doi:10.3772/j.issn.1002 - 0470.2026.08.003

混合通信调度的细粒度部分分片数据并行训练方法①

李伟民②*** 孙毓忠③*

(*中国科学院计算技术研究所处理器芯片全国重点实验室 北京 100190)

(**中国科学院大学 北京 100049)

摘 要 数据并行是分布式深度学习训练的主流技术之一,部分分片数据并行在不增加通信量时降低了传统数据并行的图形处理器(graphics processing unit,GPU)显存消耗,提升部分分片数据并行的训练效率成为学术和工业界研究的热点。本文首先揭示GPU之间不均衡的通信负载,交换更新后参数导致的长尾延迟制约了部分分片数据并行训练系统的可扩展性。设计了一个通信高效的新型部分分片数据并行训练系统,采用了一种新的细粒度状态分片方法和新型层次化并行通信方案,平衡了不同网络链路的通信负载的差异。还设计了一种高效两阶段混合通信调度算法,将更新参数传输与反向传播重叠进行。选取了5类典型的深度学习训练负载并在32个GPU上进行了系列实验,与目前最先进的部分分片数据并行训练系统相比,本文深度学习模型的训练时间减少了15.0%到85.0%。

关键词 深度学习;分布式训练;部分分片数据并行;通信调度;梯度裁剪

随着深度学习模型规模和训练数据规模增长,分布式训练成为一种提高模型训练速度的重要手段,数据并行是目前主流的分布式训练技术之一。传统数据并行(distributed data parallel,DDP)[1]中深度学习模型的模型梯度(gradients, G)和优化器状态(optimizer states,OS)在图形处理器(graphics processing unit,GPU)之间重复存储,导致各GPU之间存在大量冗余的显存开销。部分分片数据并行(partially sharded data parallel,PSDP)[2-6]是传统数据并行的变体,将G和OS分片处理,每个GPU维护深度学习模型的部分G和OS,降低了数据并行训练的GPU显存开销。

深度学习模型的参数在PSDP训练时只能在维护其对应状态(G和OS)的GPU上进行更新,其他GPU通过集体通信(collective communication,CC)从该GPU上获取新的模型参数。目前主流的PSDP训练系统[2-6]如DeepSpeed[5]在训练的反向传播(backward propagation,BP)期间基于集体通信算子reduce聚合梯度(第1阶段通信),各GPU完成参数更新后互相交换新的参数(第2阶段通信)。这种通信方案导致了以下问题:首先,reduce算子使目的地GPU的链路的负载远高于其他GPU,导致通信量在链路间分配不均衡。其次,PSDP系统无法利用BP期间空闲的带宽资源交换更新后的参数,深度学习模型任意一层新的参数只能在所有的模型参数都更新结束后才能进行交换,导致参数交换无法与BP重叠进行。

本文设计了一个新的通信高效PSDP训练系统并将其命名为Themis。针对主流PSDP训练系统中GPU间通信负载不均衡,Themis将深度学习模型每一层的G和OS都均匀分布在所有GPU上,平衡了每一次通信中各个GPU的通信负载。针对GPU集群中节点内带宽高、节点间带宽低的层次化结构,Themis采用层次化并行通信方案,减少GPU节点间

① 科技创新2030重点研发课题(2022ZD0119104)资助项目。

② 男,1997年生,博士生;研究方向:分布式深度学习系统;联系人,E-mail:liweimin19z@ict.ac.cn。

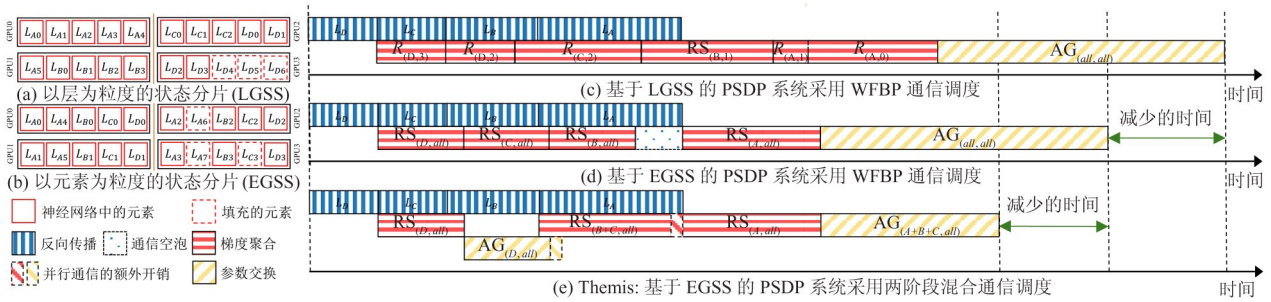
③ 通信作者,E-mail:yuzhongsun@ict.ac.cn。

(收稿日期:2025 - 05 - 07)

的通信量。深度学习模型在 Themis 上训练时，每一层参数在 GPU 间完成梯度聚合和参数更新后可以马上交换新的参数。该过程采用了一种高效两阶段混合通信调度算法，合并或拆分深度学习模型每一层的通信数据，并通过独立通信或并行通信最大化 BP 期间的通信量。Themis 实现了一种简单高效的梯度裁剪方法限制模型梯度的绝对值，稳定模型的收敛过程[7]。在 5 类具有代表性的深度学习训练负载上的实验结果表明，Themis 相比于主流 PSDP 训练系统，将模型训练时间减少 15.0% ~85.0%。

1　部分分片数据并行的基本原理

DDP 将深度学习模型所有参数的状态 G 和 OS 备份在各 GPU 上，PSDP 将 G 和 OS 分布在各 GPU 上，在不增加通信量的情况下减少 GPU 显存的使用量[3,8]。考虑由层 $L_X \in \{L_A,\cdots,L_D\}$ 组成的深度学习模型，每层分别包含 $k \in \{6,4,3,4\}$ 个元素。如图 1(a)所示，当前最先进的 PSDP 训练系统[2-6]基于深度学习模型的层为粒度进行状态分片(layer-grained states sharding，LGSS)，将 G 和 OS 分布在各个 GPU 上。层 L_A 的前 5 个元素的状态由 GPU0 维护，第 6 个元素的对应状态由 GPU1 维护，层 L_B 所有元素的状态都由 GPU1 维护，层 L_C 的状态全部由 GPU2 维护，最后层 L_D 的前 2 个元素的状态由 GPU2 维护，后 2 个元素的状态由 GPU3 维护。如图 1(a)中的 L_{D4}、L_{D5} 和 L_{D6} 所示，为了在所有 GPU 完成参数更新后使用集体通信算子 all-gather 进行参数交换，需要对层 L_D 进行填充，使每一个 GPU 维护的状态总量相等。由于每个 GPU 仅维护一部分状态，在反向传播过程中产生的梯度必须发送到对应的 GPU 进行参数更新。如图 1(c)所示，在 L_D 的反向传播结束后，L_{D2} 和 L_{D3} 的梯度被 reduce[9]到 GPU3(记作 $R_{(D,3)}$)，L_{D0} 和 L_{D1} 的梯度被 reduce 到 GPU2(记作 $R_{(D,2)}$)，该过程以无等待反向传播(wait-free backward propagation，WFBP)[10]的方式进行。在所有 GPU 完成参数更新后，各 GPU 通过 all-gather(记作 $AG_{(all,all)}$)来同步所有新的参数。



(a) 基于 LGSS 的 PSDP 系统中，每个 GPU 维护模型每层的状态量不相等；(b) 基于 EGSS 的 PSDP 系统中，每个 GPU 维护模型每层的状态量相等；(c) 基于 LGSS 的 PSDP 系统通信时，使用 reduce 进行第 1 阶段通信，所有第 1 阶段通信任务完成后通过 all-gather 完成参数交换；(d) 基于 EGSS 的 PSDP 系统通信时，使用更高效的 reduce-scatter 进行第 1 阶段通信，使得 BP 期间有更多的通信空泡；(e) Themis 是基于 EGSS 的 PSDP 系统，通过两阶段混合通信调度将参数交换与 BP 重叠，充分利用 BP 期间的通信空泡，降低通信所需的时间。R，RS 和 AG 分别代表集体通信算子 reduce，reduce-scatter 和 all-gather。R，RS 和 AG 的下标 $(\boldsymbol{X},\mathrm{GPU}_{id})$ 代表通信的内容来自深度学习模型的层 $\boldsymbol{X}$，通信的目的地为第 GPU_{id} 个 GPU，all 指在 reduce-scatter 和 all-gather 通信中所有 GPU 都是目的地 GPU。

图 1　L_{Xi} 表示深度学习模型层 X 的第 i 个元素的对应状态 G 和 OS

2　系统设计

2.1　细粒度部分分片方法

基于 LGSS 的 PSDP 系统由于通信阶段各 GPU 间存在不均衡的带宽开销从而导致整体通信效率低下：深度学习模型每层的梯度产生后，未维护该层状态的 GPU 必须通过 reduce 算子将计算出的梯度发送到维护该层状态的 GPU(目标 GPU)，目标 GPU 接收来自其他所有 GPU 发送的梯度。在该阶段通信中，目标 GPU 的网络链路将承受远高于其他 GPU 的通信压力，成为整个链路中的瓶颈，降低了整体通信效率[11]。基于 LGSS 的 PSDP 系统中，如图 1(c)

中的 $R_{(D,3)}$ 和 $R_{(D,2)}$,完成单一层的梯度传输可能需要启动多次连续的 reduce 算子,引入了额外的通信启动延迟。我们在 GPU 集群中测试了 3 种集体通信算子的性能,结果如图 2 所示。传输相同量的数据,reduce 所需时间分别是 reduce-scatter 和 all-gather 的 4.2 倍和 4.4 倍,reduce 的通信启动延迟分别是 reduce-scatter 和 all-gather 的 10.3 倍和 11.6 倍。

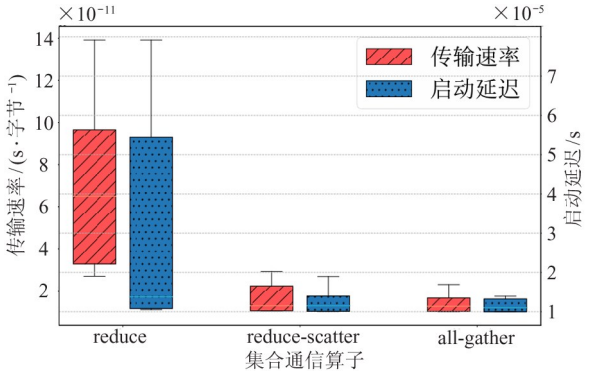


图 2　3 种集合通信算子的传输速率和启动延迟对比

Themis 基于深度学习模型每层的元素为粒度进行状态分片(element-grained states sharding, EGSS),EGSS 将深度学习模型每层的状态均匀分布在所有 GPU 上。如图 1(b)所示,与 LGSS 中每层状态最多仅由 2 个 GPU 维护(图 1(a))不同,EGSS 使每个 GPU 都维护每层模型和优化器状态的等量数据。基于 EGSS,在每层梯度计算完成后,各 GPU 拥有相同量的数据需要发送和接收,Themis 可利用更高性能的 reduce-scatter 算子聚合各 GPU 上的梯度(如图 1(d)中的 $\mathrm{RS}_{(D,all)}$)。

GPU 集群中节点内 GPU 间的互联带宽远远高于节点间的互联带宽[12],为减少节点间的通信开销,我们在 Themis 中设计了一种新型层次化并行通信(hierarchical concurrent communication, HCC)方案。不失一般性地假设 GPU 维护的状态如图 1(b)所示,GPU0 和 GPU1 位于节点 *NodeA*,GPU2 和 GPU3 位于另一个节点 *NodeB*,考虑如图 3 所示的深度学习模型的一个局部。按照模型层的自然排列顺序进行层次化通信,在 GPU0 或 GPU1 从 GPU2 或 GPU3 接收到对应于 $\{L_{A4},L_{A5},L_{B0},L_{B1}\}$(这 4 个元素的状态均由 *NodeA* 中的 GPU 维护)的梯度或更新参数后,由于 L_{A4} 和 L_{B0} 在通信缓冲区中不相邻,因此需要进行 2 次独立的通信才能完成 L_{A4} 和 L_{B0} 对应状态的交换,导致了额外的通信启动延迟。我们提出局部性可感知的元素重排序方法,在通信前确保由同一个 GPU 维护的元素在通信缓冲区中保持相邻。如图 3 所示,在交换更新参数的过程中,在通信之前重新排列元素的顺序,确保分配给同一 GPU 上的元素相邻。然后,在节点内调用一次 all-gather(图 3 中步骤②),每个 GPU 收集总数据量的 $\frac{1}{M\cdot N}$(N 表示节点的数量,M 表示单个节点中的 GPU 数量)。随后,只需向(从)另一个节点上的对应 GPU 发送(接收)(图 3 中步骤③) $\frac{1}{N}$ 的状态量。Themis 在 Pytorch 中不同的通信组(communication group,CG)之间执行集体通信算子,利用每个节点上的多个网卡在 GPU0/GPU1 与 GPU2/GPU3 之间并行地启动节点间通信[13]。

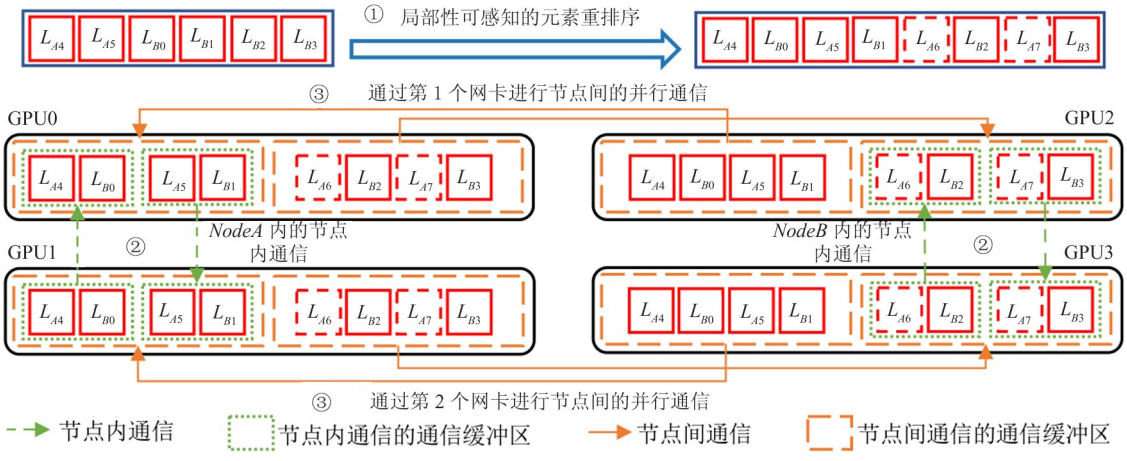


图 3　Themis 使用局部性可感知的元素重排序确保同一个 GPU 维护的元素在通信缓冲区中相邻并采用层次化并行通信降低节点间通信量

2.2　两阶段混合通信调度

针对单阶段通信的调度算法[10,14-16]在 PSDP 系统中无法充分利用 BP 阶段空闲的网络资源，导致严重的尾部延迟：针对数据并行训练的通信调度算法[10,14-16]如 WFBP[10]都是针对 DDP 系统进行设计，DDP 系统在梯度产生后只需进行一个阶段的通信即可完成迭代[1]，PSDP 系统需进行 2 个阶段的通信完成一次迭代[2-6]。如图 1(d)所示，将这类算法应用到 PSDP 系统中只能处理第 1 阶段的通信任务，交换新的参数必须在所有 GPU 都更新完自己维护的参数后才能进行。单阶段通信调度算法无法利用 PSDP 系统在 BP 过程中存在的空闲网络资源(通信空泡)进行第 2 阶段通信，导致至少一半的通信量无法与 BP 过程重叠。本研究将通信时间分解为 3 个部分，分别为：(1)与 BP 重叠的通信，(2)BP 期间的通信空泡，(3)无法与 BP 重叠的通信。本文测量了深度学习模型在不同的 PSDP 系统中训练时的通信时间，结果如图 4 所示。单阶段通信调度导致 DeepSpeed[5]中有 61.2% 的通信无法与 BP 重叠，同时导致使用 MG-WFBP 算法[15]的 Themis 存在 26.8% 的通信空泡。

本研究设计了一种高效两阶段混合通信调度算法，对通信张量进行合并[15]或分割[14]，顺序发起通信任务或并行发起通信任务[17]来最大化 BP 阶段的通信量，其中并行发起通信任务与 HCC 的并行通信不同，指通过 GPU 中不同的 CUDA 流(CUDA stream, CS)并行执行集体通信算子。

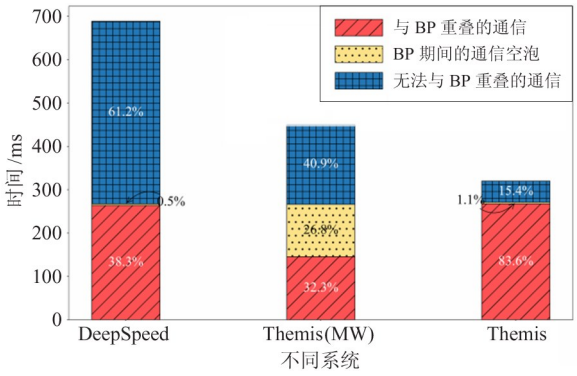


图 4　ResNet 50 训练期间通信时间分解

本文假设有一种针对单阶段通信的调度算法(例如MG-WFBP[15])，组织 Themis 中第 1 阶段的通信任务，并形成了一个包含 K 个第 1 阶段通信任务的调度方案：$\{C_1^0,\cdots,C_1^{K-1}\}$。每个任务 C_1^i 在时间戳 τ_i 开始，传输的数据量为 s^i。Themis 的第 1 阶段通信使用 reduce-scatter 算子，第 2 阶段使用 all-gather 算子。本文用公式 $t(m)=a+b\times m$[9]来建模通信的开销，其中 a 是启动延迟，b 是每字节的传输时间，m 是通信量的大小。对于并行通信的任务(如图 1(e)中的 $\mathrm{AG}_{(D,all)}$ 和 $\mathrm{RS}_{(B+C,all)}$)，本文用一个大于等于 1 的系数 γ[17]调整 b，衡量并行通信相比于非并行通信所增加的通信开销。根据图 2 所示的集体通信算子的性能，我们把 reduce-scatter 和 all-gather 这 2 个操作的系数 a 和 b 当作相等的对待。

为了在计算过程中最大化传输量，Themis 在时间戳 τ_i 时可以作出以下选择。

$D1$：不发起第 2 阶段通信任务(例如图 1(d))。

$D2$：在 C_1^i 和 C_1^{i+1} 之间插入一个第 2 阶段通信任务。

$D3$：与正在进行的任务并行执行一个新的通信任务。

设 π_i 表示第 i 种选择下的通信完成时间。在不失一般性的前提下，假设所有未完成的第 2 阶段任务在第 1 阶段任务全部完成后才开始执行，因此：

$$\pi_1=\tau_i+T_0+t_{2,all} \tag{1}$$

式中：T_0 表示时间戳 τ_i 与任务 C_1^{K-1} 完成时间戳之间的时间间隔，$t_{2,all}$ 表示在同一次通信中传输所有第 2 阶段通信量所需的时间。假设发起一个通信量为 k 的第 2 阶段任务，T_0' 表示重新调度后时间戳 τ_i 与任务 C_1^{K-1} 完成时间戳之间的时间间隔。因此：

$$\pi_2=\tau_i+T'_0+t_{2,all-k} \tag{2}$$

本研究为 $\pi_2\leqslant\pi_1$ 定义了一个条件：

$$Q1:\ T_{\mathrm{bubble}}\geqslant a \tag{3}$$

式中：T_{bubble} 表示通信空泡所占的时间间隔。

当 $Q1$ 不成立时，无法在不延长整体时间的情况下通过 $D2$ 的方式启动第 2 阶段的任务，此时只能通过 $D3$ 的方式利用通信空泡。当 $Q1$ 成立时，启动延迟可以被完全隐藏到 2 个第 1 阶段通信任务之间的通信空泡中，因此本文可以在这个通信空泡中传输大于 0 的通信量。在这种情况下，可以考虑 $D2$ 和 $D3$ 两种选择来最大化通信空泡内可以传输的通信

量。如果选择 $D2$ 的方式来填充气泡,则有:

$$a + b\,m_{D2} = T_{\text{bubble}} \tag{4}$$

式中:m_{D2} 为选择 $D2$ 时能传输的通信量。本文引入一个条件来对 $D3$ 的情况进行分类:

$$Q2:\ s^i < \frac{T_{\text{bubble}}}{(\gamma-1)b} \tag{5}$$

如果 $Q2$ 成立,即使 C_1^i 已完成,气泡也无法完全填满。那么:

$$(\gamma-1)bs^i + b(m_{D3} - s^i) = T_{\text{bubble}} \tag{6}$$

因此,可得到:

$$Q3:\ \frac{a-(\gamma-2)bs^i}{b} \geqslant 0 \tag{7}$$

如果 $Q3$ 成立,则 $m_{D3} - m_{D2} \geqslant 0$。如果 $Q2$ 不成立,则有:

$$(\gamma-1)b \times m_{D3} = T_{\text{bubble}} \tag{8}$$

现在使用一个新的条件:

$$Q4:\ a > \frac{(\gamma-2)T_{\text{bubble}}}{\gamma-1} \tag{9}$$

如果 $Q4$ 成立,则 $m_{D3} - m_{D2} \geqslant 0$。

根据以上分析,在 PSDP 训练系统中,给定一种用于第 1 阶段通信的单阶段通信调度算法,可以通过以下策略在 BP 过程中最大化交换更新后的参数量:(1)如果 BP 过程中不存在通信空泡,则无法在 BP 阶段进行参数交换;(2)如果满足条件 $Q1 \wedge Q2 \wedge \neg Q3$ 或 $Q1 \wedge \neg Q2 \wedge \neg Q4$,则应在第 1 阶段任务之间的通信空泡中插入第 2 阶段任务;(3)在其他情况下,应该与第 1 阶段通信任务并行执行第 2 阶段任务来最大化 BP 阶段的通信量。

本研究得到确定第 3 阶段任务通信类型和通信量的算法 1。其中 $B[i]$ 表示第 i 个第 1 阶段通信任务后的通信空泡大小,R 表示可以进行第 2 阶段通信的参数量。算法 1 首先(第 1 行)初始化最后一个第 1 阶段任务之后的通信空泡和更新参数的数量;然后计算第 1 阶段任务之间的通信空泡(第 2 ~ 3 行);第 4 ~ 15 行确定通信类型和通信的数据量。

算法 1 参数交换的通信方式与通信量

输入:第 1 阶段通信任务的通信量 s^i、任务启动时间戳 $\tau[i]$,衡量集体通信算子开销的系数 a,b,γ。

输出:参数交换的通信方式 $D[i]$,参数交换量 $V[i]$。

1. $B[K-1] = \text{MAX}, R = 0$
2. for $i = 0 \rightarrow K-2$ then:
3. $B[i] = \tau[i+1] - \tau[i] - a - b \times s^i$
4. for $i = 0 \rightarrow K-1$ then:
5. $R \leftarrow R + s^i$
6. if $B[i] \leqslant 0$ then:
7. $V[i] = 0$, $D[i] = D1$
8. else:
9. $Q1$ is $B[i] \geqslant a$, $Q2$ is $s^i < \frac{T_{\text{bubble}}}{(\gamma-1)b}$, $Q3$ is $\frac{a-(\gamma-2)bs^i}{b} \geqslant 0$, $Q4$ is $a > \frac{(\gamma-2)T_{\text{bubble}}}{\gamma-1}$
10. if $Q1 \wedge Q2 \wedge \neg Q3$ or $Q1 \wedge \neg Q2 \wedge \neg Q4$ then:
11. $V[i] = \min\left(\frac{B[i]-a}{b}, R\right)$, $D[i] = D2$
12. else if $Q2$ then:
13. $V[i] = \min\left(\frac{B[i]-(\gamma-1)bs^i}{b} + s^i, R\right)$, $D[i] = D3$
14. else:
15. $V[i] = \min\left(\frac{B[i]}{(\gamma-1)b}, R\right)$, $D[i] = D3$
16. $R \leftarrow R - V[i]$
17. return $V[i]$, $D[i]$

2.3 延迟梯度裁剪

Themis 的混合通信过程中包含参数更新,模型的部分参数对应的梯度聚合完成后进行这部分参数的更新,然后就可以交换新的参数。在自然语言模型(如双向编码器表示模型(bidirectional encoder representation from Transformers, BERT))训练过程中,往往使用梯度裁剪[18]来约束梯度绝对值的大小从而获得更稳定的收敛效果。梯度裁剪一般需要计算整个模型的梯度范数[18],在 Themis 中由于部分梯度聚合完成就会进行参数更新,此时无法计算整个模型的梯度范数。本文设计了一种延迟梯度裁剪方法(delayed gradient clipping, DGC),如图 5 所示,前 $n-1$ 次迭代不进行梯度裁剪,以避免在模型远离最优解时引入额外的范数计算开销。第 n 次迭代时,在反向传播结束后计算全部参数的梯度范数但不使用梯度裁剪。在第 $n+1$ 次迭代时,使用第 n 步获得的梯度范数来裁剪梯度。从此时起,模型在第

$p+1$ 步更新参数时，使用基于第 p 步完整梯度计算出的梯度范数进行梯度裁剪。

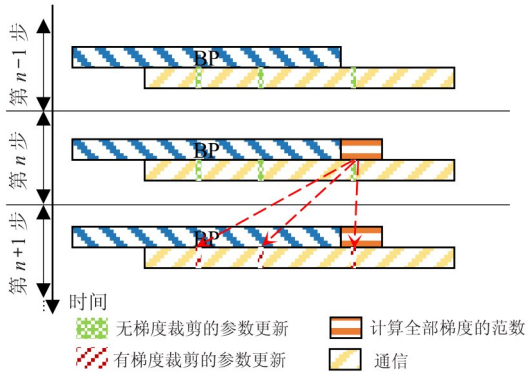


图 5　Themis 使用延迟梯度裁剪的训练过程

3　实验评估

3.1　实验环境与负载

Themis 系统基于 Pytorch 实现，实验工作基于一个包含 8 个阿里云 g1.8xlarge 实例的 GPU 集群。每个实例配备 4 个 16 GB 显存的 Nvidia Tesla V100 GPU，GPU 之间通过 NVlink 连接，实例间的带宽为 10 吉比特每秒（Gbit · s^{-1}）。实验的测试负载包含 5 个具有代表性的深度学习模型训练任务，包括在 CIFAR10 数据集上训练 ResNet 50 和 Vision Transformer（ViT）[19]，在 Wikipedia 数据集上训练 BERT，在 SQuAD 数据集上训练 XLNet[20]，以及在 Naruto-BLIP-captions[21] 数据集上训练 Stable-Diffusion-1.5[22]。

3.2　对比基线

本文将 Themis 与 DDP 训练系统 PyTorch-DDP[1] 和 PSDP 训练系统 DeepSpeed[5] 进行比较。本文还引入了额外的对比基线 Themis（MW），该基线中使用 MG-WFBP[15] 算法加速 Themis 的第 1 阶段通信，与该基线的对比可以展示两阶段混合通信调度的有效性。

3.3　Themis 高效性分析

本研究比较了不同系统训练深度学习模型的效率来验证 Themis 的高效性，结果如图 6 所示。与 DeepSpeed 相比，Themis 能将训练时间减少 19.0% 到 85.0%；相较于 PyTorch-DDP，可以减少 15.0% ~ 39.0% 的训练时间；Pytorch-DDP 由于 GPU 内存不足无法训练 Diffusion 模型。Themis 和 Themis（MW）的对比结果显示两阶段混合通信调度能使训练时间减少 9.0% ~66.0%。

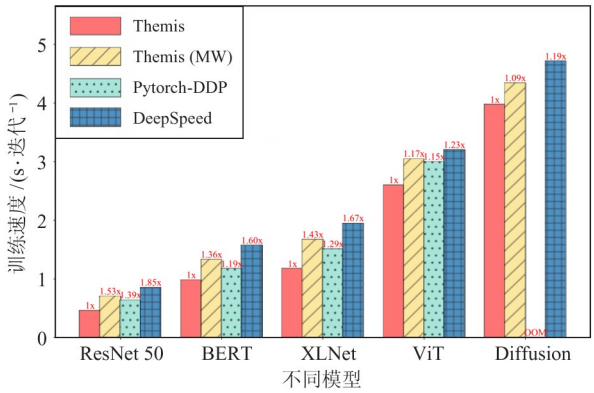


图 6　不同模型一次迭代所需时间对比

将 ResNet 50 的训练规模从 2 个节点逐渐扩展到 8 个节点来验证 Themis 的可拓展性，结果如图 7 所示。节点数量从 2 个扩展到 8 个时，Themis 的训练吞吐量增加了 3.2 倍，不使用 HCC 的 Themis（Themis w/o HCC）增长了 3.1 倍，DeepSpeed 和 Pytorch-DDP 分别只增加了 2.5 倍和 2.9 倍。

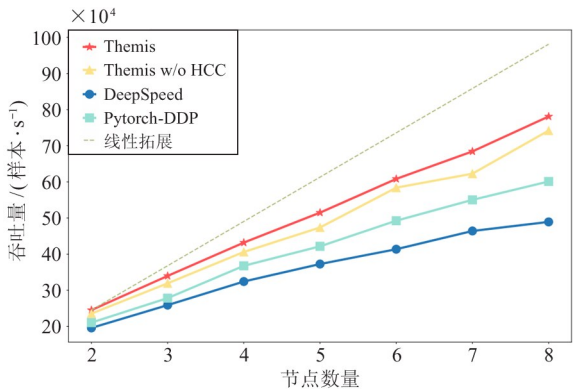


图 7　ResNet 50 不同节点数量的训练吞吐量变化

3.4　Themis 鲁棒性分析

本文比较了不同网络带宽下 ResNet 50 的训练吞吐量，结果如图 8 所示。在不同带宽条件下，Themis 始终优于 PyTorch-DDP 和 DeepSpeed，在较低带宽场景如 1 Gbit · s^{-1} 下，其吞吐量相比于 PyTorch-DDP 和 DeepSpeed 分别提高了 35.0% 和 74.0%，在较高带宽场景如 8 Gbit · s^{-1} 下，其吞吐量分别提高了 30.0% 和 65.0%。

本研究比较了 BERT 和 ResNet 50，使用不同的批量大小的训练速度，结果如表 1 所示。Themis 的

训练速度比 DeepSpeed 高出 39.0% ~51.0%。使用较大的批量大小如 128 时(较大批量会延长计算时间,弱化通信优化方法的效果),Themis 的训练速度比 DeepSpeed 快 46.0%。

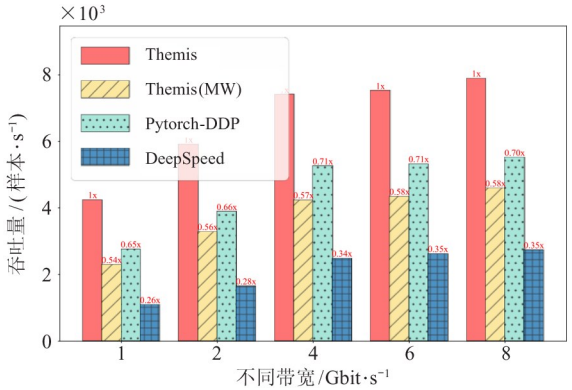


图 8　ResNet 50 在不同带宽下的训练吞吐量对比

表 1　BERT 和 ResNet 50 不同批量大小训练的每次迭代所用时间　　s

模型(批量大小)和方法	DeepSpeed	Themis	提升百分比
BERT (8)	1.28	0.67	47.7%
BERT (16)	1.31	0.70	46.6%
BERT (32)	1.57	0.96	38.9%
ResNet 50 (32)	0.56	0.27	51.6%
ResNet 50 (64)	0.69	0.35	49.6%
ResNet 50 (128)	0.85	0.46	46.0%

3.5　Themis 与梯度压缩比较

本节还引入一种典型的梯度压缩技术 TopK[23] 作为对比基线,结果如表 2 所示。梯度压缩技术可以降低梯度聚合阶段的通信量,但会引入额外的梯度压缩开销[23],梯度压缩技术在 PSDP 系统中仅能减少第 1 阶段的通信量。因此 Themis 的训练速度分别比采用 TopK 的 Pytorch-DDP(DDP + TopK)和采用 TopK 的 DeepSpeed(DS + TopK)快 6.41 倍和 8.08 倍。实验中 TopK 使 ResNet 50 和 ViT 的测试准确率分别下降了 1.6% 和 2.1%,BERT 和 XLNet 的损失值分别增加了 0.75 和 0.11,Themis 对模型的测试准确率和训练损失值均没有影响。

表 2　模型单次迭代时间对比　　s

方法和模型	ResNet 50	BERT	XLNet	ViT
Themis	0.46	0.98	1.17	2.62
DDP + TopK	3.74	5.81	7.40	13.77
DS + TopK	4.40	7.96	9.98	16.03

本文在极低带宽环境(100 兆比特每秒(Mbit · s^{-1})和 10 Mbit · s^{-1})下测试了 ResNet 50 的训练吞吐量,结果如表 3 所示。在 100 Mbit · s^{-1} 的测试中,Themis 的吞吐量比其他方法提高了 1.32 ~5.49 倍,在 10 Mbit · s^{-1} 的测试中,吞吐量提升幅度为 1.29 ~9.33 倍。

表 3　ResNet 50 训练吞吐量　　样本 · s^{-1}

方法和带宽	100 Mbit · s^{-1}	10 Mbit · s^{-1}
Themis	1 920	840
Themis(MW)	1 270	640
Pytorch-DDP	1040	570
DeepSpeed	350	90
DDP + TopK	1 450	650
DS + TopK	730	220

3.6　Themis 训练模型损失曲线分析

图 9 展示了在 Wikipedia 数据集上对 BERT 进行预训练期间使用不同梯度裁剪方法的损失曲线。在训练早期没使用梯度裁剪的 BERT 和使用常规梯度裁剪[18]的 BERT 的训练曲线几乎完全重叠,这是因为在训练早期模型离最优状态还很远,使得梯度裁剪几乎没有带来性能提升。在整个训练过程中,采用延迟梯度裁剪的 BERT 和常规梯度裁剪的 BERT 的曲线几乎完全重叠。在训练后期,使用常规梯度裁剪的 BERT 和采用延迟梯度裁剪的 BERT 的训练损失值始终低于不使用梯度裁剪的 BERT,均有超过 0.050 的优势。

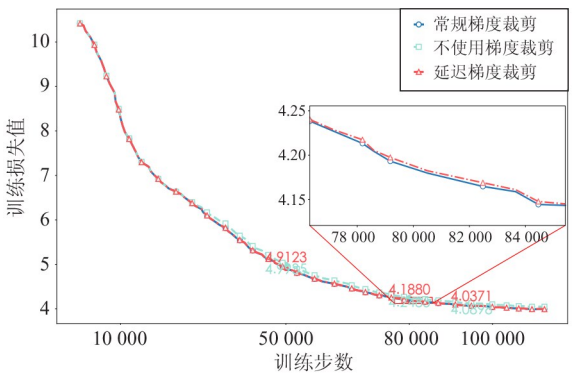


图 9　BERT 预训练损失曲线对比

图 10 展示了在 naruto-blip-captions 数据集上对 Stable-Diffusion-1.5 进行微调期间使用不同梯度裁剪方法的损失曲线。使用延迟梯度裁剪与常规梯度裁剪的训练损失曲线几乎完全重叠，与不使用梯度裁剪相比，常规梯度裁剪以及延迟梯度裁剪在训练后期能够将损失值降低约 0.004。

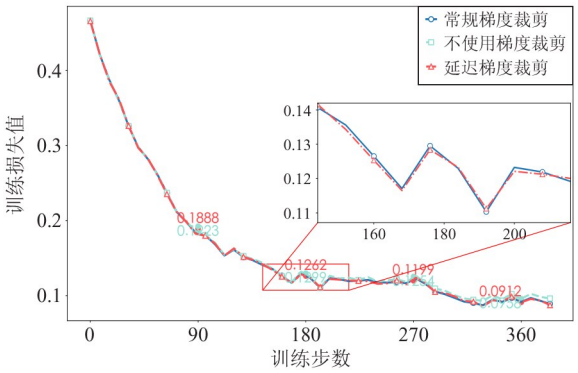


图 10　Stable-Diffusion-1.5 微调损失曲线对比

3.7　Themis 开销分析

Themis 在训练前需先测量算法 1 中所需参数 a,b,γ。本实验中，测量所需参数使用几次训练迭代即可完成，深度学习模型完整训练一次需数千至数万次训练迭代，因此该开销可以忽略不计。如图 1(a)和图 1(b)所示，EGSS 相比于 LGSS 可能会填充更多张量，导致额外显存开销。本实验测量了使用不同批量大小进行 BERT 训练时的剩余 GPU 显存，结果如图 11 所示。Themis 剩余的 GPU 显存与 DeepSpeed 基本相同且相比于 Pytorch-DDP 多约 2 000 MB，因此 EGSS 引入的额外显存开销可以忽略不计。

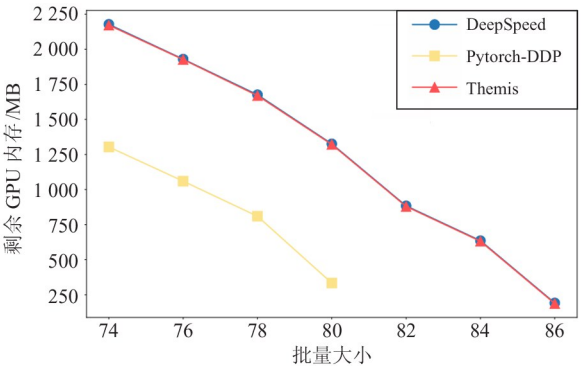


图 11　BERT 训练 GPU 剩余显存对比

4　结　论

本文提出一种新的通信高效部分分片数据并行训练系统 Themis，设计了一种以深度学习模型层元素为粒度的细粒度部分状态分片方法，使分布式训练各 GPU 维护的模型每层状态量相等，均衡了各 GPU 在通信中的通信负载，并提出了新型层次化并行通信方案，降低 GPU 节点间的通信量。为降低通信的尾部延迟，设计了一种同时进行梯度聚合与参数交换的高效两阶段混合通信调度算法，通过张量合并、切分、并行通信最大化反向传播期间的通信量；并提出一种延迟梯度裁剪策略，利用上次迭代的梯度范数对当前迭代的梯度进行裁剪，提升模型收敛稳定性。基于 5 种有代表性的深度学习模型训练负载对本文方法的评估结果表明，Themis 降低了部分分片数据并行训练的通信延迟，提升了模型训练的吞吐量，在不同节点数量、不同带宽、不同批量等场景下均有优异表现，因此具有良好的实用价值。后续研究将重点探讨如何将方法拓展到大语言模型训练，提高方法的适用范围。

参考文献

[1] Li S, Zhao Y, Varma R, et al. PyTorch distributed: experiences on accelerating data parallel training[J]. Proceedings of the VLDB Endowment, 2020, 13(12): 3005-3018.

[2] Ren J, Rajbhandari S, Aminabadi R Y, et al. ZeRO-Offload: democratizing billion-scale model training[C]// USENIX Annual Technical Conference. Online: USENIX Association, 2021: 551-564.

[3] Rasley J, Rajbhandari S, Ruwase O, et al. DeepSpeed: system optimizations enable training deep learning models with over 100 billion parameters[C]// Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Online: ACM, 2020: 3505-3506.

[4] Zhang J, He T, Sra S, et al. Why gradient clipping accelerates training: a theoretical justification for adaptivity[EB/OL]. (2020-02-10)[2025-07-09]. https://arxiv.org/pdf/1905.11881.

[5] Zhao Y, Gu A, Varma R, et al. PyTorch FSDP: experiences on scaling fully sharded data parallel[J]. Proceedings of the VLDB Endowment, 2023, 16(12): 3848-3860.

[6] Rajbhandari S, Rasley J, Ruwase O, et al. ZeRO: memory optimizations toward training trillion parameter models[C]// International Conference for High Performance Computing, Networking, Storage and Analysis. Atlanta, USA: IEEE, 2020: 1-16.

[7] Fang J, Zhu Z, Li S, et al. Parallel training of pre-trained models via chunk-based dynamic memory manage-

ment[J]. IEEE Transactions on Parallel and Distributed Systems, 2023,34(1):304 - 315.
[8] Rajbhandari S,Ruwase O,Rasley J,et al. ZeRO-infinity: breaking the GPU memory wall for extreme scale deep learning[C]//International Conference for High Performance Computing, Networking, Storage and Analysis. St. Louis, USA: ACM, 2021:1 - 14.
[9] Sarvotham S, Riedi R, Baraniuk R. Connection-level analysis and modeling of network traffic[C]//ACM SIGCOMM Workshop on Internet Measurement. San Francisco, USA: ACM, 2001:99 - 103.
[10] Zhang H, Zheng Z, Xu S, et al. Poseidon: an efficient communication architecture for distributed deep learning on GPU clusters[C]//USENIX Annual Technical Conference. Boston, USA: USENIX Association,2017:181 - 193.
[11] Li M, Andersen D G, Park J W, et al. Scaling distributed machine learning with the parameter server[C]// USENIX Symposium on Operating Systems Design and Implementation. Broomfield, USA: USENIX Association, 2014:583 - 598.
[12] Chen C, Li X, Zhu Q, et al. Centauri: enabling efficient scheduling for communication-computation overlap in large model training via communication partitioning[C]// Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. La Jolla, USA: ACM, 2024:178 - 191.
[13] Narayanan D, Shoeybi M, Casper J, et al. Efficient large-scale language model training on GPU clusters using Megatron-LM[C]// International Conference for High Performance Computing, Networking, Storage and Analysis. St. Louis, USA: ACM, 2021:1 - 15.
[14] Shi S, Chu X, Li B. MG-WFBP: efficient data communication for distributed synchronous SGD algorithms[C]// IEEE Conference on Computer Communications. Paris, France: IEEE, 2019:172 - 180.
[15] Peng Y,Zhu Y,Chen Y,et al. A generic communication scheduler for distributed DNN training acceleration[C]// Proceedings of the 27th ACM Symposium on Operating Systems Principles. Huntsville, Canada: ACM, 2019: 16 - 29.
[16] Zhang Z, Qi Q, Shang R, et al. Prophet: speeding up distributed DNN training with predictable communication scheduling[C]// Proceedings of the 50th International Conference on Parallel Processing. Lemont, USA: ACM, 2021:1 - 11.
[17] Shi S,Chu X,Li B. Exploiting simultaneous communications to accelerate data parallel distributed deep learning[C]// IEEE Conference on Computer Communications. Vancouver, Canada: IEEE, 2021:1 - 10.
[18] Levy K Y. The power of normalization: faster evasion of saddle points[EB/OL]. (2016 - 11 - 15)[2025 - 07 - 09]. https://arxiv.org/abs/1611.04831.
[19] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[EB/OL]. (2021 - 06 - 03)[2025 - 07 - 09]. https://arxiv.org/pdf/2010.11929.
[20] Yang Z, Dai Z, Yang Y, et al. XLNet: generalized autoregressive pretraining for language understanding[EB/OL]. (2020 - 01 - 02)[2025 - 07 - 09]. https://arxiv.org/pdf/1906.08237.
[21] Cervenka E. Naruto BLIP captions[EB/OL]. (2022 - 12 - 17)[2025 - 07 - 10]. https://huggingface.co/datasets/lambdalabs/naruto-blip-captions/.
[22] Rombach R, Blattmann A, Lorenz D, et al. High-resolution image synthesis with latent diffusion models[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. New Orleans, USA: IEEE, 2022:10674 - 10685.
[23] Aji A F, Heafield K. Sparse communication for distributed gradient descent[C]// Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark: ACL, 2017:440 - 445.

Hybrid communication scheduling based fine-grained partially sharded data parallel training

Li Weimin[*,**], Sun Yuzhong[*]
([*]State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
([**]University of Chinese Academy of Sciences, Beijing 100049)

Abstract

Data parallelism is a mainstream approach for distributed deep learning training. Partial sharded data parallelism reduces graphics processing unit (GPU) memory consumption without increasing communication volume. Improving its training efficiency has become a key research focus in academia and industry. This paper first reveals that the unbalanced communication overhead between GPUs and the long-tail latency caused by exchanging updated parameters limit the scalability of the partial shardeddata parallel training system. We propose a communication-efficient partial sharded data parallel training system, which employs a novel fine-grained state sharding method and a hierarchical communication scheme to balance communication loads across network links. We further design an efficient two-stage hybrid communication scheduling algorithm to overlap parameter synchronization with backpropagation by utilizing idle network resources. Experimental evaluation on five representative deep learning workloads across 32 GPUs shows that our approach reduces training time by 15.0% to 85.0% compared to the state-of-the-art partial sharded data parallel systems.

Key words: deep learning, distributed training, partially sharded data parallel, communication scheduling, gradient clipping