DOI:10.7544/issn1000-1239.2021.20190881
58(1):98-115,2021

分布式深度学习训练网络综述

朱泓睿[1,2] 元国军[1] 姚成吉[3] 谭光明[1] 王 展[1] 卢忠哲[1,2,3] 张晓扬[1,2,3] 安学军[1]

[1](中国科学院计算技术研究所 北京 100190)

[2](中国科学院大学 北京 100049)

[3](北京旷视科技有限公司 北京 100080)

(zhuhongrui@ncic.ac.cn)

Survey on Network of Distributed Deep Learning Training

Zhu Hongrui[1,2], Yuan Guojun[1], Yao Chengji[3], Tan Guangming[1], Wang Zhan[1], Hu Zhongzhe[1,2,3], Zhang Xiaoyang[1,2,3], and An Xuejun[1]

[1](*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

[2](*University of Chinese Academy of Sciences, Beijing* 100049)

[3](*Megvii Inc., Beijing* 100080)

Abstract In recent years, deep learning has achieved better results than traditional algorithms in many fields such as image, speech, and natural language processing. People are increasingly demanding training speed and data processing capabilities for deep learning. However, the calculating ability of a single server has a limit and cannot achieve human demands. Distributed deep learning training has become the most effective method to expand deep learning training computing ability. At present, distributed deep learning faces a training bottleneck due to communication problems in the network during the training process which leads the communication network to be the most influential factor. There are currently many network performance optimization researches for distributed deep learning. In this paper, the main performance bottlenecks and optimization schemes are firstly demonstrated. Then the current state-of-art ultra-large-scale distributed training architecture and methods for optimization performance are specifically analyzed. Finally, a comparative summary of each performance optimization scheme and the difficulties still existing in distributed deep learning training are given, and the future research directions are pointed out as well.

Key words distributed calculating; deep learning; communication network; performance optimization; collective communication; cluster network

摘 要 近年来深度学习在图像、语音、自然语言处理等诸多领域得到广泛应用,但随着人们对深度学习的训练速度和数据处理能力的需求不断提升,传统的基于单机的训练过程愈发难以满足要求,分布式的深度学习训练方法成为持续提升算力的有效途径.其中训练过程中节点间网络的通信性能至关重要,

收稿日期:2019-12-18;**修回日期**:2020-05-08

基金项目:中国科学院战略性先导科技专项(B类)(XDB24050200);国家自然科学基金面上项目(61972380,61702484);中国科学院计算技术研究所创新课题(20166060)

This work was supported by the CAS Strategic Priority Program(B) (XDB24050200), the General Program of the National Natural Science Foundation of China (61972380, 61702484), and the Innovation Fund from the Institute of Computing Technology, Chinese Academy of Sciences (20166060).

通信作者:元国军(yuanguojun@ncic.ac.cn)

直接影响训练性能.分析了分布式深度学习中的性能瓶颈,在此基础上对目前常用的网络性能优化方案进行综述,详细阐述了目前最新的超大规模分布式训练的体系结构、优化方法、训练环境和最有效的优化方法,最后对分布式训练仍然存在的困难进行了总结,对其未来研究方向进行了展望.

关键词 分布式计算;深度学习;通信网络;性能优化;集合通信;集群网络

中图法分类号 TP393

随着近年来深度学习算法在各类任务中取得比传统机器学习算法更好的效果,深度学习已广泛应用于图像识别[1]、音频识别[2]和自然语言处理[3]等领域.深度学习及相关交叉领域[4]的研究热度逐年提升,并在业界取得不少实际成果,应用范围也越来越广.

为了在实际应用中取得更好的训练效果,深度学习训练的数据集越来越大.标准训练物体识别的ImageNet数据集包含约128万张图片,总大小超过150GB;面向其他采用高分辨率图像或者视频专业领域的数据集甚至会高达TB乃至PB级别.另一方面神经网络结构也越来越复杂,网络层数和参数的数量不断增加,从以前的5~8层的神经网络增加到现在的上百层,甚至超过千层的神经网络也被提出并使用.

数据集规模和网络层数的增加使得深度学习的训练过程需要耗费大量的存储和计算资源.由于单台机器的算力有限,故而导致整体训练时间过长.举例来说,目前使用一块NVIDIA P100 GPU以标准数据集ImageNet[5]来训练神经网络ResNet-50[6],完成90个Epoch完整周期,就需要花费14天的时间[7].这样长耗时的训练给神经网络训练调试和研究带来了诸多不便.

为了减少深度学习训练时间,近年来以分布式计算为主的加速方法被越来越多地应用在深度学习领域.分布式的深度学习采用多台GPU服务器,通过构建高性能通信网络形成分布式深度学习计算的模式,打破了原有的算力限制,使得计算规模扩展成为可能.

现在分布式深度学习已经取得了初步进展,常见的深度学习框架[8-12]都已经开始支持分布式训练任务,并在一些时间敏感和超大计算量的应用中取得明显加速效果.但是由于分布式节点数目过多、训练参数复杂、网络环境的复杂等因素,分布式深度学习的应用推广仍然面临诸多的挑战,其中最主要的就是分布式计算下的性能问题.

本文调研了大量相关领域会议、期刊等文章,对常用的网络性能优化方案进行综述,本文的贡献主要有3个方面:

1) 分析了目前分布式深度学习所面临的问题与挑战.

2) 总结和对比了目前针对分布式深度学习的10种性能优化方案.

3) 对未来分布式深度学习训练的研究内容和方向进行了总结和展望.

1 分布式训练中的问题与挑战

分布式在深度学习训练时主要存在6方面问题:

1) 节点数增多导致的性能瓶颈

分布式深度学习在参数更新时需要对各节点的参数进行同步,这个过程涉及到通信过程.随着分布式训练中节点数目的增多,同步通信过程效率得不到保证,使得整个系统容易出现性能瓶颈.同步通信时需要保证高带宽和低延迟,这样通信时间降低,就可以通过计算和通信并行,不会拖慢整个训练过程.高带宽和低延迟是保证分布式训练加速比的基本需求.

2) 网络环境复杂、多变导致的不稳定

不同分布式集群系统的拓扑、带宽和延迟等参数各不相同,特别是对于多租户的云计算环境,网络环境复杂,且存在不小的波动.因此同一个网络性能指标在不同的集群、不同的节点甚至不同的时间点测试时都可能存在不小的差异.网络环境的复杂性一方面可能造成网络性能的降低,另一方面也增大了实验测试的难度.

3) 通信参数多变

神经网络的参数大小决定了分布式训练通信量的大小,神经网络层数决定了通信的数量,神经网络训练的速度决定了通信的频率.因此,对于不同的神经网络,通信特征差异很大,没有完美的通信方式可以适用于所有的场景.

4) 通信数据量极大

深度学习需要海量的数据作为训练和测试数据,

其数据量可能高达 TB 甚至 PB 级别.对于分布式环境来说,一方面单节点存储能力可能受到限制,另一方面多节点间数据搬运或同步至其他节点也会耗费大量时间,给分布式训练带来极大挑战.

5) 超大规模训练的应用与数据难以获取

对于传统的以 CPU 为计算核心的机群系统,深度学习应用只在特定的如分类、回归等任务中具有明显的优势,为了使得深度学习在超大规模的集群环境中使用,必须要有合适的应用场景以及足够丰富的数据源.目前没有足够的公开数据与 Benchmark 以支持更广泛领域的研究,这也为开展相关研究带来不小的挑战.

6) 训练批大小(batch size)对训练精度的影响

在大规模分布式系统中进行标准神经网络与数据集训练时,则经常会遇到由于节点数上升、batch size 上升导致的训练结果不理想的情况[13-17].原因是为了保证所有计算资源的利用率,系统会增加每一轮训练(iteration, Iter)使用的训练集数量(batch size),而当训练参数 batch size 过大时,随机梯度下降(stochastic gradient descent, SGD)算法会逐渐趋近于梯度下降(gradient descent, GD)算法(即每轮训练使用全部数据集),这会造成训练参数容易陷入局部最优值的现象,最终造成训练错误率(error rate)达不到标准值.

综上所述,尽管在不少场景下通过分布式训练可以获得更快的训练速度,解决更复杂、数据量更大的深度学习问题,但分布式深度学习发展中还是存在种种问题与挑战,归纳起来主要有 3 点:性能瓶颈、应用瓶颈与理论瓶颈.

本文将详细分析分布式深度学习中的性能问题,重点围绕网络通信方面引起的性能瓶颈.

2 研究现状

2.1 训练过程及通信特征

本节介绍目前主流的分布式深度学习训练过程及其通信行为,这将方便我们理解后续的优化策略.

神经网络是一种模仿生物大脑结构的多层神经元互连结构[18].训练过程主要是通过梯度下降[19]的原理,不断调整神经网络的各项参数,使得最终参数对于训练集数据来说尽可能接近全局最优值.在此过程中训练数据首先以前向传播的方式经过神经网络,并得到预测结果,然后比较预测结果和实际结果,再进行反向传播,从后向前依次调整神经网络参数,使得神经网络对于该数据训练结果预测越来越准确.

传统的梯度下降(GD)算法[20]一次使用所有的训练数据来进行神经网络参数训练,由于每次训练数据都是固定不变的,会使得梯度下降方向具有确定性,无法使训练过程跳出局部最优值,从而远离全局最优值.目前神经网络训练主要采用随机梯度下降[21]方法,每次从训练集中随机选取小批量(*mini-batch*)的训练数据集,增加了梯度下降过程中的随机性,从而尽可能避免了陷入局部最优值,增加了最终训练的准确性.

分布式深度学习的并行策略主要分为 2 种:数据并行[22]和模型并行[23].数据并行实际上是将 *mini-batch* 进一步进行分割,将切割后的各部分数据分配到不同的计算节点上;而当神经网络规模过大,超过了单台机器的承载能力时,通常采用模型并行策略,对神经网络进行分割,并分配到不同的计算节点上.目前超大型的神经网络并不普及,因此分布式深度学习训练主要是以数据并行为主.

对于数据并行的分布式深度学习训练来说,其具体训练流程为[8]:

1) 每个节点分别从硬盘或网络上读取总共 *mini-batch* 大小的数据并拷贝至内存;

2) 从 CPU 内存拷贝数据至 GPU 内存;

3) 加载 GPU kernel 并由前向后逐层计算(前向传播过程);

4) 计算损失函数(loss)并进行反向传播,逐层计算梯度值;

5) 同步各节点梯度值(发送自身各层梯度,并接收其他节点的各层梯度);

6) 根据同步后的梯度值更新神经网络参数.

上述 6 个步骤完成了一次神经网络训练过程(即一个迭代(iteration, Iter)).在实际训练中,需要完成多次训练,以达到最终神经网络参数训练的目的.以神经网络 ResNet-50 训练 ImageNet 为例,一次完整训练需要 90 时期(epoch),即每张图片需要被训练使用 90 次.假设训练中我们设置 *mini-batch* = 256,由于 ImageNet 共计约 128 万张图片,每一个 Epoch 需要训练约 1280 000/256 = 5 000 Iter,即 90 个 Epoch 共计 5 000 × 90 = 450 000 次.上述的训练过程共需重复约 450 000 次,如果采用 4 台机器做分布式训练,则需要 450 000/4 = 112 500 次.

上述训练流程的 6 个步骤中涉及到网络通信的部分主要是步骤 1,2,5.其中步骤 1 中如果使用本地

磁盘供应数据,则不涉及网络通信过程.步骤 2 涉及到了服务器间的通信,其需要将数据通过 PCI-e 传输到 GPU 中.步骤 5 中的通信数量和大小主要取决于神经网络的参数大小和网络层数等.在通常情况下,一个 Iter 中每个节点需要传输和接收的数据大小都等于神经网络参数的总大小,而需要传输的次数与神经网络层数相关.因此对于每一层传输的数据大小是不同的,频率间隔也与计算速度有关.对于常用于图像识别处理的卷积神经网络(convolutional neural network, CNN)[24],卷积层的参数总数比全连接层的参数要小许多,因此在反向传播的过程中,各神经网络层的通信量会呈现先大后小的不均衡现象.

不同的神经网络具有不同的层数,不同层的参数数量也不尽相同.需要注意的是,对于参数越多的神经网络,通信占比也会越大,从而更容易造成分布式训练的瓶颈.

此外,步骤 5 中也存在对机内各 GPU 之间数据进行同步操作,因此也存在设备间的通信.

2.2 关于网络通信的优化

本节介绍目前针对分布式深度学习的主流性能优化方法,详细分析优化加速原理,对比不同优化方案的利弊.

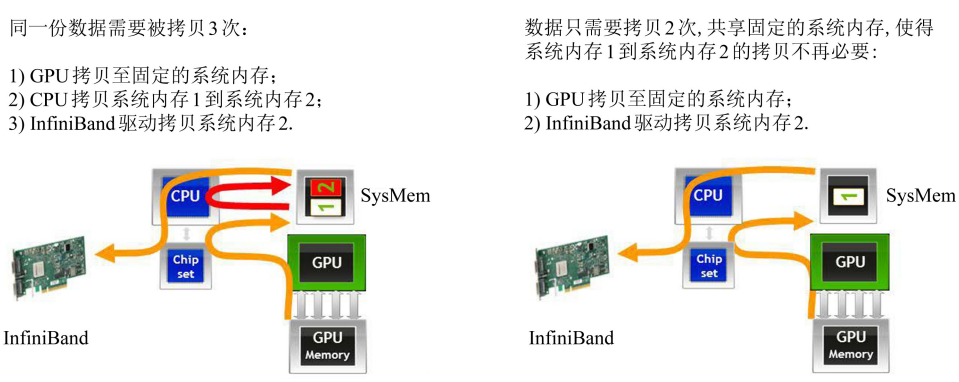
2.2.1 改善网络硬件资源

2.1 节所述分布式深度学习训练中的通信特征对于带宽、延迟具有极高的要求.因此通过改善硬件基础设施来提升网络带宽、降低延迟,是最直接有效的改进通信性能的方法.

对于机间通信的互连网络,可以采用 IB(InfiniBand)[25],OPA(Intel Omni-Path architecture)[26]等方案来实现超高的带宽和低延迟.它们通过专用的网卡、交换机和专用的协议来实现远程直接数据存取(remote direct memory access, RDMA)技术,支持微秒级延迟的点对点通信.也可以采用 TCP/IP 协议通过 RoCE(RDMA over converged Ethernet)[27]来支持 RDMA 操作,从而降低通信延迟.

对于机内通信的互连网络,通常情况下是使用 PCI-e 总线进行数据传输.常见的 GPU 插槽为 PCI-e 3.0×16,速率约为 16 GBps.

为了提升 GPU 之间的通信能力,英伟达推出了 GPU Direct 技术,其发展主要分为 4 个阶段[28].2010 年提出 SHM(GPU direct shared memory)技术,通过共享的固定主机内存提供了与第三方 PCI Express 设备驱动程序加速通信的支持,如图 1 所示;2011 年提出 P2P(GPU direct peer-to-peer)技术,允许使用高速 DMA 传输直接在 2 个 GPU 的内存(即显存)之间加载和存储数据,如图 2 所示;2013 年提出了 GDR(GPU direct RDMA)[29]技术,使网络设备可以完全绕过 CPU 主机内存直接访问 GPU 内存,如图 3 所示;2019 年提出了 GPU Direct Storage 技术,支持在 GPU 内存和存储设备(例如 NVMe 或 NVMe-oF)之间直接传输数据.



(a) 不使用GPU Direct (b) 使用GPU Direct

Fig. 1 GPU direct shared memory proposed by NVIDIA in 2010

图 1 英伟达 2010 年提出的 SHM 技术

对于 GPU Direct P2P 技术,多个 GPU 通过 PCI-e 连接到 CPU,常见的 PCI-e 3.0×16 总线的双向带宽不超过 32 GBps,随着训练数据的不断增长,PCI-e 带宽很难满足需求,逐渐成为系统瓶颈.为进一步提升多 GPU 之间的通信性能,充分发挥 GPU 的计算能力,NVIDIA 于 2016 年发布了全新的 NVLink[30]通信架构.

上述新型网络架构都可以直接提升带宽和延迟性能,可带来直接、明显的性能提升,但需要投入大量成本来更新网络设施.

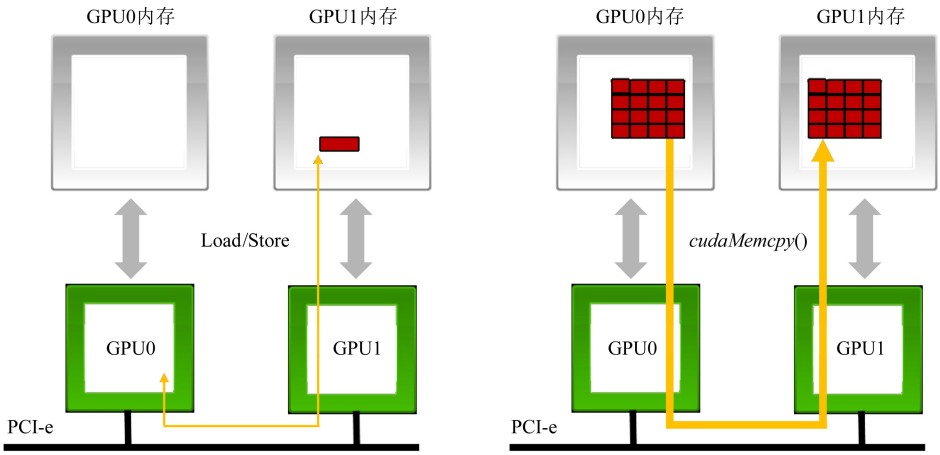


Fig. 2 GPU direct peer-to-peer proposed by NVIDIA in 2011

图 2 英伟达 2011 年提出的 P2P 技术

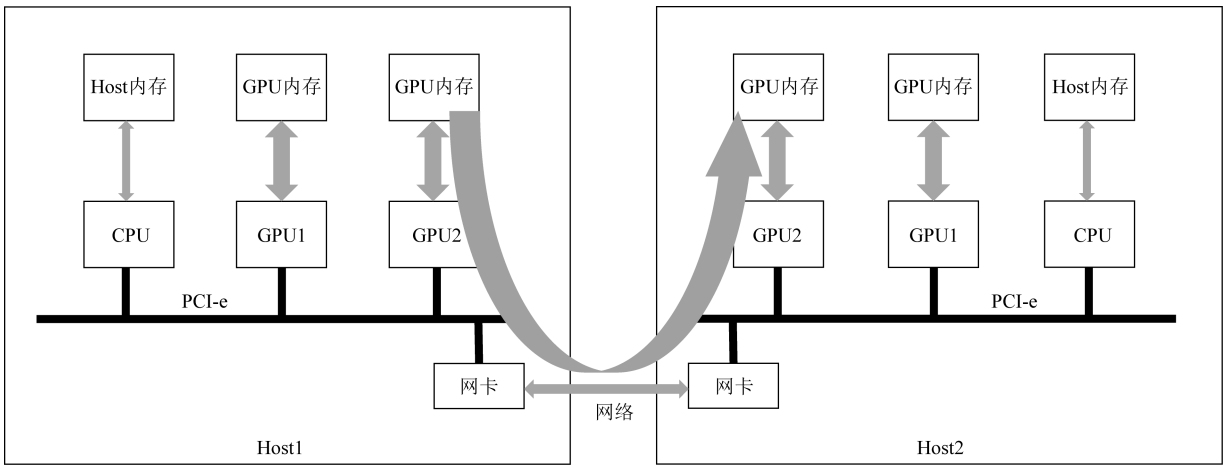


Fig. 3 GPU direct RDMA proposed by NVIDIA in 2013

图 3 英伟达 2013 年提出的 GDR 技术

2.2.2 采用 All-Reduce 集合通信

早期深度学习的分布式训练采用参数服务器(parameter server, PS)[31]的方式进行数据同步.具体方式为使用一个或多个服务器作为专门管理参数的服务器,其他节点完成一次训练后,将参数传至参数服务器;参数服务器收集各节点发送过来的参数,并归约(平均)这些参数,并将最终的结果分发返回给各计算节点;计算节点收到来自参数服务器的参数后,使用这些参数更新神经网络参数,并开始下一轮迭代.

参数服务器的结构如图 4 所示[32].其优点是算法简单、易于部署.缺点是参数服务器本身与计算

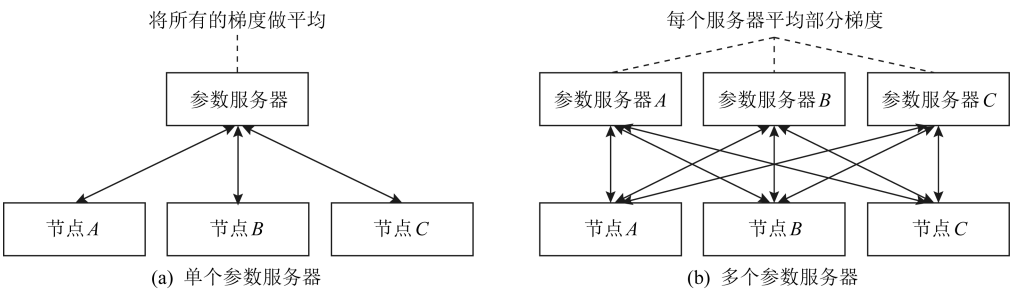


Fig. 4 Parameter server architecture

图 4 参数服务器架构

节点间需要进行 All-to-All 通信,容易产生流量拥塞;其次多机流量在链路中需要同时通过同一条链路,这种流量聚合容易占满带宽,造成带宽不足的通信瓶颈.针对以上参数服务器的缺点,研究人员提出各种参数服务器的改进办法,例如增加参数服务器数量、使用多级参数服务器等,但仍然不能彻底解决参数服务器通信瓶颈的问题.

目前主流的同步方式开始摒弃参数服务器,而采用类似于 MPI[33] 中的集合通信方式.其中 All-Reduce 集合通信[34-36]是最适合于该场景的集合通信方式,分为 Reduce-Scatter 和 All-Gather 这 2 个步骤.All-Reduce 对需要通信的数据进行分片划分成若干个片段(chunk),在 Reduce-Scatter 步骤中将 chunk 分发到其他节点,并从其他计算节点收集到一个完整的 chunk(归约了其他所有节点的该 chunk 数据);然后在 All-Gather 过程中发送这个完整的 chunk 并收集其他节点的其他完整 chunk,这样就能收集到所有通信数据的归约结果.如图 5 所示:

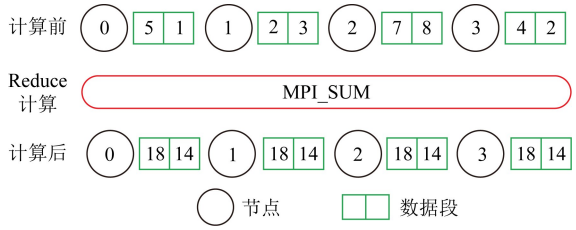


Fig. 5 Architecture of MPI All-Reduce[37]

图 5 MPI All-Reduce 的架构[37]

All-Reduce 集合通信方法在传统并行分布式计算中运用广泛,在超算环境下具有长久的积累,具备成熟的并行算法和编译环境.相对于参数服务器,All-Reduce 集合通信摒弃了类似参数服务器的中心管理节点,实现了去中心化,在通信过程中节点间关系平等,不存在流量共同流向中心节点造成的拥塞问题.但是采用 All-Reduce 后,由于没有了中心管理节点,给备灾容错、任务调度以及异步通信方法等带来挑战.

2.2.3 增加并行度

在 2.1 节中我们分析了分布式训练深度学习的过程.为了减少通信或计算的开销,应当尽可能得让不存在依赖关系的步骤实现并行化,从而减小时间消耗.

文献[38]中详细给出了深度学习训练中的并行度分析,重点围绕计算操作(operators)、网络和训练 3 个方面进行研究.其中计算并行加速方法主要是神经网络本身计算时通过使用矩阵运算来大幅提高并行度,这也是深度学习中 GPU 替代 CPU 进行训练的优势;网络方面的并行加速方法包括数据并行(data parallelism)、模型并行(model parallelism)和层流水(layer pipelining)的方式,其中数据并行为目前最常用的多机并行方式,而模型并行通过将完整的神经网络拆分到不同的计算节点进行计算,适合于神经网络参数特别大的场合,另外模型并行在一定程度上也可以减小单节点的计算任务;对于层流水方式,可以减小单计算节点的参数存储量,但并不能直接提高并行度,因为神经网络训练无论是前向传播还是反向传播都是逐层推进的.

对于实际训练过程,同时实现计算与通信并行是一个极其有效的优化方案.通信操作发生在每次反向传播计算该层神经网络完成的时刻,而反向传播进行下一次(即前一层)计算时不依赖通信后的结果,因此反向传播和参数(梯度)同步可以并行执行.文献[39]中给出了训练过程的有向无环图(DAG),如图 6 所示:

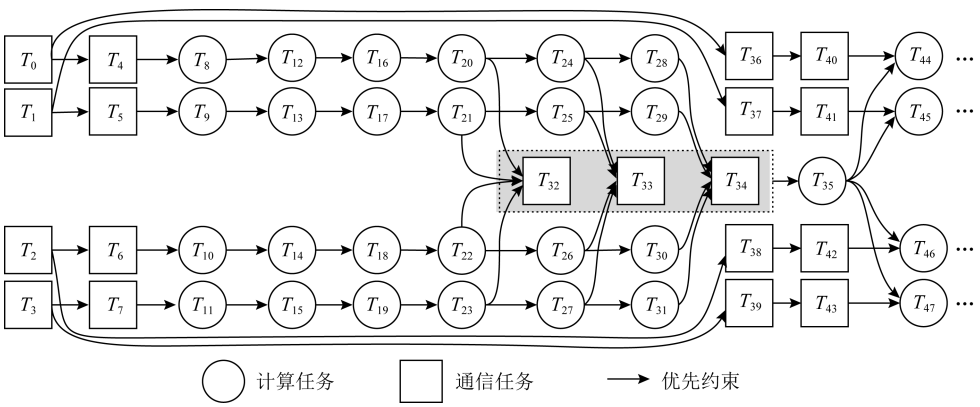


Fig. 6 DAG of distributed deep learning training

图 6 分布式深度学习训练的有向无环图

图 6 所示是 4 个计算节点进行 3 层神经网络训练的一个 Iter 完整的流程图.其中 T_0 到 T_3 为各节点获取数据的过程;T_4 到 T_7 为设备间通信(主要是 CPU 内存到 GPU 内存搬运)过程;T_8 到 T_{19} 为前向传播的 3 层神经网络过程,其中 T_8 到 T_{11} 为第 1 层,T_{12} 到 T_{15} 为第 2 层,T_{16} 到 T_{19} 为第 3 层;T_{20} 到 T_{31} 为反向传播的 3 层神经网络参数计算过程,其中 T_{20} 到 T_{23} 为第 3 层,T_{24} 到 T_{27} 为第 2 层,T_{28} 到 T_{31} 为第 1 层;T_{32} 到 T_{34} 分别是神经网络第 3 层到第 1 层参数或梯度的同步通信过程;过程 T_{35} 是完成参数同步后使用同步后的参数更新神经网络参数的计算过程,此过程实际上是每个节点自身计算完成的.

在不考虑并行时,一次神经网络训练的时间应为

$$t_{\text{total}}=t_{\text{io}}+t_{\text{H2D}}+t_{\text{forward}}+t_{\text{backward}}+t_{\text{comm}}+t_{\text{update}},$$

其中,t_{io} 是数据从硬盘读取到 CPU 内存的时间,t_{H2D} 是设备间通信(CPU 内存到 GPU 内存拷贝)的时间,t_{forward} 是前向传播计算时间,t_{backward} 是反向传播计算时间,t_{comm} 是参数同步的通信时间,t_{update} 是参数同步后更新时间.

利用该有向无环图我们可以看出,无逻辑依赖的部分可通过并行的方式来提高整体效率.因此,在设计深度学习框架的时候,应该尽可能将通信部分通过并行的方式隐藏在计算的过程中.目前,各深度学习框架的设计各不相同,有的侧重并行,有的并行度一般但计算速度快.在理想的情况下,每一次神经网络训练的时间应为

$$t_{\text{ideal}}=t_{\text{forward}}+t_{\text{backward}}+t_{\text{comm}}^{(1)}+t_{\text{update}}^{(1)},$$

其中,$t_{\text{comm}}^{(1)}$ 和 $t_{\text{update}}^{(1)}$ 代表神经网络第 1 层的同步时间和参数更新时间.在理想的训练过程中,节省了数据准备时间 t_{io} 以及设备间通信(CPU 内存搬运到 GPU 内存)时间 t_{H2D},因为数据可以在 GPU 计算时提前搬运到 GPU 内存中;而前向传播和反向传播是逐层计算的,因此无法并行;而通信和更新参数可以隐藏在下一个反向传播层中,因此除了最后一次之外都可以与反向传播计算并行执行.

增加并行度的方法更多的是与深度学习框架开发相关的并行优化.如果将训练中的并行方案做好,可以在理想网络状况下将通信时间尽可能隐藏在计算时间内,极大减小了通信时间.缺点是依赖于深度学习框架的开发程度,因此自主开发优化的难度较大.

2.2.4 同步通信算法优化

同步通信最容易成为分布式训练的瓶颈,因此直接针对 All-Reduce 的优化也是一个受关注的热点问题.

集合操作 All-Reduce 早期集成在消息传递接口(message passing interface, MPI)[40]中,因此针对 All-Reduce 的研究已经比较成熟.

目前使用最广泛的 All-Reduce 算法是 Ring All-Reduce,其最早由百度公司应用在分布式深度学习训练中[41],并在实验环境取得较好的性能.目前,该算法已被主流的深度学习通信框架采用,例如 NCCL[42], Horovod[32], gloo[43], GPU-MPI[44] 等.其具体原理如图 7 所示:

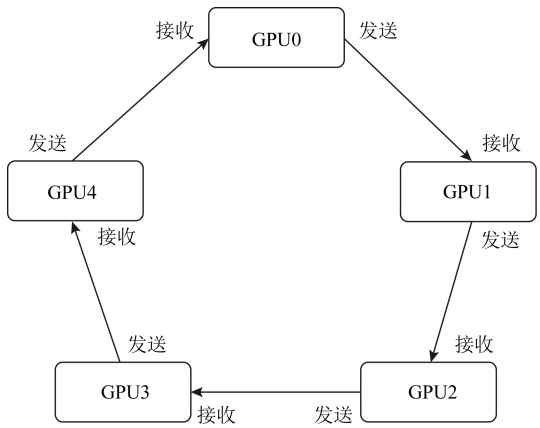


Fig. 7 Ring All-Reduce structure

图 7 Ring All-Reduce 的基本结构

各计算单元在逻辑上构成一个环状拓扑,仅与左右 2 边的单元通信.将所需同步的数据按照节点数 N 平均切分成 N 个 chunk,在 Reduce-Scatter 过程每个节点每次轮询传输一个 chunk,并将接收到的 chunk 做归约操作.当完成 $N-1$ 次传输后,每个计算单元会收集到一个完整的 chunk.之后 All-Gather 过程重复 $N-1$ 次传输后,每个计算单元都会收到所有的完整归约的 N 个 chunk.

对于节点数达到数百或数千的分布式训练任务来说,Ring All-Reduce 可能会遇到通信瓶颈.这是由于当通信节点数目增多时,Ring 的传输次数快速上升,而单次传输的数据数量迅速减小,导致大量的小容量包在环形结构上高频传输,导致最终平均传输效率降低.

为了优化超大规模下 All-Reduce 的传输效率,许多超大规模实验中开始使用多维环状(torus)的结构来替代单维 Ring 结构,其中每一维度都是完整的环形结构,通过提升维度,可以减小每一维度的环大小,最终显著减少 All-Reduce 中传输的次数.

NCCL 在 2.4 版本中提出了双向二叉树(double binary tree)[45],它使用 2 个二叉树结构,使得每一条链路单次传输的时候没有链路聚合而且双向带宽都能得到充分利用.利用二叉树的结构显著降低了 All-Reduce 中的传输次数.在实验环境下双向二叉树取得了比传统 Ring 和多维度 Ring 都更出色的性能,尤其是在节点规模比较大时,双向二叉树性能优势更明显.其基本结构如图 8 所示:

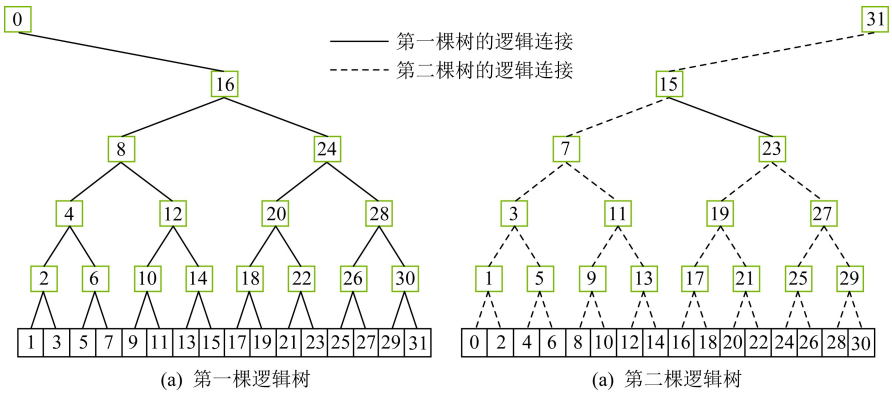


Fig. 8 Double binary tree structure

图 8 双向二叉树的基本结构

除此之外,还有许多其他常见的 All-Reduce 算法.例如 Recursive Doubling,Recursive Halving and Doubling,Binary Blocks 等.文献[46]对比了在不同配置环境(节点数、通信包大小)下各种 All-Reduce 算法的性能,图 9 给出了不同 All-Reduce 算法中通信节点数随通信量的变化.从此可以看出,不同网络环境下不同的 All-Reduce 性能存在不小差异,因此在选择 All-Reduce 通信算法的时候需要考虑具体配置环境,包括数据包大小、节点数、网络状况等诸多因素.因此如何在特定网络环境下快速选择出最优的 All-Reduce 算法并不容易.

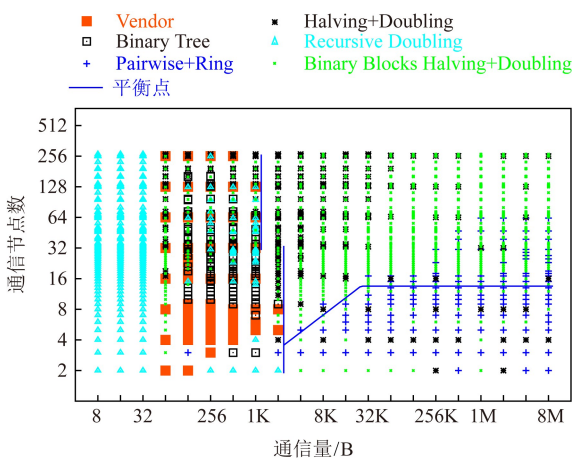


Fig. 9 Performance of different All-Reduce Algorithms

图 9 不同 All-Reduce 算法的性能对比

2.2.5 张量融合技术

在 2.2.4 节中提到不同 All-Reduce 算法在不同环境下性能是有差异的,那么如何优化使得同一种 All-Reduce 算法在不同环境下尽可能获得较好的性能是一个挑战.

在实际训练环境下,通常最直接影响一个 All-Reduce 算法性能的主要是小包的通信性能.无论一个通信框架多么完善,除去本身带宽限制导致的延迟之外,还存在诸多因素引起的网络本身的延迟;而对于大包来说,该延迟造成的影响相对于整体传输时间来说较小,更容易接近物理极限带宽;相比之下小包对延迟效应更敏感.

张量融合(tensor fusion)技术[47]被提出并应用于解决同步通信时传输量不均等的问题.其原理是将通信的包进行拆分、拼合,从而实现相同大小的数据包传输.

Horovod 中实现张量融合的步骤为[48]:

1) 判定哪些张量准备开始做 Reduce 操作,选择最初几个可以适配配置标准尺寸的且同数据类型的张量;

2) 为标准尺寸通信单元分配内存;

3) 将选定的张量拷贝至标准尺寸通信单元;

4) 对标准尺寸通信单元执行 All-Reduce;

5) 将同步后的数据从标准尺寸单元拷贝至输出张量;

6) 重复以上过程直到本轮没有多余的张量需要传输.

根据上述的过程我们可以看到,尽管张量融合技术能够保证数据包的一致性,但需要额外完成 2 次

数据的内存拷贝工作,这同样也会带来性能损耗;在训练框架中进行张量融合,会使得原本应该实时进行同步的 All-Reduce 操作变成异步的模式;除此之外,如何选定标准尺寸通信单元的大小也是实际使用该方法时所面临的问题.

2.2.6 使用异步通信

尽管目前使用同步通信已经成为分布式深度学习训练的主流通信方式,但异步通信作为有效提升训练效率的方法,也一直是研究热点.

目前异步通信的实现都是基于参数服务器(parameter server)结构.传统的同步通信过程中,每一个同步 Iter,参数服务器需要收集到所有节点的参数数据,并完成归约操作后才能发送给各节点完成本轮通信.由于不同节点数据到达参数服务器的时间可能不一样,参数服务器必须要等待至最慢的节点数据到达之后才能完成整个过程,这个等待过程会导致通信性能及训练效率的下降;而异步通信中,每个节点数据到达参数服务器后,可以根据异步设置,不需等待或者仅等待少量节点参数到达,就可以直接由参数服务器返回数据,这将在每个训练 Iter 中节省大量参数同步时间.

尽管异步训练方法实现了更好的并行性,但其并非严格的梯度下降过程,因此会导致训练精度下降.对于同步 SGD,各计算节点的梯度值是进行累加/平均操作,在数学上与单节点训练相同数据是等同的;而对于异步 SGD,同一 Iter 中每个节点从参数服务器上获取的参数不能保证相同,相当于是在不同的位置做梯度下降,这种不精准的梯度被称为浑浊梯度(stale gradients)[49].浑浊梯度的问题会造成训练始终与同步 SGD 具有误差,导致最终可能无法获得同步 SGD 那样精准的结果.

文献[50]中提及到一种弹性平均 SGD(elastic averaging SGD)方法,基于一个数学公式来减小异步通信引起的浑浊梯度问题.但是该方法仍然使用参数服务器的结构,无法避免多对一通信时造成的带宽瓶颈问题.

文献[51]中提出了流言 SGD(gossiping SGD)方法,使用流言协议(gossip 协议)来实现非中心化的异步同步策略.这种方式具有良好的扩展性和容错性,同时可以实现去中心化.但该方法也存在不少缺点,譬如同步延迟不确定、延迟更高、存在消息冗余等,即数据会重复发送,从而浪费了带宽.

采用异步通信的方式可以实现在节点间训练速度不平衡时,传输快的节点不需要等待慢的节点数据到达,因此节省了同步时的通信时间.但是目前来说异步通信都是基于参数服务器结构的,还没有人尝试在 All-Reduce 中实现异步结构,此外异步通信还会在一定程度上造成训练精度的丢失.

2.2.7 量化压缩与稀疏化压缩

为了降低通信时间占比,一种直接且有效的方法是降低通信量(大小和数量).深度学习本身是一种模糊计算,因此对于数据精度的要求并不像科学计算那样高,一定程度上可以通过有损压缩的方法来显著降低通信量.目前针对深度学习数据压缩的方法主要有量化压缩与稀疏化压缩 2 种.

量化压缩来源于 Google 的文章[52],通过使用 8 b 或 16 b 的整数类型来替代 32 b 的浮点数类型.这种使用低精度的计算方法很大程度节约了内存与存储的开销,同时在可以使用单指令流多数据流 SIMD 的设备可以显著增加训练的速度.而对于分布式训练来说,低 bit 的数据传输量成倍减小,从而大幅节约了通信时间.同时在一些无浮点数支持的低功耗嵌入式设备中,利用低 bit 进行训练或推理都是一个很好的选择.

量化压缩是一种有损压缩,因此在选择压缩位数时也需要平衡考虑.目前 16 b,8 b 量化压缩已经在实际系统训练中被使用,实验表明有限的有损压缩不会显著影响训练精准度.也有研究表明更低位数的量化压缩,例如 3 值压缩(−1,0,1)[53-54]甚至 1 b 压缩[55],在某些场景也能满足使用的需求.

稀疏化压缩是另一种有损压缩的方法.在神经网络参数更新时,并不是每一次都会更新所有的参数,有的参数更新的多,有的参数更新的少.稀疏化压缩是通过尽可能减小不更新的参数传输或更新较小的参数传输,来保障重要参数参与通信同步的同时显著降低通信量.

最早稀疏化压缩由文献[56]提出,该文基于阈值(thresholding)方式来选择传输数据,仅当梯度值大于某个固定的阈值时才进行传输,这种方式的压缩率达到了 846～2 871 倍,极大地减少了通信量.但由于这种阈值是个静态值,在不同情形下需要手动进行调整.文献[57]在此基础上做了若干改进,选择了正比例和负比例梯度更新的固定比例;文献[58]提出了梯度下降以基于绝对值通过单个阈值稀疏梯度;还有研究为了保持收敛速度,梯度下降需要添加归一化[59]等.但是以上压缩基本都在小训练集(例如 MNIST 训练)和简单的神经网络(例如全连接

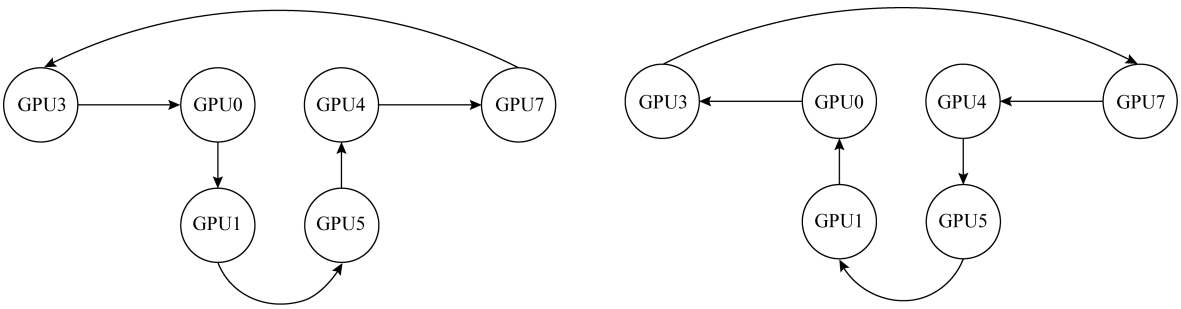
网络)做测试.文献[60]中提出的 DGC 使用了多种策略混合调整,最终使得稀疏化压缩在复杂神经网络(例如 ResNet-50,AlexNet)中也取得了很好的加速,且没有显著影响计算精确率.

总结来说,量化压缩与稀疏化压缩方法可以有效减少通信量,从而减小通信时间.但压缩过程中也存在计算时间消耗,通过作为一种有损压缩方案会一定程度降低训练结果的精准度.

2.2.8 与拓扑相关的网络优化

现实中分布式训练环境非常复杂,不同的网络环境差异极大,利用网络拓扑信息对通信做针对性的优化,可以充分利用带宽资源,从而实现整体训练效率的提升.

文献[61]中针对多计算节点间通信带宽利用率低的问题,提出利用多个不同路径的生成树,尽可能将每一条链路每个方向的带宽充分利用.这种策略针对存在多条通路的网络环境效果很好,例如 DGX-1 机内网络,因为该策略充分利用了标准 All-Reduce 通信例如 Ring 中没有利用到的链路,减少了单链路的传输量.图 10 显示的是 Ring All-Reduce 针对链路的利用图;图 11 中显示的是算法将通信分为 3 条链路后的利用图.

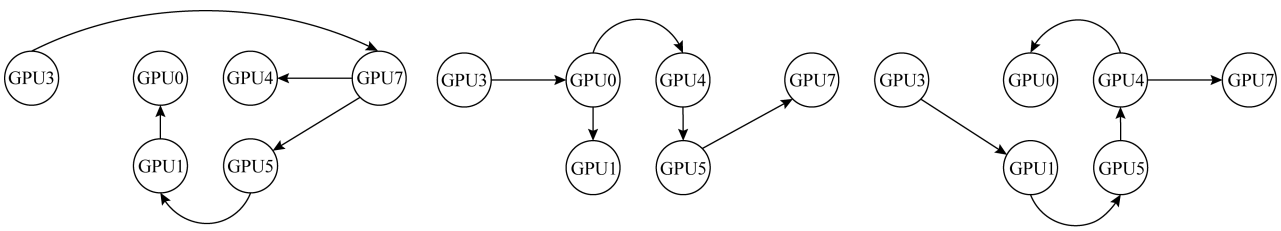


(a) Ring All-Reduce 通信逻辑环 1　　(b) Ring All-Reduce 通信逻辑环 2

测试环境:NCCL 6-GPU Ring-Allreduce. 在GPU1与GPU3, GPU5与GPU7, 以及GPU0与GPU4之间的连接没有被用到.

Fig. 10 Path for Ring All-Reduce

图 10 Ring All-Reduce 的链路利用图



(a) Blink 第一条链路　　(b) Blink 第二条链路　　(c) Blink 第三条链路

Fig. 11 Three tree paths for one All-Reduce

图 11 All-Reduce 切分成 3 条链路

目前分布式训练中一般会将通信方式设置为多级架构(hierarchical),普遍都会针对机内通信和机间通信做不同处理.图 12 是典型的 All-Reduce 在机内、机间的混合方式.其在机内是环状的通信方式完成第一次 All-Reduce,接着在机间通信时仅由一个计算单元与其他节点通信,也是采用环状的 All-Reduce 通信方式,最后再将最终 All-Reduce 结果通过机内广播至各计算单元.

与拓扑相关的网络优化可以充分利用环境的特征,寻找最适合的通信手段,从而达到提升性能的效果.但是如何及时完成拓扑发现、如何调整以达到最优效果都是其困难所在.目前对其的研究仍处于起步阶段,有待进一步研究.

2.2.9 专用系统设计

在传统通信网络无法有效解决通信瓶颈问题时,根据应用训练环境进行专用的系统设计成为一种可行且实用的途径.

对于分布式深度学习来说,通信部分是影响其加速比的最大瓶颈,而其中又以参数或梯度的同步操作为主.针对深度学习中 All-Reduce 采用专用的

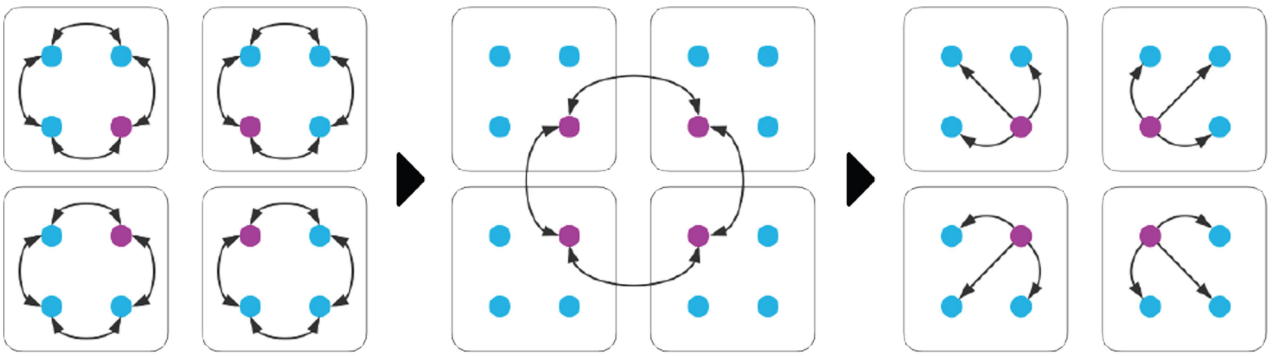


Fig. 12 Hierarchical All-Reduce structure

图 12 多级的 All-Reduce 结构

软硬件系统设计是一种切实可行的方案.

Mellanox HPC-X[62] 中使用了 Mellanox 的专用交换网络,如 Mellanox 网卡(CORE-Direct 引擎和硬件标签匹配)和交换机(支持 Mellanox SHARP 加速引擎).通过采用 SHARP(可扩展分层聚合和缩减协议)技术将集合操作由 CPU 卸载至交换网络.由于采用了聚合节点却又不存在多条链路数据聚合到一个链路中,减少了链路中的数据聚合,缓解了网络拥塞;另一方面通过将集合操作卸载到网络,也减少了集合操作的时间,释放宝贵的计算资源用于计算而不是耗费在通信处理上,减少了网络传输的数据量.

在目前最顶尖的超级计算机 Summit 中[63],也针对集合通信使用了在网计算的优化方案.Summit 使用了混合的架构,每个节点包含了 2 个 22 核心的 IBM POWER9 CPU 以及 6 个 NVIDIA Tesla V100 GPU.超算中共包含约 3 400 个节点,采用无阻塞胖树架构互联,基于 Mellanox Infiniband EDR 实现存储以及节点间 200 Gbps 的带宽连接.在整个系统设计中,针对通信架构使用了在网计算进行加速,可以用于常见的 MPI,SHMEM/PGAS 等通信框架,在网络交换机上实现归约等集合操作.

其他面向深度学习的在网计算方法包括:网卡类计算卸载,例如 Mellanox ConnectX(NIC),Cary Aries NIC,IBM BlueGene 系列等;交换机类计算卸载,例如 G-Net Switch,Mellanox SwitchIB-2,P4 Switch 等.其中网卡类计算卸载无法减少集合操作在网络中的流量,仅能替代 CPU 或 GPU 完成 All-Reduce 中的求和工作;而交换机类计算卸载既可以实现 All-Reduce 过程中的计算任务,又可以减少链路聚合从而缓解网络拥塞.

使用在网计算等专用系统设计可以从物理层面直接改善分布式深度学习中网络拥塞的问题,是一种高效且扩展性强的方案,其缺点是需要专用的硬件和配套软件,具有较高成本.

2.2.10 减小通信占比

不同的神经网络具有不同的参数大小,因此其计算/通信比例是不相同的.通过改变神经网络的类型,可以降低通信在整个分布式训练中的比例,从而降低通信开销.

在典型神经网络的网络层中,主要分为卷积层、激活函数、池化层和全连接层等类型,其中涉及到大量参数和计算的有卷积层和全连接层.由于卷积层每层的参数只有若干个卷积核,并且全连接层 2 层之间所有神经元都有连接,故而全连接层的参数量远远大于卷积层.

以 VGG-16[64] 为例,Conv1-1,输入为 224×224×3,64 个 3×3 filter,输出 feature map 为 224×224×64,其参数量为 3×3×3×64=1728.同理 Conv2-1,输入为 224×224×64,128 个 3×3 filter,输出 feature map 为 112×112×128,Conv2-1 的参数量 3×3×64×128=73 728.而对于全连接层来说,VGG-16 的第一个全连接层中,最后一次卷积得到的 feature map 为 7×7×512,展开成一维向量为 1×4 096,因此其参数量为 7×7×512×4 096=102 760 448 个参数.因为分布式训练过程中通信的主要内容就是参数或参数的梯度同步,因此全连接层占比多的网络,其通信/计算比就更大,更容易因为通信问题造成瓶颈.

故而在选择和优化神经网络时,选择卷积层比例多而全连接层比例少的网络可以更有效地减少通信占比.常见的神经网络及其计算量(GFLOPs)和参数量(即通信量)统计如表 1 所示:

Table 1 Parameters for Common Neural Networks

表 1 常见神经网络的参数

名称	类型	参数数量	模型尺寸/MB	计算量/GFLOPS
AlexNet	CNN	60 965 224	233	0.7
GoogleNet	CNN	6 998 552	27	1.6
VGG-16	CNN	138 357 544	528	15.5
VGG-19	CNN	143 667 240	548	19.6
ResNet50	CNN	25 610 269	98	3.9
ResNet101	CNN	44 654 608	170	7.6
ResNet152	CNN	60 344 387	230	11.3
Eng Acoustic Model	RNN	34 678 784	132	0.035
TextCNN	CNN	151 690	0.6	0.009

通过减小通信占比来优化通信时间是一种被动的优化方法,因为研究人员通常是以数据为驱动的而不是以性能为驱动的.因此,只有在一些极端情况例如终端计算时,才会采用这样被动的优化方案,例如在低功耗设备上等,通过缩减模型同时尽可能保证结果精确度.

2.3 关于文件传输的优化

关于分布式深度学习中关于文件传输、数据供应的优化,相关的研究并不多.主要有 2 方面原因:

1) 并非所有应用都会使数据存储面临瓶颈.对于一些标准数据集而言,其数据集大小是有限的.例如 ImageNet 标准集,其大小在 150 GB 左右.对于分布式训练来说,完全可以做到每个节点都拷贝一份数据存放在本地使用.

2) 并非所有的集群架构都存在存储瓶颈.某些集群架构中将存储做成了集中存储的存储池,并且拥有专门的网络线路来支撑数据传输,这样无论训练数据有多大,都不会达到集中存储池的上限.此外某些集群还针对文件传输建立了一套独立的网络用来传输数据,大大降低了文件传输对通信网络的干扰.

需要着重对文件传输进行优化的场景主要是超大数据(一般是非传统数据集)用于深度学习的训练过程.例如在文献[65]中,训练数据达到了 3.5 TB,这不仅对存储带来了压力,对数据的传输也带来了巨大压力.由于深度学习中并不需要每个节点都使用全部的数据,只需要在一个 Epoch 过程中,所有节点共计使用过一次全部数据即可.故而文献[65]提出一种分布式文件分级系统,将文件划分为多段,供不同的节点训练使用.同时每个节点使用多线程并行读取数据,使用 8 线程时将数据读取速度提高了 6.7 倍.该文使用这种方法,可以实现在 3 min 内完成在 summit 上对 1 024 个节点的数据供应,以及 7 min 对 4 500 个节点的数据供应.

总体来说,关于文件存储、传输等是一项与实际使用相关性很高的工作.针对其优化更适合在实际系统中遇到实际问题后按需解决,目前还没有成熟通用的在分布式深度学习训练中分布式供应数据的系统.

2.4 目前超大规模训练实例

2.4.1 传统数据集训练

本节详细介绍目前已有的若干分布式深度学习的超大规模训练实例,分析它们的规模以及如何保证超大规模下的通信性能.

在这些训练实例中,最常用的性能测试集是使用 ResNet-50 训练 ImageNet 数据集.ResNet 在 2015 年被提出,在 ImageNet 比赛中获得了分类任务的第 1 名,并运用在检测、分隔和识别等诸多领域,Alpha zero 中也使用了 ResNet,是目前最常使用的一种 CNN 网络.使用一块 NVIDIA M40 GPU 和 ResNet-50 训练 ImageNet 训练集,训练 90 个 Epoch 的完整过程需要 10^{18} 个单精度浮点操作,需要花费约 14 天.

文献[66]基于 256 个 GPU 实现了 1 h 内完成训练 ImageNet/ResNet-50 的成绩.实验中使用了以 Facebook's Big Basin[67] 为主的 GPU 服务器.每个服务器含有 8 个 NVIDIA Tesla P100 GPU,相互之间采用 NVLink 连接,其每台服务器本地有高达 3.2 TB 的 SSD 硬盘,网络连接使用 Mellanox ConnectX-4 50 Gbps 的以太网卡和 Wedge100 以太网交换机.在软件方面使用了 Caffe2 作为训练框架,Gloo 作为通信框架.该实验具备了优秀的硬件设施系统,而没有针对分布式训练做其他方面的性能优化.实验结果表明,在使用 256GPU(32 机)时,其单卡效率可以达到 8GPU(1 机)的约 90%.

文献[68]中使用 1 024 个 GPU 完成了 15 min 使用 ResNet-50 训练 ImageNet 的成绩.其使用 Preferred Networks 提供的 MN-1 cluster,包含 128 个节点,每个节点包含 2 个 Intel Xeon E5-2667 CPU、256 GB 内存和 8 个 NVIDIA Tesla P100 GPU,节点间使用 Mellanox Infiniband FDR 实现互联.在软件方面其使用了 Chainer/ChainerMN 作为训练框架,NCCL 和 Open MPI 作为通信库,亮点在于通信时使用了半精度浮点数来减少通信量.

腾讯中实现了 6.6 min 训练 ImageNet/ResNet-50 的成绩[69].文献[69]中的集群含有 256 个节点,每个节点有 8 块 NVIDIA Tesla P40 GPU.节点间使用 Mellanox ConnectX-4 100 Gbps 的以太网卡,使用 RoCEv2 实现 RDMA,同时使用了 GDR 技术在节点间的 GPU 实现数据交换.软件方面使用了 TensorFlow 作为框架,NCCL 和 OpenMPI 作为通信库.亮点在于使用了混合精度的模式来提高单 GPU 的吞吐,同时使用了多维混合的 All-Reduce 方法来降低通信延迟,在此基础上还使用了张量融合的技术来减少小包在通信中的开销.

索尼公司在文献[70]中实现了 224 s 训练 ImageNet/ResNet-50 的成绩.其使用了 ABCI cluster,包含 1 088 个节点,每个节点含有 4 块 NVIDIA Tesla V100 GPU、2 个 Xeon Gold 6148 CPU 和 376 GB 内存.节点内 GPU 使用 NVLink2 连接,节点间使用 InfiniBand EDR 实现互联.软件方面其使用了 Neural Network Libraries(NNL)作为训练框架,NCCL 和 OpenMPI 作为通信库.亮点在于使用了 2D-Torus 的改善 All-Reduce 方法,降低了普通 Ring All-Reduce 的延迟.

除了使用 GPU 集群实现的训练,也有使用超算 CPU 资源实现的分布式深度学习训练.CPU 相比 GPU 并行计算能力更弱,因此需要更多核心的并行计算才能达到相同效果,这增加了分布式的节点数,对通信过程带来更多的挑战.例如文献[71]中使用 2 048 个 Intel Xeon Platinum 8160 处理器,训练 90 个 Epoch 的 ImageNet/ResNet-50 花销 20 min,64 个 Epoch 时花费 14 min,但是在文章中没有对性能做针对性优化.

总的来说,在传统数据集的超大规模训练上,目前已经基本做到了性能的极限水平.在利用目前最先进的显卡、网络设备等支持下,训练完整的 90 个 Epoch ImageNet/ResNet-50 只需要不到 4 min 的时间.

2.4.2 非传统数据集训练

目前越来越多的新型超算开始采用 GPU 集群架构,因此也越发适用于超大规模分布式深度学习任务.这使得某些传统超算任务可以与深度学习建立联系,使用深度学习的方法来达到更好的效果.

文献[65]中使用超算集群来实现气候分析.其在 Piz Daint 集群中利用 Tiramisu 网络,使用 5 300 块 P100 GPU,达到了 21.0PF/s 的吞吐率和 79.0% 的并行效率;另外其在 Summit 集群中利用 DeepLabv3+网络,使用 27 360 块 V100 GPU,达到了 325.8PF/s 的吞吐率和 90.7%的并行效率.

AlphaGo[72]围棋 AI 程序早期使用 176 个 GPU 的分布式系统来实现版本 AlphaGo Fan 的训练.后来在 AlphaGo Lee 中使用 48 个 TPU 的分布式训练来实现围棋 AI 的训练.直到最新的 AlphaGo Zero,才开始使用单机 4TPU 的结构来降低功耗.

总的来说,深度学习在非传统应用如图像、音频和自然语言处理等方面处于刚刚起步的状态,随着未来深度学习的发展以及行业的需求,非传统应用会越来越广泛.

3 未来的研究问题与挑战

随着深度学习技术的不断进步,未来越来越多的应用会使用深度学习算法.对于某些超大型的数据集和计算任务,分布式深度学习提供了良好的解决方案.在超算领域,分布式深度学习为传统计算任务提供了新的解决思路,改善了传统模型的准确性.

目前针对分布式深度学习的性能优化有诸多方法,如何将这些方法利用在训练环境中,并缓解性能瓶颈成为一个挑战.常见的优化方案及其优缺点总结如表 2 所示.

我们根据目前各优化方案的研究及其成果对表 2 的10 种方案做出关于成熟度的分类.其中较为成熟可靠的方案有:改善网络硬件资源、采用 All-Reduce 集合通信、增加并行度、改善同步通信算法、使用张量融合技术;而使用异步通信、量化压缩与稀疏化压缩、与拓扑相关的网络优化、专用系统设计、减小通信占比等方案目前仍处于研究初期或存在较大的缺陷.

尽管目前针对分布式深度学习训练的性能优化方法有很多,但是目前已有的科研方案中各系统展现了很强的独立性,各系统采用了不同的优化方案并实现了不同的性能效果.研究之间相互没有统一性.最主要的原因是由于目前的优化方案不完全成熟,训练环境差异大,因而未来分布式深度学习仍然面临诸多挑战.

展望未来,分布式深度学习性能优化方面具体面临的困难总结为 6 点:

1) 对于目前分布式深度学习训练的推广,缺乏一套综合性设计方案.目前已有的深度学习框架、通信框架、通信算法、硬件资源等种类众多,却没有保障分布式的性能,以及使用便利性.因此应该利用

相对完善的优化方案,形成系统的深度学习框架系统和性能评测系统,设计出一套在绝大多数环境场景下都能保证分布式训练性能的系统方案.实现一套综合性设计方案可以使得广大用户、中小型公司与科研单位充分利用分布式加速的好处,是目前分布式深度学习普及的重中之重.

Table 2 Comparison for the 10 Optimization Methods

表 2 10 种优化方案的对比

优化方案	优点	缺点
改善网络硬件资源	从物理上直接提升带宽,降低延迟	成本高(需要新设备及使用维护开销)
采用 All-Reduce 集合通信	减少带宽聚合	不利于集中管理
增加并行度	通过重叠计算与通信减少通信独立时间	实现与深度学习框架有关,修改困难
改善同步通信算法	通过选取更合适的算法来减少通信时间	不同环境下算法选择不固定
使用张量融合技术	减少小通信包的通信时间占比	有额外的计算开销,会造成异步
使用异步通信算法	减小由于计算时间差导致的节点等待	只支持 PS 结构,另外会造成浑浊梯度影响训练精度
量化压缩与稀疏化压缩	大幅度降低通信量	造成数据信息丢失,影响训练精度
与拓扑相关的网络优化	充分利用拓扑带宽	限定应用范围,系统、算法实现难
专用系统设计	减少带宽聚合,减少节点计算量	硬件成本高
减少通信占比	调整神经网络来降低同步参数量	在多数环境下性能因素并非调整神经网络的主导因素

2) 缺乏针对其他典型环境下的设计方案.除去综合性的设计方案之外,应该对于各典型环境针对其特点做出常用的系统方案.目前常见的分布式深度学习训练环境主要有:小型局域网、云计算环境、超算环境、私有集群等.其中不同环境在网络带宽、用户数、稳定性、计算节点数等有显著的不同,因此对于各典型的训练环境来说应该采用针对其特点采用不同的优化策略,形成不同的优化设计方案.该方案可以在综合性系统之上保证分布式训练在各环境中性能进一步提升.

3) 缺乏已有算法综合性的性能比较.目前各系统及其优化方案独立性强,相关的比较研究较为匮乏,因此目前还没有综合性的优化算法性能比较.因此在针对环境进行优化设计时,需要将所有算法都进行实验,这加大了系统设计的复杂程度以及时间成本.

4) 一部分新优化算法不够成熟,需要进一步研究.表 2 中优化方案中一部分算法仍处于研究的初始阶段,例如与拓扑相关的网络优化方案;也有目前为止存在很多问题的优化方案,例如稀疏化压缩等;也有一些无法或不便于大规模推广,例如专用系统设计等.因此需要继续改善这些优化方法或提出新的算法,这必然是一个需要研究积淀的长期过程.

5) 缺乏对于极端环境的容错、冗余方案.目前研究实例中的实现方案大多处在理想的实验环境中,例如不存在多用户的干扰,大规模训练中也不需要考虑服务器宕机等情况.而实际生产实验环境中,环境要更加复杂,会出现各种各样的问题导致分布式训练性能降低、中断等.因此如何实现对于复杂、极端环境的容忍,多机训练任务中断的恢复、多机任务的管理等成为一个挑战.

6) 缺乏算法参数选择机制的研究.目前已有的性能优化方案中通常涉及到多参数、多算法等问题,而现有实现主要是根据实验中的实验数据和结果人工调整参数以达到最佳性能效果.然而不同的参数对于不同的环境来说通常是不通用的,因此使用实验中的参数不一定能达到最优的性能效果,需要重新实验来确定参数.这对于非研究人员来说是一个繁琐的过程,增加了训练的时间和精力成本.因此如何确定一个相对更通用的参数或如何实现自动选择参数将成为未来研究的挑战.

由于近些年来贸易战等政策原因,国外针对我国技术封锁越来越严重,未来我国需要在深度学习方面多做技术与科研储备.

目前来说,国内的深度学习发展主要分为 2 方面,一方面是由普通高校、研究所等科研人员针对算法应用层面不断研究;另一方面是由国内顶尖 IT 企业、AI 公司和较有实力的科研单位从底层系统到上层算法的全方面研究.其中前者由于近些年来深度学习的火热程度,国内研究投入力度大,取得成果在国际上也处于领先水平;而后者由于国内在深度学习领域起步相对欧美国家晚,在底层架构、性能优化等方面基础相对薄弱.

分布式深度学习作为未来深度学习扩展性能的重要方法,必然会得到越来越多的重视.例如现代超算集群中,越来越多支持 GPU 集群计算用于支持深度学习.国内在系统结构方面相对较弱,而在新型计算结构(例如 GPU)等积累更少.未来应该重视自主研究的计算器件、深度学习框架,多做研究实践积累和市场推广,利用市场效应带动应用推广,从而实现良性循环.未来国内理想的超算环境应该部署自己的向量计算加速器、使用自产神经网络框架,甚至在专用网络器件等设备中也实现国产化.

针对短期内的性能优化,应该着重分析目前主要分布式环境,可以构建基础数学模型,对环境进行模拟和测试,分析不同环境下分布式训练时可能会遇到的问题以及可能使用哪些优化方案来解决,分析出主要环境的特点以及相对通用的分布式优化方案;设计出一套相对智能的分布式深度学习框架,可以通过测试不同环境下的通信、计算等特征从而自动选择可行的优化策略,使得不同用户在不同环境下都能取得较好的分布式加速性能.

针对长期的性能优化工作,一方面应该继续对各类优化算法进行研究,改善目前已有优化算法的不足,提出新的算法,以及进行实际系统中算法间的性能比较;另一方面应该加大在底层硬件例如低功耗向量加速器件、网络通信设备、专用加速器件等研发生产,不断改善底层软件框架例如深度学习框架、通信库等使之愈发成熟.

总体来说,分布式深度学习还处于发展的初始阶段.目前绝大多数的深度学习训练还是处于单机训练的阶段.但是随着未来更多应用的需求,深度学习对算力的要求越来越高,分布式深度学习会得到越来越多的关注.而分布式深度学习的性能优化是维持分布式训练加速不可或缺的重要手段,未来在分布式训练中必然起到至关重要的作用.

4 总 结

分布式深度学习突破了单机训练的算力瓶颈,为未来深度学习的发展提供了充足的算力支撑,可以为未来相关应用的发展打下重要基础.然而分布式训练中会由于通信等问题造成性能瓶颈,因此关于分布式深度学习的性能优化显得尤为重要.本文阐述了分布式深度学习发展中可能遇到的问题与挑战,分析了当前网络通信性能中制约分布式训练的主要因素,总结和对比了 10 种优化分布式深度学习网络性能的方案,最后针对未来分布式深度学习研究提出 6 个研究挑战与未来方向.

我们认为未来分布式深度学习将得到更广泛的应用,针对分布式深度学习的具体优化方案会更加完善.在未来常用的环境中有更成熟、完善的方案来支撑分布式深度学习应用的发展.未来值得我们进一步探讨在分布式深度学习中引入各类性能优化技术.

参 考 文 献

[1] Zhang Chenlin, Zhang Hao, Wei Xiushen, et al. Deep bimodal regression for apparent personality analysis [C] // Proc of European Conf on Computer Vision. Berlin: Springer, 2016: 311-324

[2] Lee H, Pham P, Largman Y, et al. Unsupervised feature learning for audio classification using convolutional deep belief networks [C] //Proc of Advances in Neural Information Processing Systems. New York: Curran Associates, 2009: 1096-1104

[3] Young T, Hazarika D, Poria S, et al. Recent trends in deep learning based natural language processing [J]. IEEE Computational Intelligence Magazine, 2018, 13(3): 55-75

[4] Ivanitskaya L, Clark D, Montgomery G, et al. Interdisciplinary learning: Process and outcomes [J]. Innovative Higher Education, 2002, 27(2): 95-111

[5] Deng Jia, Dong Wei, Socher R, et al. ImageNet: A large-scale hierarchical image database [C] //Proc of IEEE Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2009: 248-255

[6] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-ResNet and the impact of residual connections on learning [C] //Proc of the 31st AAAI Conf on Artificial Intelligence. Menlo Park, CA: AAAI, 2017: 4278-4284

[7] You Yang, Zhang Zhao, Hsieh C J, et al. ImageNet training in minutes [J]. arXiv preprint, arXiv: 1709.05011, 2017

[8] Abadi M, Barham P, Chen Jianmin, et al. TensorFlow: A system for large-scale machine learning [C] //Proc of the 12th Symp on Operating Systems Design and Implementation (OSDI 16). Berkeley, CA: USENIX Association, 2016: 265-283

[9] Ketkar N. Deep Learning with Python: Introduction to Pytorch [M]. Berkeley, CA: Apress, 2017

[10] Jia Yangqing, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding [C] // Proc of the 22nd ACM Int Conf on Multimedia. New York: ACM, 2014: 675-678

[11] Gulli A, Pal S. Deep Learning with Keras [M]. Birmingham: Packt Publishing Ltd, 2017

[12] Seide F, Agarwal A. CNTK: Microsoft's open-source deep-learning toolkit [C] //Proc of the 22nd ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2016: 2135-2135

[13] Keskar N S, Mudigere D, Nocedal J, et al. On large-batch training for deep learning: Generalization gap and sharp minima [J]. arXiv preprint, arXiv:1609.04836, 2016

[14] Codreanu V, Podareanu D, Saletore V. Scale out for large minibatch SGD: Residual network training on ImageNet-1K with improved accuracy and reduced time to train [J]. arXiv preprint, arXiv:1711.04291, 2017

[15] Devarakonda A, Naumov M, Garland M. Adabatch: Adaptive batch sizes for training deep neural networks [J]. arXiv preprint, arXiv:1712.02029, 2017

[16] Goyal P, Dollár P, Girshick R, et al. Accurate, large minibatch SGD: Training ImageNet in 1 hour [J]. arXiv preprint, arXiv:1706.02677, 2017

[17] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift [J]. arXiv preprint, arXiv:1502.03167, 2015

[18] Hansen L K, Salamon P. Neural network ensembles [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1990, 12(10): 993-1001

[19] Rosenblatt F. The perceptron, a perceiving and recognizing automation project para [R]. New York: Cornell Aeronautical Laboratory, 1957

[20] Burges C, Shaked T, Renshaw E, et al. Learning to rank using gradient descent [C] //Proc of the 22nd Int Conf on Machine learning. New York: ACM, 2005: 89-96

[21] Bottou L. Large-scale machine learning with stochastic gradient descent [C] //Proc of COMPSTAT'2010. Berlin: Springer , 2010: 177-186

[22] Tarditi D, Puri S, Oglesby J. Accelerator: Using data parallelism to program GPUs for general-purpose uses [J]. ACM SIGPLAN Notices, 2006, 41(11): 325-335

[23] Jia Zhihao, Zaharia M, Aiken A. Beyond data and model parallelism for deep neural networks [J]. arXiv preprint, arXiv:1807.05358, 2018

[24] LeCun Y, Bengio Y, Hinton G. Deep learning [J]. Nature, 2015, 521(7553): 436-444

[25] Pfister G F. High Performance Mass Storage and Parallel I/O: An Introduction to the InfiniBand Architecture [M]. New York: Wiley Press, 2001: 617-632

[26] Birrittella M S, Debbage M, Huggahalli R, et al. Intel® Omni-path architecture: Enabling scalable, high performance fabrics [C] //Proc of the 23rd IEEE Annual Symp on High-Performance Interconnects. Piscataway, NJ: IEEE, 2015: 1-9

[27] Beck M, Kagan M. Performance evaluation of the RDMA over Ethernet (RoCE) standard in enterprise data centers infrastructure [C] //Proc of the 3rd Workshop on Data Center-Converged and Virtual Ethernet Switching of Int Teletraffic Congress. Piscataway, NJ: IEEE, 2011: 9-15

[28] NVIDIA.NVIDIA GPUDirect [EB/OL]. 2019[2019-04-25]. https://developer.nvidia.com/gpudirect

[29] Potluri S, Hamidouche K, Venkatesh A, et al. Efficient inter-node MPI communication using GPUDirect RDMA for InfiniBand clusters with NVIDIA GPUs [C] //Proc of the 42nd Int Conf on Parallel Processing. Piscataway, NJ: IEEE, 2013: 80-89

[30] Foley D, Danskin J. Ultra-performance Pascal GPU and NVLink interconnect [J]. IEEE Micro, 2017, 37(2): 7-17

[31] Li Mu, Andersen D G, Park J W, et al. Scaling distributed machine learning with the parameter server [C] //Proc of the 11th USENIX Symp on Operating Systems Design and Implementation (OSDI 14). Berkeley, CA: USENIX Association, 2014: 583-598

[32] Sergeev A, Del B M. Horovod: Fast and easy distributed deep learning in TensorFlow [J]. arXiv preprint, arXiv: 1802.05799, 2018

[33] Barker B. Message passing interface (MPI): High performance computing on stampede [CP/OL]. 2015[2019-10-20]. https://computing.llnl.gov/tutorials/mpi/

[34] Patarasuk P, Yuan Xin. Bandwidth optimal all-reduce algorithms for clusters of workstations [J]. Journal of Parallel and Distributed Computing, 2009, 69(2): 117-124

[35] Patarasuk P, Yuan Xin. Bandwidth efficient all-reduce operation on tree topologies [C] //Proc of 2007 IEEE Int Parallel and Distributed Processing Symp. Piscataway, NJ: IEEE, 2007: 1-8

[36] Faraj A. Performing an allreduce operation on a plurality of compute nodes of a parallel computer: US, 8375197 [P]. 2013-02-12

[37] Wesley Bland. MPI Tutorial [EB/OL].[2019-10-20]. http://mpitutorial.com/tutorials/mpi-reduce-and-allreduce/

[38] Ben-Nun T, Hoefler T. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis [J]. ACM Computing Surveys, 2019, 52(4): 65-65

[39] Shi Shaohuai, Wang Qiang, Chu Xiaowen, et al. A DAG model of synchronous stochastic gradient descent in distributed deep learning [C] //Proc of the 24th IEEE Int Conf on Parallel and Distributed Systems (ICPADS). Piscataway, NJ: IEEE, 2018: 425-432

[40] Gropp W, Thakur R, Lusk E. Using MPI-2: Advanced Features of the Message Passing Interface [M]. Cambridge, MA: MIT Press, 1999

[41] Andrew G. Bringing HPC techniques to deep learning [EB/OL]. 2017 [2018-04-21]. http://research. baidu. com/bringing-hpc-techniques-deep-learning

[42] NVIDIA. NVIDIA collective communications library(NCCL) [EB/OL]. 2019[2019-10-20]. https://developer.nvidia. com/nccl

[43] Facebook. Collective communications library with various primitives for multi-machine training (gloo) [EB/OL]. 2019 [2019-10-20]. https://github.com/facebookincubator/gloo

[44] Yang C T, Huang C L, Lin Cheng Fang. Hybrid CUDA, OpenMP, and MPI parallel programming on multicore GPU clusters [J]. Computer Physics Communications, 2011, 182(1): 266-269

[45] NVIDIA.NVIDIA NCCL 2.4 [EB/OL]. 2019[2019-10-20]. https://devblogs. nvidia. com/massively-scale-deep-learning-training-nccl-2-4/

[46] Thakur R, Rabenseifner R, Gropp W. Optimization of collective communication operations in MPICH [J]. The International Journal of High Performance Computing Applications, 2005, 19(1): 49-66

[47] Zadeh A, Chen Minghai, Poria S, et al. Tensor fusion network for multimodal sentiment analysis [J]. arXiv preprint, arXiv:1707.07250, 2017

[48] Horovod. Horovod Tensor Fusion [EB/OL]. 2019[2019-10-20]. https://github. com/horovod/horovod/blob/master/docs/tensor-fusion.rst

[49] Zhang Wei, Gupta S, Lian Xiangru, et al. Staleness-aware ASYNC-SGD for distributed deep learning [J]. arXiv preprint, arXiv:1511.05950, 2015

[50] Zhang Sixin, Choromanska A E, LeCun Y. Deep learning with elastic averaging SGD [C] //Proc of Advances in Neural Information Processing Systems. New York: Curran Associates, 2015: 685-693

[51] Jin P H, Yuan Qiaochu, Iandola F, et al. How to scale distributed deep learning? [J]. arXiv preprint, arXiv:1611.04581. 2016

[52] Alvarez R, Prabhavalkar R, Bakhtin A. On the efficient representation and execution of deep acoustic models [J]. arXiv preprint, arXiv:1607.04683, 2016

[53] Lim H, Andersen D G, Kaminsky M. 3LC: Lightweight and effective traffic compression for distributed machine learning [J]. arXiv preprint, arXiv:1802.07389. 2018

[54] Wen Wei, Xu Cong, Yan Feng, et al. TernGrad: Ternary gradients to reduce communication in distributed deep learning [C] //Proc of Advances in Neural Information Processing Systems. New York: Curran Associates, 2017: 1509-1519

[55] Sattler F, Wiedemann S, Müller K R, et al. Sparse binary compression: Towards distributed deep learning with minimal communication [J]. arXiv preprint, arXiv: 1805.08768, 2018

[56] Strom N. Scalable distributed DNN training using commodity GPU cloud computing [C] //Proc of the 16th Annual Conf of the Int Speech Communication Association. New York: ACM, 2015: 1488-1492

[57] Dryden N, Moon T, Jacobs S A, et al. Communication quantization for data-parallel training of deep neural networks [C] //Proc of the 2nd Workshop on Machine Learning in HPC Environments (MLHPC). Piscataway, NJ: IEEE, 2016: 1-8

[58] Aji A F, Heafield K. Sparse communication for distributed gradient descent [J]. arXiv preprint, arXiv: 1704.05021, 2017

[59] Ba J L, Kiros J R, Hinton G E. Layer normalization [J]. arXiv preprint, arXiv:1607.06450, 2016

[60] Lin Yujun, Han Song, Mao Huizi, et al. Deep gradient compression: Reducing the communication bandwidth for distributed training [J]. arXiv preprint, arXiv:1712.01887, 2017

[61] Wang Guanhua, Shivaram V, Amar P, et al. Blink: Fast and generic collectives for distributed ML [J]. arXiv preprint, arXiv:1910.04940, 2019

[62] Graham R L, Bureddy D, Lui P, et al. Scalable hierarchical aggregation protocol (SHArP): A hardware architecture for efficient data reduction [C] //Proc of the 1st Int Workshop on Communication Optimizations in HPC (COMHPC). Piscataway, NJ: IEEE, 2016: 1-10

[63] Laanait N, Romero J, Yin Junqi, et al. Exascale deep learning for scientific inverse problems [J]. arXiv preprint, arXiv:1909.11150, 2019

[64] Qassim H, Verma A, Feinzimer D. Compressed residual-VGG16 CNN model for big data places image recognition [C] //Proc of the 8th IEEE Annual Computing and Communication Workshop and Conf (CCWC). Piscataway, NJ: IEEE, 2018: 169-175

[65] Kurth T, Treichler S, Romero J, et al. Exascale deep learning for climate analytics [C] //Proc of the Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2018: 649-660

[66] Lee K. Introducing big basin: Our next-generation AI hardware [EB/OL]. 2017 [2019-10-20]. https://code.facebook.com/posts/1835166200089399/introducing-big-basin

[67] Akiba T, Suzuki S, Fukuda K. Extremely large minibatch SGD: Training ResNet-50 on ImageNet in 15 minutes [J]. arXiv preprint, arXiv:1711.04325, 2017

[68] Jia Xianyan, Song Shutao, He Wei, et al. Highly scalable deep learning training system with mixed-precision: Training ImageNet in four minutes [J]. arXiv preprint, arXiv:1807.11205, 2018

[69] Mikami H, Suganuma H, U-chupala P, et al. ImageNet/ResNet-50 training in 224 seconds [J]. arXiv preprint, arXiv:1811.05233, 2018

[70] You Yang, Zhang Zhao, Hsieh C J, et al. Speeding up ImageNet training on supercomputers [OL]. 2018[2019-10-20]. https://people. eecs. berkeley. edu/~ youyang/publications/sysml2018.pdf

[71] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge [J]. Nature, 2017, 550(7676): 354-359

Zhu Hongrui, born in 1994. PhD candidate. Student member of CCF. His main research interests include distributed deep learning, data center network, network simulator.

朱泓睿,1994 年生.博士研究生,CCF 学生会员.主要研究方向为分布式深度学习、数据中心网络、网络模拟器等.

Yuan Guojun, born in 1983. PhD, associate professor, master supervisor. Member of CCF. His main research interests include computer architecture, high performance computing, optical network, distributed deep learning and system interconnection.

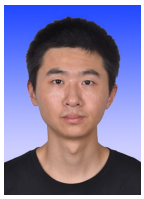
元国军,1983 年生.博士,副研究员,硕士生导师,CCF 会员.主要研究方向为计算机体系结构、高性能计算、光网络、分布式深度学习、系统互连等.

Yao Chengji, born in 1991. Master. His main research interests include deep learning framework, distributed training.

姚成吉,1991 年生.硕士.主要研究方向为深度学习框架、分布式训练等.

Tan Guangming, born in 1980. PhD, professor, PhD supervisor. Member of CCF. His main research interests include parallel computing, domain-specific architecture, big data.

谭光明,1980 年生.博士,研究员,博士生导师,CCF 会员.主要研究方向为并行计算、特定领域架构、大数据等.



Wang Zhan, born in 1986. PhD, associate professor. Member of CCF. His main research interests include high performance computing, the interconnection network of high performance computer, and system virtualization.

王　展,1986 年生.博士,副研究员,CCF 会员.主要研究方向为高性能计算、高性能计算机互连网络和系统虚拟化等.

Hu Zhongzhe, born in 1993. PhD candidate. Student member of CCF. His main research interests include high performance computing, parallel & distributed learning algorithms.

户忠哲,1993 年生.博士研究生,CCF 学生会员.主要研究方向为高性能计算、并行与分布式深度学习算法.

Zhang Xiaoyang, born in 1993. PhD candidate. Student member of CCF. His main research interests include deep learning performance optimization and acceleration, computer architecture, high performance computing.

张晓扬,1993 年生.博士研究生,CCF 学生会员.主要研究方向为深度学习性能优化与加速、计算机体系结构、高性能计算等.

An Xuejun, born in 1966. PhD, professor, PhD supervisor. Member of CCF. His main research interests include HPC architecture, HPC network, graph computing accelerator, SoC and system interconnection.

安学军,1966 年生.博士,正高级工程师,博士生导师,CCF 会员.主要研究方向为高性能计算机体系结构、高性能计算网络、图计算加速器、片上系统、系统互连等.