分布式训练系统及其优化算法综述

王恩东[3] 闫瑞栋[1),2),3)] 郭振华[3] 赵雅倩[2),3)]

[1)](山东海量信息技术研究院 济南 250101)

[2)](浪潮(北京)电子信息产业有限公司 北京 100875)

[3)](浪潮电子信息产业股份有限公司 济南 250101)

摘 要 人工智能利用各种优化技术从海量训练样本中学习关键特征或知识以提高解的质量,这对训练方法提出了更高要求.然而,传统单机训练无法满足存储与计算性能等方面的需求.因此,利用多个计算节点协同的分布式训练系统成为热点研究方向之一.本文首先阐述了单机训练面临的主要挑战.其次,分析了分布式训练系统亟需解决的三个关键问题.基于上述问题归纳了分布式训练系统的通用框架与四个核心组件.围绕各个组件涉及的技术,梳理了代表性研究成果.在此基础之上,总结了基于并行随机梯度下降算法的中心化与去中心化架构研究分支,并对各研究分支优化算法与应用进行综述.最后,提出了未来可能的研究方向.

关键词 分布式训练系统;(去)中心化架构;中心化架构算法;(异)同步算法;并行随机梯度下降;收敛速率

中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2024.00001

A Survey of Distributed Training System and Its Optimization Algorithms

WANG En-Dong[3] YAN Rui-Dong[1),2),3)] GUO Zhen-Hua[3] ZHAO Ya-Qian[2),3)]

[1)](*Shandong Massive Information Technology Research Institute*, *Jinan* 250101)

[2)](*Inspur (Beijing) Electronic Information Industry Co., Ltd*, *Beijing* 100875)

[3)](*Inspur Electronic Information Industry Co., Ltd*, *Jinan* 250101)

Abstract Artificial intelligence employs a variety of optimization techniques to learn key features or knowledge from massive samples to improve the quality of solutions, which puts forward higher requirements for training methods. However, traditional single-machine training cannot meet the requirements of storage and computing performance, especially since the size of datasets and models continue to increase in recent years. Therefore, a distributed training system with the cooperation of multiple computing nodes has become one of the hot topics in computation-intensive and storage-intensive applications such as deep learning. Firstly, this survey introduces the main challenges (e.g., dataset/model size, computing performance, storage capacity, system stability, and privacy protection) of single-machine training. Secondly, three key problems including partition, communication, and aggregation are proposed. To address these problems, a general framework of a distributed training system including four components (e.g., partition component, communication component, optimization component, aggregation component) is summarized. This paper pays attention to the core technologies in each component and reviews the existing representative research progress. Furthermore, this survey focuses on the parallel stochastic gradient descent algorithm and its variants, and categorizes them into the branches of centralized and decentralized

收稿日期:2022-08-31;在线发布日期:2023-03-14.本课题得到山东省自然科学基金项目(ZR2021QF073)资助.**王恩东**,硕士,教授,中国工程院院士,中国计算机学会(CCF)会士,主要研究领域为计算机体系结构设计、关键技术研究与实现、人工智能.E-mail:wangendong.hsslab@gmail.com.**闫瑞栋**(通信作者),博士,高级工程师,中国计算机学会(CCF)会员,主要研究方向为算法设计与分析、分布式系统优化.E-mail:yanruidong@inspur.com.**郭振华**,博士,高级工程师,中国计算机学会(CCF)会员,主要研究方向为计算机体系结构设计、人工智能.**赵雅倩**,博士,高级工程师,中国计算机学会(CCF)会员,主要研究方向为计算机体系结构设计、人工智能.

architecture respectively. In each branch, a line of synchronous and asynchronous optimization algorithms has been revisited. Furthermore, it introduces three representative applications which consist of heterogeneous environment training, federated learning, and large model training in distributed systems. Finally, the following two future research directions are proposed. For one thing, an efficient distributed second-order optimization algorithm will be designed, and for another, a theoretical analysis method in federated learning will be explored.

Keywords distributed training system; decentralized algorithms; centralized algorithms; (a) synchronous algorithms; parallel stochastic gradient descent; convergence rate

1 引 言

近年来,大数据、云计算、人工智能、高性能计算以及互联网技术取得了突破性进展,促进了计算机视觉、自然语言处理、自动驾驶等领域的发展,并对学术界和产业界产生了深远影响.例如,美国 OpenAI 公司近期研发了智能聊天机器人 ChatGPT,通过深入理解与学习用户语言模式实现对话,协助用户完成诗歌创作、邮件撰写及内容检索等任务.人工智能的成功得益于以下三个方面:一是各领域产生的文本、语音、图像、视频等海量数据,为人工智能算法和模型提供原材料;二是研究人员从海量样本中抽取特征与知识,运用理论优化分析等手段提出高精度、可扩展的各类算法与模型;三是诸如 TensorFlow[1]、PyTorch[2]、CNTK[3]以及 MXNet[4]等开源计算框架和平台,为算法部署与实施提供了统一接口以及算力支撑.

在以往样本稀缺的条件下,基于有限样本的单机训练成为主要解决策略.然而,随着数据密集型与计算密集型应用的持续增加,传统的单机训练策略在数据存储、任务实时处理等方面愈发捉襟见肘.主要面临如下挑战:

(1)大数据集及大模型挑战.深度学习的解决方案依赖于规模庞大的数据和模型.例如,在数据集方面,用于图像分类任务的 ImageNet[5]数据集包含 1400 多万张图像,涉及超过 20 000 个类别.在模型方面,微软的主题模型 LightLDA[6]参数规模超过 200 亿.研究人员不得不陷入基于大数据集训练大模型的困境.例如,OpenAI[7]基于 WeText 数据集训练参数规模超过 1750 亿的自回归语言模型 GPT-3[8].谷歌基于 C4[9]数据集训练参数规模为 1100 亿的 T5 模型.文献[10]表明使用单个 M40 GPU 在 ImageNet 数据上训练 ResNet-50 模型 100 个 epoch 至少花费 14 天.文献[11]采用单个 GPU 基于 ImageNet 训练模型至少需要 7 天.综上所述,采用传统单机训练策略无法满足大模型训练任务的要求,特别是训练周期过长对于时间敏感型任务难以适用.不同于单机训练策略,研究人员正朝着分布式训练策略的方向积极探索.例如,微软近期开源了名为 DeepSpeed[12]的分布式深度学习模型训练优化工具,提升模型训练效率.

(2)计算性能挑战.训练大规模深度学习模型如 M6[13]、GLM[14]、Switch Transformer[15]等势必对算力资源与设备提出了更高的要求.虽然 CPU、GPU[16]、TPU[17]、FPGA[18]以及 ASIC[19]等各种算力资源与技术库[20-21]相继出现并加速模型训练,但高性能计算仍面临挑战.截至目前,训练策略朝着纵向扩展与横向扩展两个方向演化.纵向扩展[22-23]是指提升单个算力设备的计算性能.例如,英伟达推出的 Titan V 和 Tesla V100 等产品使深度学习模型训练速度较普通 CPU 提升 47 倍.横向扩展则是在训练系统中添加额外的算力设备,通过设计并行或分布式算法将复杂训练任务分配到不同设备,提升系统整体性能.然而,横向扩展方法不可避免地涉及到子系统间资源配置、执行调度、设备间信息交换及聚合等问题.上述问题的解决对从业人员要求具备一定的技术积累.

(3)存储挑战.随着数据和模型规模的爆发式增长,单个设备有限存储容量无法满足存储需求,仅依赖单机训练策略是不可行的.因此,作为单机训练的扩展,分布式训练成为主流训练策略.它的核心思想是以数据或模型并行等方式将计算任务拆分成若干部分,并把拆分后各个部分分配至不同的计算节点进行处理,从而达到"分而治之"的目的.然而,分布式训练不可避免地涉及以下技术难题[24]:一是如何对数据或模型进行有效拆分;二是如何实现计算节点或子模型间的高效通信.上述问题的解决需同

时考虑数据集、模型及计算节点特性等多方面因素，相关研究有待进一步完善.

(4)系统稳定性挑战. 稳定性是存储密集型和计算密集型系统不可忽视的特性. 一般而言，稳定性是指计算机系统或存储子系统在部件、硬件或软件出现故障时仍能继续工作而不中断服务、不丢失数据或危及安全的能力. 单机训练过程中一旦仅有的设备发生故障，训练任务就会中止. 因此，单机训练的稳定性大打折扣. 相反，分布式训练方案依靠多个计算节点间的协作实现系统稳定性与可靠性. 目前，研究人员已提出各类稳定性方法[25]：① 冗余，即添加额外设备或进程使训练系统具备克服故障的能力. 例如，文献[26]提出了一种带备份计算节点的分布式训练方法，其优势在于以下两点：一是相较于无备份节点方法，该方法加快了训练速度从而缩短了训练时长；二是该方法提高可靠性的同时保证了算法精度. ② 负载均衡，即避免部分计算节点闲置而其它计算节点过载，从而优化训练系统响应和执行时长. ③ 检查点，即保存应用程序的多个快照状态以便在出现故障时可利用快照重新启动. 此外，训练系统稳定性研究仍然面临存储开销、质量测试及故障检测等挑战.

(5)隐私保护挑战. 隐私保护是分布式训练系统中的重要问题之一. 谷歌在 2016 年提出了联邦学习[27](Federated Learning)的概念用以解决安卓手机终端用户在本地更新模型的问题，保障终端数据及个人数据的隐私安全. 然而，隐私保护策略一定程度上会降低模型训练的精度，从而影响训练效率. 因此，构建高效可扩展的数据安全共享机制，实现隐私安全保护和模型效率的平衡，是当前业界亟需攻克的科学问题.

综上所述，本文是为深度学习等相关领域从业人员提供分布式训练系统及其优化算法研究进展的概述与总结. 尽管目前在分布式训练系统领域已有部分研究成果(如表 1 所示)，但它们在算法设计和理论分析层面关注较少. 因此，本文重点介绍分布式训练系统及其优化算法，梳理基于并行随机梯度下降的分布式优化算法研究成果，阐述算法设计与理论分析. 主要贡献包括以下四个方面：

(1)分析了单机训练方案所面临的数据与模型、存储、计算性能以及系统稳定性等挑战，引出分布式训练的必要性，并从算法理论优化分析角度梳理现有研究成果.

(2)总结了分布式训练系统的通用框架及其主要构成组件，分析了各组件中的技术难题.

(3)归纳总结了基于并行随机梯度下降的中心化和去中心化架构算法研究分支. 针对各研究分支，阐述了各类优化算法理论性质及应用.

(4)基于现有研究进展及挑战，提出了分布式训练系统未来可能的研究方向.

表 1　分布式训练系统相关综述论文汇总

文献	文章侧重点	理论分析
[28]	机器学习算法分析	√
[29]	机器学习算法、架构、生态	×
[30]	分布式机器学习训练算法	×
[11]	深度学习算法、分布式框架	×
[31]	深度学习算法、应用	×
[32]	深度学习开源框架	×
[33]	优化算法、应用	×
[34]	优化算法	×
[35]	通信优化算法分析	√
[36]	通信优化算法、系统架构	×

2　分布式训练系统框架

本节首先分析分布式训练系统需要解决的三个关键问题. 在此基础之上，提出分布式训练系统的通用框架及其主要构成组件. 其次，介绍分布式训练系统优化问题的数学表达形式.

2.1　分布式训练系统框架及关键问题

分布式训练系统是由多个计算节点构成的网络，并且各个计算节点可由一台或多台主机构成. 分布式训练系统尚无统一的定义，不同研究机构给出了各自的理解与解释. 例如，微软对分布式训练系统的定义为“训练模型的工作负载被分配到多个工作节点，并且这些工作节点通过并行的工作方式加速模型训练. 分布式训练不仅用于机器学习模型，而且适用于计算和时间密集型任务”①. 再如，亚马逊对分布式训练系统的定义为“分布式训练采用数据并行或模型并行的方式实现模型高效训练. 其中，数据并行是指原始数据集被拆分成多个子数据集并分配到不同计算节点，再由神经网络或机器学习算法处理，最后将各个局部计算结果聚合. 模型并行是指大模型首先被拆分成若干子模型并分配至相应计算节点，再由各计算节点进行处理，最后将子模型计算中间结果进行聚合”②.

由上述定义可知，分布式训练借鉴“分而治之”的策略. 首先，系统将数据或模型进行拆分并分配到

① https://docs.microsoft.com/en-us/azure/machine-learning
② https://docs.aws.amazon.com/sagemaker/latest/dg

不同计算节点.其次,各计算节点基于自身的局部数据或模型执行训练任务.最后,系统将局部训练中间结果进行聚合.综上所述,分布式训练系统需要考虑如下三个关键问题:

(1)划分问题,即如何有效地对数据集或模型进行拆分,并将拆分后的子数据集或子模型分配至适合的计算节点.

(2)通信问题,即为了确保模型训练的精确性与参数一致性,不同计算节点间在迭代过程中需要信息交互.因此,如何现实计算节点间的高效通信尤为重要.

(3)聚合问题,即如何对各计算节点产生的局部或中间结果进行有效聚合或融合,并输出分布式训练任务的最终解.

基于上述三个关键问题,本文提出分布式训练系统的通用框架及其组件(如图1所示).在图1中,分布式训练系统包括划分组件、优化组件、通信组件以及聚合组件.划分组件负责数据或模型的拆分任务,并将拆分后的子数据集或子模型部署到相应计算节点.优化组件提供各类优化算法供各计算节点调用.通信组件负责各计算节点间数据传输与信息同步.聚合组件负责将各计算节点产生的中间训练结果进行聚合,并输出训练任务的全局解.上述四个组件各司其职,协同完成训练任务.

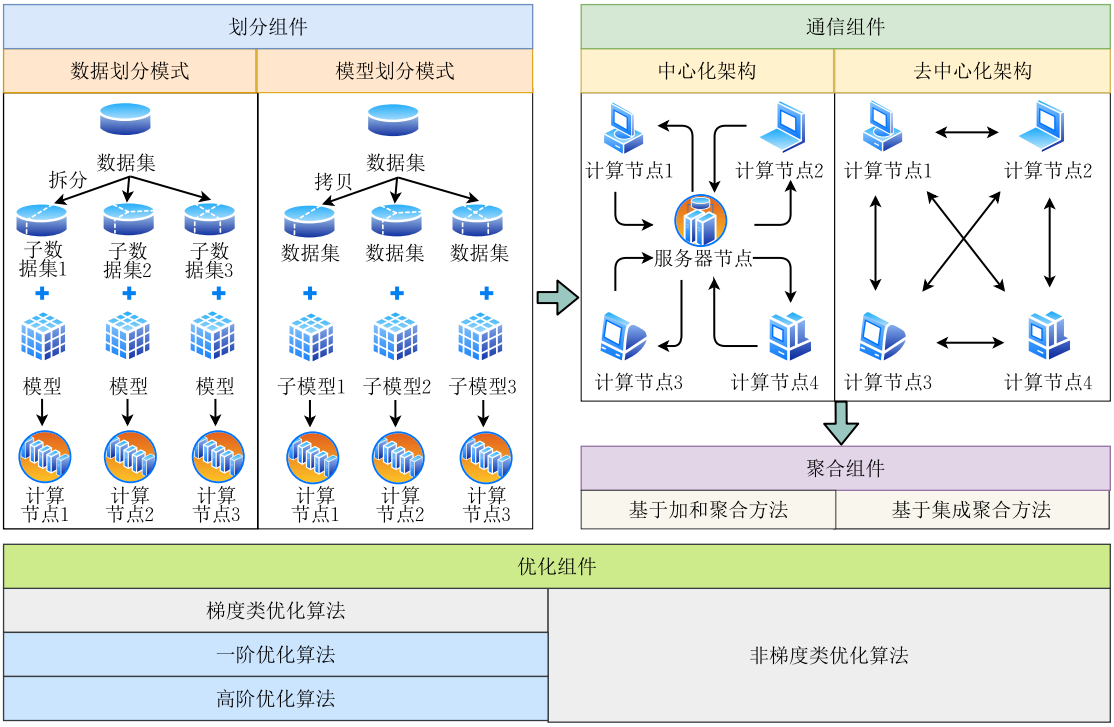


图1 分布式训练系统通用框架及其组件

2.2 分布式训练系统优化问题数学形式

机器学习或深度学习的各类复杂应用采用分布式训练技术求解时,可描述为如下形式的有限和优化问题:

$$f=\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}):=\frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}),$$

其中,局部函数 $f_i(\cdot)$ 定义如下:

$$f_i:=\mathbb{E}_{\xi_i\sim\mathcal{D}_i}[F_i(\boldsymbol{x};\xi_i)],$$

其中,i 表示计算节点标号,n 表示全体计算节点数目并且 $[n]=\{1,2,3,\cdots,n\}$,$\boldsymbol{x}$ 表示一个 d 维参数向量,$\xi_i\sim\mathcal{D}_i$ 表示计算节点 i 随机采样得到的样本 ξ_i 服从局部数据集 $\mathcal{D}_i$($\mathcal{D}_i\subseteq\mathcal{D}$),$\mathcal{D}$ 表示全体样本数据集并且 $|\mathcal{D}|=m$,$F_i(\boldsymbol{x};\xi_i)$ 表示计算节点 i 基于随机样本 ξ_i 在参数 $\boldsymbol{x}$ 处的损失函数,$\mathbb{E}[\cdot]$ 表示期望运算操作符.

3 分布式训练系统主要组件

本节介绍分布式训练系统划分组件、通信组件、优化组件以及聚合组件的工作原理,阐述组件间的内在逻辑关联,通过具体实例讲解各组件在分布式训练系统中的应用情况.

3.1 划分组件

划分组件主要包含数据并行、模型并行以及混合并行模式.下面分别介绍上述三种并行模式.

3.1.1 数据并行

训练数据可分为隐私数据与公开数据.基于隐

私数据的分布式训练通常采用联邦学习框架，实现训练数据输入、预测标签输出、模型参数等信息的隐私保护. 本小节侧重基于公开数据的分布式训练并行模式，因此仅讨论公开数据的并行模式. 具体而言，由于单个计算节点无法存储和处理规模日益增长的海量数据，因此需要将大规模数据集拆分成若干个子数据集后进行处理，使得计算节点执行训练任务变得可行. 换言之，数据集的拆分与数据并行息息相关. 根据文献[37]的定义，数据并行是并行化的一种形式，它通过多个并行计算节点细分数据集实现分割计算. 数据并行将数据分配至不同的并行计算节点，并且各计算节点执行相同的计算模型. 此外，数据并行模式按照不同的拆分策略可分为基于样本的数据并行与基于样本维度的数据并行[38].

(1) 基于样本的数据并行. 假定分布式训练系统数据集共包含 m 个数据样本和 n 个计算节点. 基于样本的数据并行主要包括有放回地随机采样及局部(全局)置乱采样两种方法.

有放回地随机采样方法将原始数据集作为采样空间，通过有放回地随机采样获取若干样本子集，并且各样本子集包含一定数目的样本. 例如，系统中各计算节点执行 m/n 次有放回地随机采样后获取上述样本构成的样本子集. 有放回地随机采样方法的优势在于保持样本子集数据分布与原始数据集分布一致性. 该方法的不足之处主要体现在以下两个方面：一是样本空间巨大导致采样成本较高；二是无法保证所有样本均被采样.

局部(全局)置乱采样将原始有序数据集作为采样空间，首先打乱数据集样本间的原始排列顺序，而后基于乱序的样本集进行采样并获取样本子集. 图 2 提供了置乱采样的一个示意图. 全局置乱采样包括以下三个主要步骤：

步骤 1. 将原始升序排列的 m 个样本随机打乱顺序.

步骤 2. 将乱序的样本集随机划分成 n 个样本子集.

步骤 3. 根据各计算节点存储容量为其分配相应的样本子集.

此外，各个计算节点的样本子集在训练过程中也可采用局部置乱的方式改变数据样本分布情况. 与有放回地随机采样相比，局部或全局置乱采样方法成本低并且能够充分利用全体数据样本，避免了一些关键样本未被采样的问题. 然而，该方法的不足之处在于各样本子集数据分布情况与原始数据集分布情况存在差异，使得后续算法理论分析变得相对困难.

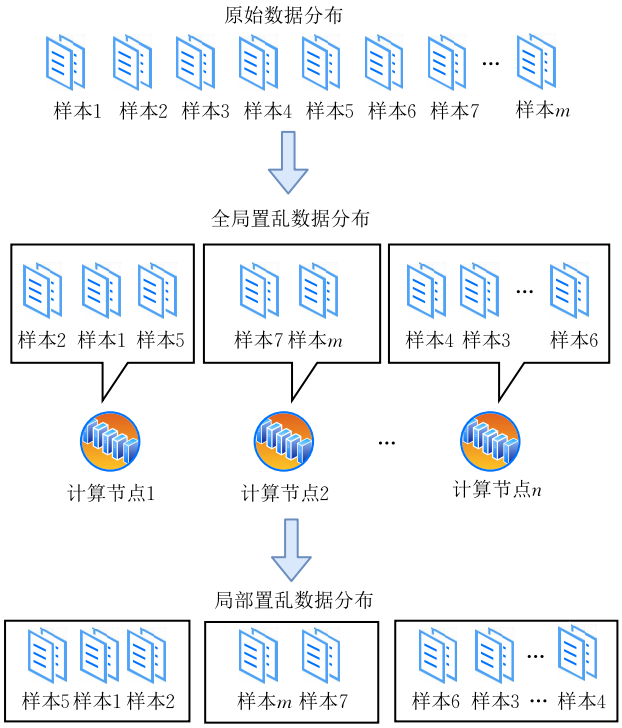


图 2　全局置乱及局部置乱方法

(2) 基于样本维度的数据并行. 假定数据集包含 m 个样本且各个样本具有 d 维属性或特征，分布式训练系统包含 n 个计算节点. 基于样本维度的数据并行则是从样本属性维度出发，将 m 个样本按照不同的属性进行拆分，并把拆分后的样本子集分配至相应的计算节点. 图 3 展示了基于样本维度的数据并行方法. 在图 3 中，各个 d 维样本按照属性维度划分后分配至 n 个计算节点. 例如，属性维度 1 分配至计算节点 1，属性维度 2 和属性维度 3 分配至计

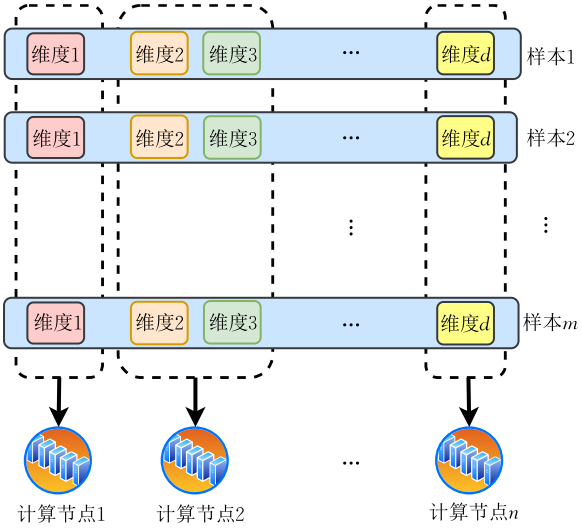


图 3　基于样本维度的数据划分

算节点 2,属性维度 d 分配至计算节点 n. 一般地,样本维度的数据并行与模型性质以及优化算法的耦合度较高. 如果目标函数线性可分并且计算一些维度的偏导数或梯度开销较小,则可以选择基于样本维度的数据并行对样本进行划分.

3.1.2 模型并行

如果训练任务模型过大且无法通过单机方式实现存储,则需要对模型进行拆分. 根据文献[35,39]的定义,模型并行基于特定的划分规则将原始模型拆分成若干子模型,并且将拆分后的子模型分配至不同的计算节点. 由于神经网络模型的特殊性,即神经网络模型的分层结构使其在应用模型并行方面具有显著优势. 神经网络的模型并行按照不同的拆分方式可以分为水平拆分、垂直拆分以及混合拆分.

图 4 分别提供了神经网络的水平拆分、垂直拆分以及混合拆分示意图. 图 4 的神经网络包含一个 4 层结构,分别为输入层、隐藏层 1、隐藏层 2 以及输出层(不同层使用不同颜色表示),箭头表示数据或参数传递方向. 在图 4(a)中,该神经网络模型被垂直拆分为两个不重叠的子模型,并且分别被分配至计算节点 1 和计算节点 2 之上. 通过观察不难发现,在垂直拆分方式下各个子模型均包含了神经网络的所有层,即子模型同时包含不同颜色的层次结构. 在图 4(b)中,神经网络模型被水平拆分为 3 个子模型,并且分别被分配至计算节点 1、计算节点 2 和计算节点 3 之上. 在该种拆分方式下各个子模型仅包含了神经网络的部分层,即子模型仅包含若干相同颜色的部分层次结构. 图 4(c)展示了垂直拆分与水平拆分相结合的混合拆分方式,其中虚线框内部分表示分配至同一计算节点的子模型,箭头表示数据或参数传递方向. 特别地,加粗箭头表示子模型间数据或参数通信过程. 在图 4(c)中,节点 1 和节点 2 构成的子模型与节点 3 和节点 4 构成的子模型可视为水平拆分. 节点 1 和节点 3 构成的子模型与节点 2 和节点 4 构成的子模型可视为垂直拆分.

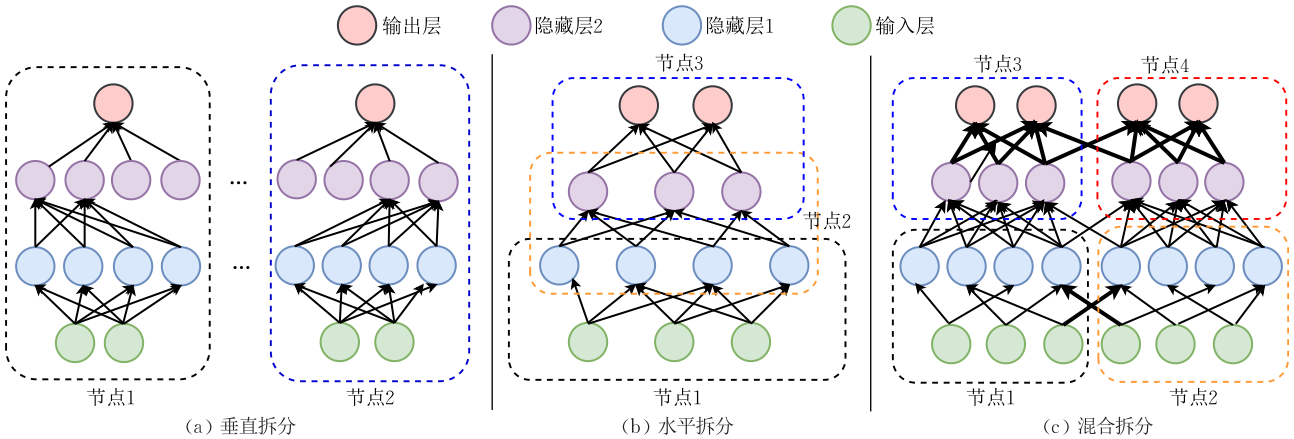


图 4 神经网络

神经网络的模型并行会产生如何选择拆分方式的问题,即在实际模型训练过程中应如何选择适合的神经网络拆分方式. 一般而言,神经网络的模型并行需要考虑神网络层数与各层神经元数目之间的关系. 具体而言,如果神经网络各层的神经元数目较小而层数却很大,此时应选择水平拆分方式. 相反地,如果神经网络各层的神经元数据较大而神经网络层数却很小,此时应选择垂直拆分方式更为合理. 此外,模型并行面临以下两个挑战:

(1)现有研究已经证明拆分神经网络模型属于 NP-完全问题[40]. 因此,设计具有良好的近似拆分理论算法十分困难.

(2)深层神经网络各层参数分布不均衡,各层之间计算依赖性较强,这些因素导致神经网络模型并行愈发困难. 虽然目前针对上述挑战研究人员提出了一些解决方法,但是大多采用工程实验手段,理论研究有待进一步完善.

3.1.3 混合并行

一方面,数据并行虽然将整个数据集拆分,但要求各计算节点保存完整的模型备份. 另一方面,模型并行将原始神经网络模型拆分,但各个计算节点仍需保存完整的数据集. 为了兼顾数据并行和模型并行的优势,研究人员提出了一种混合并行的模式. 顾名思义,混合并行就是同时将数据并行模式与模型并行模式结合起来,使其能够应用于更复杂的模型训练任务. 混合并行的优势在于同时利用了数据并行和模型并行,使得数据或模型的拆分方式灵活多样. 但混合并行也一定程度上引入额外的通信开销,因此降低通信开销是混合并行模式的重要目标. 典型的混合并行模式包括文献[41-42]. 文献[41]构建

了一个名为 SOYBEAN 的系统，该系统执行自动并行化并且将现有深度学习系统前端捕获的串行数据流图转化为最优的并行数据流图. 文献[42]提出了一种基于多 GPU 的卷积神经网络并行化训练算法. 该算法主要思想是利用了卷积层的数据并行性和全连接层的模型并行性，从而设计一种混合并行模式的训练方法. 此外，读者可参考混合并行的其它相关文献[43-44].

3.2　通信组件

分布式训练系统与传统单机训练的最主要差别在于利用多个计算节点间的协同合作加速完成训练任务，因此通信成为分布式训练系统不可或缺的环节. 然而，由于硬件设备、网络带宽和传输速率等因素的影响，分布式训练系统计算节点间的通信往往成为瓶颈，严重制约了训练性能. 在这种情况下，通信组件试图设计合理、高效的通信机制，减少通信开销. 通信机制不仅要考虑硬件系统层面的限制约束，还要兼顾软件算法层面的优化问题. 本小节将从通信内容、通信拓扑以及通信同步方式三个方面介绍分布式训练系统的通信组件.

3.2.1　通信内容

通信内容与并行模式有关. 在数据并行中，各个计算节点使用本地数据进行模型训练. 为了达到全局模型一致性的目的，各计算节点需要同其它计算节点进行通信以获得其它计算节点的局部模型参数，进而实现全局模型参数一致性. 区别于数据并行，模型并行模式中各计算节点使用相同的数据来训练不同的子模型. 例如，在神经网络模型训练过程中，一个计算节点的迭代必须依赖于其它节点的中间计算结果或输出，此时需要进行通信才能获得其它节点训练的中间结果. 总而言之，数据并行模式的通信内容主要包括模型参数及其更新，而模型并行模式的通信内容则是各个计算节点产生的中间结果.

为降低分布式系统通信开销，减少系统节点间的数据通信量，研究学者针对通信内容开展了量化压缩、稀疏化压缩、低秩分解等研究工作[45-47].

量化压缩[48]源自谷歌研发团队，通过采用低比特整数类型替换 32 比特浮点数类型，从而减少参数或梯度的通信量，代表性算法包括 signSGD[49]以及 1-bit SGD[50]. signSGD 方法将梯度向量中负元素量化为-1，其它元素量化为 1. 1-bit SGD 方法则把 32 比特的浮点数压缩为 1 比特，进而压缩了 32 倍的通信量. 压缩算法虽然能够显著减少通信量，但它们通常会引起模型精度的损失. 为了解决精度损失问题，文献[51]设计了一种带有误差反馈的压缩算法，尽可能地降低压缩算法对精度的影响. 此外，研究学者近期将低比特压缩技术与自适应算法相结合，提出了 1-bit Adam[52]、1-bit LAMB[53]以及 0/1 Adam[54]等算法. 1-bit Adam 算法包括热身阶段和压缩阶段. 在热身段中，根据预先设定的迭代次数执行原始 Adam 算法[55]，直到方差项趋于稳定. 在压缩阶段，保持方差项不变并对动量项实施带有补偿措施的压缩操作，从而加速模型训练. 1-bit LAMB 算法则是在压缩阶段为神经网络的不同层设置不同的学习率. 0/1 Adam 算法在 1-bit Adam 算法框架之上，根据方差是否稳定来选用 1-bit All-Reduce 操作或 All-Reduce 操作.

注意到神经网络模型的参数更新通常是不均衡的. 换言之，模型训练迭代过程中，并非所有参数更新情况相同，有些参数更新多而有些参数更新少. 稀疏化压缩尽可能减少不更新的参数传输并且选择重要参数进行传输，从而降低通信量的同时保障精度，代表性方法包括 Random-k[56]、Sparse Gradient[57]等. Random-k 算法随机选择一个 d 维向量中的 k 个元素并将这 k 个元素置为 1，其余 $d-k$ 个元素置为 0. 为保证压缩的无偏性，压缩后的向量需乘以一个缩放因子 d/k. 在 Sparse Gradient 中，各计算节点 i 通过 $u_t^i=u_{t-1}^i+\epsilon g_t^i$（$\epsilon>0$，$g_t^i$表示节点 i 的局部梯度）维护局部梯度，累计梯度计算方式为 $v_t^i=\beta_1 u_{t-1}^i+u_t^i$，其中，$\beta_1>0$. 只有当梯度 v 中第 j 个元素 $v[j]<-\theta$ 或者 $v[j]>\theta$（θ 表示一个用户定义的阈值），$v[j]$才会被通信. 此外，文献[58]将 Random-k 等多种稀疏化方法应用于二阶梯度信息的压缩，并提出了基于高效通信机制的近似海森矩阵重构算法.

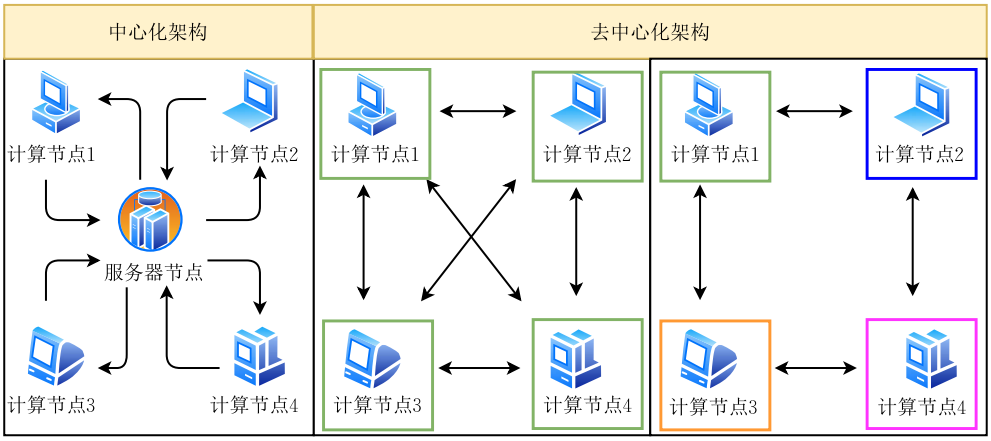
低秩分解重在探索参数矩阵的低秩结构，通过各类矩阵分解方法将原始大规模参数矩阵分解为多个规模较小的矩阵乘积[59]. 通信过程中实际传输的是分解后的小矩阵，传输完成之后再重新恢复为原始矩阵. 例如，一个矩阵 $\boldsymbol{A}\in\mathbb{R}^{m\times l}$可以分解为两个低秩矩阵 $\boldsymbol{A}_1\in\mathbb{R}^{m\times r}$和 $\boldsymbol{A}_2\in\mathbb{R}^{r\times l}$的乘积，其中，$r\ll\min\{m,l\}$.

值得注意的是采用量化压缩、稀疏化压缩以及低秩分解等方法，虽然可以降低通信量，但是也会引起额外的压缩与解压缩的开销. 因此，在实际训练过程中，需权衡上述压缩方法引入的额外开销与低通信量带来的收益，以便获得更好的性能.

3.2.2　通信拓扑

分布式训练系统计算节点间的协同合作依赖于通信. 例如，在训练一个神经网络任务中，不同的层被部署到不同的计算节点，前向或后向传播跨层传

递参数或梯度信息描述了各计算节点之间的通信过程.然而,不同的分布式系统架构产生了不同的通信方式,即分布式训练网络拓扑架构决定了通信方式.一般而言,分布式训练系统的通信拓扑架构是指各个计算节点之间的连接方式,分别包括物理拓扑和逻辑拓扑.物理拓扑主要包括 Fat-Tree 和 BCube[60]等在内的多种拓扑.文献[60]从理论和实践的角度讨论了上述两种物理拓扑对分布式系统训练性能的影响.逻辑拓扑包括中心化架构和去中心化架构(如图 5 所示).本文侧重分布式训练系统的逻辑拓扑.



(a) 参数服务器架构 (b) All-Reduce架构 (c) Gossip架构

图 5 分布式训练系统通信拓扑

(1)中心化架构.中心化架构的分布式训练系统是具有一个中心主节点来协调各个计算节点的架构.典型实例是参数服务器(Parameter Server,PS)架构[39,61-62].在参数服务器架构中存在两种角色:计算节点(Worker)和服务器节点(Server).计算节点主要负责如下操作:①基于其局部数据样本完成本地训练任务;②通过客户端接口与服务器节点进行通信,即从服务器节点处下载最新的全局模型参数,并将其自身的局部参数上传到服务器节点处.服务器节点作为参数服务器架构的核心部件主要完成如下操作:①接收并聚合各个计算节点发送来的局部梯度或模型参数等信息;②更新全局模型参数并返回至各个计算节点处.另外,需要指出的是参数服务器架构在计算节点和服务器节点之间逻辑上采用基于二部图的通信拓扑.换言之,通信只发生在服务器节点与计算节点之间,而计算节点之间不存在直接通信.

图 5(a)为参数服务器的架构示意图,包括 4 个计算节点和 1 个参数服务器节点.各个计算节点与参数服务器节点双向连接,箭头表示数据传递方向.然而,计算节点之间并无直连.在该架构中,各个计算节点均可使用服务器节点提供的参数独立地开展训练任务.具体而言,各个计算节点通过以下两种操作与服务器节点进行通信:一是下载(PULL)操作,即计算节点从服务器节点处获取全局参数;二是上传(PUSH)操作,即计算节点向服务器节点发送本地参数.为了便于说明,本文首先假设分布式训练系统是以数据并行模型执行的.然后,以文献[61]为例,详细讨论基于参数服务器架构的关键流程及不同角色在该架构中的分工.

文献[61]提出了一种基于参数服务器架构的分布式训练框架,制定了计算节点和服务器节点的工作流程.计算节点与服务器节点的工作流程涉及 *LoadData*()、*WorkerIterate*()以及 *ServerItearte*()函数(如图 6(b)所示).*LoadData*()函数完成计算节点的采样过程,各计算节点并发地从数据集中随机挑选一定数目样本构成样本子集.*WorkerIterate*()函数负责计算节点的梯度计算以及计算节点与服务器节点间的通信.*ServerItearte*()函数完成服务器节点对所有计算节点的局部信息的聚合,并对全局参数执行更新操作.

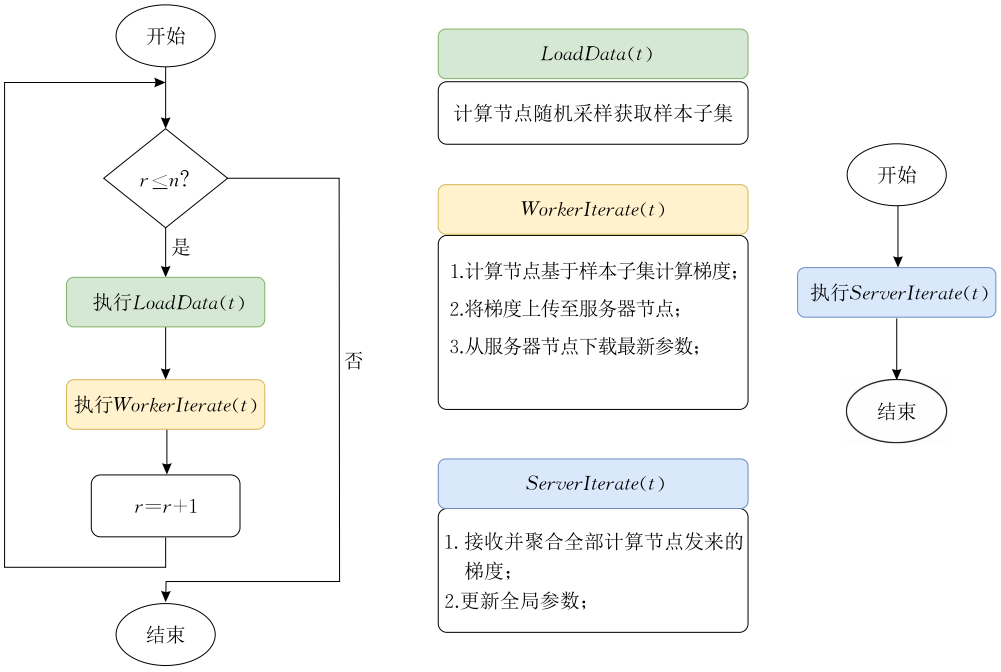
计算节点的工作流程(如图 6(a)所示)主要包括以下三个步骤:

步骤 1. 判断循环迭代条件.如果满足判断条件,则执行步骤 2.否则,循环结束.

步骤 2. 执行 *LoadData*()函数获取随机样本并生成样本子集.

步骤 3. 首先执行 *WorkerIterate*()函数并计算随机样本梯度,其次计算节点将局部梯度信息上传至服务器节点,最后并从服务器节点下载最新的全局模型参数.

服务器节点的工作流程(如图 6(c)所示)包括循环迭代执行 *ServerIterate*()函数,接收并聚合所有计算节点发送来的局部梯度等信息,完成全局模型的参数更新.



(a) 计算节点工作流程　　(b) 三个关键函数　　(c) 服务器节点工作流程

图 6　参数服务器架构工作流程

讨论:一方面,中心化架构训练系统具有设计简单、计算开销低、资源调度灵活及易于扩展等优点.此外,中心化架构的分布式训练系统支持各种异步更新策略(下一小节中介绍).另一方面,中心化架构训练系统的瓶颈主要体现在服务器节点的通信拥塞问题.这是因为各个计算节点均与服务器节点进行通信,大量计算节点会占用过多的通信带宽导致拥塞.目前,中心化架构的分布式训练系统被广泛应用于各领域,主要有 Parameter Server[63]、Petuum[64]、DistBelief[39]、DMTK/Multiverso[65]、GeePS[66]、SparkNet[67]、Project Adam[68]以及 Poesidon[69]等.

(2)去中心化架构.为了缓解中心化架构的通信拥塞问题,研究人员提出了去中心化架并设计诸如全规约(All-Reduce)和 Gossip 等复杂通信机制.All-Reduce 机制为了确保模型参数的一致性,要求各个计算节点与其它所有计算节点进行通信,并以广播的方式将本地信息传递给其它计算节点.各计算节点以上述方式获取所有计算节点的信息,从而实现全局信息同步.与 All-Reduce 机制不同,Gossip 机制中的各个计算节点仅与其邻居节点进行通信,进而提升通信效率.下面介绍上述两种去中心化通信机制.

All-Reduce 机制.系统内计算节点间通过全规约(All-Reduce)操作[70]实现信息交换,全体计算节点在无中心服务器节点的情况下进行通信.全规约操作包括规约-分发(Reduce-Scatter)步骤以及全聚合(All-Gather)步骤.在规约-分发步骤中,首先根据计算节点的数量将梯度及参数等信息分割成若干个大小相同的数据块(Chunk).然后,各个计算节点将各自数据块发送给其它计算节点,同时接收其它计算节点发来的数据块.在全聚合步骤中,各个计算节点替换并发送完整的数据块至其它计算节点.依赖于规约-分发以及全聚合两个步骤,所有计算节点实现参数同步并保持全局模型参数一致性(如图 5(b)所示,本文使用相同的颜色表示计算节点之间参数的全局一致性).

与参数服务器架构相比,全规约机制放弃了中心服务器节点,规避了服务器节点的通信拥塞问题,实现了去中心化,已广泛应用于各类分布式训练系统.然而,全规约机制也存在以下两点不足:一是由于缺乏中心服务器节点的全局统一调度与管理,需要引入大量的计算节点间的协调策略,从而导致通信开销仍是挑战.二是全规约机制不支持异步通信,从而一定程度上降低了训练系统信息同步方式的灵活性.

由于拓扑架构在通信过程中发挥至关重要的作用,因此本文关注全规约机制中计算节点间的拓扑结构.以全规约机制为例,由于各个计算节点均需与其余计算节点交换信息,因此一个由 n 个计算节点构成的完全图网络通信复杂度大约为 $O(n^2)$.随着计算节点数量的增加,通信开销增长巨大.为了降低通信开销,研究人员提出了一系列改进方法,包括基

于环状拓扑的 Ring-All-Reduce 机制、树形拓扑及蝴蝶拓扑等.

在众多改进方法中,Ring-All-Reduce 无疑是最具影响力的架构之一. 常见的主流深度学习通信框架,如 NCCL、Horovod[71]、Gloo、GPU-MPI[72] 均采用了环状拓扑(如图 7 所示). 在该架构中,各计算节点在逻辑上排列成一个环形拓扑,各个计算节点连接有一个左邻居节点和一个右邻居节点,并且各个计算节点只允许与这两个邻居进行通信. 具体而言,各个计算节点只向右邻居发送信息并且从左邻居接收信息. 类似于全规约机制,环状拓扑全规约机制同样包括分发-规约(Reduce-Scatter)步骤和全聚合(All-Gather)步骤. 下面分析环状拓扑全规约机制的通信开销.

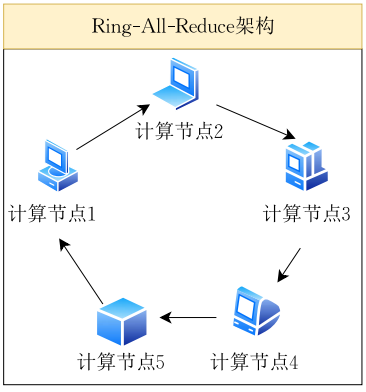


图 7 Ring-All-Reduce 架构

讨论:假定 τ 表示计算节点之间通信延迟,D 表示待通信数据块大小,B 表示计算节点间通信带宽,C 表示处理一字节数据计算耗时,n 表示计算节点数目. 一方面,环状拓扑全规约机制的规约-分发步骤需要执行 $n-1$ 次迭代,各次迭代的通信耗时为 $\tau+D/(nB)$,并且计算耗时为 $(D/n)\times C$. 因此,规约-分发阶段总消耗时长为

$$(n-1)\times[\tau+D/(nB)]+(n-1)\times[(D/n)\times C].$$

另一方面,全聚合步骤需执行 $n-1$ 次迭代,各次迭代的通信耗时为 $\tau+D/(nB)$ 并且不存在计算耗时. 因此,全聚合耗时为 $(n-1)\times[\tau+D/(nB)]$.

综上所述,环状拓扑全规约机制的总消耗时长为

$$2(n-1)\times[\tau+D/(nB)]+(n-1)\times[(D/n)\times C].$$

基于上述分析,环状拓扑全规约机制的优势在于充分利用了网络带宽. 但是,随着计算节点数目的不断增加,逻辑环长度也会相应增加,从而加剧通信延迟. 为了改进上述问题,腾讯研发团队提出了分层的环状拓扑全规约机制[73]. 首先,该机制将计算节点划分为若干个分组. 然后,依次采用分组内规约(Reduce)、分组间全规约(All-Reduce)、分组内广播(Broadcast)等步骤实现计算节点间通信. 分层机制充分利用了分组内节点间的高带宽,同时弱化分组间低网络带宽的影响. 此后,索尼公司研发团队提出了 2D-Torus 算法[74]、谷歌提出了 2D-Mesh 算法[75]. 表 2 列举了常见的全规约算法及其改进算法.

表 2 基于 All-Reduce 机制的通信架构汇总

通信架构或算法名称	算法总耗时	算法特点
Reduce+Broadcast[76]	$2(\tau+D/B)+nDC$	中心节点规约全体计算节点发送的信息,并以广播的方式发送至各个计算节点
Recursive halving and Doubling[76]	$2(\log_2 n)(\tau+D/B+DC)$	改进 Reduce+Broadcast 算法
Butterfly[77]	$(\log_2 n)(\tau+D/B+DC)$	改进 Recursive halving and Doubling 算法
Ring-All-Reduce[78]	$2(n-1)[\tau+D/(nB)]+(n-1)[(D/n)C]$	充分利用网络带宽,计算节点数目增加导致算法耗时显著增加
分层 Ring-All-Reduce[79]	$(m-1)[(D/m)C]+2(\tau+D/B)+nDC+2(m-1)[\tau+D/(mB)]$	利用分组内高带宽,弱化分组间低网络带宽的影响
2D-Torus[74]	$2(s-1)[\tau+D(sB)+(s-1)(D/s)]+2(m+1)[\tau+D(mB)]+(m-1)[(D/m)C]$	分组内规约-分发,分组间全规约,分组内全聚合

Gossip 机制. Gossip 机制是一种用于解决分布式平均问题的方法[80-81]并且不要求保持全局模型一致性(如图 5(c)所示,不同颜色表示计算节点不同的局部模型). 该机制对性能较差的计算节点具有一定的容错性. 与全规约机制相比,该机制中各计算节点仅与邻居节点进行信息交互,从而加快通信. 另外,为保证训练精度,该机制的挑战是如何保持模型参数的一致性,即“共识问题”(Consensus Problem)[82-83].

3.2.3 通信同步方式

分布式系统的通信同步方式直接决定了模型参数及梯度等信息交互的过程,从而影响分布式算法的收敛性质. 一般地,通信同步方式主要包括同步算法和异步算法.

(1)同步算法. 在同步算法中,如果分布式训练系统中的一个计算节点完成当前轮次迭代训练后,必须等待其它计算节点完成它们各自当前轮次迭代训练之后,所有计算节点才能共同处理下一轮次迭

代训练任务. 例如，典型的同步算法是整体同步并行(Bulk Synchronous Parallel，BSP)算法[84]. 具体而言，在 BSP 算法中，当一个计算节点完成当前迭代训练任务后，需要通过各种通信拓扑逻辑与其它计算节点同步模型参数或梯度等信息. 然后，全部计算节点以相同的“起点”进入下一轮次迭代训练过程. 为了保证以相同的“起点”进行训练，该算法引入了一个全局同步障碍(Synchronization Barrier). 同步障碍要求那些处理能力较强且迭代速度快的计算节点在同步障碍处强制停止，并等待其它处理能力较弱且迭代速度慢的计算节点完成其当前轮次迭代训练任务后，系统才会执行下一轮次迭代训练任务.

同步算法不仅可应用于全规约架构，而且可应用于参数服务器架构. 全规约架构中的同步算法要求所有计算节点以相同的迭代训练速度执行 All-Reduce 操作并获取全局模型参数，从而维护模型一致性. 参数服务器架构中的同步算法则要求所有计算节点首先将局部参数或梯度发送至服务器节点，再由服务器节点聚合并更新全局模型，最后将全局参数返回至各计算节点. 另外，在使用整体同步并行算法实现基于数据并行模式的分布式训练时，需考虑将通信同步方式与特定的优化技术相结合. 例如，将同步随机梯度下降算法[85]与模型平均算法[86]相结合的训练方式.

讨论：一方面，同步算法的优点是维护计算节点间模型参数的全局一致性，确保分布式算法收敛性与精确性. 另一方面，同步算法的缺点是容易产生“拖累者”问题[87]. 拖累者问题是指分布式训练系统采用同步通信时性能取决于迭代速度最慢的计算节点，若系统中计算节点之间性能差异较大，同步算法很容易产生速度较快的计算节点等待速度较慢的计算节点的现象. 随着计算节点数目的增加，该问题愈发严重.

(2) 异步算法. 在异步算法中，当系统中的一个计算节点完成当前轮次迭代训练后，它可以继续执行下一轮次迭代训练而无需等待其它计算节点. 异步算法可进一步细分为跨计算节点的多机异步通信和单计算节点内的多线程异步通信.

跨计算节点的多机异步通信. 如果将一台机器视为一个计算节点，则多机通信意味着信息交互是跨计算节点的. 为了便于说明，本文以参数服务器架构为例介绍跨节点多机异步通信的工作原理. 在跨节点多机异步训练过程中，由于各个计算节点仅与服务器节点通信，因而计算节点可按照自身迭代速度独立执行本地训练与参数更新. 具体而言，当一个计算节点完成当前轮次梯度计算后，它将局部梯度发送至服务器并从服务器处获取当前全局参数. 之后，继续进行下一轮次迭代而无需等待其它计算节点. 异步通信虽然避免了计算节点间不必要的等待，但会导致梯度过时. 换言之，计算节点之间由于迭代训练速度的不同产生的新旧梯度会导致模型收敛缓慢甚至不收敛. 例如，在相同时间内，计算节点 A 执行 100 次迭代训练，而计算节点 B 只迭代了 1 次. 如果将计算节点 B 产生的旧梯度发送至服务器节点并且服务器节点使用该梯度更新全局参数，此时计算节点 A 使用该陈旧的全局参数会减缓分布式训练整体的收敛速度.

单计算节点内的多线程通信. 由于内存访问的速度远远超过跨计算节点的网络传输速度，因而单机多线程异步通信适用于规模较小的数据集. 在这种通信方式中，多个线程同时访问存储在共享内存中的模型参数可能会导致冲突. 因此，如何避免冲突是单节点内多线程通信的关键. 当前，研究人员根据不同的应用场景提出了有锁和无锁的并行训练算法. 例如，Hogwild! 算法[88]利用稀疏性设计了无锁的通信方式，提高吞吐量的同时尽可能地避免冲突.

(3) 同步算法与异步算法的平衡. 基于上述分析和讨论，同步算法和异步算法各有利弊. 迭代速度较慢的计算节点使用同步算法会产生拖累者问题，而异步算法则会导致计算节点间梯度过时，这些问题都会直接影响系统性能. 为了解决上述问题，研究人员提出了一种兼顾同步和异步通信的延时同步并行(Stale Synchronous Parallel，SSP)算法[89]. 该算法的主要思想是控制迭代速度最快与最慢计算节点之间的迭代间隔，算法要求两者迭代间隔不能超过预先设定的阈值. 具体而言，只要任意两个计算节点间的迭代间隔不超过该阈值，各个计算节点独立开展训练且不与其它计算节点通信. 一旦任意两个计算节点的迭代间隔大于该阈值则会触发算法的等待机制. 也就是说，算法会迫使迭代速度较快的计算节点停止工作并且等待其它迭代较慢的计算节点，从而避免梯度过时问题的产生. 此外，延时同步并行算法的难点在于如何自动选择一个恰当的阈值. 一方面，较小的阈值容易造成计算节点频繁的等待与资源闲置. 另一方面，较大的阈值容易陷入梯度过时的困境. 综上所述，设计一种动态自适应的阈值设定方法尤为重要.

3.3　优化组件

3.3.1　术语

优化组件为分布式训练系统提供各类优化算法. 本小节首先介绍一些必要的术语，而后将优化组

件总结为梯度类优化算法以及非梯度类优化算法两类.为方便后续讨论,本文给出如下定义.

定义 1. 凸函数.对于任意向量 $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^d$,若函数 $f:\mathbb{R}^d\to\mathbb{R}$ 满足如下条件:

$$f(\boldsymbol{x})-f(\boldsymbol{y})\geq\nabla f(\boldsymbol{y})^{\top}(\boldsymbol{x}-\boldsymbol{y}),$$

其中,$\nabla f(\boldsymbol{y})$表示函数在参数 $\boldsymbol{y}$ 处梯度并且$\top$表示转置运算操作,则称函数 f 是凸函数.

凸函数的性质在于局部最优即为全局最优,因而被广泛应用于机器学习和深度学习的各种场景.作为凸函数的特例,强凸函数的定义如下.

定义 2. α-强凸函数.对于任意向量$\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^d$,若存在一个常数 $\alpha>0$ 使得函数 $f:\mathbb{R}^d\to\mathbb{R}$ 满足如下条件:

$$f(\boldsymbol{x})-f(\boldsymbol{y})\geq\nabla f(\boldsymbol{y})^{\top}(\boldsymbol{x}-\boldsymbol{y})+\frac{\alpha}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2,$$

其中,$\nabla f(\boldsymbol{y})$表示函数在参数 $\boldsymbol{y}$ 处梯度,$\|\cdot\|$表示向量 ℓ_2 范数,则称 f 是 α-强凸函数.

收敛性分析是客观评价优化算法收敛速率的重要工具.因此,本文给出如下的收敛速率定义.

定义 3. 收敛速率.令 f^* 和 f^t 分别表示最优值和算法在第 t 次迭代后返回值,则收敛速率定义为

$$\lim_{t\to\infty}\frac{|f^{t+1}-f^*|}{|f^t-f^*|^q}=\boldsymbol{R},$$

当 $t\to\infty$,不同的 q 影响收敛速率 $\boldsymbol{R}$.

(1)当 $q=0$ 且 $\boldsymbol{R}$ 收敛至 0,表明算法具有次线性收敛速率;

(2)当 $q=1$ 且 $\boldsymbol{R}$ 收敛至 $\aleph$ 且 $\aleph\in(0,1]$,表明算法具有线性收敛速率;

(3)当 $q>1$ 且 $\boldsymbol{R}$ 收敛至 $\aleph$ 且 $\aleph\in(0,1]$,表明算法具有超线性收敛速率.

定义 4. Lipschitz-连续.对于任意向量 $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^d$,若存在一个常数 $\boldsymbol{L}>0$ 使得函数 $f:\mathbb{R}^d\to\mathbb{R}$ 满足如下条件:

$$\|f(\boldsymbol{x})-f(\boldsymbol{y})\|\leq\boldsymbol{L}\|\boldsymbol{x}-\boldsymbol{y}\|,$$

则称函数 f 是 Lipschitz 连续.

定义 5. Lipschitz-连续梯度.对于任意向量 $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^d$,若存在一个常数 $\boldsymbol{L}>0$ 使得函数 $f:\mathbb{R}^d\to\mathbb{R}$ 满足如下条件:

$$\|\nabla f(\boldsymbol{x})-\nabla f(\boldsymbol{y})\|\leq\boldsymbol{L}\|\boldsymbol{x}-\boldsymbol{y}\|,$$

则称函数 f 具有 Lipschitz 连续梯度.

定义 6. 无偏梯度.对于任意向量 $\boldsymbol{x}\in\mathbb{R}^d$ 并且 $i\in[n]$,函数 f_i 基于随机样本 $\xi\sim\mathcal{D}_i$ 在参数向量 $\boldsymbol{x}$ 处的梯度值为 $\nabla f_i(\boldsymbol{x};\xi)$.如果该梯度的期望值等于全梯度 $\nabla f(\boldsymbol{x})$,即

$$\mathbb{E}_{\xi\sim\mathcal{D}_i}[\nabla f_i(\boldsymbol{x};\xi)]=\nabla f(\boldsymbol{x}),$$

称 $\nabla f_i(\boldsymbol{x};\xi)$ 为无偏随机梯度.

3.3.2 优化算法分类

常见的优化算法分为梯度类优化算法以及非梯度类优化算法.在梯度类优化算中,又可进一步细分为一阶优化方法和高阶优化方法.然而,一些目标函数未必存在梯度或导数,因此存在一些非梯度类优化算法.本文重点关注梯度类的一阶优化算法与二阶优化算法.

常见的一阶算法包括随机梯度下降(Stochastic Gradient Descent,SGD)算法[90]、Adam 算法[91]以及 LAMB 算法[92]等.SGD 算法在各次迭代中使用单样本随机梯度代替全梯度,从而有效降低计算开销.然而,SGD 算法的随机梯度会引入额外的方差,导致算法收敛速率放缓.此外,研究学者提出了基于自适应学习率的 Adam 算法,该算法同时考虑了梯度信息的一阶矩与二阶矩.近期,文献[92]基于权重与梯度之比,为神经网络的不同层设置不同的学习率,并提出了 LAMB 算法,使得该算法可应用于大规模神经网络模型训练任务.

二阶算法主要包括 Newton 算法[93]、自然梯度下降(Natural Gradient Descent,NGD)算法[94]及其近似算法.Newton 算法在各次迭代中要求计算一个海森矩阵.因此,相较于一阶算法而言,Newton 算法具有更快的理论收敛速率.然而,由于计算海森矩阵及其逆矩阵的开销过大,导致二阶算法在实际训练中的难以适用.NGD 算法则利用 Fisher 信息矩阵实现对海森矩阵的近似,以降低计算开销.但在大多情况下,Fisher 信息矩阵的计算代价仍然较高.为此,研究学者提出了基于 Kronecker-Factored Approximate Curvature(K-FAC)的方法[95].K-FAC 算法从以下两个方面近似 Fisher 信息矩阵.一是近似对角矩阵.神经网络的各层对应于一个小规模的矩阵,并且这些小规模矩阵按照顺序组成 Fisher 信息矩阵的近似对角矩阵.二是各个小规模矩阵可以利用 Kronecker 乘积性质进一步分解为两个规模更小的矩阵乘积.通过上述两方面的近似,降低二阶算法的计算开销,从而加速训练过程.

3.4 聚合组件

分布式训练系统聚合组件将各个计算节点产生的中间结果进行聚合,从而输出最终的训练结果.研究表明有效的聚合方法会加速训练过程.聚合组件包括基于加和操作的聚合以及基于集成的聚合.

3.4.1 基于加和操作的聚合

基于加和操作的聚合方法常见于数据并行模式.当全体计算节点完成训练任务后,聚合组件基于

特定的聚合逻辑将计算节点产生的中间结果进行聚合. 聚合逻辑一般包括全加和操作与部分加和操作. 下面以参数服务器架构为例，介绍上述两种聚合逻辑的区别与联系.

全加和操作为不同计算节点赋予不同的权重并且对全体计算节点产生的中间结果执行加权求和操作，代表性文献包括模型平均(Model Averaging，MA)算法[86]、BMUF[96]算法以及 ADMM 算法[97]. 该聚合方式的优点是复杂性较低且易于实现，缺点是使用同步并行通信时易产生拖累者问题.

为了克服全加和操作的不足，研究人员又提出了部分加和操作，包括带备份节点的同步算法、异步 ADMM 算法以及去中心化算法. 带备份节点的同步算法采取以空间换时间的策略. 文献[26]指出，聚合额外大约百分之五计算节点的中间结果能够有效提升算法精确性. 异步 ADMM 算法则是控制最大延迟来聚合部分计算节点中间结果，从而尽可能避免受到不精确信息的影响. 去中心化算法则聚合少量邻居节点中间结果，降低全规约机制的通信开销，提升训练速度.

3.4.2　基于集成的聚合

基于集成的聚合用于解决神经网络模型训练. 文献[98]指出简单地对各计算节点的局部中间结果进行平均并不能保证聚合后的全局模型优于局部模型. 因此，提出了一种融合压缩的方法 EC-DNN. 此外，文献[99]表明基于投票的聚合发挥了重要作用. 与单机训练策略相比，该算法在几乎不损失精度的前提下，保证模型训练过程快速收敛.

3.5　小　结

第 3 节首先总结了分布式训练系统的通用框架，主要涵盖了划分组件、通信组件、聚合组件以及优化组件. 其次介绍了各组件的工作原理. 其中，划分组件描述分布式训练方案以数据并行、模型并行或混合并行的方式执行. 通信组件规定了分布式训练系统的逻辑通信拓扑和信息同步方式. 优化组件对计算节点的局部训练以及全局模型更新进行优化. 聚合组件则是对各个计算节点产生的中间结果有效聚合. 各组件协同配合，共同完成训练任务.

4　基于并行随机梯度下降算法研究进展

近年来国内外学者在分布式训练优化算法领域开展了广泛研究并取得重要进展，催生出一系列以并行随机梯度下降(Parallel Stochastic Gradient Descent，PSGD)算法[85]为代表的突破性成果. 该算法的主要贡献是将经典的串行随机梯度下降算法实现了并行化，促进了分布式并行优化算法的发展. 本节回顾了 PSGD 算法的分布式训练优化算法及其改进方法. 同时，以 PSGD 算法为开端，总结了中心化架构算法和去中心化架构算法两个研究分支. 在各研究分支中，重点关注以下两个方面：(1) 优化问题及目标函数的性质. (2) 算法及其理论分析结果. 综上所述，图 8 展示了基于 PSGD 算法的中心化架构以及去中心化架构研究脉络.

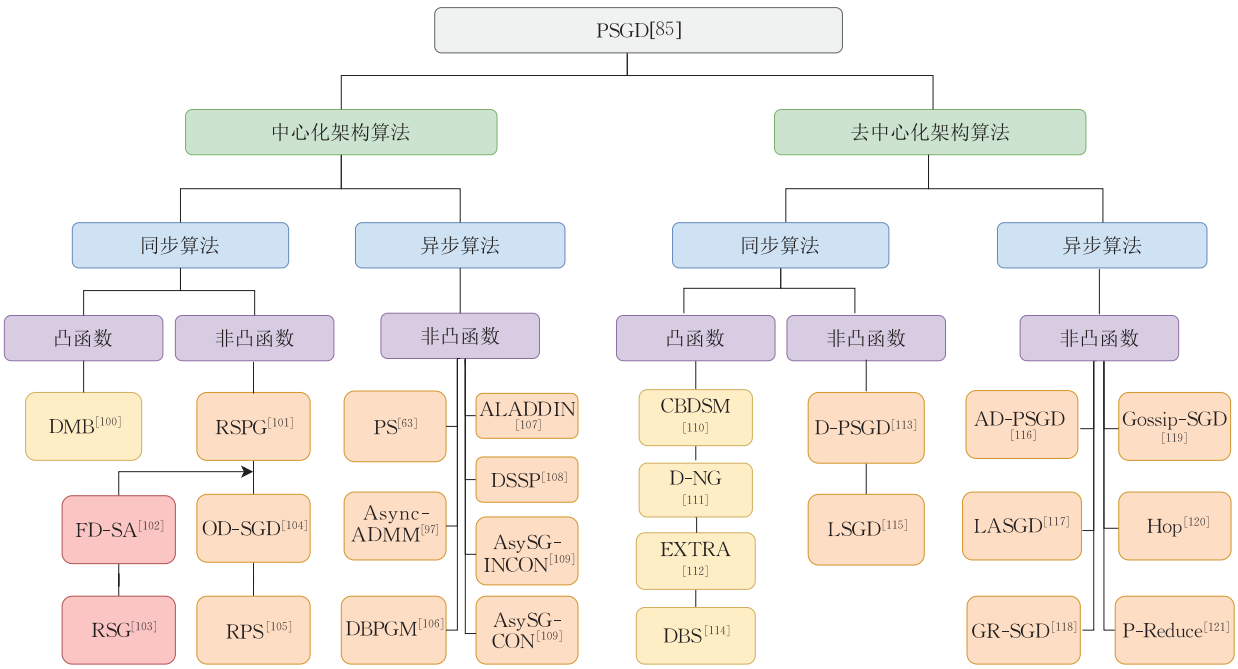


图 8　并行随机梯度下降算法及其改进算法研究脉络

4.1 并行随机梯度下降算法

随着可用数据的增加,并行机器学习已成为一个亟需解决的问题.文献[85]提出了首个并行随机梯度下降 PSGD 算法,以解决凸目标函数的经验风险最小化问题.与先前关于并行优化算法的研究不同,该算法引入了一种收缩映射方法量化参数收敛速率的渐近极限.PSGD 算法提供了渐进线性收敛速率且不存在严格的延迟限制,从而增强了算法在分布式场景中的可用性.算法流程如图 9 所示.

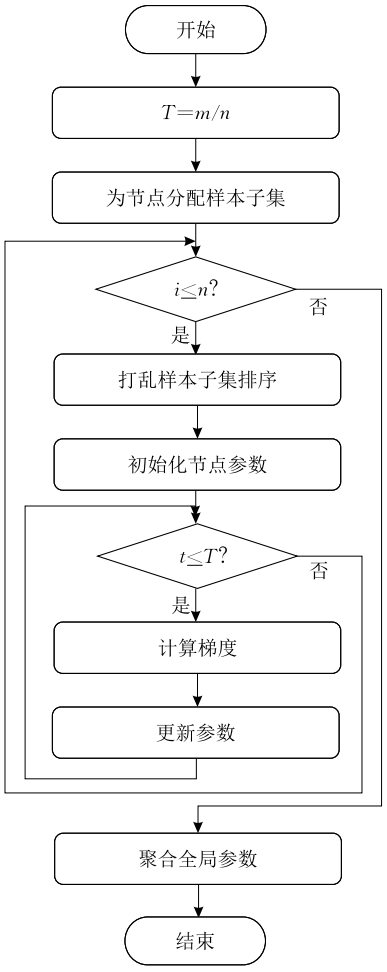


图 9 并行随机梯度下降算法流程图

PSGD 算法要求输入学习率 γ,计算节点数 n 及样本总数 m,并且输出模型参数 $\boldsymbol{x}$.首先,算法随机为各个计算节点分配一个包含 T 个样本的集合$\mathcal{D}_i$.而后,各个计算节点并行地执行如下双循环结构:在外循环结构中,各计算节点随机打乱集合$\mathcal{D}_i$中样本排序并初始化局部参数;在内循环结构中,各计算节点基于样本集合$\mathcal{D}_i$中的随机样本 ξ_t^i计算函数 $f_i^t(\boldsymbol{x}_t^i;\xi_t^i)$与梯度 $\nabla f_i^t(\boldsymbol{x}_t^i;\xi_t^i)$,并更新局部参数 $\boldsymbol{x}_{t+1}^i$.最后,算法聚合全体计算节点局部参数 $\boldsymbol{x}=(1/n)\sum_{i=1}^{n}\boldsymbol{x}_T^i$,并将其平均值作为全局参数.

并行随机梯度下降算法的主要贡献在于将串行的随机梯度算法实现了并行化,并保障了算法的渐进线性收敛速率.继并行随机梯度下降算法之后,研究人员开展了随机梯度并行化的研究.本文将研究进展按照通信架构归纳为中心化架构算法和去中心化架构算法两个分支.下面介绍各研究分支近期关键进展.

4.2 中心化架构算法

以参数服务器架构为代表的中心化架构算法广泛应用于机器学习及深度学习等领域.中心化架构算法按照信息同步方式的不同,又可分为同步算法和异步算法.

4.2.1 同步算法

基于中心化架构的同步并行随机梯度下降算法的关键迭代步骤如下.

首先,各计算节点下载保存在服务器节点处的全局模型参数,并使用该全局模型参数进行梯度计算.

其次,计算节点将各自的局部梯度上传到服务器节点处.

最后,服务器节点聚合所有计算节点的梯度并更新全局模型参数.

考虑到目标函数的性质的不同,研究人员提出了各种中心化架构同步算法求解凸优化与非凸优化问题.这些算法不仅从实验层面验证了算法的有效性,而且从理论层面[100]分析了算法的收敛速率[101].下面分别介绍凸目标函数与非凸目标函数的研究进展.

(1)凸目标函数算法.文献[100]研究了一个随机在线预测问题,其目标为最小化全体样本预测值与观测值差距的总和.为了解决上述问题,作者提出分布式小批量(Distributed Mini-Batch,DMB)算法并将串行在线预测算法转化为分布式算法.作者证明该算法返回解在光滑且凸损失函数及随机输入条件下渐近最优.

(2)非凸目标函数算法.文献[101]研究一类带有约束条件的随机非凸复合优化问题.众所周知,随机规划(Stochastic Programming,SP)最早可追溯到经典随机逼近(Stochastic Approximation,SA)问题,并且保证了求解强凸随机规划问题的渐近最优收敛速率.然而,文献[102]指出随机规划算法实际性能有待提升.此外,虽然文献[103]提出了一个随机梯度(Randomized Stochastic Gradient,RSG)算法,但该算法仅适用于无约束非凸随机规划问题.因此,不能直接应用于求解文献[101]中的问题.综上所述,文献[101]将随机规划方案与随机近似算法相

结合以提升算法性能.

与 RSG 算法不同，文献[101]提出了一种随机投影梯度(Randomized Stochastic Projected Gradient，RSPG)算法，并且计算复杂度与 RSG 算法类似. RSPG 算法不足之处在于返回解稳定性较差. 因此，作者提出了一种两阶段算法 2-RSPG. 此后，作者在 2-RSPG 算法的基础上设计了一种仅利用随机零阶信息的投影算法 2-RSPG-V. 数值结果表明，2-RSPG 和 2-RSPG-V 算法能够有效解决解的稳定性问题.

近期，文献[104]为了解决同步随机梯度下降算法迭代慢的问题，提出了一步延迟随机梯度下降(One-step Delay SGD，OD-SGD)算法. 该算法采用参数服务器架构，并且各次迭代均包括服务器节点的全局更新操作与计算节点的局部更新操作. 计算节点的局部更新操作又可进一步细分为 Warm-up 阶段、Switching 阶段及 OD-SGD 阶段. Warm-up 阶段中计算节点执行同步随机梯度并行 SSGD 过程. Switching 阶段中计算节点获取和备份当前全局参数，并对其在下次迭代中进行更新. OD-SGD 阶段中计算节点广播其更新后的局部参数，且无需等待从服务器节点下载的最新全局参数. 综上所述，OD-SGD 算法实现了加速训练.

图 10 与图 11 比较了 OD-SGD 算法与 SSGD 算法在迭代过程中的运行时间. 假设计算和通信开销均为 3 个单位时间. 使用 SSGD 进行训练时，当前迭代的通信过程结束后才会开始下一次迭代. 因此，各次总迭代开销为 5 个单位时间(如图 10 所示). 然而，使用 OD-SGD 进行训练时，由于其计算与通信具有更高的并发性，使得 OD-SGD 算法各次迭代总开销仅为 3 个单位时间(如图 11 所示).

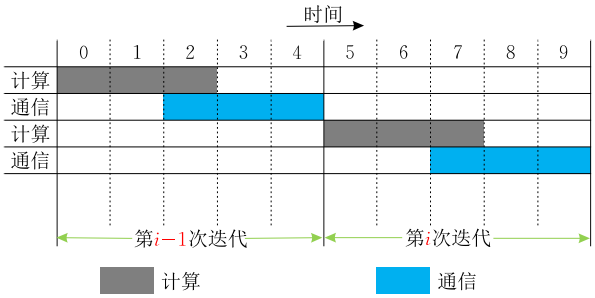


图 10　SSGD 算法

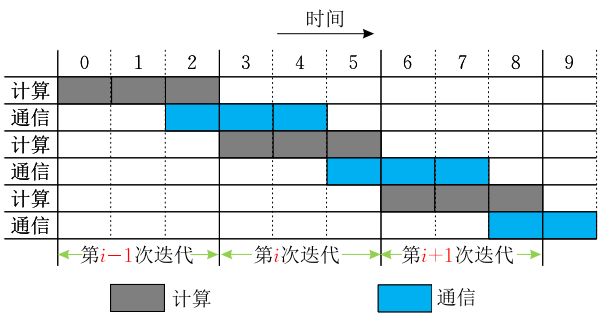


图 11　OD-SGD 算法

现有大部分研究假定通信过程是可靠的. 文献[105]则考虑能否设计容忍训练过程中网络不可靠的机器学习系统. 假设给定一个标准的分布式参数服务器架构，当计算节点和服务器节点之间的通信内容以一个非零概率被丢弃，则是否存在一个算法仍然收敛，并且该算法以何种速度收敛. 为此，文献[105]提出了基于参数切分的 RPS 算法. 首先，算法理论分析证明了不可靠网络上的分布式学习算法可以达到与可靠网络上的集中式或分布式学习算法相当的收敛速率. 其次，作者证明了数据丢包率的影响随着参数服务器节点数量的增加而减小. 最终，RPS 算法应用于不可靠网络场景并实现深度神经网络的训练任务.

4.2.2　异步算法

区别于同步算法，异步算法中各计算节点完成当前轮次迭代训练后无需等待其它计算节点，因而提升了计算资源的利用率. 近年来，异步并行优化算法在解决非凸目标函数领域中取得了关键进展[63,106-108].

非凸目标函数算法. 文献[63]研究非凸复合优化问题. 目标函数由一个非凸连续可微函数 $f(\boldsymbol{x})$ 与一个非光滑的凸函数 $h(\boldsymbol{x})$ 构成. 为了解决上述问题，作者设计了如下的近端梯度算子：

$$Prox_{\gamma}(\boldsymbol{x})=\arg\min_{\boldsymbol{y}\in X}h(\boldsymbol{y})+\frac{1}{2\gamma}\|\boldsymbol{x}-\boldsymbol{y}\|^2.$$

针对复合优化目标函数，首先对连续可微函数 $f(\boldsymbol{x})$ 求梯度并采用 SGD 算法更新参数；其次，对更新后的参数计算近端梯度算子并且作为最终参数. 例如，在第 t 次迭代中给定学习率 $\gamma_t>0$，基于近端梯度算法的参数更新规则如下：

$$\boldsymbol{x}_{t+1}=Prox_{\gamma}[\boldsymbol{x}_t-\gamma_t\nabla f(\boldsymbol{x}_t)].$$

此外，作者证明了基于近端梯度策略可以使算法收敛至驻点.

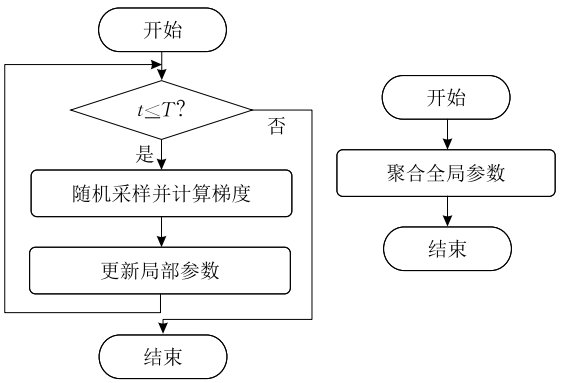
文献[107]提出了一种新型中心化训练算法 ALADDIN. 该算法在服务器节点与计算节点之间采用非对称通信方式，解决服务器节点的通信瓶颈问题. ALADDIN 算法设计了新型 PS-triggered 本地参数更新策略及 Lazy 全局参数更新策略，以缓解方差增加的问题. 通过收敛性分析，作者证明了该算法在非凸目标函数下具有线性加速比. 此外，

针对延时同步并行 SSP 算法静态阈值设置难题，文献[108]提出了一种自适应动态同步阈值算法(Dynamic Stale Synchronous Parallel，DSSP)．算法在运行时自适应地调整各次迭代的同步阈值，以减少速度较快计算节点对全局模型参数同步的等待时间，从而增加迭代吞吐量并加快收敛．

讨论：与同步并行算法相比，异步并行算法允许各计算节点在无需等待其它计算节点的前提下独立计算，大大降低了系统等待开销．文献[109]指出，异步并行算法很大程度上未能提供非凸优化问题的收敛性质和加速特性的理论分析结果．为填补相应理论空白，文献[109]分别研究了非共享内存和共享内存两种条件下分布式异步并行算法设计与理论分析．

4.2.3 非共享内存的分布式异步算法

文献[109]在非共享内存条件下研究了“星状”网络拓扑(中心化架构)的异步更新策略，并提出了 AsySG-CON 算法．算法流程如图 12 所示．



(a) 计算节点工作流程 (b) 服务器节点工作流程

图 12 AsySG-CON 算法流程图

算法输入全局参数 $\boldsymbol{x}_0$，学习率 $\gamma_{t=1,\cdots,T}$ 及小批量样本容量 M，并输出模型参数 $\boldsymbol{x}$．在参数服务器架构中，各计算节点并行地执行如下操作：首先，随机采取 M 个样本 $\{\xi_1,\xi_2,\cdots,\xi_M\}$ 并计算梯度．其次，根据 $\boldsymbol{x}_{t+1}=\boldsymbol{x}_t-\gamma_t\sum_{i=1}^{M}\nabla F(\boldsymbol{x}_{t,i};\xi_{t,i})$ 更新局部参数．最后，将梯度上传至服务器节点．服务器节点接收计算节点发送的局部梯度并更新全局参数．注意到计算节点更新参数为原子操作．换言之，两个节点不同时读取或修改参数．原子操作保证了全局参数的一致性．此外，作者证明了 AsySG-CON 算法的次线性收敛速率．

4.2.4 共享内存的分布式异步算法

文献[109]在共享内存条件下研究了异步更新策略并提出了 AsySG-INCON 算法．算法流程与 AsySG-CON 算法类似．注意到 AsySG-INCON 算法是一种无锁实现方案．共享内存中存储的全局参数，允许所有的计算节点在不使用锁的情况下同时读取和修改．AsySG-INCON 算法中各计算节点并发迭代执行如下操作．

(1) 读取操作，以无锁的方式从共享内存中读取参数 $\hat{\boldsymbol{x}}_{t,i}$ 至局部存储器．

(2) 计算操作，基于样本 $\xi_{t,i}$ 和参数 $\hat{\boldsymbol{x}}_{t,i}$ 计算随机梯度 $\nabla F(\hat{\boldsymbol{x}}_{t,i};\xi_{t,i})$．

(3) 更新操作，以无锁的方式更新全局参数 $\boldsymbol{x}_{t+1}=\boldsymbol{x}_t-\gamma\sum_{i=1}^{M}\nabla F(\hat{\boldsymbol{x}}_{t,i};\xi_{t,i})$．

然而，基于无锁的设定导致各计算节点读取到的参数可能产生不一致的现象．因此，作者提出了如下参数校对方法：

$$\hat{\boldsymbol{x}}_t=\boldsymbol{x}_t-\sum_{j\in J(t)}(\boldsymbol{x}_{j+1}-\boldsymbol{x}_j),$$

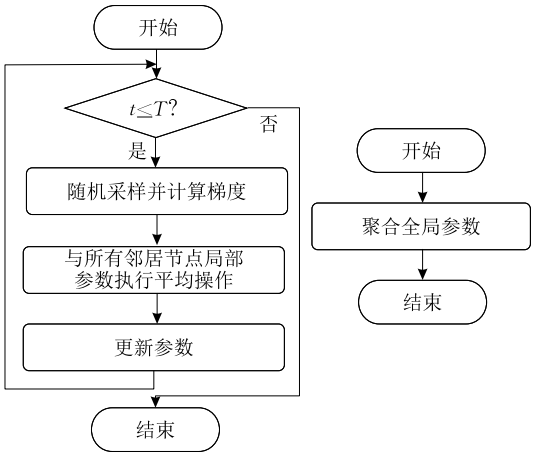
其中，$\hat{\boldsymbol{x}}_t$ 表示读取到校对后的全局模型参数，$\boldsymbol{x}_t$ 表示实际共享内存中的全局模型参数，并且迭代次数集合满足 $J(t)\subset\{t-1,t-2,\cdots,0\}$．该参数校对方法的主要思想是，全局模型参数与真实模型参数之间的误差，可由相邻两次迭代过程中真实参数之差累计量近似．与非共享内存环境不同的是，AsySG-INCON 算法中采用固定的学习率．此外，作者给出了算法次线性收敛的理论结果．

4.3 去中心化架构算法

在中心化架构中，中心节点的通信拥塞问题将随着计算节点数量的增加而愈发严重．不同的是，去中心化架构算法依赖于计算节点之间复杂的通信机制以规避中心化架构中的通信拥塞．去中心化架构算法按照通信方式可细分为同步算法和异步算法．

4.3.1 同步算法

基于次梯度法[110]或快速次梯度法[111]等去中心化架构优化方法假设次梯度有界．上述算法的缺点是当采用固定学习率时，无论目标函数是否可微，算法仅收敛于最优解附近的次优解而不是最优解．与中心化架构算法相比，去中心化架构算法通常具有更多限制性的假设使得其收敛速度较差[112]．为了解决这个问题，文献[113]首次提供了去中心化架构算法优于中心化架构算法的理论证据．作者针对凸目标函数，研究与文献[109]相同的随机优化问题，并提出了去中心化并行随机梯度下降(Decentralized Parallel Stochastic Gradient Descent，D-PSGD)算法．算法流程如图 13 所示．



(a) 计算节点工作流程　　(b) 服务器节点工作流程

图 13　D-PSGD 算法流程图

D-PSGD 算法输入局部参数 $\boldsymbol{x}_0$，学习率 γ，权重矩阵 $\boldsymbol{W}$ 以及总迭代次数 T，并且输出参数 $\boldsymbol{x}_T$. 各计算节点并行执行如下操作：首先，计算节点 i 在第 t 次迭代中随机选择一个样本 $\xi_{t,i}$ 并计算随机梯度 $\nabla F_i(\boldsymbol{x}_{t,i};\xi_{t,i})$. 其次，各计算节点与其所有邻居节点进行参数平均，即 $\boldsymbol{x}_{t+1/2,i}=\sum_{j=1}^{n}W_{ij}\boldsymbol{x}_{t,j}$. 最后，更新局部参数 $\boldsymbol{x}_{t+1,i}=\boldsymbol{x}_{t+1/2,i}-\gamma\nabla F_i(\boldsymbol{x}_{t,i};\xi_{t,i})$. 服务器节点负责聚合所有计算节点的局部参数并输出全局参数 $\boldsymbol{x}_T=\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_{T,i}$.

去中心化分布式训练系统可抽象为一个带权无向图 $G=(V,W)$，其中 V 表示 n 个计算节点构成的集合，$W\in\mathbb{R}^{n\times n}$ 表示一个对称的双随机矩阵. 矩阵的各行与各列之和均为 1. $w_{ij}\in W$ 表示计算节点 j 对计算节点 i 的影响程度. $w_{ij}=0$ 表明计算节点 j 与计算节点 i 之间不存在直接相连的边. 在 D-PSGD 算法中，所有计算节点通过共同的时间锁进行参数同步. 换言之，D-PSGD 算法属于同步算法范畴. 计算节点 i 维护其本地参数 $\boldsymbol{x}_{t,i}$. 例如，在第 t 次迭代中，计算节点 i 根据其本地参数 $\boldsymbol{x}_{t,i}$ 和样本 $\xi_{t,i}$ 计算随机梯度 $\nabla F_i(\boldsymbol{x}_{t,i};\xi_{t,i})$. 当一个计算节点遇到同步屏障时，它将本地参数与其所有邻居节点参数做平均并更新本地局部参数.

需要指出的是：(1) 虽然 D-PSGD 算法中仅给出了使用单个随机样本计算随机梯度的情况，但是当使用小批量样本计算随机梯度时，并不会对 D-PSGD 算法的理论结果产生任何影响；(2) 作者声明了 D-PSGD 算法具有次线性收敛速率；(3) D-PSGD 算法有效减少了计算节点的通信开销. 例如，在 D-PSGD 算法中各个计算节点的通信开销大致为 $O(deg(G))$，其中 $deg(G)$ 表示网络 G 的度数. 中心化架构算法的通信开销为 $O(n)$，其中 n 表示全体计算节点的数目. 节点度数一般远小于全体节点数.

注意到 D-PSGD 算法在迭代中采用固定数量的样本子集. 近期，文献[114]则从小批量样本容量的角度出发并提出动态样本容量(Dynamic Batch Size，DBS)算法. 具体而言，DBS 算法首先根据上一轮迭代情况对各计算节点的性能进行评估. 其次，根据计算节点当前轮次的性能动态调整小批量样本容量以及数据集划分情况，从而提高分布式集群的整体利用率. 令 ρ_i^j 表示第 i 个计算节点在第 j 个 epoch 中的样本容量比例，即 $\rho_i^j=|\mathcal{D}_i^j|/|\mathcal{D}|$，且 $\sum_{i=1}^{n}\rho_i^j=1$. 令 t_i^j 表示第 i 个计算节点在第 j 个 epoch 中的训练时长. 则计算节点 i 的性能评估模型如下：

$$Perf_i^j=\rho_i^j/t_i^j,$$

基于上述性能评估模型，样本容量 M 的动态调整策略如下：

$$M_i^{j+1}=\left(Perf_i^j\Big/\sum_{i=1}^{n}Perf_i^j\right)\times M_i^j,$$

此外，作者证明了 DBS 算法在强凸函数下具有线性收敛速率.

文献[115]基于分层或分组机制提出了面向去中心化架构的同步算法(Layered SGD，LSGD)，如图 14 所示. 在图中，圆形代表计算节点，三角形代表局部服务器节点(通信器)，矩形阴影代表一个层次或分组. 在同一分组内，局部服务器节点与计算节点之间执行 All-Reduce 操作(实线连接). 不同分组局部服务器节点间执行 All-Reduce 操作(虚线连接). 与全体计算节点构成同一分组的同步算法相比，LSGD 算法不同分组之间可以实现并行计算. 此外，与环状拓扑相比，LSGD 算法采用分层的特性有助于降低通信延迟.

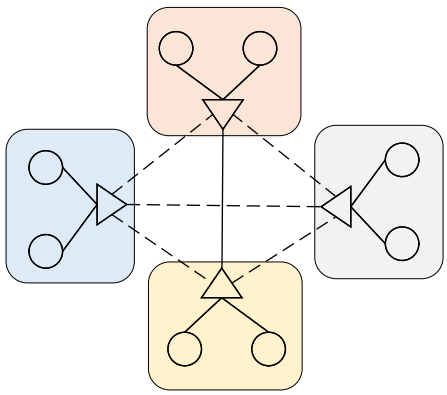


图 14　LSGD 算法通信拓扑

4.3.2 异步算法

虽然 D-PSGD 算法在收敛速度方面与中心化架构算法具有相似的特性,但由于各个计算节点均需要同其所有邻居节点在当前迭代中进行信息同步,从而导致 D-PSGD 算法的通信开销仍然较高. 为了克服上述问题,文献[116]设计了异步去中心化并行随机梯度下降(Asynchronous Decentralized Parallel Stochastic Gradient Descent, AD-PSGD)算法并应用于非凸目标函数.

在 AD-PSGD 算法中,n 个计算节点构成了无向图 $G=(V,E)$,其中 $V=[n]$ 表示全体计算节点构成的集合,E 表示计算节点间所有边构成的集合. 各个计算节点拥有各自的局部数据样本,并且关联一个非凸损失函数. 首先,计算节点 i 随机采样 M 个样本 $\{\xi_{t,1}^i,\xi_{t,2}^i,\cdots,\xi_{t,M}^i\}$ 并且计算梯度. 其次,计算节点随机选择一个邻居节点并与该节点进行参数平均操作. 相比于 D-PSGD 算法,由于 AD-PSGD 算法仅选择一个邻居节点进行参数平均,因而有效降低了通信开销. 此外,作者证明了算法具有次线性收敛速率.

文献[117]提出了局部异步随机梯度下降(Local Asynchronous SGD, LASGD)算法. 该算法兼顾高效去中心化 All-Reduce 以及异步训练的灵活性. 在 LASGD 中,节点间采用异步通信方式,因而无需等待其它节点. LASGD 算法节点间使用同步 All-Reduce 操作进行通信,同时异步计算下一个小批量样本梯度. 算法通过重叠通信过程与计算过程,实现更高的精度和更快的训练.

文献[118]基于去中心化的 Ring-All-Reduce 与 Gossip 机制提出了 Gossip Ring SGD(GR-SGD)算法. 该算法通过控制节点间的通信次数实现异步通信,解决了 Ring-All-Reduce 中由节点速度变慢导致的通信等待问题. 针对非凸目标函数,GR-SGD 算法提供了次线性收敛速率. 此外,文献[119]在 Gossip SGD 基础之上,设计了通信管理节点. 该通信管理节点的主要作用在于评估计算节点的计算时长与通信时长. 根据获取的评估时长,GR-SGD 算法执行通信调度. 这种异步的调度策略可实现计算与通信重叠,从而加速模型训练.

4.4 小 结

本节回顾了分布式训练系统中基于并行随机梯度下降算法的一系列关键研究进展. 首先,分析并讨论了并行随机梯度下降算法的主要特性与贡献. 其次,将后续研究工作按照不同的通信拓扑架构归纳为中心化算法和去中心化算法两个研究分支. 最后,梳理了各个研究分支近年来具有代表性的研究成果.

5 分布式训练优化算法应用

本节介绍分布式训练优化算法的三个典型应用:(1)面向异构环境的分布式训练算法;(2)面向联邦学习的分布式训练算法;(3)面向大模型的分布式训练算法.

5.1 面向异构环境的分布式训练算法与应用

由于计算设备呈现出多样化发展态势,研究人员逐步探索并研究异构环境下的分布式训练策略[120-122]. 文献[120]提出了一个异构感知的去中心化分布式训练方法. 在该方法中,作者研究了去中心化训练的迭代间隙特性. 这一特性直接影响训练方法的正确性以及通信开销. 作者通过分析计算节点之间的通信拓扑结构给出了迭代间隙的上界,并改进了 NTIFY-ACK 算法[123].

根据先前章节的介绍,AD-PSGD 算法中各个计算节点随机选择一个邻居进行参数平均可能导致参数更新冲突. 例如,两个计算节点同时请求一个共同邻居节点进行参数平均,这会导致参数更新冲突. 为了避免冲突,两个节点的请求必须串行执行,但是串行执行会影响算法的效率. 考虑到上述问题,文献[121]设计了一种兼顾 All-Reduce 操作的高效性以及 AD-PSGD 容错性的算法,并构建了一个高性能的异构感知异步去中心化训练系统 Ripple. 该系统包括以下三个关键技术.

第一个关键技术称之为局部全规约(Partial All-Reduce, P-Reduce)算法,如图 15 所示. 算法要求各计算节点循环执行如下过程. 各计算节点初始化局部参数 $\boldsymbol{x}_i$ 并判断算法是否收敛. 如果算法不收敛,则执行内循环. 在内循环迭代中,各计算节点首先随机选择样本 ξ_i 并根据局部参数 $\boldsymbol{x}_i$ 计算梯度. 其次,各计算节点使用公式 $\boldsymbol{x}_i=\boldsymbol{x}_i-\eta_t\nabla F(\boldsymbol{x}_i;\xi_i)$ 进行参数更新. 最后,分组生成器随机生成一个包含计算节点 i 的分组 $\mathcal{G}$ 并对组内计算节点的局部参数求平均 $\hat{\boldsymbol{x}}_{\mathcal{G}}=\frac{1}{|\mathcal{G}|}\sum_{\forall i\in\mathcal{G}}\boldsymbol{x}_i$.

注意到 AD-PSGD 算法随机生成只包含两个成员的分组并且这两个成员进行参数平均. P-Reduce 算法则生成成员数大于 2 的分组并且同一分组内成员进行参数平均操作. 因此,P-Reduce 算法一定程度上减少了冲突发生的概率.

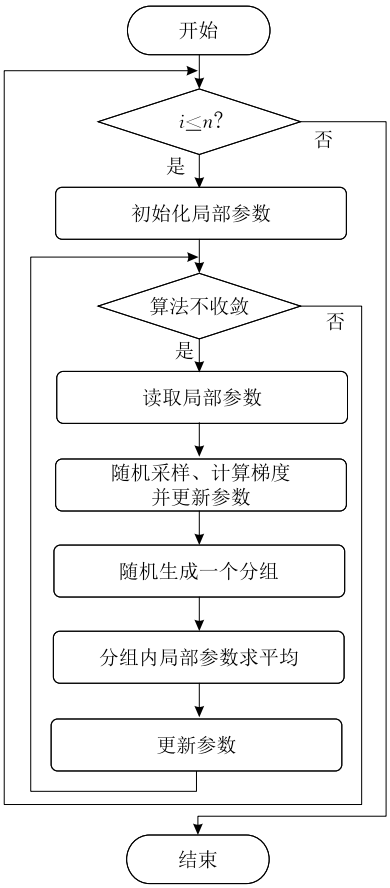


图 15　P-Reduce 算法流程图

第二个关键技术称之为分组生成器(Group Generator,GG)用以生成参数平均的基本单元.下面通过一个具体实例介绍分组生成器的工作原理.

案例分析:图 16 中包含 6 个计算节点,分别记为 N_1,N_2,N_3,N_4,N_5,N_6.假设在初始时刻,计算节点 N_1 和 N_3 已经完成了各自的迭代任务并要求执行参数平均操作.此时,N_1 和 N_3 分别向分组生成器发送参数平均请求.上述两个请求分别记为 Step1 和 Step2.

分组生成器为所有计算节点维护一个锁结构,用来指示当前各计算节点是否正在执行局部全规约操作.该锁结构由一个 6 维向量构成并且各个维度均是布尔变量 **1** 或 **0**.如果一个计算节点对应锁结构标记为 **1**,则说明该节点正在执行局部全规约操作.如果一个计算节点对应锁结构标记为 **0**,则说明该节点未执行局部全规约操作.在本实例中,假设所有的计算节点在初始时刻均未执行局部全规约操作,即所有计算节点对应锁结构均标记为 **0**.

一方面,当分组生成器首先接收到计算节点 N_1 发送的请求,它将随机生成一个包含节点 N_1 且大小不固定的分组.例如,分组生成器随机生成包含计算节点 N_1,N_3 以及 N_5 的三元组 $\{N_1,N_3,N_5\}$(记为 Step3).同时,分组生成器在 Step4 中将计算节点

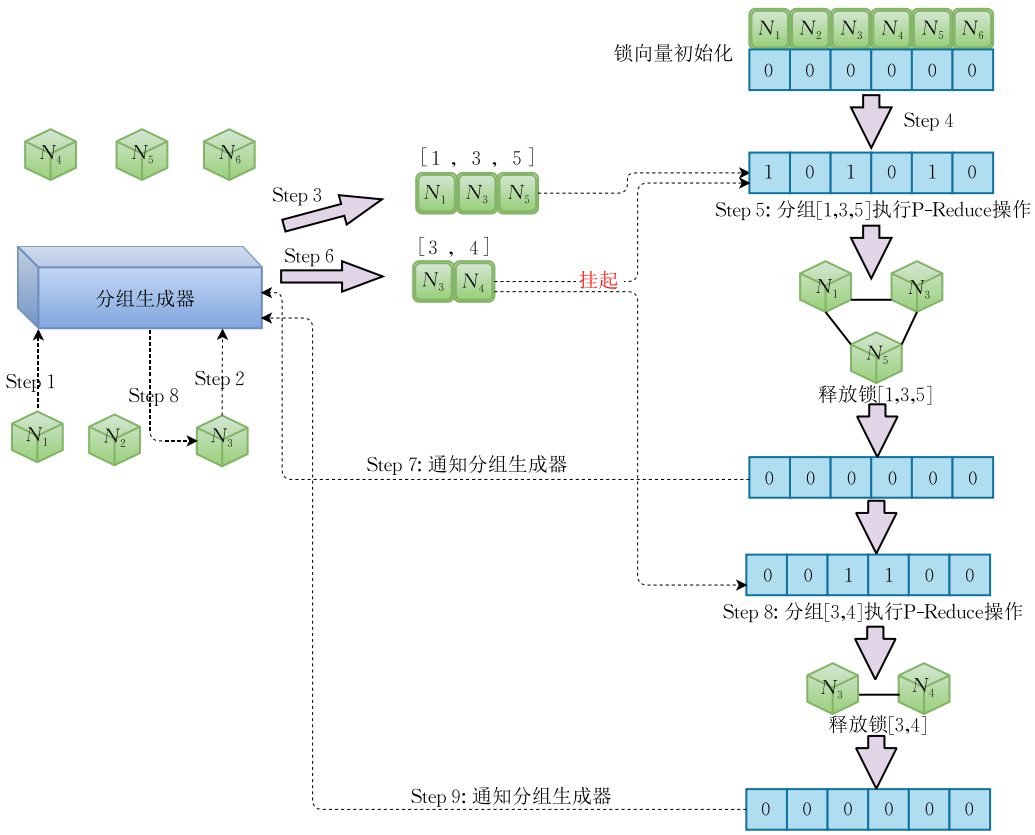


图 16　分组生成器工作步骤

N_1,N_3以及 N_5对应的锁结构标记为 **1**,即[**1**,**0**,**1**,**0**,**1**,**0**].之后,分组生成器通知上述 3 个计算节点在 Step5 中执行局部全规约操作.另一方面,分组生成器也接收到计算节点 N_3发送的请求,并且在 Step6 中随机生成一个包含计算节点 N_3的二元组{N_3,N_4}.三元组{N_1,N_3,N_5}与二元组{N_3,N_4}具有重叠的节点 N_3,这表明两个参数请求过程将产生冲突.因此,为了保证参数平均的正确性,局部全规约操作必须按照先后顺序执行.在三元组{N_1,N_3,N_5}未完成局部全规约操作之前,二元组{N_3,N_4}将被挂起.三元组{N_1,N_3,N_5}在 Step7 执行完操作之后通知分组生成器.分组生成器立刻释放计算节点 N_1,N_3以及 N_5对应的锁结构,并且标记锁结构为[**0**,**0**,**0**,**0**,**0**,**0**].紧接着,分组生成器处理被挂起的二元组{N_3,N_4}并标记锁结构为[**0**,**0**,**1**,**1**,**0**,**0**].二元组{N_3,N_4}在 Step8 中执行局部全规约操作.最后,计算节点 N_3和 N_4执行完操作之后,通知分组生成器并在 Step9 中释放相应锁结构.类似地,如果有多个计算节点向分组生成器发送参数同步请求,重复执行上述 Step1~9,直到所有请求被处理完成为止.

第三个关键技术称之为冲突避免策略.分组生成器产生的分组结果可能导致不同分组间参数更新冲突.为了解决这个问题,作者提出了基于静态调度的策略(如图 17 所示).图中包含 8 计算节点,分别记为 N_1,N_2,…,N_8.这些节点以一定迭代间隔被周期性地分成若干分组.在本实例中迭代间隔周期为 2,并且在各个迭代周期内相同颜色的计算节点表示它们被分配到相同的分组.这种预先制定的静态调度规则使得分组之间不会产生冲突.因此,所有分组的局部全规约操作可并发执行,从而提升参数平均执行效率.

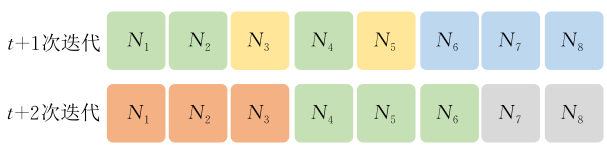


图 17 节点间的静态分组调度表

虽然静态分组调度表可以完全避免冲突,但它存在以下三个挑战:一是静态分组调度表周期大小难以确定.周期过小容易造成频繁生成随机分组,增大系统开销.周期过大则导致局部全规约操作灵活性差;二是静态分组调度表的存储需要占用额外的空间.如果各个计算节点均存储同一调度表会造成存储空间开销过大以及信息冗余;三是静态分组调度扩展性差.调度表预先规定了所有计算节点的分组信息与局部全规约执行规则.一旦系统中一个计算节点宕机或新节点加入,静态分组调度表将无法处理.为了解决静态分组调度表的上述挑战,作者基于异构环境设计了一种智能化随机分组生成方法,实现了冲突避免与性能的有效平衡.

众所周知,分布式训练系统依赖特定的通信协议并利用众多计算节点,达到加速计算的目的.然而,运行上述定制化通信协议需要可靠的高速网络.但是,高速网络通常只能在专用集群中使用.相比之下,基于云或联邦学习的分布式训练任务大多运行在设备不可靠以及网络带宽不稳定的环境.因此,现有训练任务大多使用参数服务器或基于 Gossip 的平均协议.文献[122]提出了一种迭代平均协议 Moshpit SGD 以解决上述异构设备或网络因素的限制.作者证明了该协议对分布式优化的有效性,并且提供了算法理论收敛速率分析结果.此外,实验表明,异构环境下 Moshpit SGD 算法在 ImageNet 数据上训练 ResNet-50 的速度比基于 Gossip 协议的同类方法提高了 1.3 倍.

5.2 面向联邦学习的分布式训练算法与应用

随着数据孤岛现象的加剧及个人隐私保护重视程度的逐步提高,集中学习的应用模式备受制约.联邦学习作为分布式机器学习框架的特例,在不公开用户隐私数据的前提下,联合多个参与方共同训练模型[124-125].联邦学习包括参与方以及训练数据样本集(包含样本 ID、样本属性与样本标签).由于一个参与方不会向其它参与方公开其私有数据,所以联邦学习较传统集中式学习精度稍差.考虑到参与方的样本属性、样本标签以及样本 ID 的不同,联邦学习分为横向联邦学习、纵向联邦学习以及联邦迁移学习.

横向联邦学习面向参与方之间拥有较多共同样本属性且拥有较少共同样本 ID 的场景.例如,两个地区金融机构的用户群体分别来自各自所在的地区,用户 ID 重叠现象较少.但是,由于两机构同属金融领域,其业务属性重叠现象相对较多.代表性联邦学习开源框架有谷歌的 TensorFlow Federated、OpenMined 的 Pysyft 等.

纵向联邦学习面向参与方之间拥有较多共同样本 ID 且拥有较少共同样本属性的场景.例如,同一地区金融机构的用户群体与医疗机构的用户群体,用户 ID 重叠现象较多.但是,由于两机构分属不同应用领域,其业务属性重叠现象相对较少.代表性联邦学习开源框架有百度的 PaddleFL 等.

联邦迁移学习面向参与方样本属性和样本 ID

均不同的场景. 例如，A 地区的金融机构与 B 地区的医疗机构用户群体，样本 ID 与样本属性差异巨大，可能存在少量的重叠. 联邦迁移学习的关键是利用有限少量的共有样本集合，学习不同属性空间的公共表示. 代表性联邦学习开源框架有微众银行的 FATE 等.

联邦学习面临的安全攻击手段主要包括数据投毒[126]、模型投毒[127]及后门攻击[128]. 在数据投毒中，攻击者通过在数据获取阶段有意识地投放错误或伪造数据来降低数据可用性，以达到降低模型精度的目的. 近期，研究学者提出了基于生成对抗网络(Generative Adversarial Nets，GAN)的投毒算法 PoisonGAN[129]. 该算法添加恶意方模型更新比例系数，以放大生成数据的毒害效果. 模型投毒则直接对模型参数进行修改. 由于聚合服务器对参与方发送的信息无法辨别真伪，这为攻击者提供有利条件. 攻击者直接修改或伪造参与方的信息并发送至服务器，破坏模型聚合结果. 后门攻击是指攻击者在模型训练过程中采用不同方式植入一些后门. 如果这些后门未被触发，被攻击的模型表现正常. 一旦这些后门被触发，模型的输出则变为攻击者预先设定的标签，从而实现恶意攻击.

针对联邦学习面临的风险，研究学者提出了多方安全计算、差分隐私以及加密等隐私保护策略. 多方安全计算(Secure Muti-party Computation，SMC)[130]是多个互不信任的参与方在保护各自隐私信息前提下，协同建模的过程. 例如，参与方 A 与参与方 B 共同计算一个双方事先约定好的函数 $f(x_A, x_B) = (y_A, y_B)$. 在计算过程中，参与方 A 仅知晓其 y_A 且无法获知参与方 B 的数据 x_B. 也就是说，多方安全计算同时保证输入数据的隐私安全性以及计算的正确性. 近期，研究人员基于公私钥管理策略提出了函数加密技术[131]及秘密共享技术[132]，从而实现了多方安全计算.

差分隐私(Differentical Privacy)通过在参与方数据集上添加噪声或扰动，并保证在一定概率范围内，攻击者无法由参与方发布的信息推测用户隐私[133]. 差分隐私可使用隐私预算(Privcay Budget)调节隐私保护的等级. 如果隐私预算值较小，则差分隐私机制需为相似的数据输入提供非常相似的输出，增加攻击者推测用户隐私的难度，从而提供更高等级的隐私保护. 相反，如果隐私预算值较大，则差分隐私机制提供相对较低等级的隐私保护. 在噪声添加方面，文献[134]依据梯度下降方向灵活动态调整并添加噪声，从而提升模型性能. 文献[135]为满足差分隐私需求，在参与方本地训练迭代中多次添加高斯噪声. 研究表明将差分隐私应用于联邦学习不会产生过多的额外计算开销. 但是，算法一定程度上降低了模型精度. 因此，需在隐私预算与模型精度间寻求平衡.

联邦学习中的模型加密算法采用同态加密(Homomorphic Eencryption，HE)[136]等方式实现隐私保护. 同态加密允许参与方利用密文进行训练并且训练结果也是密文，同时保证解密后结果与明文训练结果保持一致. 文献[137]提出了一种公钥加密与私钥解密方法，该方法的全体参与方共享一对公私钥. 参与方利用公钥对上传更新进行加密，并利用本地私钥对服务器聚合后的全局模型进行解密. 然而，上述方法仍存在一定的风险. 如果攻击者与服务器相互勾结，攻击者即可还原参与方上传的信息. 为解决这个问题，文献[138]提出本地公钥与共享公钥相结合的二次加密算法. 参与方首先使用本地公钥加密，其次使用共享公钥进行二次加密，从而实现更强的隐私保护. 文献[139]则提出了共享私钥的拆分方法. 该方法将拆分后的私钥由多个不同的参与方共同保管，以增加攻击方的攻击难度. 基于加密算法的隐私保护方案现阶段仅支持相对简单的聚合，同时加密算法也带来了一定的通信与计算成本.

5.3　面向大模型的分布式训练算法与应用

近些年，从谷歌提出的机器翻译 Transformer 模型[140]、语言表征 BERT 模型[141]、视觉 ViT 模型[142]、混合专家 MoE 模型[143]到超大规模稀疏语言 Switch Transformer 模型[15]，模型参数规模不断飙升至千亿乃至万亿级别. 与此同时，大模型对计算资源以及内存需求提出了巨大挑战. 据 OpenAI 报告，模型规模 3.5 个月翻一倍，而 GPU 显存则需要 18 个月才能翻倍. 受限于 GPU 显存，单个 GPU 无法再容纳更大规模模型参数. 因此，面向大模型的分布式训练还需在内存优化方面做进一步探索. 现有内存优化技术主要包括激活重新计算(Activation Re-computing)以及基于高效内存的优化器等.

激活重新计算技术会标记神经网络中少量算子. 前传过程仅保留这些被标记算子的激活值而其它算子的激活直接被释放，后传过程使用被标记的算子重新计算并更新梯度. 激活重计算技术是一种典型的以时间换空间的策略. 在重计算时，同样需权衡计算时间开销. 为此，文献[144]提出了一种名为 Activation/Gradient Checkpointing 的亚线性内存优化的后向重计算技术，旨在降低训练中间激活值的显存瓶颈. 此外，OpenAI 开源了重计算工具包

Gradient-Checkpointing，并且以增加 20%的训练时长为代价换取训练 10 倍的模型①. 旷世科技的深度学习开源框架 MegEngine 融合亚线性显存优化方法与移动边界、块合并、块分裂等其它工程手段，在计算与存储资源受限条件下，可使用更大批量样本训练更深的神经网络模型②. 近期，文献[145]指出重计算中存在大量不必要的冗余，因此提出了选择性激活重算和序列并行算法. 该算法结合张量并行等技术，在训练 GPT-3 模型方面较传统重计算方法实现了 29%的速度提升.

在高效内存优化器设计方面，深度学习框架如 DeepSpeed 集成了内存优化工具 ZeRO[12]. 该工具通过分割模型状态到不同计算单元，以解决数据并行模式下内存不足的问题. ZeRO 将模型状态进行三个层级的分割. 一是优化器状态(Optimizer States)分割，将优化器状态分割并把分割后的分片分配不同的模块，各模块单独处理对应的分片，避免优化器状态的冗余表示. 二是优化器状态与梯度状态(Gradients States)分割，在优化器状态分割之上添加梯度信息的分割，减少冗余梯度内存占用. 三是化器状态、梯度状态及参数(Parameters)分割，在前两者分割基础之上进行参数切分. 此外，文献[146]首先分析了经典 Adam 优化器的显存占用情况，并指出梯度的一阶矩和二阶矩规模与模型参数本身规模相同. 因此，对于大模型训练而言，梯度的一阶矩与二阶矩所消耗的显存空间较大. 其次，提出了一个改进的优化器 AdaFactor 以达到节省显存的目的. 作为降低显存消耗的关键措施，AdaFactor 摒弃了梯度的一阶矩，并且采用基于矩阵低秩分解的方式求解梯度二阶矩的近似. 近期，诸如 1-bit Adam[52] 以及 1-bit LAMB[53] 等优化器也从降低内存占用的角度提出相应策略，从而提升分布式训练性能.

5.4 小 结

本节回顾了分布式训练优化算法的三个典型应用，包括面向异构环境、联邦学习以及大模型场景. 针对上述应用场景，梳理了近期研究进展，并分析了算法设计的核心思想.

6 未来工作

本文概述了广泛应用于机器学习及深度学习等领域的分布式训练系统，总结了分布式训练系统的通用框架，归纳并梳理了各类优化算法. 虽然现有研究成果从不同的角度提升分布式训练系统性能，但仍需对以下挑战进行探索.

(1) 设计高效分布式二阶优化算法. 一方面，现有分布式训练主要采用一阶优化算法，如 SGD 算法和 Adam 算法及其变种算法. 相比之下，二阶优化算法如 K-FAC 应用于深度学习模型训练则较少. 二阶优化算法利用梯度的高阶信息通常比一阶优化算法具有更高的精度，同时具有更快的理论收敛速率. 文献[147]展示了在实现 top-1 验证精度条件下，二阶优化算法迭代次数仅是一阶优化算法迭代次数的 1/3. 然而，计算海森矩阵或 Fisher 信息矩阵的成本较高. 这是因为二阶算法在迭代过程中涉及大规模矩阵求逆等复杂运算. 高昂的计算成本限制了二阶优化算法在泛化、超参数调优和深度神经网络优化等重要问题上的应用[147]. 另一方面，由于二阶优化算法本身的复杂程度高，使得其理论分析相对困难. 此外，二阶优化算法应用于分布式训练应考虑通信开销的问题. 综上所述，未来的二阶优化算法可开展 Fisher 信息矩阵的逼近、理论收敛速率分析、高效矩阵求逆以及分布式通信策略设计等方面的研究工作.

(2) 构建联邦学习场景下的理论分析方法. 联邦学习作为一种特殊的分布式训练模式，使得多个参与方在不共享数据的情况下共同训练模型. 当前，联邦平均(Federated Averaging)是服务器聚合多个参与方信息的一种常用算法，但是普遍缺乏理论保障. 尽管文献[148]分析了非独立同分布数据集上联邦平均算法的收敛速率，并证明了强凸且光滑问题的收敛速率. 然而，目前针对联邦学习非凸问题的理论研究工作相对较少. 此外，鉴于隐私保护是联邦学习不可忽视的重要前提，而高标准的隐私保护策略却会降低分布式协同训练的精度. 所以，构建高效可扩展的分布式数据安全共享机制，实现数据隐私安全保护和模型精度间的平衡，是今后重要研究方向之一.

7 结 论

本文在充分调研和深入分析的基础上，对分布式训练系统近期研究进展进行了综述. 首先，分析了传统单机训练策略面临的数据存储与模型计算等挑战. 为了解决上述挑战，分布式训练系统应运而生.

① https://github.com/cybertronai/gradient-checkpointing
② https://github.com/MegEngine

其次，总结了分布式训练系统的通用框架，包括四个主要组件：(1) 划分组件，即研究如何拆分数据集或模型. 比较了数据并行、模型并行及混合并行为代表的典型拆分模式的异同；(2) 通信组件，即研究分布式训练系统中不同计算节点之间如何实现信息交互的问题. 重点围绕通信内容、通信拓扑以及通信同步方式等方面，详细阐述通信组件中存在的关键技术难题，并梳理了现有研究关键进展；(3) 优化组件，即作为分布式算法优化核心工具，为理论分析提供坚实基础. 本文分类简述了一阶优化算法与二阶优化算法；(4) 聚合组件，即研究如何对各计算节点产生的局部或中间结果进行融合并输出训练结果. 在此基础之上，本文侧重并行随机梯度下降算法及其改进方法. 以通信架构、计算节点间信息同步方式、以及优化目标函数性质等维度梳理主要研究成果，并将其总结为中心化架构和去中心化架构的研究分支. 针对各研究分支中的重要文献进行了深入的分析，并讨论了收敛速率及复杂性等理论性质. 此外，介绍了面向异构环境、联邦学习及大模型三个场景的典型应用. 最后，本文结合分布式训练系统研究现状，提出了未来可能的研究方向，期待对相关领域的研究人员有所启发.

参　考　文　献

[1] Abadi M, Barham P, Chen J, et al. TensorFlow: A system for large-scale machine learning//Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation. Savannah, USA, 2016: 265-283

[2] Paszke A, Gross S, Chintala S, et al. Automatic differentiation in PyTorch//Proceedings of the 31st Conference on Neural Information Processing Systems. Long Beach, USA, 2017: 1-4

[3] Seide F, Agarwal A. CNTK: Microsoft's open-source deep-learning toolkit//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, USA, 2016: 2135-2135

[4] Chen T, Li M, Li Y, et al. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274, 2015

[5] Deng J, Dong W, Socher R, et al. ImageNet: A large-scale hierarchical image database//Proceedings of the 22nd IEEE Conference on Computer Vision and Pattern Recognition. Miami, USA, 2009: 248-255

[6] Yuan J, Gao F, Ho Q, et al. LightLDA: Big topic models on modest computer clusters//Proceedings of the 24th International Conference on World Wide Web. Florence, Italy, 2015: 1351-1361

[7] Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners. OpenAI Blog, 2019, 1(8): 9

[8] Brown T B, Mann B, Ryder N, et al. Language models are few-shot learners. arXiv preprint arXiv: 2005.14165, 2020

[9] Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 2020, 21: 1-67

[10] You Y, Zhang Z, Hsieh C, et al. 100-epoch ImageNet training with AlexNet in 24 minutes. arXiv preprint arXiv:1709.05011, 2017

[11] Chahal K S, Grover M S, Dey K, et al. A hitchhiker's guide on distributed training of deep neural networks. Journal of Parallel and Distributed Computing, 2020, 137: 65-76

[12] Rajbhandari S, Rasley J, Ruwase O, et al. Zero: Memory optimizations toward training trillion parameter models//Proceedings of the 32nd International Conference for High Performance Computing, Networking, Storage and Analysis. Atlanta, USA, 2020: 1-16

[13] Lin J, Men R, Yang A, et al. M6: A Chinese multimodal pretrainer. arXiv preprint arXiv: 2103.00823, 2021

[14] Du Z, Qian Y, Liu X, et al. All NLP tasks are generation tasks: A general pretraining framework. arXiv preprint arXiv: 2103.10360, 2021

[15] Fedus W, Zoph B, Shazeer N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. arXiv preprint arXiv:2101.03961, 2021

[16] NVIDIA R. Corporation, Nvidia Tesla V100 GPU Architecture. White Paper, 2017

[17] Room C. Tensor processing unit. Machine Learning, 2021, 15(54): 13

[18] Zhang C, Li P, Sun G, et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks//Proceedings of the 23rd ACM/SIGDA International Symposium on Field-programmable Gate Arrays. Monterey, USA, 2015: 161-170

[19] Smith M J S. Application-Specific Integrated Circuits. Reading, UK: Addison-Wesley, 1997: 7

[20] Gropp W, Gropp W D, Lusk E, et al. Using MPI: Portable Parallel Programming with the Message-Passing Interface. Cambridge, USA: MIT Press, 1999

[21] Lawson C L, Hanson R J, Kincaid D R, et al. Basic linear algebra subprograms for Fortran usage. ACM Transactions on Mathematical Software, 1979, 5(3): 308-323

[22] Bergstra J, Breuleux O, Bastien F, et al. Theano: A CPU and GPU math compiler in python//Proceedings of 9th Python in Science Conference. Austin, USA, 2010, 1: 3-10

[23] Raina R, Madhavan A, Ng A Y. Large scale deep unsupervised learning using graphics processors//Proceedings of the 26th Annual International Conference on Machine Learning. Montreal, Canada, 2009: 873-880

[24] Liu Tie-Yan, Chen Wei, Wang Tai-Feng, et al. Distributed Machine Learning: Theories, Algorithms, and Systems. Beijing: China Machine Press, 2018(in Chinese)

(刘铁岩，陈薇，王太峰等. 分布式机器学习：算法，理论与实践. 北京：机械工业出版社，2018)

[25] Kaur H，Sharma R. To improve fault tolerance in distributed computing system — A review. International Journal of Computer Science and Mobile Computing，2015，4(8)：330-334

[26] Chen J，Pan X，Monga R，et al. Revisiting distributed synchronous SGD. arXiv preprint arXiv:1604.00981，2016

[27] Yang Q，Liu Y，Cheng Y，et al. Federated learning. Synthesis Lectures on Artificial Intelligence and Machine Learning，2019，13(3)：1-207

[28] Xing E P，Ho Q，Xie P，et al. Strategies and principles of distributed machine learning on big data. Engineering，2016，2(2)：179-195

[29] Verbraeken J，Wolting M，Katzy J，et al. A survey on distributed machine learning. ACM Computing Surveys，2020，53(2)：1-33

[30] Peteiro-Barral D，Guijarro-Berdinas B. A survey of methods for distributed machine learning. Progress in Artificial Intelligence，2013，2(1)：1-11

[31] Pouyanfar S，Sadiq S，Yan Y，et al. A survey on deep learning：Algorithms，techniques，and applications. ACM Computing Surveys，2018，51(5)：1-36

[32] Mayer R，Jacobsen H A. Scalable deep learning on distributed infrastructures：Challenges，techniques，and tools. ACM Computing Surveys，2020，53(1)：1-37

[33] Yang T，Yi X，Wu J，et al. A survey of distributed optimization. Annual Reviews in Control，2019，47：278-305

[34] Sun S，Cao Z，Zhu H，et al. A survey of optimization methods from a machine learning perspective. IEEE Transactions on Cybernetics，2020，50(8)：3668-3681

[35] Tang Z，Shi S，Chu X，et al. Communication-efficient distributed deep learning：A comprehensive survey. arXiv preprint arXiv：2003.06307，2020

[36] Shi S，Tang Z，Chu X，et al. A quantitative survey of communication optimizations in distributed deep learning. IEEE Network，2020，35(3)：230-237

[37] Merelli I，Mosca E，Milanesi L. Parallel Computing，Data Parallelism. New York，USA：Springer，2013

[38] Meng Q，Chen W，Wang Y，et al. Convergence analysis of distributed stochastic gradient descent with shuffling. Neurocomputing，2019，337：46-57

[39] Dean J，Corrado G S，Monga R，et al. Large scale distributed deep networks//Proceedings of the 26th International Conference on Neural Information Processing Systems. Lake Tahoe，USA，2012：1223-1231

[40] Mayer R，Mayer C，Laich L. The TensorFlow partitioning and scheduling problem：It's the critical path!//Proceedings of the 1st Workshop on Distributed Infrastructures for Deep Learning. Las Vegas，USA，2017：1-6

[41] Wang M，Huang C，Li J. Unifying data，model and hybrid parallelism in deep learning via tensor tiling. arXiv preprint arXiv:1805.04170，2018

[42] Krizhevsky A. One weird trick for parallelizing convolutional neural networks. arXiv preprint arXiv:1404.5997，2014

[43] Jia Z，Zaharia M，Aiken A. Beyond data and model parallelism for deep neural networks//Proceedings of the 2nd Conference on Machine Learning and Systems. Stanford，USA，2019，1：1-13

[44] Wang M，Huang C，Li J. Supporting very large models using automatic dataflow graph partitioning//Proceedings of the 14th European Conference on Computer Systems. Dresden，Germany，2019：1-17

[45] He X，Xue F，Ren X，et al. Large-scale deep learning optimizations：A comprehensive survey. arXiv preprint arXiv：2111.00856，2021

[46] Xu H，Ho C Y，Abdelmoniem A M，et al. Compressed communication for distributed deep learning：Survey and quantitative evaluation. Technical Report，King Abdullah University of Science and Technology，Kingdom of Saudi Arabia，2020

[47] Zhu Hong-Rui，Yuan Guo-Jun，Yao Cheng-Ji，et al. Survey on network of distributed deep learning training. Journal of Computer Research and Development，2021，58(1)：98-115 (in Chinese)
(朱泓睿，元国军，姚成吉等. 分布式深度学习训练网络综述. 计算机研究与发展，2021，58(1)：98-115)

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

[49] Bernstein J，Wang Y X，Azizzadenesheli K，et al. signSGD：Compressed optimisation for non-convex problems//Proceedings of the 35th International Conference on Machine Learning. Stockholm，Sweden，2018：560-569

[50] Seide F，Fu H，Droppo J，et al. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs//Proceedings of the 15th Annual Conference of the International Speech Communication Association. Singapore，2014：1058-1062

[51] Zheng S，Huang Z，Kwok J. Communication-efficient distributed blockwise momentum SGD with error-feedback//Proceedings of the 33rd Conference on Neural Information Processing Systems. Vancouver，Canada，2019，11450-11460

[52] Tang H，Gan S，Awan A A，et al. 1-bit Adam：Communication efficient large-scale training with Adam's convergence speed//Proceedings of the 38th International Conference on Machine Learning. Virtual Event，2021：10118-10129

[53] Li C，Awan A A，Tang H，et al. 1-bit LAMB：Communication-efficient large-scale large-batch training with lamb's convergence speed. arXiv preprint arXiv:2104.06069，2021

[54] Lu Y，Li C，Zhang M，et al. Maximizing communication efficiency for large-scale training via 0/1 Adam. arXiv preprint arXiv:2202.06009，2022

[55] Kingma D P，Ba J. Adam：A method for stochastic optimization. arXiv preprint arXiv:1412.6980，2014

[56] Stich S U, Cordonnier J B, Jaggi M. Sparsified SGD with memory//Proceedings of the 32nd Conference on Neural Information Processing Systems. Montréal, Canada, 2018: 4452-4463

[57] Sun H, Shao Y, Jiang J, et al. Sparse gradient compression for distributed SGD//Proceedings of the 24th Database Systems for Advanced Applications. Chiang Mai, Thailand, 2019: 139-155

[58] Islamov R, Qian X, Richtarik P. Distributed second order methods with fast rates and compressed communication//Proceedings of the 38th International Conference on Machine Learning. 2021: 4617-4628

[59] Jain P, Jin C, Kakade S M, et al. Streaming PCA: Matching matrix Bernstein and near-optimal finite sample guarantees for Oja's algorithm//Proceedings of the 33rd International Conference on Machine Learning. New York, USA, 2016: 1147-1164

[60] Wang S, Li D, Geng J, et al. Impact of network topology on the performance of DML: Theoretical analysis and practical factors//Proceedings of the 38th IEEE Conference on Computer Communications. Paris, France, 2019: 1729-1737

[61] Li M, Andersen D G, Park J W, et al. Scaling distributed machine learning with the parameter server//Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation. Broomfield, USA, 2014: 583-598

[62] Jiang J, Cui B, Zhang C, et al. Heterogeneity-aware distributed parameter servers//Proceedings of the 22nd ACM International Conference on Management of Data. Chicago, USA, 2017: 463-478

[63] Li M, Zhou L, Yang Z, et al. Parameter server for distributed machine learning//Proceedings of the 27th Conference on Neural Information Processing Systems. Lake Tahoe, USA, 2013: 1-10

[64] Xing E P, Ho Q, Dai W, et al. Petuum: A new platform for distributed machine learning on big data. IEEE Transactions on Big Data, 2015, 1(2): 49-67

[65] Elk A. Distributed machine learning toolkit: Big data, big model, flexibility, efficiency. 2019[2019-03-07]. http//www.dmtk.io

[66] Cui H, Zhang H, Ganger G R, et al. GeePS: Scalable deep learning on distributed GPUs with a GPU-specialized parameter server//Proceedings of the 11th European Conference on Computer Systems. London, UK, 2016: 1-16

[67] Moritz P, Nishihara R, Stoica I, et al. SparkNet: Training deep networks in spark. arXiv preprint arXiv:1511.06051, 2015

[68] Chilimbi T, Suzue Y, Apacible J, et al. Project Adam: Building an efficient and scalable deep learning training system//Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation. Broomfield, USA, 2014: 571-582

[69] Zhang H, Zheng Z, Xu S, et al. Poseidon: An efficient communication architecture for distributed deep learning on GPU clusters//Proceedings of the 13th USENIX Annual Technical Conference. Santa Clara, USA, 2017: 181-193

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

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

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

[73] Jia X, Song S, He W, et al. Highly scalable deep learning training system with mixed-precision: Training ImageNet in four minutes. arXiv preprint arXiv:1807.11205, 2018

[74] Tanaka Y, Kageyama Y. ImageNet/ResNet-50 training in 224 seconds. arXiv preprint arXiv:1811.05233, 2019

[75] Ying C, Kumar S, Chen D, et al. Image classification at supercomputer scale. arXiv preprint arXiv:1811.06992, 2018

[76] Hasanov K, Lastovetsky A. Hierarchical redesign of classic MPI reduction algorithms. The Journal of Supercomputing, 2017, 73(2): 713-725

[77] Ben-Nun T, Hoefler T. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. ACM Computing Surveys, 2019, 52(4): 1-43

[78] Gibiansky A. Bringing high performance computing techniques to deep learning, 2017[2018-04-21]. http://research.baidu.com/bringing-hpc-techniques-deep-learning

[79] Cho M, Finkler U, Kung D. BlueConnect: Novel hierarchical all-reduce on multi-tiered network for deep learning//Proceedings of the 2nd Conference on Machine Learning and Systems. Stanford, USA, 2019: 241-251

[80] Colin I, Bellet A, Salmon J, et al. Gossip dual averaging for decentralized optimization of pairwise functions//Proceedings of the 33rd International Conference on Machine Learning. New York, USA, 2016: 1388-1396

[81] Kempe D, Dobra A, Gehrke J. Gossip-based computation of aggregate information//Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science. Cambridge, USA, 2003: 482-491

[82] Aybat N S, Wang Z, Lin T, et al. Distributed linearized alternating direction method of multipliers for composite convex consensus optimization. IEEE Transactions on Automatic Control, 2017, 63(1): 5-20

[83] Carli R, Fagnani F, Frasca P, et al. Gossip consensus algorithms via quantized communication. Automatica, 2010, 46(1): 70-80

[84] Cheatham T, Fahmy A, Stefanescu D, et al. Bulk synchronous parallel computing a paradigm for transportable software//Proceedings of the 28th Annual Hawaii International Conference on System Sciences. Wailea, USA, 1995: 268-275

[85] Zinkevich M, Weimer M, Smola A J, et al. Parallelized stochastic gradient descent//Proceedings of the 24th Conference on Neural Information Processing Systems. Vancouver, Canada, 2010: 2595-2603

[86] McDonald R, Hall K, Mann G. Distributed training strategies for the structured perceptron//Proceedings of the 11th Annual Conference of North American Chapter of the Association for Computational Linguistics. Los Angeles, USA, 2010: 456-464

[87] Tandon R, Lei Q, Dimakis A G, et al. Gradient coding: Avoiding stragglers in distributed learning//Proceedings of the 34th International Conference on Machine Learning. Sydney, Australia, 2017: 3368-3376

[88] Niu F, Recht B, Ré C, et al. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent//Proceedings of the 25th Conference on Neural Information Processing Systems. Granada, Spain, 2011: 693-701

[89] Ho Q, Cipar J, Cui H, et al. More effective distributed ml via a stale synchronous parallel parameter server//Proceedings of the 27th Conference on Neural Information Processing Systems. Lake Tahoe, USA, 2013: 1223-1231

[90] Bottou L, Curtis F E, Nocedal J. Optimization methods for large-scale machine learning. SIAM Review, 2018, 60(2): 223-311

[91] Kim H S, Kang J H, Park W M, et al. Convergence analysis of optimization algorithms. arXiv preprint arXiv:1707.01647, 2017

[92] You Y, Li J, Reddi S, et al. Large batch optimization for deep learning: Training BERT in 76 minutes. arXiv preprint arXiv:1904.00962, 2019

[93] Polyak B T. Newton's method and its use in optimization. European Journal of Operational Research, 2007, 181(3): 1086-1096

[94] Amari S. Differential-Geometrical Methods in Statistics. Berlin, Germany: Springer Science and Business Media, 2012

[95] Pauloski J G, Huang L, Xu W, et al. Deep neural network training with distributed K-FAC. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(12): 3616-3627

[96] Chen K, Huo Q. Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering//Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing. Shanghai, China, 2016: 5880-5884

[97] Boyd S, Parikh N, Chu E. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning, 2011, 3(1): 1-122

[98] Sun S, Chen W, Bian J, et al. Ensemble-compression: A new method for parallel training of deep neural networks//Proceedings of the 6th Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Skopje, Macedonia, 2017: 187-202

[99] Meng Q, Ke G, Wang T, et al. A communication-efficient parallel algorithm for decision tree. arXiv preprint arXiv: 1611.01276, 2016

[100] Dekel O, Gilad-Bachrach R, Shamir O, et al. Optimal distributed online prediction using mini-batches. Journal of Machine Learning Research, 2012, 13(1): 165-202

[101] Ghadimi S, Lan G, Zhang H. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. Mathematical Programming, 2016, 155(1/2): 267-305

[102] Spall J C. Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control. New York, USA: John Wiley and Sons, 2005

[103] Ghadimi S, Lan G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Mathematical Programming, 2016, 156(1/2): 59-99

[104] Yu E, Dong D, Xu Y, et al. CP-SGD: Distributed stochastic gradient descent with compression and periodic compensation. Journal of Parallel and Distributed Computing, 2022, 169: 42-57

[105] Yu C, Tang H, Renggli C, et al. Distributed learning over unreliable networks//Proceedings of the 36th International Conference on Machine Learning. Long Beach, USA, 2019: 7202-7212

[106] Li M, Andersen D G, Smola A J, et al. Communication efficient distributed machine learning with the parameter server//Proceedings of the 28th Conference on Neural Information Processing Systems. Montreal, Canada, 2014, 27: 19-27

[107] Ko Y, Choi K, Jei H, et al. ALADDIN: Asymmetric centralized training for distributed deep learning//Proceedings of the 30th ACM International Conference on Information and Knowledge Management. 2021: 863-872

[108] Zhao X, An A, Liu J, et al. Dynamic stale synchronous parallel distributed training for deep learning//Proceedings of the IEEE 39th International Conference on Distributed Computing Systems. Dallas, USA, 2019: 1507-1517

[109] Lian X, Huang Y, Li Y, et al. Asynchronous parallel stochastic gradient for nonconvex optimization//Proceedings of the 29th Conference on Neural Information Processing Systems. Montreal, Canada, 2015, 28: 2737-2745

[110] Matei I, Baras J S. Performance evaluation of the consensus-based distributed subgradient method under random communication topologies. IEEE Journal of Selected Topics in Signal Processing, 2011, 5(4): 754-771

[111] Jakovetic D, Xavier J, Moura J M F. Fast distributed gradient methods. IEEE Transactions on Automatic Control, 2014, 59(5): 1131-1146

[112] Shi W, Ling Q, Wu G, et al. EXTRA: An exact first-order algorithm for decentralized consensus optimization. SIAM Journal on Optimization, 2015, 25(2): 944-966

[113] Lian X, Zhang C, Zhang H, et al. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent//Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, USA, 2017: 5336-5346

[114] Ye Q, Zhou Y, Shi M, et al. DBS: Dynamic batch size for distributed deep neural network training. arXiv preprint arXiv:2007.11831, 2020

[115] Yu K, Flynn T, Yoo S, et al. Layered SGD: A decentralized and synchronous SGD algorithm for scalable deep neural network training. arXiv preprint arXiv:1906.05936, 2019

[116] Lian X, Zhang W, Zhang C, et al. Asynchronous decentralized parallel stochastic gradient descent//Proceedings of the 35th International Conference on Machine Learning. Stockholm, Sweden, 2018: 3043-3052

[117] Avidor T, Israel N T. Locally asynchronous stochastic gradient descent for decentralised deep learning. arXiv preprint arXiv:2203.13085, 2022

[118] Zhou J, Tu J, Ren D L. An asynchronous distributed training algorithm based on gossip//Proceedings of the 7th International Conference on Big Data Analytics. Hyderabad, India, 2022: 214-217

[119] Oguni H, Shudo K. Communication scheduling for gossip SGD in a wide area network. IEEE Access, 2021, 9: 77873-77881

[120] Luo Q, Lin J, Zhuo Y, et al. Hop: Heterogeneity-aware decentralized training//Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems. Providence, USA, 2019: 893-907

[121] Luo Q, He J, Zhuo Y, et al. Prague: High-performance heterogeneity-aware asynchronous decentralized training//Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. Lausanne, Switzerland, 2020: 401-416

[122] Ryabinin M, Gorbunov E, Plokhotnyuk V, et al. Moshpit SGD: Communication-efficient decentralized training on heterogeneous unreliable devices//Proceedings of the 35th Conference on Neural Information Processing Systems. 2021, 34: 18195-18211

[123] Kadav A, Kruus E. ASAP: Asynchronous approximate data-parallel computation. arXiv preprint arXiv: 1612.08608, 2016

[124] Gu Yu-Hao, Bai Yue-Bin. A survey on security and privacy of federated learning models. Journal of Software, 2020, 30(7): 1-32(in Chinese)
(顾育豪，白跃彬. 联邦学习模型安全与隐私研究进展. 软件学报，2020，30(7)：1-32)

[125] Yang Q, Liu Y, Chen T, et al. Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology, 2019, 10(2): 1-19

[126] Biggio B, Nelson B, Laskov P. Poisoning attacks against support vector machines//Proceedings of the 29th International Conference on Machine Learning. Edinburgh, UK, 2012: 1467-1474

[127] Shejwalkar V, Houmansadr A. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning//Proceedings of the 28th Network and Distributed System Security Symposium, 2021: 18

[128] Nguyen T D, Rieger P, Miettinen M, et al. Poisoning attacks on federated learning-based IoT intrusion detection system//Proceedings of the 27th Workshop Decentralized Internet of Thing Systems and Security. San Diego, USA, 2020: 1-7

[129] Zhang J, Chen B, Cheng X, et al. PoisonGAN: Generative poisoning attacks against federated learning in edge computing systems. IEEE Internet of Things Journal, 2020, 8(5): 3310-3322

[130] Yao A C. Protocols for secure computations//Proceedings of the 23rd Annual Symposium on Foundations of Computer Science. Chicago, USA, 1982: 160-164

[131] Xu R, Baracaldo N, Zhou Y, et al. HybridAlpha: An efficient approach for privacy-preserving federated learning//Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security. London, UK, 2019: 13-23

[132] Khazbak Y, Tan T, Cao G. MLGuard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning//Proceedings of the 29th International Conference on Computer Communications and Networks. Toronto, Canada, 2020: 1-9

[133] Xiong Ping, Zhu Tian-Qing, Wang Xiao-Feng. A survey on differential privacy and applications. Chinese Journal of Computers, 2014, 37(1): 101-122(in Chinese)
(熊平，朱天清，王晓峰. 差分隐私保护及其应用. 计算机学报，2014，37(1)：101-122)

[134] Huang X, Ding Y, Jiang Z L, et al. DP-FL: A novel differentially private federated learning framework for the unbalanced data. World Wide Web Journal, 2020, 23(4): 2529-2545

[135] Wu M, Ye D, Ding J, et al. Incentivizing differentially private federated learning: A multidimensional contract approach. IEEE Internet of Things Journal, 2021, 8(13): 10639-10651

[136] Li Zong-Yu, Gui Xiao-Lin, Gu Ying-Jie, et al. Survey on homomorphic encryption algorithm and its application in the privacy-preserving for cloud computing. Journal of Software, 2018, 29(7): 1830-1851(in Chinese)
(李宗育，桂小林，顾迎捷等. 同态加密技术及其在云计算隐私保护中的应用. 软件学报，2018，29(7)：1830-1851)

[137] Chai D, Wang L, Chen K, et al. Secure federated matrix factorization. IEEE Intelligent Systems, 2020, 36(5): 11-20

[138] Fang C, Guo Y, Wang N, et al. Highly efficient federated learning with strong privacy preservation in cloud computing. Computers and Security, 2020, 96: 101889

[139] Fang C, Guo Y, Hu Y, et al. Privacy-preserving and communication-efficient federated learning in Internet of Things. Computers and Security, 2021, 103: 102199

[140] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need//Proceedings of the 31st Conference on Neural

Information Processing Systems. Long Beach, USA, 2017: 5998-6008

[141] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018

[142] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16×16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020

[143] Lepikhin D, Lee H J, Xu Y, et al. GShard: Scaling giant models with conditional computation and automatic sharding. arXiv preprint arXiv:2006.16668, 2020

[144] Chen T, Xu B, Zhang C, et al. Training deep nets with sublinear memory cost. arXiv preprint arXiv:1604.06174, 2016

[145] Korthikanti V, Casper J, Lym S, et al. Reducing activation recomputation in large transformer models. arXiv preprint arXiv:2205.05198, 2022

[146] Shazeer N, Stern M. Adafactor: Adaptive learning rates with sublinear memory cost//Proceedings of the 35th International Conference on Machine Learning. Stockholm, Sweden, 2018: 4596-4604

[147] Ueno Y, Osawa K, Tsuji Y, et al. Rich information is affordable: A systematic performance analysis of second-order optimization using K-FAC//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Diego, USA, 2020: 2145-2153

[148] Li X, Huang K, Yang W, et al. On the convergence of FedAvg on non-IID data. arXiv preprint arXiv:1907.02189, 2019

WANG En-Dong, M. S., professor, academician of Chinese Academy of Sciences. His research interests include design, key technology and engineering implementation of computer architecture and artificial intelligence.

YAN Rui-Dong, Ph. D., senior engineering. His research interests include algorithm design and analysis, and distributed system optimization.

GUO Zhen-Hua, Ph. D., senior engineering. His research interests include computer architecture design and artificial intelligence.

ZHAO Ya-Qian, Ph. D., senior engineering. Her research interests include computer architecture design and artificial intelligence.

Background

In recent years, the rapid development of machine learning and deep learning technologies have promoted applications in various fields. However, as the scale of datasets and models continues to increase, traditional single-machine training strategy is difficult to apply. It faces the following challenges:

(1) The challenge of large datasets and models. For example, training ImageNet on a single machine with a modern GPU needs one week. This is time-consuming.

(2) The challenge of computing resources. Training large models such as M6, GLM, and Switch Transformer inevitably put forward higher requirements on computing resources.

(3) The challenge of storage capacity. With the increasing amount of data and model size, they cannot be fully stored by the limited storage capacity of a single machine, resulting in that training by a single machine being impractical.

(4) The challenge of system stability. The stability cannot be ignored in communication-intensive, computation-intensive and storage-intensive training systems.

(5) The challenge of privacy protection. Privacy protection has become a precondition for data analysis and model training in training systems.

In summary, the traditional way of single-machine training is frustrating because it's hard to meet the requirements of various tricky challenges. In contrast, the collaboration of multi-machines also known as distributed training, as an alternative training way, has become a mainstream learning approach. It does not only leverage various software-level methods but also hardware-level methods to accelerate distributed training effectively.

From the systematic aspect, this survey introduces three key issues of distributed training systems including partition, communication, and aggregation. According to the above issues, this paper proposes a framework for the distributed training system. This framework includes the following four components: partition component, communication component, optimization component, and aggregation component. In addition, we discuss the core technique of each component and recent research progress.

From the algorithmic aspect, this survey focuses on the parallel stochastic gradient descent (PSGD) algorithm and its variants. The main contribution of this algorithm is to parallelize the traditional serial stochastic gradient descent (SGD) algorithm. We summarize the research progress into two branches: the centralized architecture algorithms and the decentralized architecture algorithms. For each research branch, we analyze the algorithm design and convergence properties. Finally, this survey proposes future research directions for the distributed system.

This work is supported by the Shandong Provincial Natural Science Foundation, China No. ZR2021QF073.