,一次迭代训练时间 t_{iter} 由前向计算损失时间、反向逐层计算梯度时间、层梯度压缩时间和压缩后梯度通信时间 4 部分组成,表示如下

$$t_{\text{iter}}=t_{\text{f}}+\sum_{l=1}^{L}t_{\text{b}}^{l}+\sum_{l=1}^{L}t_{\text{s}}^{l}+t_{\text{c}}^{\text{no}} \tag{2}$$

式中:t_{iter} 表示一次迭代总时间;t_{f} 表示前向计算时间;t_{b}^{l} 表示层 l 的反向计算时间;t_{s}^{l} 表示层 l 的稀疏化时间;t_{c}^{no} 表示没有重叠的通信时间。τ_{b}^{l}、τ_{s}^{l}、τ_{c}^{l} 计算公式分别为

$$\tau_{\text{b}}^{l}=\begin{cases}t_{\text{f}}, & l=L\\ \tau_{\text{s}}^{l+1}+t_{\text{s}}^{l+1}, & 1\leqslant l<L\end{cases} \tag{3}$$

$$\tau_{\text{s}}^{l}=\tau_{\text{b}}^{l}+t_{\text{b}}^{l} \tag{4}$$

$$\tau_{\text{c}}^{l}=\begin{cases}\tau_{\text{s}}^{l}+t_{\text{s}}^{l}, & l=L\\ \max\{\tau_{\text{s}}^{l}+t_{\text{s}}^{l},\ \tau_{\text{c}}^{l+1}+t_{\text{c}}^{l+1}\}, & 1<l\leqslant L\end{cases} \tag{5}$$

式中:τ_{b}^{l} 表示层 l 开始计算梯度的时刻,取值为前向计算结束的时刻或者其前一层$(l+1)$稀疏化结束时刻;式(4)表示层 l 的开始稀疏化的时刻 τ_{s}^{l},取值为层 l 的梯度计算结束时刻;式(5)中,τ_{c}^{l} 表示层 l 开始通信的时刻,它由层 l 的稀疏化结束时刻和其前一层$(l+1)$的通信结束时刻决定。

3.1.2　合并过程分析

层梯度合并过程中,将多个层梯度张量合并到一起通信,虽然会缓解通信延迟开销,但是会推迟前一层或者几层梯度开始稀疏化和通信的时刻,如图 3a 中,层 L 计算完梯度并压缩后会在时刻 1 开始该层梯度通信,图 3b 中将层 L 和层$(L-1)$合并一起通信,会将层 L 开始梯度通信的时刻延迟到时刻 2。如果将所有层梯度合并到一起发送,则通信延迟开销最低,从而使得通信成本最低,但这种通信方式需要计算完所有层梯度才开始通信,不能实现梯度计算和通信并行执行。因此,层梯度合并的目标是在训练过程中尽可能将层梯度张量合并的同时,实现梯度通信和计算的最佳重叠。

将合并层定义为:如果在时刻 τ 处,将层 l 的梯度合并到层$(l-1)$,并不压缩和通信层 l 的梯度,则层 l 为合并层,表示为 $l\oplus(l-1)$。用符号 l_m 表示合并层,用符号 l_n 表示普通层,则

$$\begin{cases}l=l_m\\ \text{s.t.}\begin{cases}l_m>1\\ t_{\text{c}}^{l}=t_{\text{s}}^{l}=0\\ d^{l-1}=d^{l}+d^{l-1}\\ \tau_{\text{b}}^{l-1}=\tau_{\text{b}}^{l}+t_{\text{b}}^{l}\end{cases}\end{cases} \tag{6}$$

式中:$l_m>1$ 表明第 1 层不能为合并层,因为没有前

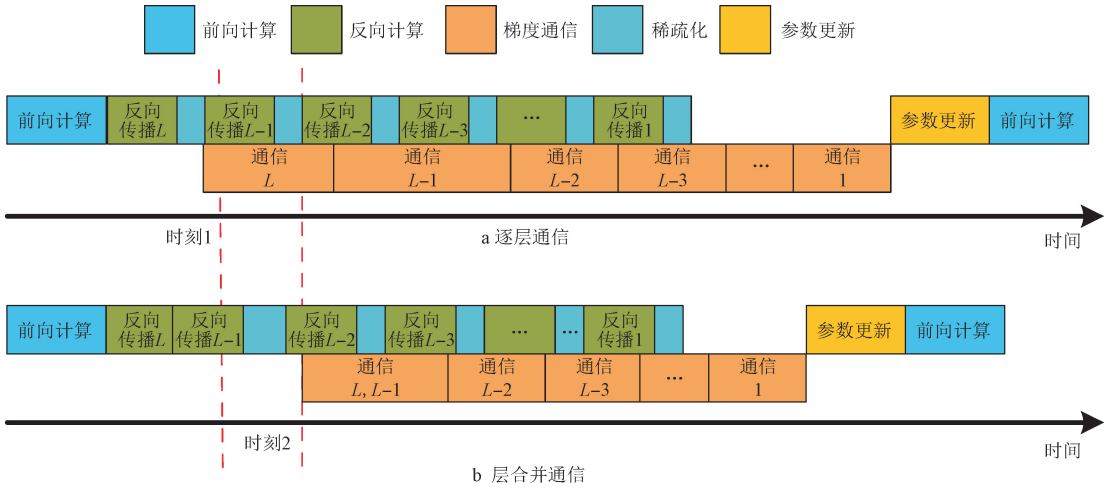

图 3 层梯度合并通信示意图

Fig. 3 Illustration of layer gradient aggregation communication

一层要合并；$t_c^l = t_s^l = 0$ 表明 l_m 这一层梯度计算完成后没有压缩和通信；$d^{l-1} = d^l + d^{l-1}$ 表明 l_m 这一层的梯度张量 d^l 将会累积到前一层张量 d^{l-1} 中；$\tau_b^{l-1} = \tau_b^l + t_b^l$ 表示层$(l-1)$的梯度计算可以在层 l 梯度计算完成后立即开始。

对于一个具有 L 层的深度神经网络模型，若某层 l 属于普通层或者合并层，则所有的层梯度合并可能的组合 O 有 2^{L-1} 种。在对梯度分层通信过程中，由于反向计算梯度是从后往前逐层进行的，因此一次迭代结束时间 t_{iter} 为第 1 层梯度通信结束时刻，可写为

$$t_{iter} = \tau_c^1 + t_c^1 \tag{7}$$

式中：τ_c^1 是第 1 层开始通信的时刻；t_c^1 是第 1 层通信的时间。式(7) 表示一次迭代结束时刻 t_{iter} 等价于第 1 层梯度通信结束时刻。同时由式(5) 可知，第 1 层的通信开始时刻由第 1 层的梯度稀疏化结束时刻和第 2 层通信结束时刻决定，第 2 层中的通信开始时刻由第 2 层中的梯度稀疏化结束时刻和第 3 层通信结束时刻决定，如此类推，直到第 L 层开始计算梯度为止。

3.2 优化目标

给定一个具有 L 层的深度神经网络模型 M，优化目标是从所有的层梯度可能的组合 O 中找到一种组合 $m \in O$，使得一次迭代时间 t_{iter} 最小，在最小化通信延迟开销的同时达到最佳的梯度计算与梯度通信的重叠，用公式表示为

$$\begin{cases} \min\ t_{iter} = t_f + \sum_{i=1}^{L} (\tau_c^l + t_c^l) \\ \text{s.t.}\ \ m \in O \end{cases} \tag{8}$$

式中：τ_c^l 表示层 l 开始通信的时刻；t_c^l 表示层 l 通信的时间；$\tau_c^l + t_c^l$ 表示计算层 l 通信结束的时刻。层 l 开始通信的时刻 τ_c^l，由层$(l+1)$的通信时间或层 l 的稀疏化时间决定，即 $\tau_c^l = \max\{\tau_c^{l+1} + t_c^{l+1}, \tau_s^l + t_s^l\}$。式(8) 的目标是找到一个最优的组合 m 最小化迭代时间 t_{iter}。

3.3 层梯度合并算法

在对优化目标迭代求解过程中，关键是确定合并一个层是否能减少迭代时间 t_{iter}，如果可以减少迭代时间，则将该层设为合并层，否则，将该层保持为普通层。对于任意层 l，若将其梯度合并后总的迭代时间变为 t'_{iter}，如果使得 $t'_{iter} < t_{iter}$，则将层 l 的梯度合并，否则不合并。图 4 描述了具体迭代规划合并过程。

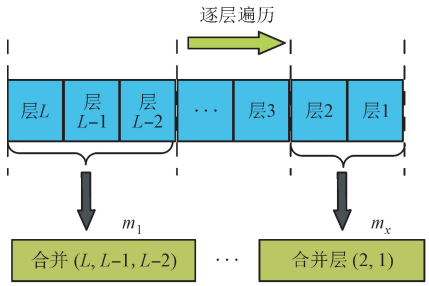

图 4 层梯度合并过程

Fig. 4 Layered gradient aggregation process

假设已经将层 L、$L-1$、$L-2$ 进行了合并，形成层梯度组合 $m_1 = [L, L-1, L-2]$，则在该组合中，当层 L 和层$(L-1)$的梯度计算完成后，没有立即压缩和通信，而是将层 L 和层$(L-1)$的梯度张量累积到层$(L-2)$中，即 $d^{L-2} = d^{L-2} + d^{L-1} + d^L$。接着，从层

$(L-3)$ 开始,对剩余部分层继续进行迭代规划,形成下一个层梯度组合 $m_j(1\leqslant j\leqslant x)$。当所有层都遍历完之后,所有层被划分构成一个层梯度组合 $m=\{m_1,m_2,\cdots,m_x\}$。在之后的训练过程中,将每个组合 m_j 中所包含的若干层进行一次通信即可。

具体合并过程实现如算法 2 所示,其中函数 CommTime( )表示返回层 l 的通信时间,函数 TopKTime( )表示返回层 l 稀疏化的时间,函数 CommStartTime( )和 CompStartTime( )是预先设置用于计算层 l 梯度开始通信时刻 τ_c 和梯度计算的起始时刻 τ_b;函数 Merge( )执行梯度合并操作,将预定义的普通层更改为合并层。

算法 2　最优层梯度合并算法

```
1  初始化 m[1,…,L] = {l_n}
2  for l=L to 2 do
3      μ = τ_b[l] + t_b[l] + t_b[l−1]
4      if t'_iter < t_iter do
5          Merge(t_b, t_s, t_c, d[l])
6          τ_b2, τ_s2, t_s2 = CompStartTime (t_b[1,…,
           L], d[1,…,L], τ_b[l] + t_b[l]);
7          τ_b[1,…,L] = τ_b2, τ_s[1,…,L] = τ_s2;
8          τ_c, t_c = CommStartTime(t_s, τ_c, d[l]);
9          m[l] = l_m;
10     end if
11 end for
12 return m;
13 Procedure Merge(t_b, t_s, t_c, l)
14     t_b[l−1] = t_b[l−1] + t_b[l];
15     d[l−1] = d[l−1] + d[l];
16     t_c[l−1] = CommTime(d[l−1]);
17     t_s[l−1] = Top K Time(d[l−1]);
```

整个合并算法主要执行步骤如下。

步骤 1　初始化神经网络所有层为普通层,通过函数 CommStartTime()和 CompStartTime()记录层 l 梯度开始通信和计算的起始时刻。

步骤 2　从最后一层 L 开始向前逐层遍历,对于遍历到的每一层 l,判断合并该层到前一层$(l-1)$与不合并该层是否可以减少迭代时间。

步骤 3　如果可以减少迭代时间,则调用 Merge()函数将预定义的普通层更改为合并层,将该层 l 的梯度张量累积到前一层梯度张量中,然后将层号推入层梯度组合 m_j 中。

步骤 4　如果不能减少迭代时间,则将该层梯度与其前一层不合并,并从前一层开始创建一个新的层梯度组合 m_{j+1},继续往前逐层判断。

步骤 5　当所有层遍历结束后,求得最优的组合方式 m。

算法从第 L 层开始逐层往前扫描直到第 2 层为止,需要 $O(L)$ 步,对于每次扫描,如果层 l 是可以合并的层,则需要计算该层梯度计算和通信开始时间,复杂度为 $O(L)$,因此整个算法的复杂度为 $O(L^2)$。

3.4　层合并后训练执行流程

通过对层梯度合并的分析和求解,得到最优的层梯度组合方式 $m=[m_1,m_2,\cdots,m_x]$,将求得的最优层梯度合并方式组合 m 应用于迭代训练过程中。层合并后训练执行流程如下所示。

步骤 1　每个计算节点初始化共享同步队列 Q,存储每层的层号,计算每层的参数大小等相关变量;

步骤 2　在服务器节点(Rank 0)调用算法 2 求最优层梯度合并方式 m,并向所有计算节点广播 m;

步骤 3　每个计算节点得到层梯度组合 m,初始化相关变量,迭代的读取小批量数据进行前向传播并求损失;

步骤 4　根据损失进行反向传播计算得到当前层随机梯度 $g_i^{n,l}$,然后根据合并后的层梯度组合 m_j 判断是否立即压缩和通信该层梯度。

步骤 5　将层梯度组合 m_j 中的层号推入共享队列,并将 m_j 中包含的多个层压缩后的梯度 $C(g_i^{n,l})$ 一起发送给参数服务器。

步骤 6　参数服务器收到所有计算节点发送的层梯度组合 m_j 的多个层梯度,求和取平均值后,更新服务器节点的模型参数。

步骤 7　计算节点从服务器节点拉取更新后的模型参数,更新本地模型,进入下一轮迭代训练。

4　实　验

4.1　实验设置

(1)硬件环境。实验使用北京超算中心的 N40 分区,采用 4 个节点,其中 1 个为参数服务器节点,另外 3 个为计算节点。服务器节点和计算节点之间通过 RoCE 2×25 Gb/s(RDMA 协议)建立连接。每个节点各配备 1 块 CPU 和 GPU 卡,硬件配置相关参数为:CPU 为 AMD EPYC 7402 (48C)@2.8 GHz,内存 512 GB,显卡为 NVIDIA® GeForce® RTX 4090,显存 24 GB。

(2)软件环境。并行训练使用 Horovod 分布式深度学习框架,进行上层分布式梯度计算和通信等封装,底层采用 PyTorch 深度学习框架。操作系统为 Red Hat 4.8.5,Horovod 版本为 0.28.1,PyTorch 版

本为 2.0.1,CUDA 版本为 11.8,Python 版本为 3.8,NCCL 版本为 2.18.1。

(3)网络模型及数据集。分别用 ResNet-20 和 VGG16 两种网络模型在 CIFAR-10 数据集、用 ResNet-50 网络模型在 ImageNet 数据集子集上进行验证。其中,CIFAR-10 数据集由 10 个类别的60 000 张 32×32 像素的彩色图片组成,其中包含50 000张训练图片和 10 000 张测试图片。ImageNet 数据子集由截取的 100 个类的 135 000 张图片组成,并裁剪为 224×224 像素,其中一个训练类包含 1 300 张图片,一个测试类包含 50 张图片。

(4)实验参数设置。在同步数据并行框架下,使用分布式随机梯度下降优化算法,在 CIFAR-10 数据集共迭代训练 140 轮(Epoch)。学习率采用阶梯式设置,前 80 轮设置为 0.1,80～120 轮设置为 0.01,120～140 轮设置为 0.001,批量大小设置为 32。在 ImageNet 数据集共迭代训练 80 轮,前 30 轮学习率设置为 0.1,30～60 轮设置为 0.01,60～80 轮设置为 0.001,批量大小设置为 64。

4.2 实验结果与分析

4.2.1 并行训练精度与损失对比

为了验证本文方法(DLGS)在提升模型训练精度和降低训练损失方面的有效性,在 0.1 压缩率下分别使用 ResNet-20、VGG16 和 ResNet-50 模型测试不同压缩方法的训练精度和训练损失。ResNet-20 模型的训练精度和训练损失分别如图 5 和图 6 所示,VGG16 模型的训练精度和训练损失分别如图 7 和图 8 所示,ResNet-50 模型的训练精度和训练损失分别如图 9 和图 10 所示。不同压缩方法的最终训练精度和训练损失对比如表 1 所示。

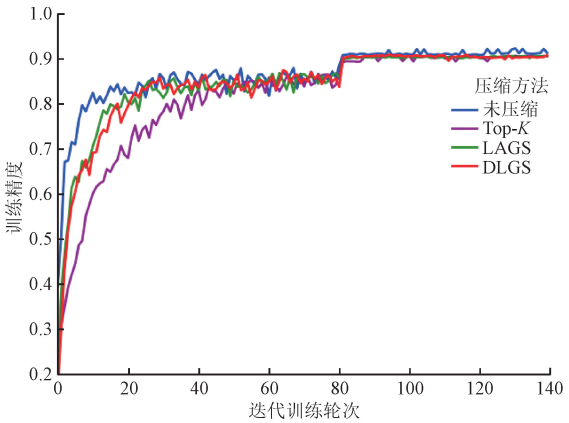

图 5 不同压缩方法在 ResNet-20 模型上的训练精度对比

Fig. 5 Training accuracy comparison for different compression methods on ResNet-20 model

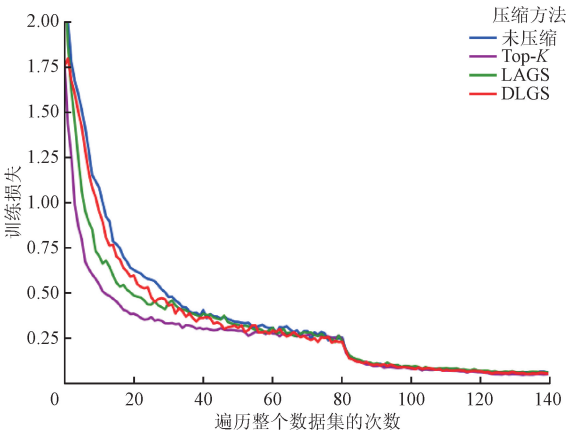

图 6 不同压缩方法在 ResNet-20 模型上的训练损失对比

Fig. 6 Training loss comparison for different compression methods on ResNet-20 model

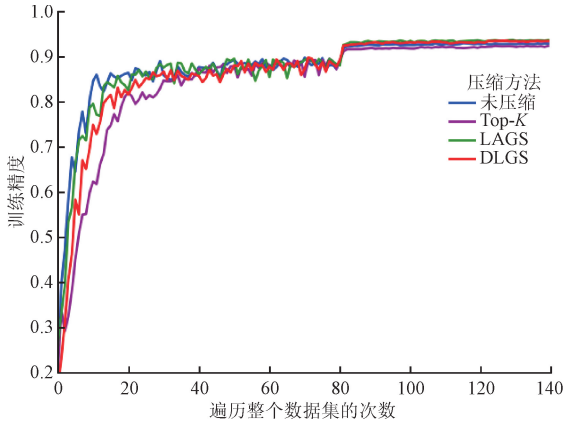

图 7 不同压缩方法在 VGG16 模型上的训练精度对比

Fig. 7 Training accuracy comparison for different compression methods on VGG16 model

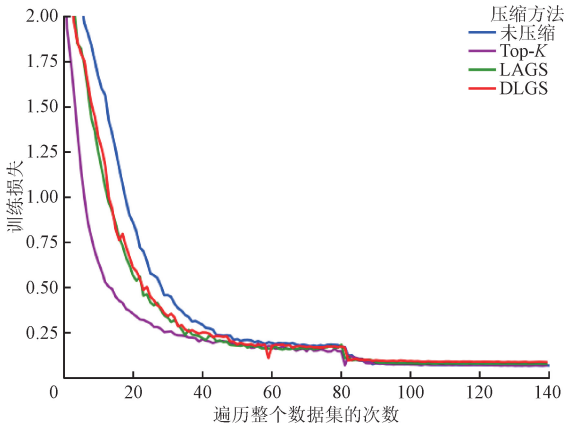

图 8 不同压缩方法在 VGG16 模型上的训练损失对比

Fig. 8 Training loss comparison for different compression methods on VGG16 model

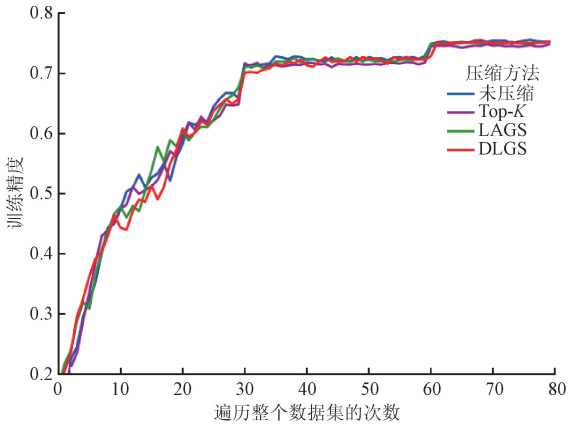

图 9　不同压缩方法在 ResNet-50 模型上的训练精度对比

Fig. 9　Training loss comparison for different compression methods on ResNet-50 model

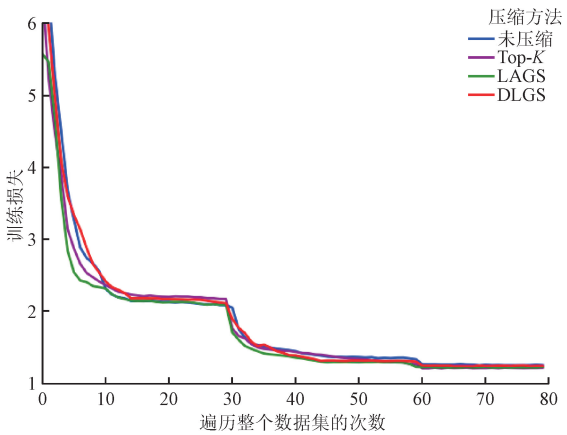

图 10　不同压缩方法在 ResNet-50 模型上的训练损失对比

Fig. 10　Training loss comparison for different compression methods on ResNet-50 model

表 1　不同压缩方法的最终训练精度和训练损失对比

Table 1　Comparison of final training accuracy and loss for different compression methods

压缩方法	ResNet-20 模型		VGG16 模型		ResNet-50 模型	
	最终训练损失	最终训练精度	最终训练损失	最终训练精度	最终训练损失	最终训练精度
未压缩	0.043 1	0.910 2	0.062 2	0.931 9	1.198 6	0.755 2
Top-K	0.054 4	0.903 7	0.063 2	0.921 6	1.243 3	0.748 1
LAGS	0.051 6	0.907 2	0.072 5	0.926 4	1.215 9	0.753 6
DLGS	0.048 4	0.906 6	0.082 2	0.927 2	1.203 2	0.753 3

从图 5 和图 7 可以看出，与 Top-K、LAGS、未压缩等方法相比，本文 DLGS 方法在训练开始时与其他各压缩方法训练精度差异较小，随着迭代次数的增加，本文方法训练精度提升效果仅次于未压缩方法，在第 40 轮时训练精度趋于稳定，在第 80 轮时，各压缩方法精度都有显著的提升，主要是因为在第 80 轮后学习率衰减为 0.01。从图 6 和图 8 的训练损失曲线可以看出，各压缩方法在训练过程中训练损失迅速降低，之后趋于稳定，在第 80 轮后损失又有了明显的下降，主要是因为 80 轮后学习率衰减为原来的 1/10。图 9 和图 10 表明，本文方法在 ResNet-50 模型上的训练精度和训练损失同其他两种模型上的类似。从表 1 可以看出，本文方法在 3 种网络模型上的训练精度和训练损失方面差距不大，十分接近于未压缩方法，表明本文方法可以在减少训练时间的同时保证模型精度。

4.2.2　训练耗时对比

为评估在提升训练速度上的有效性，在 0.1 和 0.01 两种压缩率下，对模型 ResNet-20、VGG16 和 ResNet-50 的训练耗时进行对比，不同压缩方法训练耗时如表 2 所示。从表 2 可知，本文 DLGS 方法在两种压缩率下的训练耗时均少于其他压缩方法，主要原因是本文方法通过阈值重用以及梯度稀疏化对多个层梯度进行了合并通信，减少了模型训练过程中的压缩开销和通信开销，从而加速了模型整体训练过程。本文 DLGS 方法在 0.01 的压缩率下训练时间和非压缩方法相比，最大加速比达到了 1.99 倍；在 VGG16 模型上 DLGS 同其他压缩方法相比，加速比提高不明显，主要是因为 VGG16 输入分辨率较小，梯度计算时间较小而参数量较大。本文方法更倾向于合并多个层的梯度，导致梯度计算和梯度通信重叠程度相对较低。

4.2.3　压缩性能分析

为了验证本文方法在提升模型训练速度方面的有效性，将一次迭代的训练时间分解为计算时间、稀疏化时间和非重叠的通信时间，并在 0.1 和 0.01 两种压缩率下，比较不同压缩方法一次迭代的各部分训练时间，结果如图 11～图 13 所示。

表 2 不同压缩方法训练耗时对比

Table 2 Training time comparison of different compression methods

模型	压缩率	压缩方法	耗时/s	加速比
ResNet-20	0.01	未压缩	1 517	1.00
		Top-K	1 044	1.45
		LAGS	976	1.55
		DLGS	762	1.99
	0.1	未压缩	1 517	1.00
		Top-K	1 296	1.21
		LAGS	1 181	1.32
		DLGS	936	1.56
VGG16	0.01	未压缩	2 496	1.00
		Top-K	1 706	1.46
		LAGS	1 544	1.62
		DLGS	1 384	1.81
	0.1	未压缩	2 496	1.00
		Top-K	1 964	1.27
		LAGS	1 826	1.37
		DLGS	1 632	1.53
ResNet-50	0.01	未压缩	33 906	1.00
		Top-K	22 287	1.52
		LAGS	23 462	1.44
		DLGS	20 733	1.62
	0.1	未压缩	38 518	1.00
		Top-K	29 404	1.31
		LAGS	26 782	1.42
		DLGS	24 146	1.59

从图 11～图 13 可以看出，在 3 种训练模型上，由于未压缩方法的通信量很大，通信时间远大于计算时间，导致一次迭代的训练总时间比其他方法的总时间长；在 0.1 和 0.01 两种压缩率下，由于 Top-K 方法减少了通信量进而减少了通信时间，因此比未压缩方法的训练时间短；LAGS 方法在两种压缩率下稀疏化时间比 Top-K 方法多，主要原因是每层调用 Top-K 函数导致稀疏化开销增加，但该方法重叠了计算和通信，故在整体训练时间上少于 Top-K 方法。本文 DLGS 方法在两种压缩率下耗时均最短，一方面是因为本文方法减少了通信量从而减少了通信时间，且在这个过程中计算和通信重叠，将部分通信开销隐藏在计算中；另一方面是因为本文方法通过阈值重用减少了稀疏化的时间开销，并通过层梯度合并方法降低了通信延迟开销，从而进一步降低了整个训练迭代的时间。

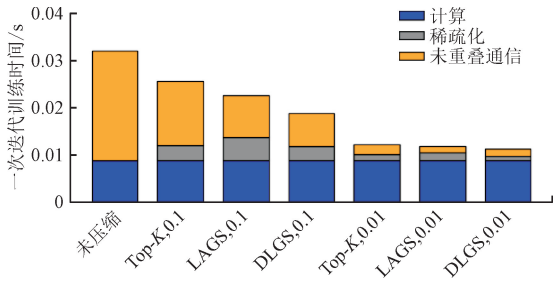

图 11 ResNet-20 模型上不同压缩方法训练时间组成

Fig. 11 Composition of training time for different compression methods on ResNet-20 model

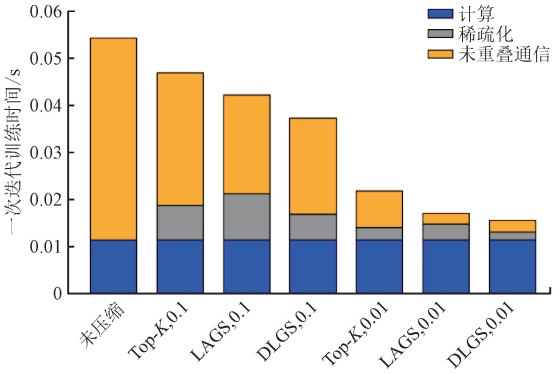

图 12 VGG16 模型上不同压缩方法训练时间组成

Fig. 12 Composition of training time for different compression methods on VGG16 model

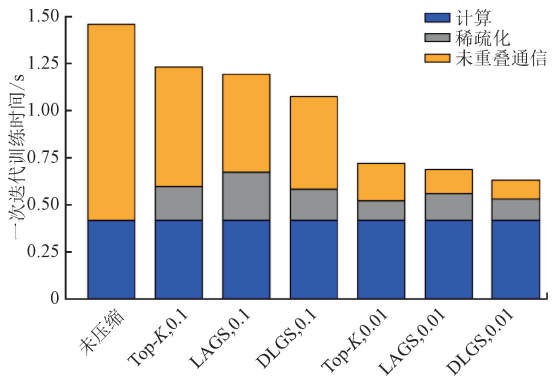

图 13 ResNet-50 模型上不同压缩方法训练时间组成

Fig. 13 Composition of training time for different compression methods on ResNet-50 model

5 结 论

本文提出了一种动态分层梯度稀疏化及梯度合并优化方法，用于解决基于参数服务器的数据并行优化方法加速大规模深度神经网络模型训练过程中存在的通信开销大、训练耗时长等问题。首先，通过在迭代过程中动态调整每一层的压缩阈值，有效减少了通信量，使得层梯度压缩及梯度计算与通信可

以流水线并行执行，重叠了部分通信时间。其次，利用动态规划对层梯度合并时的通信耗时、稀疏化计算及层梯度计算时间进行权衡优化，求解出最优的层合并组合，将多层小尺度梯度张量合并为一层进行通信。最后，将最优的层合并组合应用于具体的训练迭代过程。实验结果表明，本文所提方法在并行训练精度、训练耗时、压缩效果方面和已有方法相比都有明显的优势，可以在保证训练精度的情况下，进一步减少通信量，最大化计算和通信的重叠，最终降低深度神经网络模型训练耗时，提高模型的训练速度。

参考文献：

[1] 朱泓睿，元国军，姚成吉，等. 分布式深度学习训练网络综述 [J]. 计算机研究与发展，2021，58(1)：98-115.
ZHU Hongrui，YUAN Guojun，YAO Chengji，et al. Survey on network of distributed deep learning training [J]. Journal of Computer Research and Development，2021，58(1)：98-115.

[2] 王帅，李丹. 分布式机器学习系统网络性能优化研究进展 [J]. 计算机学报，2022，45(7)：1384-1411.
WANG Shuai，LI Dan. Research progress on network performance optimization of distributed machine learning system [J]. Chinese Journal of Computers，2022，45(7)：1384-1411.

[3] 巨涛，赵宇阳，刘帅，等. 面向图片识别的深度学习模型并行优化方法 [J]. 西安交通大学学报，2023，57(1)：141-151.
JU Tao，ZHAO Yuyang，LIU Shuai，et al. A parallel optimization method of deep learning model for image recognition [J]. Journal of Xi'an Jiaotong University，2023，57(1)：141-151.

[4] LI Mu，ANDERSEN D G，PARK J W，et al. Scaling distributed machine learning with the parameter server [C]//Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation. Berkeley，CA，USA：USENIX Association，2014：583-598.

[5] 朱虎明，李佩，焦李成，等. 深度神经网络并行化研究综述 [J]. 计算机学报，2018，41(8)：1861-1881.
ZHU Huming，LI Pei，JIAO Licheng，et al. Review of parallel deep neural network [J]. Chinese Journal of Computers，2018，41(8)：1861-1881.

[6] BOTTOU L，CURTIS F E，NOCEDAL J，et al. Optimization methods for large-scale machine learning [J]. SIAM Review，2018，60(2)：223-311.

[7] 高赫然，吴恒，许源佳，等. 面向深度学习训练的内存交换机制综述 [J]. 软件学报，2023，34(12)：5862-5886.
GAO Heran，WU Heng，XU Yuanjia，et al. Survey on memory swapping mechanism for deep learning training [J]. Journal of Software，2023，34(12)：5862-5886.

[8] 巨涛，刘帅，王志强，等. 深度神经网络模型任务切分及并行优化方法 [J/OL]. 北京航空航天大学学报：1-18[2024-04-13]. https://doi.org/10.13700/j.bh.1001-5965.2022.0731.
JU Tao，LIU Shuai，WANG Zhiqiang，et al. Task segmentation and parallel optimization of DNN model [J/OL]. Journal of Beijing University of Aeronautics and Astronautics：1-18 [2024-04-13]. https://doi.org/10.13700/j.bh.1001-5965.2022.0731.

[9] YAN Guangfeng，LI Tan，WU Kui，et al. Killing two birds with one stone：quantization achieves privacy in distributed learning [J]. Digital Signal Processing，2024，146：104353.

[10] 陈世达，刘强，韩亮. 降低分布式训练通信的梯度稀疏压缩方法 [J]. 浙江大学学报(工学版)，2021，55(2)：386-394.
CHEN Shida，LIU Qiang，HAN Liang. Gradient sparsification compression approach to reducing communication in distributed training [J]. Journal of Zhejiang University(Engineering Science)，2021，55(2)：386-394.

[11] 王恩东，闫瑞栋，郭振华，等. 分布式训练系统及其优化算法综述 [J]. 计算机学报，2024，47(1)：1-28.
WANG Endong，YAN Ruidong，GUO Zhenhua，et al. A survey of distributed training system and its optimization algorithms [J]. Chinese Journal of Computers，2024，47(1)：1-28.

[12] STROM N. Scalable distributed DNN training using commodity GPU cloud computing [C]//Proceedings of the Interspeech 2015. Paris，France：International Speech Communication Association，2015：1488-1492.

[13] DRYDEN N，MOON T，JACOBS S A，et al. Communication quantization for data-parallel training of deep neural networks [C]//2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC). Piscataway，NJ，USA：IEEE，2016：1-8.

[14] CHEN C Y，CHOI J，BRAND D，et al. AdaComp：adaptive residual gradient compression for data-parallel distributed training [C]//Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Ed-

ucational Advances in Artificial Intelligence. Palo Alto, CA, USA: AAAI Press, 2018: 2827-2835.

[15] RENGGLI C, ASHKBOOS S, AGHAGOLZADEH M, et al. SparCML: high-performance sparse communication for machine learning [C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New York, USA: Association for Computing Machinery, 2019: 11.

[16] SATTLER F, WIEDEMANN S, MÜLLER K R, et al. Sparse binary compression: towards distributed deep learning with minimal communication [C]//2019 International Joint Conference on Neural Networks (IJCNN). Piscataway, NJ, USA: IEEE, 2019: 1-8.

[17] SHI Shaohuai, WANG Qiang, ZHAO Kaiyong, et al. A distributed synchronous SGD algorithm with global top-k sparsification for low bandwidth networks [C]//2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). Piscataway, NJ, USA: IEEE, 2019: 2238-2247.

[18] LIN Yujun, HAN Song, MAO Huizi, et al. Deep gradient compression: reducing the communication bandwidth for distributed training [EB/OL]. (2020-06-23)[2024-01-01]. https://arxiv.org/abs/1712.01887.

[19] SHI Shaohuai, TANG Zhenheng, WANG Qiang, et al. Layer-wise adaptive gradient sparsification for distributed deep learning with convergence guarantees [M]//Frontiers in Artificial Intelligence and Applications. Amsterdam, Netherlands: IOS Press, 2020: 1467-1474.

[20] SHI Shaohuai, CHU Xiaowen, CHEUNG K C, et al. Understanding top-K sparsification in distributed deep learning [EB/OL]. (2019-11-20) [2024-01-01]. https://arxiv.org/abs/1911.08772.

[21] GUO Bingjun, LIU Yazhi, ZHANG Chunyang. A partition based gradient compression algorithm for distributed training in AIoT [J]. Sensors, 2021, 21(6): 1943.

[22] ABDELMONIEM A M, AHMED A, ALOUINI M S, et al. An efficient statistical-based gradient compression technique for distributed training systems [J]. Proceedings of Machine Learning and Systems, 2021, 3: 297-322.

[23] CHMIEL B, BEN-URI L, SHKOLNIK M, et al. Neural gradients are near-lognormal: improved quantized and sparse training [EB/OL]. (2020-10-12) [2024-01-01]. https://arxiv.org/abs/2006.08173.

[24] SONG Yingjie, AI Yongbao, XIAO Xiong, et al. HCEC: an efficient geo-distributed deep learning training strategy based on wait-free back-propagation [J]. Journal of Systems Architecture, 2024, 148: 103070.

[25] ZHANG Hao, ZHENG Zeyu, XU Shizhen, et al. Poseidon: an efficient communication architecture for distributed deep learning on GPU clusters [C]//Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference. Berkeley, CA, USA: USENIX Association, 2017: 181-193.

[26] JAYARAJAN A, WEI Jinliang, GIBSON G, et al. Priority-based parameter propagation for distributed DNN training [J]. Proceedings of Machine Learning and Systems, 2019, 1: 132-145.

[27] PENG Yanghua, ZHU Yibo, CHEN Yangrui, et al. A generic communication scheduler for distributed DNN training acceleration [C]//Proceedings of the 27th ACM Symposium on Operating Systems Principles. New York, USA: Association for Computing Machinery, 2019: 16-29.

[28] 吴艳霞, 梁楷, 刘颖, 等. 深度学习 FPGA 加速器的进展与趋势 [J]. 计算机学报, 2019, 42(11): 2461-2480.
WU Yanxia, LIANG Kai, LIU Ying, et al. The progress and trends of FPGA-based accelerators in deep learning [J]. Chinese Journal of Computers, 2019, 42(11): 2461-2480.

(编辑 亢列梅)