分布式机器学习平台与算法综述

舒 娜[1] 刘 波[1] 林伟伟[2] 李鹏飞[1]
(华南师范大学计算机学院 广州 510631)[1] (华南理工大学计算机科学与工程学院 广州 510640)[2]

摘 要 分布式机器学习研究将具有大规模数据量和计算量的任务分布式地部署到多台机器上,其核心思想在于"分而治之",有效提高了大规模数据计算的速度并节省了开销。分布式机器学习作为机器学习最重要的研究领域之一,受到各界研究者的广泛关注。鉴于分布式机器学习的研究意义和实用价值,文中系统综述了分布式机器学习的主流平台Spark,MXNet,Petuum,TensorFlow及PyTorch,并从各个角度深入总结、分析对比其特性;其次,从数据并行和模型并行两方面深入阐述了机器学习算法的分布式实现方式,而后依照整体同步并行模型、异步并行模型和延迟异步并行模型3种方法对机器学习算法的分布式计算模型进行概述;最后,从平台性能改进研究、算法优化、模型通信方式、大规模计算下算法的可扩展性和分布式环境下模型的容错性5个方面探讨了分布式机器学习在未来的研究方向。

关键词 大数据,分布式机器学习,机器学习,算法分析,并行计算

中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.03.002

Survey of Distributed Machine Learning Platforms and Algorithms

SHU Na[1] LIU Bo[1] LIN Wei-wei[2] LI Peng-fei[1]
(School of Computer, South China Normal University, Guangzhou 510631, China)[1]
(School of Computer Science and Technology, South China University of Technology, Guangzhou 510640, China)[2]

Abstract Distributed machine learning deploys many tasks which have large-scale data and computation in multiple machines. For improving the speed of large-scale calculation and less overhead effectively, its core idea is "divide and conquer". As one of the most important fields of machine learning, distributed machine learning has been widely concerned by researchers in each field. In view of research significance and practical value of distributed machine learning, this paper gave a summarization of mainstream platforms like Spark, MXNet, Petuum, TensorFlow and PyTorch, and analyzed their characteristics from different sides. Then, this paper made a deep explain for the implementation of machine learning algorithm from data parallel and model parallel, and gave a view of distributed computing model from bulk synchronous parallel model, asynchronous parallel model and delayed asynchronous parallel model. Finally, this paper discussed the future work of distributed machine learning from five aspects: improvement of platform, algorithms optimization, communication of networks, scalability of large-scale data algorithms and fault-tolerance.

Keywords Big data, Distributed machine learning, Machine learning, Algorithm analysis, Parallel computing

1 引言

分布式机器学习(Distributed Machine Learning)是机器学习当前最热门的研究领域之一,尤其是随着"大数据"概念的兴起,数据爆炸式增长,我们迎来了崭新的"big data[1]"时代。大数据具有五大特征:大数据量(Volume)、多类型(Variety)、低价值密度(Value)、高时效(Velocity)和数据在线(Online)。其中,数据在线是大数据区别于传统数据最显著的特征,这要求对数据进行实时处理。传统的机器学习注重在单机中处理数据的速度,而庞大的数据存储和计算在单机上是远远做不到的,且硬件支持的有限性使得在单机上做大数据处理显得十分吃力,将计算模型分布式地部署到多台、多类型机器上进行同时计算是必要的解决方式。

分布式机器学习的目标是将具有庞大数据和计算量的任务分布式地部署到多台机器上,以提高数据计算的速度和可扩展性,减少任务的耗时。随着数据和计算量的不断攀升,数据处理不但要求实时性,而且要求准确高效,近几年除了硬件实力不断提升,软件支持和算法优化也在同步提高,尤其是分

到稿日期:2018-04-06 返修日期:2018-06-28 本文受国家自然科学基金项目(61772205),广东省科技计划项目(2017B010126002,2017A010101008,2017A010101014,2017B090901061,2016B090918021,2016A010101018,2016A010119171),广州市南沙区科技计划项目(2017GJ001)资助。

舒 娜(1993-),女,硕士生,主要研究方向为分布式机器学习;刘 波(1968-),男,博士,教授,主要研究方向为云存储技术、云计算与大数据技术;林伟伟(1980-),男,博士,教授,CCF高级会员,主要研究方向为云计算、大数据,E-mail:linww@scut.edu.cn(通信作者);李鹏飞(1993-),男,硕士生,主要研究方向为Docker容器调度、云计算。

布式机器学习的研究已经成功应用于图像识别、语音识别、自然语言处理、机器翻译和知识表达推理等领域。早在 2005 年,Apache 就实现了 Hadoop 分布式系统的基础架构,其核心设计——HDFS 分布式文件系统为海量数据提供了存储空间,而 MapReduce 为海量数据提供了计算支持,有效提高了大数据的处理速度。Apache Spark,是由 UC Berkeley AMP 实验室开源的类 Hadoop 并行框架,应用于数据挖掘和机器学习领域,且在迭代优化计算方面的应用效果更好。分布式机器学习经历了近 20 年的发展,国内外涌现出了大量的研究工作,且取得了显著成效。文献[2]从分布式机器学习的程序部署与执行、任务的通信方式与通信内容 4 个方面综合总结了分布式机器学习平台和算法的设计原则。文献[3]分析并总结了基于大数据的机器学习算法与并行算法的研究现状。文献[4]比较了当下主流分布式机器学习平台的规模和可用性,分析了这些平台的容错性和瓶颈,并对比了其在手写体数据集上的效果。文献[5]回顾了并行机器学习算法的研究现状与应用情况,并展望了其发展趋势。文献[6]回顾了机器学习领域中一些流行的算法和优化技术,并重点阐述了分布式机器学习的相关平台与算法的现状、应用及未来的发展趋势。

研究者通常从分离的角度考虑分布式机器学习平台与算法的发展。事实上,平台研究与算法设计对分布式机器学习性能的提升相辅相成,平台研究要结合算法的可行性,算法设计同样需要考虑在平台上的执行效果。鉴于分布式机器学习的理论研究和现实意义,本文系统综述了分布式机器学习的主流平台,并结合它们各自的特点做了深入总结和分析。本文第 2 节概述了分布式机器学习平台及其相应的模型;第 3 节阐述了机器学习算法的分布式实现;第 4 节分析了分布式机器学习在未来的研究趋势;最后对全文进行总结。

2 分布式机器学习平台

2.1 基于数据流模型的 Spark 平台

2.1.1 数据流模型

数据流模型(DataFlow Graph Model)通常把计算即符号表达式抽象成数据流图。数据流图为有向无环图(Directed Acyclic Graph,DAG),图中每个结点表示一个操作,边代表一个结点与另一个结点的依赖关系。例如,在一个 DAG 中,从结点 A 到结点 B 有一条边,说明结点 B 是依赖于结点 A 的一个计算结果。有向无环图的运算类似于树的遍历,入度为零的结点可在任意时间执行,结点之间的关系通过弧唯一连接,因此数据流模型又可理解为数据驱动模型。

尽管将符号表达式抽象成数据流图较为简单,但 DAG 合理调度以最小化数据缓存量并减少计算延迟却非常关键。Bouakaz 等[7]提出的从数据驱动到优先级驱动的 DAG 调度转换能有效解决此问题,却未考虑 DAG 的分布式执行情况,而传统的数据流模型必须综合它在大数据情况下的准确性、延迟和开销。Akidau 等[8]以 CAP 定理为基础解决此问题,且考虑不同的应用领域需要达到的效果不同。例如,在电商领域,允许牺牲一小部分准确性来达到数据的快速处理与分析;而在金融行业,必须对数据进行准确的在线分析;但对于医疗大数据,必须保证百分之百的完整性。当然,针对数据流模型,领域专家也提出了很多改进方法,例如,基于 Apache Spark 百度设计的异构分布式深度学习平台不仅实现了平台的分布式运行,更引入了参数服务器思想实现了数据流模型的分布式数据并行和模型并行。

2.1.2 Spark

Spark 是由 UC Berkeley AMP 实验室开源的分布式大数据处理平台,其生态系统如图 1 所示。

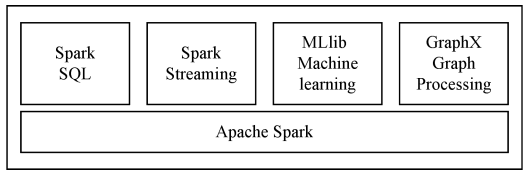

图 1 Spark 生态系统

Fig. 1 Spark ecosystem

Spark SQL 支持使用 SQL 和 Hiveql 多类数据库函数;Spark Streaming 基于 Spark 的微批处理引擎,结合了传统批处理和流式计算的优点,支持各类数据源导入;GraphX 在大规模图像数据处理中应用广泛;分布式机器学习库 MLlibN 包含了大量的机器学习算法,可处理分类、回归、聚类和降维等各类问题。随着 MLlibN 不断发展,其在算法优化和各类型结构化数据处理上的性能提升明显[9]。

Spark 是一种典型的数据流模型计算框架,属于符号式编程计算平台。Spark 通常将计算图建模成 DAG 有向无环图,每个顶点代表一个弹性分布式数据集(RDD),顶点之间的边表示对 RDD 的一个运算,一个 RDD 可包含多种运算。例如,有一条边 E 从顶点 A 指向顶点 B,表示 RDD B 是在 RDD A 上执行运算 E 得到的结果。RDD 上的运算包括变换(Transformation)和动作(Action),变换又包含映射(Map)、过滤(Filter)和连接(Join),动作则表示对变换后的数据进行一系列计算操作。

Spark RDD 拥有分区列表、分片函数以及各 RDD 间的依赖关系列表这 3 个主要属性。Spark 通过计算图中各 RDD 的依赖关系将其划分到不同的逻辑分区,每个分区执行一个任务。Spark 的执行过程如图 2 所示,DAG 被编译为 Stage,每个 Stage 作为一个任务并行执行。

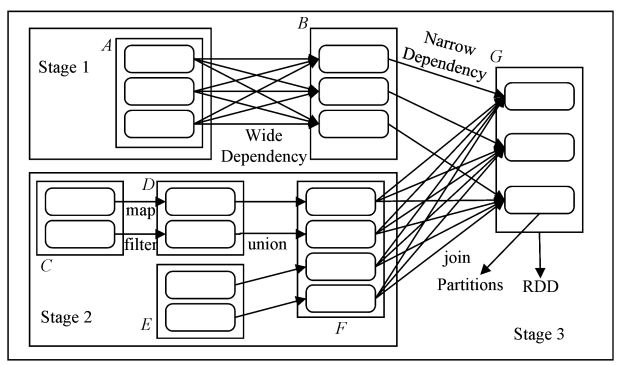

图 2 Spark 的执行过程

Fig. 2 Execution process on Spark

$A-G$ 表示计算图中的顶点 RDD,其中 A 和 B 之间存在

宽依赖。宽依赖是一对多的依赖关系,常见的宽依赖有 shuffle 依赖。而 C 和 D 之间则是窄依赖关系。窄依赖为一对一依赖,结点之间的依赖关系对平台的性能有着极大的影响。在容错性方面,Spark 支持 Lineage 机制和 checkpoint 机制。在 Lineage 机制的容错机制中,若当前结点执行错误,则窄依赖只需重新执行与当前结点有关的一个父结点,而宽依赖需重新执行与之有关联的所有结点。

2.2　基于参数服务器模型的 Petuum 平台

2.2.1　参数服务器模型

随着大数据时代的到来,在分布式机器学习各领域中的数据量和数据维度都呈指数增长[10]。数据流模型在处理大规模、高维度的大数据时,巨大的参数规模使得模型训练异常耗时,如此密集的计算工作和频繁的数据通信需要一个简洁、高效的模型来解决。

2010 年,Alexander 等[11]最早提出参数服务器的概念,参数服务器模型(Parameter Server Model)将参数划分给各工作结点,提高了大规模参数模型训练的性能。参数服务器一经提出,便受到学术界和工业界的高度关注,并被广泛应用。参数服务器模型的架构如图 3 所示。

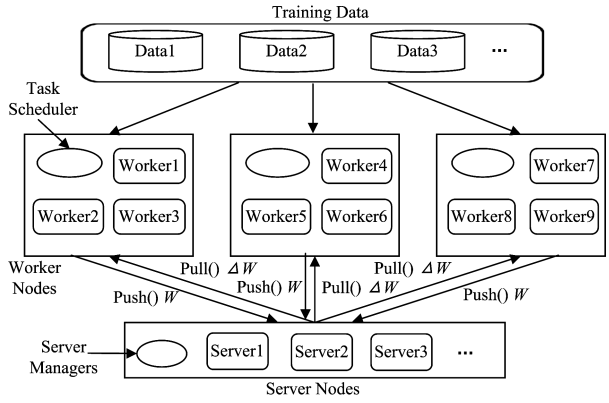

图 3　参数服务器模型

Fig. 3　Parameter server model

模型的参数服务器结点中包含一个服务器管理者结点,该管理者结点负责维护服务器各结点元数据的一致性,管理服务器各结点的生命周期以及分配资源。工作结点负责处理数据,一个工作结点处理一部分训练数据,计算局部梯度并进行数据分析,各工作结点间互不通信。任务调度管理器为对应工作结点下的机器分配工作负载并监督机器处理的进程,增删工作结点时,任务调度器为它们重新分配未完成的任务。工作结点之间独立运行,可实现多租户模式,不同类型的应用可在不同工作结点中并行运行,同类型的应用也可同时分配到多个工作结点中同步执行以提高并行性。

参数服务器模型由工作结点和服务器端参数服务器结点组成,工作结点和服务器结点都各自包含多台机器。服务器结点下的每台机器都包含一部分全局的共享参数,如图 4 所示。

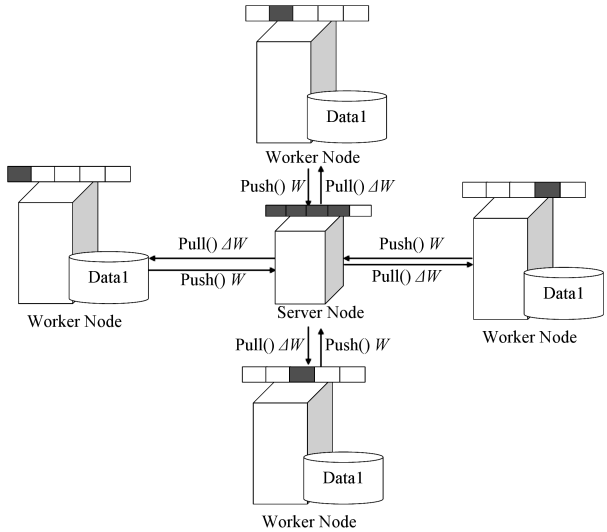

图 4　参数服务器模型参数共享图

Fig. 4　Parameter-shared diagram of parameter server model

参数最初采用分布式 Memcached 存储。参数服务器根据模型耦合关系将网络模型划分成 P 个部分,每个工作结点分配到部分参数,且独立地计算和更新部分参数。为保证模型的可靠性和可扩展性,服务器结点互相通信并进行参数的备份和迁移。工作结点和服务器结点通过 Push 和 Pull 通信,工作结点通过 Push 将梯度参数传递给服务器结点,服务器结点修正和更新参数 Pull 给工作结点。参数服务器模型自出现至今不断得到优化,Li 等[12-14]分 3 个阶段对参数服务器在大规模分布式机器学习领域的应用进行了分析,表 1 对比了其在不同阶段的特点。第一代[12]中侧重对参数服务器的分布式进行优化,有效地平衡了算法的收敛速度,以及系统的有效性、可扩展性和高度容错性;其被广泛应用于 LVM[15]、分布式推理图[16]、深度学习[17]领域。第二代[13]中主要解决工作结点和服务器结点的数据异步通信问题,并通过迭代处理训练数据优化模型。第三代[14]中设计了一种新算法 Asynchronous Block Proximal Gradient Method,其在保证算法收敛率的同时解决了目标函数非凸和不平滑的问题。

表 1　参数服务器各发展阶段的对比

Table 1　Comparison of parameter server in each development period

特 点	第一代	第二代	第三代
易用性	高性能和多线程的线代操作,促进了训练集与参数的运算。	参数以稀疏向量或矩阵的方式存储,线代数据类型有高性能的多线程库。	—
通信方式	结点间异步通信、同步计算。	异步通信,减少了网络拥塞和开销。	异步通信
扩展性	在不重启的状态下动态增加新结点。	在不重的启状态下动态增加新结点。	弹性可扩展
容错性	数据副本体系结构将数据存储在多个服务器结点上,若一结点失败,另一结点自动执行计算。	秒级修复非灾难性机器故障,时钟矢量确保网络故障后正常运行。	数据副本,瞬时数据迁移
一致性	—	松弛有度的一致性进一步减少了同步代价和延迟	导步任务依赖、任务依赖图和多种用户

2.2.2 Petuum

2014 年,卡耐基梅隆大学 Xing 等研发的 Petuum[18],是一个基于参数服务器模型,专门针对机器学习算法的分布式计算平台。与其他分布式机器学习平台有所不同,Petuum 更注重优化大数据计算在大规模分布式集群上花费的通信和等待时间,并为用户提供了数据并行、模型并行、数据和模型并行 3 种可编程方式。研究者认为,大规模分布式机器学习在处理过程中主要存在两方面问题。

(1)研究发现,将单机式机器学习模型迁移到分布式集群上,计算速度和集群机器数量不呈线性相关;相反,当机器增加到一定数量时,计算速度反而降低。在分布式机器学习平台中,数据计算需要花费更多的通信时间和资源等待时间。Petuum 采用 SSP(Stale Synchronous Parallel)[15]并行方式优化了此问题,与 BSP(Bulk Synchronous Parallel)必须所有 Worker 结点执行完才可更新一次 $\Delta\theta$ 值不同,SSP 为工作结点设置一个阈值 S,每个工作结点在计算时,不必等到最慢的结点计算完才更新参数 $\Delta\theta$ 值,只要当最快的结点与最慢的结点之间相差 S 时,最快的结点便自动阻塞以等待最慢的结点执行结束。

(2)大规模机器学习的“大”主要体现在两个方面:1)数据量大,如社交网络[19];2)数据模型大,如深度残差网络[20]、图像识别的卷积神经网络[21]。对此,Petuum 平台实现了数据并行和模型并行。Petuum 平台由 Parameter Server,Scheduler 和 Workers 3 个组件组成。Parameter Server 通过分布式共享内存 API 将用户输入的数据划分给各 Worker 以实现数据并行;Scheduler 负责分配模型参数;Worker 结点负责更新各模型参数,以实现模型并行,Worker 结点下的 machine 接收更新参数并迭代执行计算。Petuum 实现了模型并行的 LDA、MF、Lasso 回归和分布式机器学习数据并行 4 类算法,解决了大规模分布式机器学习问题。

在参数服务器模型的基础上,为了更好地实现容错,Petuum 平台参数服务器采用 ESSP (Eager Stale Synchronous Parallel)模型[22],既保证了模型的收敛速度,又有效减少了网络同步和数据通信的开销。Petuum 下的 PMLlib 库包含了现今流行的多种算法,如聚类、卷积神经网络[21]、随机森林[23]、逻辑回归[24]和支持向量机,解决了机器学习领域的众多问题。

2.3 基于混合模型的 MXNet 与 TensorFlow 及 PyTorch 平台

2.3.1 混合模型

经过长期的实践应用,研究者深感数据流和参数服务器模型本身各有利弊。GoogleBrain 团队在 DistBelief 的基础上研发了 TensorFlow[25],将数据流和参数服务器搭配使用,衍生出了混合模型(DataFlow and Parameter Server Model)。为了取得更快的速度、更高的移植性和灵活性,TensorFlow 采用 DataFlow+Parameter Server 的混合模型。与传统的数据流模型和参数服务器模型均不同,在混合模型中,是将计算任务抽象成有向图(但不是 DAG 有向无环图,而是一个可变状态的有向循环图)。

计算图中结点表示操作 operations(ops),有些 ops 控制流结点被用来计算梯度以更新网络模型中的变量;边表示结点间相互依赖的多维矩阵(张量 tensor)。混合模型中加入了参数服务器的思想进行训练,模型由 3 个主要部分组成:Client,Master,Worker。按网络图层将模型逐层划分给不同的 Worker 结点,并将参数服务器与数据流图相结合。混合模型图的结构如图 5 所示。

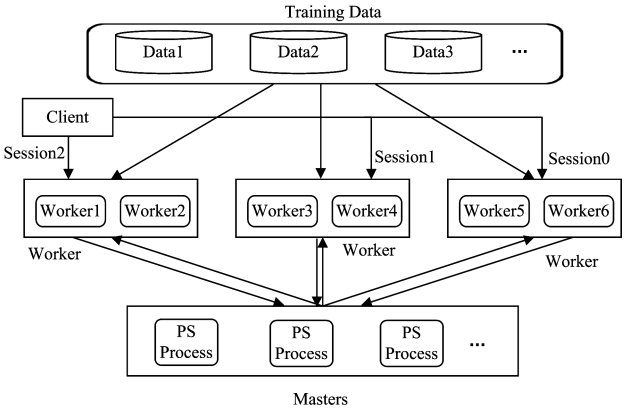

图 5 混合模型图的结构

Fig. 5 Structure of mixed model graph

Client 负责将符号表达式抽象成计算图,并将计算请求发送至服务器结点;Worker 结点负责处理从服务器结点发送过来的计算请求;而 Master 负责任务调度,并确保每个 Worker 的运行需求。Worker 结点内部使用多线程消息传递机制通信,机器内部线程设置 send 结点和 receive 结点,在运行时 send 结点和 receive 结点实现跨设备传输数据,确保所需的数据从源设备到目的设备只转换一次,保证了数据同步结果的一致性。不同 Worker 结点通过 session 交互,减少了 Worker 结点之间的数据传递开销。

2.3.2 MXNet

MXNet 是由李沐等联合研发的轻量级分布式机器学习平台,既属于符号计算框架,又属于命令式框架,由多状态数据流循环计算图组成,是训练参数服务器模型的平台。2017 年,研究者对该平台做了一系列改进以方便用户使用,用户只需考虑数据量和计算量并构建好模型,无须关注硬件和平台内部结构。MXNet 加入动态接口 Gluon 和 ONNX 开放体系。Gluon 高级动态接口比低级接口更简单易用,支持 JIT 编译的动态计算图,灵活且高效。Gluon 接口的引入使得 MXNet 的性能有了极大的提升:1)节省计算资源,提高了资源利用率;2)计算时间随机器和 GPU 扩展线性增加,且在单机上的效率也明显提高;3)同时支持命令式和符号式编程;4)支持静态计算图和动态计算图,且两种状态图可自由切换;5)SGD 优化器加速了模型的收敛。

MXNet 系统架构如图 6 所示,从上到下分别为嵌入的主语言、编程接口、命令和符号表达式编程模式实现、各类硬件支持。NDArray 是 MXNet 存储和处理数据的工具,类似于 Numpy 的多维数组,支持 CPU/GPU 和分布式云架构上的异步计算和自动求导。Autograd 包添加了优秀的损失函数,与 PyTorch 类似,可对模型参数自动求导,即时进行反向梯度传播。

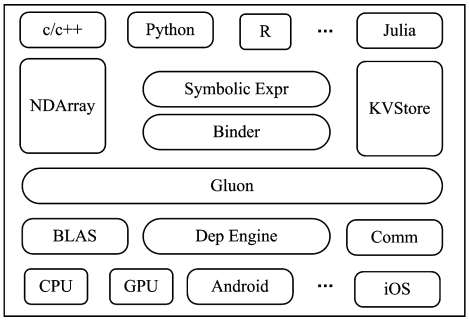

图 6　MXNet 架构

Fig. 6　MXNet architecture

模型结构:MXNet 采用 ndarray 和 autograd 库实现整个神经网络模型,网络由 Blocks 组成,Blocks 是接收输入并产生输出的单元,整个网络仅有一层。将 MXNet 网络模型表示成一个计算图,图 7 所示是多层感知机的部分计算图,其中圆圈表示变量,方框表示操作子,操作子可以是简单的符号操作,也可以是网络模型的一层操作,从接收输入到产生输出,箭头表示数据依赖关系,实线箭头表示正向传播过程,虚线箭头表示求导后的反向传播过程。

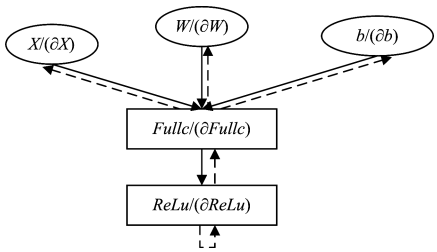

图 7　MXNet 的模型计算

Fig. 7　Model computation for MXNet

MXNet 支持训练参数服务器模型。KVStore 是基于参数服务器的两层数据通信结构:第一层是服务器管理单机内部的多个设备之间的通信,由于单机内部的带宽基本保持一致,因此采用强一致性模型;第二层的服务器管理机器之间通过网络通信,由于网络受各机器之间带宽占用和资源分配的影响会有所不同,因此采用弱一致性模型。Serialization 机制中的模型存储类似容错机制中的检查点文件,可以保存和对比训练过的模型,从而无须从头训练就能选取迭代中效果最好的模型,节省了大量时间和资源,并提高了平台的容错性。

2.3.3　TensorFlow

Google 于 2011 年推出了第一代分布式机器学习处理系统——DistBelief,后来通过研发在此基础上推出了第二代系统——TensorFlow,旨在实现大规模分布式机器学习模型的应用和部署。DistBelief 在参数服务器模型上实现了数据并行和模型并行,用 Downpour SGD 异步随机梯度下降和 Sandblaster 分布式批量优化算法训练大规模分布式模型,在 ImageNet 上训练了比先前大 30 倍的网络模型并取得了很好的效果,被广泛应用于计算机视觉、文本分类和语音识别等领域。TensorFlow 支持在各类环境下执行分布式机器学习程序,包括移动设备、Windows 和 CPU/GPU 等大规模计算系统。TensorFlow 以其速度快、扩展性好、灵活性高的特点,以及具备计算图和数据分析的一系列可视化工具,而受到各行业的广泛关注。TensorFlow 与 DistBelief 的最大区别在于 DistBelief 是由相对复杂的 layers 组成,而 TensorFlow 在数据流图中是以 operates 作为结点。TensorFlow 客户端通过备份技术和核心数据流图的执行实现并行,与其他设备协调更新服务器端的共享参数。

TensorFlow 平台的总体架构采用 DataFlow+Parameter Server 的形式,将计算抽象成由顶点和边集合组成的有向图(Directed Graph)。在 TensorFlow 中,顶点表示一个操作实例,每个顶点有 0～N 个出度或入度,图中的边为 tensor(张量-任意维度数组)。tensor 在 TensorFlow 中表示操作单元,是一个操作而不是一个值,不能进行赋值操作。TensorFlow 中还有特殊边——边中没有数据流过,但它表示控制依赖的源顶点必须在控制依赖执行目的顶点执行前完成执行。在模型中插入控制依赖来强制控制执行顺序,可管控内存利用率的峰值。

TensorFlow Client 之间通过 session 进行交互,支持计算图中结点或边的动态增加。Client 用于构建数据流计算图,并与 Master 和 Worker 进程进行通信,与 Master 的通信通过 session 完成。一个 Worker 结点可能包含一台或多台设备,设备负责执行被分配的任务。一对一的 Worker 和 Device 比较简单,而一对多的情况则需考虑计算图结点对应设备的分配问题和不同设备间数据的通信问题。Worker 结点的主要任务是把计算图映射到对应可用的设备中,并估算图中每个结点从输出到输入的 tensor 的计算时长。Worker 下的结点采用启发式贪婪算法进行部署,该算法既可基于不同操作类型进行静态分配,又可根据早期计算图执行的真实值做决策。计算图通常被划分为多个子计算图集合,因此 Arvind 等[26]提出了子图并行执行、多次迭代的方法。TensorFlow 系统实现了数据并行、模型并行训练和同步流水线模型计算。数据并行训练采用 SGD 加速算法收敛,对于同一批次的实例,模型分布在不同的设备上完成计算。DLSTM[27]中的 sequence to sequence learning[28]在机器翻译经典数据集中的 BLEU 分数提升明显。同步流水线模型计算与异步数据并行类似,区别在于同步流水线模型计算的并行性发生在同一设备中,而非不同设备的计算图中。文献[29]以消息传递接口(MPI)作为 TensorFlow 在分布式存储子系统的并行化通信接口,成功地将 TensorFlow 的执行扩展到大规模集群上,缓解了分布式内存的限制。

2.3.4　PyTorch

PyTorch 是由 Facebook 于 2017 年研发的轻量级分布式机器学习平台,是 Python 开源科学计算库 NumPy 的替代者,支持 GPU 且具有更高级的性能,为深度学习研究平台提供了最高的灵活性和最快的速度。PyTorch 与 TensorFlow 类似,二者网络模型的符号表达式都被抽象成计算图(Computational Graph),也都提供了损失函数反向模式自动微分方法来反向求导传播参数。不同的是,PyTorch 更适合小规模项目,且其计算图是动态的,故被称为"动态计算图",在运行时动态构建。而对于 TensorFlow,一个计算图通常先经过 Python"编译",然后再运行。众所周知,在机器学习中,同一个网络模型需要经过反复迭代才能达到最好的效果,在 TensorFlow 中,先定义一个计算图,每次迭代只是 feed 不同的数据,反复执行同一个计算图;相反,PyTorch 每次迭代都重新

构建一个计算图，这加速了网络模型的收敛。

PyTorch 的主要特点包括：1）包含类似于 Numpy 的 N 维 tensor，Numpy 尽管是一个非常完美的框架，但不能使用 GPU 加速计算，而 Tensor 在 GPU 加速上做得相当出色；2）AutoGrad包内的自动微分方法可快速构建和训练神经网络模型，并迅速反向传播梯度；3）良好的数据加载 API 设计，使得数据的并行加载非常简单。PyTorch 网络模型被抽象成计算图，其中 variable 是结点，边代表从某个输入 tensor 到产生输出 tensor 的函数。variable 和 tensor 没有本质区别，只是 variable 被放入计算图中进行正向、反向传播和自动求导。variable 包含 3 部分：类似于 tensor 的 variable. data、梯度计算 variable. grad 和获取 tensor 操作的 variable. creator。动态计算图可加速模型收敛，静态计算图可提前优化，也便于将一些计算子图融合或分布式地部署到 GPU 和其他机器上。

2018 年 4 月，Facebook 宣布将 Caffe2 并入 PyTorch，这对目前分布式机器学习平台的格局产生了巨大影响。随着 PyTorch 在各界应用的不断推广，研究者也发现了其相对于 TensorFlow 的不足之处。Caffe 的研发者贾扬清表示，PyTorch 优秀的前端以及 Caffe2 强大的后端相结合将能大大提升开发性能。

Caffe[30]是一个非常优秀的深度学习框架，可跨平台训练卷积神经网络和其他深度学习网络模型，可在异构平台之间无缝切换，且其在云环境下的部署也十分简便，是目前计算机视觉界最流行的工具之一。基于分层模型的体系结构使得 Caffe 对卷积神经网络和其他深度网络模型的训练变得简单。分层模型有利有弊，不能像 TensorFlow 一样支持细粒度网络层，因此 Caffe 在循环网络和语言建模方面的表现很差。但相较于其他平台，Caffe 完全基于 C++实现，无障碍的硬件切换使其适用于科研和大规模工业领域；另外，Caffe 提供了参考模型，从而节省了预训练时间，加速了模型收敛。

Caffe 体系结构大致可分为：数据存储、模型分层、构建网络、训练网络、模型微调。根据需要将数据从 CPU 复制到 GPU，消除了 CPU/GPU 之间操作的计算和通信开销。为了确保正向和反向数值的传递，Caffe 本质上按照神经网络层结构进行分层，采用 SGD 算法优化训练模型，且数据以 mini-batch 形式读入，开发者可根据数据特征微调模型结构。

2017 年 4 月，Facebook 开源了完全重构和升级 Caffe 的 Caffe2。Caffe2 更注重模块化，并且轻量、扩展性好，支持多 GPU 运行，在移动端和大规模部署上表现卓越。Caffe2 延续了对视觉类问题的处理，增加了对有助于自然语言处理和时序预测的 RNN 和 LSTM 网络的支持。在 Caffe 中，每层网络模型的权重、偏差都是以 layer 级别存储，这使得网络模型层的修改非常复杂。Caffe2 细化并调整了网络模型层，基本计算单位为 operator（表示计算逻辑），将权重、偏差参数从层中剥离出来单独存储。另外，Caffe2 扩展了 Blobs 数据类型，不再只针对图像数据，应用范围更加广泛。

虽然目前 PyTorch 不支持 Keras，TF. Learn，TensorFlow-slim 等类型的库，但其自带的 torch. nn 包同样包含了一系列组件，可完成比网络模型原始计算图更高级的抽象和模型训练。在模型优化方面，PyTorch 也做得非常好，不再使用简单的 SGD 算法手动调整数据以保证参数的可用性，而是采用更加复杂的优化器，如 AdaGrad，RMSProp 和 Adam，来优化网络模型。在控制流和权重共享上，使用全连接 ReLu 网络，通过多次重用同一模块实现内层权值的共享。

2.4 平台比较

分布式机器学习主要从以下 4 个方面取得突破：1）模型优化；2）算法优化；3）系统优化；4）适用场景。到目前为止，分布式机器学习平台按照编程方式可分为深嵌入符号式平台和浅嵌入非符号式（命令式）平台。符号式平台中的计算被定义为一个向量计算符的符号图，在一个符号表达式计算中先将其转换成对应的计算图，再进行赋值和计算；而命令式则要求变量在计算前已被赋值，并立即执行计算表达式。现有的符号计算平台有 Theano 和 CNTK 等；命令式计算平台有 Spark，Torch 和 Caffe 等；而当前比较流行的 MXNet 和 TensorFlow 均采用符号式+命令式的混合编程方式。前文介绍了当前比较流行的几种分布式机器学习平台，表 2 对这几个平台的各种特点进行了详细对比。随着版本的更新和新平台的出现，分布式机器学习平台的适用场景越发广泛，MXNet，Petuum 和 TensorFlow 等在图像识别、语音识别、自然语言处理等应用上都取得了较好的成效。在计算机视觉类应用领域的翘首当属 Caffe，Facebook 于 2017 年 4 月推出的 Caffe2 成功弥补了其在 RNN 和 LSTM 等循环网络模型上的缺陷。另外，MXNet，Caffe2 和 TensorFlow Lite 平台不仅支持在 CPU 和多 GPU 上运行，更将分布式机器学习从云端搬到线下的 iOS 和 Android 移动端，让分布式机器学习无处不在。各编程接口支持的语言也越来越多。

表 2 分布式机器学习平台的性能对比

Table 2 Performance comparison of distributed machine learning platforms

	Spark	MXNet	Petuum	TensorFlow	PyTorch
适用场景	分类/回归/降维	分类问题/序列/NLP	分类问题/推荐	分类问题/NLP/序列预测	图像识别/手写体
适用规模	大规模/跨平台	轻量级/可移植	中、大规模	大规模/跨平台/嵌入式	轻量级/小规模
编程语言	Scala/Java/Python	C++/Python/Scala	C++	C++/Python	Python
API	NumPy/Spark API	动态 GLUON	YARN/HDFS	TFServing/动态 EagerExecution	NumPy/SciPy
库	MLlib/GraphX	Keras	PMLlib	Keras/TF·Learn/tensorSlim	Module 组件
支持网络	SVM/CF/ALS	CNN/RNN/LSTM/GAN	CNN/VGGVet/AlexNet	CNN/DNN/RNN(神经网络 API)	CNN/RNN/LSTM
支持平台	CPU/GPU	CPU/GPU/移动端	CPU/GPU	CPU/GPU/iOS/Windows/移动端	CPU/GPU
编程方式	命令式	符号式+命令式	符号式	符号式+命令式	符号式
优化方法	—	异步 SGD	Mini-Batch SGD/CD	SGD	AdaGrad/RMSprop
计算图	静态计算图	静态/动态计算图	静态计算图	非循环静态/动态计算图	动态计算图
一致性	—	SSP/ASP	SSP/BSP	SSP	异步多进程
通信	消息传递机制	KVStore	KVStore	多线程/消息传递机制	消息传递机制
并行方式	数据并行/模型并行	数据并行	数据并行/模型并行	同步、异步数据并行/模型并行	数据并行
容错	Lineage/checkpoint	checkpoint	迭代收敛算法	Log file	Module 组件重用
数据存储	RDD	NDArray	Key-Value	Tensor	Tensor

相对于 PyTorch,TensorFlow 更适合大规模项目的部署,且其良好的跨平台性和嵌入式部署使得它拥有众多支持者。TensorFlow 自带的可视化工具 TensorBoard 可随时查看机器学习训练过程中数据的变化和模型的训练曲线与验证结果。另外,其对高级 API 的支持和平台成熟性也是 PyTorch 目前所不能及的。而将 Caffe2 并入 PyTorch 这一重大举措能结合两者的优势,为开发者提供更多便捷。

在网络模型支持上各平台也都大同小异,MXNet 支持的 GAN 生成式对抗神经网络受到了众多爱好者的欢迎。PyTorch 也在试图支持各种网络模型和 keras 库。值得一提的是,前面几种平台都是将表达式构建成静态计算图,而 TensorFlow 和 PyTorch 支持动态计算图。静态计算图在执行过程中,在同一个计算图中运行不同的数据;动态计算图每次迭代重新生成计算图,提升了网络模型的性能。同样,在收敛算法的优化上,其他平台采用较传统的 SGD 算法和 CD 算法,而 PyTorch 采用 AdaGrad,RMSprop 和 Adam 优化器进行优化。为了加快模型的速度,现有的模型训练并行方法有数据并行、模型并行和数据+模型并行 3 种。数据并行是在不同的机器上运行同一网络模型;模型并行是将模型划分到不同的机器上,再用同一批数据进行训练。将模型按照依赖关系划分到不同的机器上,被划分到不同机器上的模型的耦合性较低。Petuum 和 TensorFlow 实现了数据并行和模型并行。在分布式机器学习中,即使是配置相同的机器,由于网络带宽、资源争用,也会导致在相同模型下训练出的结果存在差异,因此平台的容错性也很重要。

3　机器学习算法的分布式实现

机器学习平台的实现需要机器学习算法的分布式支持。目前分布式机器学习应用广泛,遍布计算机视觉、语音识别、自然语言处理、推荐系统等领域[31]。机器学习算法按学习方式可分为监督学习、半监督学习、非监督机器学习和强化学习。监督式学习和半监督式学习在分类和回归问题上应用时具有较好的效果,非监督学习方式在关联规则和聚类问题上表现突出,而强化学习方式更适合处理与环境有交互的问题,如机器人控制和自动驾驶。文献[3]主要介绍了大数据下的机器学习算法,从大数据特征选择、分类、聚类及关联分析等方面阐述了传统算法的弊端及机器学习算法在各适用领域的发展情况。本节概述了机器学习算法的分布式实现方式,并从整体同步并行、异步并行和延迟异步并行模型方面阐述了分布式计算模型。

3.1　机器学习算法的分布式实现方式

计算量庞大和数据通信密集的机器学习问题在资源有限的单机上无法得到解决。机器学习算法的分布式实现既可加快计算速度,又可达到模型的线性扩展。其实现思想就是划分大规模网络模型并将其分配给不同的机器进行分布式计算,在每次迭代完后对各机器的计算结果进行融合,直至算法收敛。机器学习算法的分布式实现的主要障碍在于:1)分布式计算中各机器之间的通信延迟较单机下的通信延迟多几倍;2)分布式计算下容易因宕机或各机器计算结果的差异影响模型收敛,因此容错性也是算法分布式实现的主要瓶颈。分布式机器学习在实现时需要将分布式机器学习平台、机器学习算法以及分布式优化算法三者结合才能达到分布式计算的理想效果。模型训练可简化如下:

$$\min f(W), f(W)=\frac{1}{n}\sum_{i=1}^{N} f_i(W) \tag{1}$$

目标函数 $f_i(\cdot)$ 是第 i 次迭代的代价函数值,模型通过不断迭代找到最佳拟合训练数据的参数值 W,从而使得模型效果最优。机器学习算法的分布式实现主要分两部分:1)计算工作负载的分布式划分,针对不同的模型特性,机器学习算法在集群上的分布式实现方式主要有数据并行和模型并行;2)如何确保各机器之间的计算模型,以达到算法一致性。

如图 8 所示,Worker0 和 Worker1 实现了数据并行,Machine0－Machine3 实现了模型并行,每个 Task 完成一部分计算和传输操作。单箭头表示参数的获取与更新,双向箭头表示机器之间的通信,通常有 gRPC 和 RDMA 等数据通信机制。

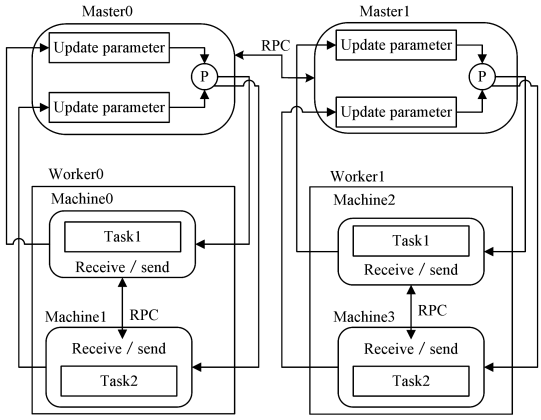

图 8　分布式计算模型

Fig. 8　Distributed computation model

3.1.1　数据并行

机器学习是一个迭代收敛的过程,即给定数据 D 和目标函数 $\mathcal{L}$,不断迭代更新参数 θ,直到 θ 达到终止条件。

$$\theta^{(t+1)}=\theta^{(t)}+\varepsilon\Delta_{\mathcal{L}}(\theta^{(t)}, D^{(t)}) \tag{2}$$

其中,上标 t 是算法的第 t 次迭代计数;θ 是模型参数;$\Delta()$ 为更新函数,将当前状态下的 $\theta^{(t)}$ 和 $D^{(t)}$ 作为输入,计算并返回更新值,再将更新值和当前参数状态值 $\theta^{(t)}$ 相加,从而将参数更新到新的状态 $\theta^{(t+1)}$;ε 为学习率。数据并行参数的更新公式如下:

$$\theta^{(t+1)}=\theta^{t}+\varepsilon\sum_{p=1}^{P}\Delta_{\mathcal{L}}(\theta^{(t)}, D_p^{(t)}) \tag{3}$$

数据并行如图 9 所示,将数据划分为 P 部分并分配给工作结点,第 p 份数据记为 D_p,每个工作结点独立地计算更新函数 $\Delta()$,最后将结果累加,从而将参数更新到新的状态 $\theta^{(t+1)}$。

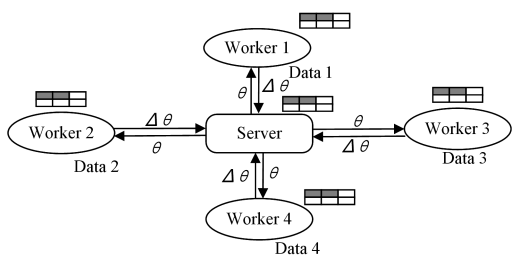

图 9　数据并行示意图

Fig. 9　Data parallel

3.1.2 模型并行

模型并行如图 10 所示，模型并行参数的更新公式如下：

$$\theta^{(t+1)}=\theta^{(t)}+Con(\{\Delta_p(\theta^t,S_p^{(t)}(A^{(t)})\}_{p=1}^{P}) \quad (4)$$

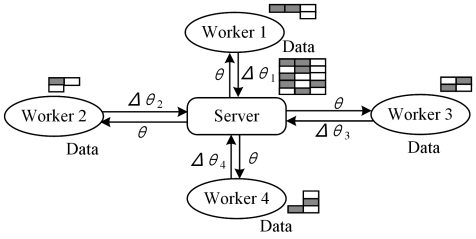

图 10 模型并行示意图

Fig. 10 Model parallel schematic

在分布式机器学习模型并行中，将模型 θ 分割为 P 部分并划分给工作结点进行分布式计算，更新函数 $\Delta()$，由于每个 Δ_p 影响多维参数 θ 的不同元素，因此 Δ_p 表示工作结点 p 的更新状态值。模型并行需要一个调度函数 $S_p^{(t)}(A)$ 以限制 $\Delta_p()$ 更新相对应的工作结点 Worker1，Worker2，Worker3，… 的参数 θ_{j1}，θ_{j2}，θ_{j3}，…。

3.2 机器学习算法的分布式计算模型

阻碍机器学习算法的分布式实现模型效率呈线性增长的主要因素在于多台机器下模型参数的更新以及通信耗时，即既要求不同机器上的网络参数独立计算与更新，又须保证新一轮迭代时各机器获取的参数基本一致。受硬件性能、资源占用、网络带宽等多因素的影响，同一模型在不同机器下的计算速度不同，而每次迭代需要获取所有的参数并进行更新，这对模型的计算效率产生了影响。为此，学者们提出了众多分布式机器学习并行计算模型与优化算法。并行计算模型是分布式算法实现的基础，模型参数更新方式有同步更新和异步更新。同步更新可保证算法的收敛率，但由于计算资源层次不齐，导致资源利用率较低；异步更新无需等待其他机器的参数更新，提高了资源利用率，但无法保证收敛效果。在分布式机器学习计算模型中，常见的并行模型有整体同步并行模型 BSP、异步并行模型 ASP 和延迟异步并行模型 SSP，更有如 Petuum 平台采用的既能保证收敛速度又能达到最优效果的 ESSP 模型。

3.2.1 整体同步并行模型

整体同步并行模型(Bulk Synchronous Parallel Model)由一系列计算工作结点和参数管理结点组成，各工作结点下的机器负责一部分数据计算并将参数 Push 到对应的参数管理结点，参数管理结点接收到所有工作端结点的参数后进行整合，并统一同步更新各工作结点参数进行新一轮迭代计算。整体同步并行模型的计算模型如图 11 所示。

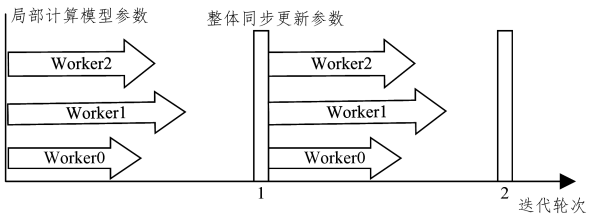

图 11 整体同步并行模型

Fig. 11 Bulk synchronous parallel model

较快的 Worker 结点执行完后，需要等待最慢的结点执行完，参数管理端接收到全部参数后，才进行同步更新。BSP 模型的优点在于模型可达到强一致性同步，准确率高；缺点在于每次迭代较快的计算结点都须等待慢结点执行，因此浪费了大量的计算资源，且效率较低。

3.2.2 异步并行模型

异步并行模型(Asynchronous Parallel Model)由一系列计算工作结点和参数管理结点组成，其计算模型如图 12 所示。以互相不干扰、互不等待为执行原则，各工作结点负责一部分计算，将这部分参数 Push 到相应的参数管理结点进行参数同步更新，执行下一次迭代计算。ASP 模型能达到最高的计算资源利用率，但各部分工作结点计算完后直接进行参数更新，这会导致在每次迭代计算中每个工作结点获取到不同的参数，造成模型计算结果不一致，从而导致模型的准确率降低。

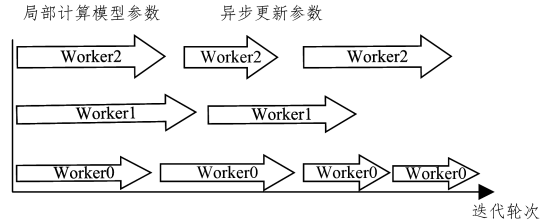

图 12 异步并行模型

Fig. 12 Asynchronous parallel model

3.2.3 延迟异步并行模型

延迟异步并行模型(Stale Synchronous Parallel Model)由一系列计算工作结点、参数管理结点和延迟参数 s 组成，它是结合了 BSP 优点和 ASP 优点的并行计算模型，其计算模型如图 13 所示。

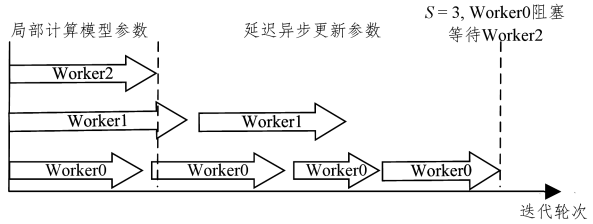

图 13 延迟异步并行模型

Fig. 13 Stale synchronous parallel model

各工作结点以不同的迭代轮数计算并更新参数，同时模型设置了一个阈值 s，当计算结点与最慢的计算结点之间的迭代次数的差值达到 s 时，阻塞该结点，等待最慢的结点执行，直到差值小于 s 再继续执行计算并进行参数更新。SSP 在尽量保证资源利用率的同时提高了模型结果的一致性，是目前很受开源分布式平台欢迎的分布式算法计算模型。

机器学习与传统算法最大的区别在于，需要多次迭代优化目标函数以达到模型最优，因此优化算法在参数调整取值中扮演着重要角色。亢良伊等[32]详细概述了单线程机器学习优化算法的原理，并根据适用于不同类型的目标函数，从并行执行和分布式执行两个角度进行了深入阐述。

4 分布式机器学习研究的未来趋势

分布式机器学习作为当下最热门的研究领域之一，吸引了越来越多学术界和工业界人士的目光。尽管分布式机器学

习已成功应用于许多领域,但目前仍存在许多亟待解决的问题。文献[33]简单阐述了分布式机器学习的发展趋势,提出主要从算法设计和算法优化技术的角度来提高性能。作者认为分布式机器学习未来的发展趋势主要包括以下几个方向。

(1)平台性能的改进。尽管目前分布式机器学习平台与日俱进,但各平台的性能参差不齐。首先,各平台任务的管理与调度、资源的控制和分配依然过多依赖于人工,这直接造成了资源的浪费和平台的低效;其次,各平台的应用领域仍具有较大的局限性,各大平台试图支持更多的网络模型,以扩大应用范围;再者,各平台之间尚未实现开放的生态体系,导致分布式机器学习的生态圈过于碎片化。文献[34]设计的鲲鹏分布式计算平台将大规模分布式系统和优化算法结合,既支持有向无环图和多种数据同步,又实现了强大的容错功能,将复杂的通信及调度过程包装成 API,从而快速实现模型同步。文献[35]设计的 NSML 实现了计算可视化、GPU 自动分配、模型参数快照,开发者只需注重模型设计即可。

(2)算法的优化。不同的平台由于特性不同,所支持的算法存在差异,且算法在不同领域的实际性能也有所差别,如在图像识别领域有较好应用的深度卷积神经网络在语音识别、机器翻译领域应用时效果不明显,因此算法的设计需要更强的普适性。其次,算法的应用不仅需要考虑时间复杂度和空间复杂度,在分布式环境下,也需要保证数据通信完整性与结果一致性。Hinton[36] 提出了一种新算法(神经胶囊网络),推翻了卷积神经网络,认为神经胶囊网络是更贴合人脑功能的高级网络,该网络目前在 MINIST,CIFAR10 数据集上的验证效果良好。文献[37]基于完全卷积结构的 LSTM 网络在 4D 医疗图片的分类应用上的准确率有较大提升,文献[38]使用深度强化学习来解决 TSP 问题,在车辆路径方面优于中规模启发式算法。文献[39]使用编码技术解决在模型训练中数据通信和矩阵计算两个阶段占用大量时间的问题,提高了机器学习算法的效率。

(3)模型的通信方式。分布式机器学习中,多计算结点的部署实现算法的并行化在减少计算时间的同时,也带来了模型准确率降低和通信开销增加的问题。通信开销受模型大小和维度、网络资源、结点之间的距离等多因素影响,目前众多学者致力于该问题的研究。文献[8]遵循 CAP 定理设计了一系列方法,在不影响模型准确率的情况下最小化通信开销;基于分层模型结构设计,寻找无依赖关系的计算与通信操作,将其重叠执行以减少模型开销。

(4)大规模计算下算法的可扩展性。分布式机器学习与传统机器学习的主要区别在于可扩展性,分布式机器学习希望增加的机器数与并行处理的数据量呈线性相关,多机器下的计算导致集群与数据通信占用了大部分时间,将计算和通信分离,二者并行运行将会减少大量时间,实现线性可扩展。

(5)分布式环境下模型的容错性。在分布式机器学习中,我们需要保证结果的一致性与有效性,因此分布式机器学习与传统的顺序执行不同,需要保证任一机器的宕机或计算结果出错不会影响总体效果。在分布式机器学习中,模型训练通常在离线状态下完成,传统意义上的容错是指一台机器宕机或执行任务失败,重启机器或任务不影响总体结果即可。而在大规模集群中,单台机器的重启不仅浪费资源,更破坏了模型同步的平衡。例如,支持分布式计算的 MPI 因缺乏容错机制,导致在工作结点很多的情况下效率大打折扣。

随着分布式机器学习的不断发展,分布式平台层出不穷,主流的分布式平台支持的硬件类型越来越多,从云端迁移到移动端,然而各个框架过于碎片化,ONNX 开放的生态体系倡导平台互通互用,使得共享成为一种新趋势。除了有形的技术不断进步外,无形的互联网文化也亟需成文的法律与管理机制。由 Google 领头的 federating learning 分散人工智能算法致力于保护用户数据隐私,加强了用户数据隐私保护。现有的算法仍然存在一些亟待解决的问题,如快速训练、模型小型化、训练平台多元化、预训练精准化都是目前需要努力突破的问题。至 2017 年,ImageNet 比赛正式宣告结束,新的标杆 COCO 已接棒出发,简单的图像分类、物体识别的效果已经趋于完美,未来希望计算机可以在识别物体的基础上推断出物体的动作,理解图片中的环境,结合 NLP 给出相应的图片描述并进行复杂的 QAs 以及自动驾驶。

结束语　分布式机器学习作为机器学习中的重要分支,近几十年来得到了越来越多的关注,分布式机器学习平台和算法得到了飞速发展。本文从分布式视角出发,详细概述了大数据环境下分布式机器学习的重要性和必要性,从数据流模型、参数服务器模型和混合模型 3 个方面全面分析了 Spark,MXNet,Petuum,TensorFlow 和 PyTorch 平台,并从多个角度对比了它们的性能;然后概述了机器学习算法的分布式实现方式以及分布式计算模型,旨在说明分布式机器学习的发展是平台与算法共同优化的结果;最后从平台研究、算法设计、可扩展性、通信方式和容错性方面讨论了分布式机器学习研究未来的发展趋势,以期为后续研究者提供重要参考。

参 考 文 献

[1] PRESS G. A very short history of big data[EB/OL]. https://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/#3cf546e65a18.

[2] XING E P,HO Q,XIE P,et al. Strategies and principles of distributed machine learning on big data[J]. Engineering,2016,2(2):179-195.

[3] HE Q,LI N,LUO W J,et al. A survey of machine learning algorithms for big data[J]. Pattern Recognition and Artificial Intelligence,2014,27(4):327-336. (in Chinese)
何清,李宁,罗文娟,等. 大数据下的机器学习算法综述[J]. 模式识别与人工智能,2014,27(4):327-336.

[4] ZHANG K,ALQAHTANI S,DEMIRBAS M. A Comparison of Distributed Machine Learning Platforms[C]// 2017 26th International Conference on Computer Communication and Networks (ICCCN). IEEE,2017:1-9.

[5] LIU B,HE J R,GENG Y J,et al. Recent advances in infrastructure architecture of parallel machine learning algorithms[J]. Computer Engineering and Applications,2017,53(11):31-38. (in Chinese)
刘斌,何进荣,耿耀君,等. 并行机器学习算法基础体系前沿进展综述[J]. 计算机工程与应用,2017,53(11):31-38.

[6] WANG Z,LIAO J,CAO Q,et al. Friendbook:a semantic-based friend recommendation system for social networks[J]. IEEE Transactions on Mobile Computing,2015,14(3):538-551.

[7] BOUAKAZ A,TALPIN J P,VITEK J. Affine data-flow graphs for the synthesis of hard real-time applications[C]// 2012 12th International Conference on Application of Concurrency to System Design (ACSD). IEEE,2012:183-192.

[8] AKIDAU T,BRADSHAW R,CHAMBERS C,et al. The dataflow model:a practical approach to balancing correctness,latency,and cost in massive-scale,unbounded,out-of-order data processing[J]. Proceedings of the VLDB Endowment,2015,8(12):1792-1803.

[9] MENG X,BRADLEY J,YAVUZ B,et al. Mllib:Machine learning in apache spark[J]. The Journal of Machine Learning Research,2016,17(1):1235-1241.

[10] LU J,WU D,MAO M,et al. Recommender system application developments:A survey[J]. Decision Support Systems,2015,74(C):12-32.

[11] ALEXANDER M,NARAYANAMURTHY S. An architecture for parallel topic models[J]. Proceedings of the VLDB Endowment,2010,3(1):703-710.

[12] LI M,ZHOU L,YANG Z,et al. Parameter server for distributed machine learning[C]// Big Learning NIPS Workshop. 2013.

[13] LI M. Scaling Distributed Machine Learning with the Parameter Server[C]// International Conference on Big Data Science and Computing. ACM,2014.

[14] LI M,ANDERSEN D G,SMOLA A J,et al. Communication efficient distributed machine learning with the parameter server [C]// Advances in Neural Information Processing Systems. 2014:19-27.

[15] HO Q,CIPAR J,CUI H,et al. More effective distributed ml via a stale synchronous parallel parameter server[C]// Advances in neural information processing systems. 2013:1223-1231.

[16] AHMED A,SHERVASHIDZE N,NARAYANAMURTHY S,et al. Distributed large-scale natural graph factorization[C]// Proceedings of the 22nd International Conference on World Wide Web. ACM,2013:37-48.

[17] DEAN J,CORRADO G,MONGA R,et al. Large scale distributed deep networks[C]// Advances in neural information processing systems. 2012:1223-1231.

[18] XING E P,HO Q,DAI W,et al. Petuum:A new platform for distributed machine learning on big data[J]. IEEE Transactions on Big Data,2015,1(2):49-67.

[19] DROR G,KOENIGSTEIN N,KOREN Y,et al. The yahoo! music dataset and kdd-cup'11[C]// Proceedings of KDD Cup 2011. 2012:3-18.

[20] HE K,ZHANG X,REN S,et al. Deep residual learning for image recognition[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016:770-778.

[21] KRIZHEVSKY A,SUTSKEVER I,HINTON G E. Imagenet classification with deep convolutional neural networks[C]// Advances in neural information processing systems. 2012:1097-1105.

[22] DAI W,KUMAR A,WEI J,et al. High-Performance Distributed ML at Scale through Parameter Server Consistency Models[C]// 29th AAAI Conference on Artificial Intelligence (AAA-15). 2015:79-87.

[23] LIAW A,WIENER M. Classification and regression by random Forest[J]. R News,2002,2(3):18-22.

[24] HOSMER J D W,LEMESHOW S,STURDIVANT R X. Applied logistic regression[M]. New York:John Wiley & Sons,2013.

[25] ABADI M,BARHAM P,CHEN J,et al. TensorFlow:A System for Large-Scale Machine Learning[J]. arXiv:1605.08695,2016.

[26] ARVIND,CULLER D E. Dataflow Architectures[J]. Annual Review of Computer Science,2010,1(1):225-253.

[27] SAK H,SENIOR A,BEAUFAYS F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling[C]// Fifteenth Annual Conference of the International Speech Communication Association. 2014.

[28] SUTSKEVER I,VINYALS O,LE Q V. Sequence to sequence learning with neural networks[C]// Advances in neural information processing systems. 2014:3104-3112.

[29] VISHNU A,SIEGEL C,DAILY J. Distributed tensorflow with MPI[J]. arXiv:1603.02339,2016.

[30] JIA Y,SHELHAMER E,DONAHUE J,et al. Caffe:Convolutional architecture for fast feature embedding[C]// Proceedings of the 22nd ACM International Conference on Multimedia. ACM,2014:675-678.

[31] GOODFELLOW I,BENGIO Y,COURVILLE A. Deep learning [M]. Massachusetts:MIT press,2016.

[32] KANG L Y,WANG J F,LIU J,et al. Survey on parallel and distributed optimization algorithms for scalable machine learning [J]. Journal of Software,2018,29(1):109-130. (in Chinese)
亢良伊,王建飞,刘杰,等. 可扩展机器学习的并行与分布式优化算法综述[J]. 软件学报,2018,29(1):109-130.

[33] LIU T Y,CHEN W,WANG T. Distributed machine learning:Foundations,trends,and practices[C]// Proceedings of the 26th International Conference on World Wide Web Companion. International World Wide Web Conferences Steering Committee,2017:913-915.

[34] ZHOU J,DING Y,et al. KunPeng:Parameter Server based Distributed Learning Systems and Its Applications in Alibaba and Ant Financial[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM,2017:1693-1702.

[35] SUNG N,KIM M,JO H,et al. NSML:A Machine Learning Platform That Enables You to Focus on Your Models[J]. arXiv:1712.05902.

[36] SABOUR S,FROSST N,HINTON G E. Dynamic routing between capsules[C]// Advances in Neural Information Processing Systems. 2017:3859-3869.

[37] GAO Y,PHILLIPS J M,ZHENG Y,et al. Fully convolutional structured LSTM networks for joint 4D medical image segmentation[C]// 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). IEEE,2018:1104-1108.

[38] NAZARI M,OROOJLOOY A,SNYDER L V,et al. Deep Reinforcement Learning for Solving the Vehicle Routing Problem [J]. arXiv:1802.04240.

[39] LEE K,LAM M,PEDARSANI R,et al. Speeding up distributed machine learning using codes[J]. IEEE Transactions on Information Theory,2017,PP(99):1.