Journal of Computer Applications
计算机应用,2017,37(11):3039-3047,3052
ISSN 1001-9081
CODEN JYIIDU
2017-11-10
http://www.joca.cn

文章编号:1001-9081(2017)11-3039-09
DOI:10.11772/j.issn.1001-9081.2017.11.3039

大数据下的典型机器学习平台综述

焦嘉烽,李　云*
(南京邮电大学 计算机学院,软件学院,网络空间安全学院,南京 210003)
(*通信作者电子邮箱 liyun@njupt.edu.cn)

摘　要:由于大数据海量、复杂多样、变化快,传统的机器学习平台已不再适用,因此,设计一个高效的、通用的大数据机器学习平台成为目前的研究热点。通过介绍和分析机器学习算法的特点以及大规模机器学习的数据和模型并行化,引出常见的并行计算模型。简单介绍了整体同步并行模型(BSP)、SSP 并行计算模型以及 BSP、SSP 模型与 AP 模型的区别,主要介绍了基于这些并行模型的典型的机器学习平台和这些平台的优缺点,并指出各个平台最适合处理何种大数据问题。最后从采用的抽象数据结构、并行计算模型、容错机制等方面对典型的机器学习平台进行了总结,并提出一些建议和展望。

关键词:大数据;机器学习平台;并行计算模型;参数服务器
中图分类号:TP311　　**文献标志码**:A

Review of typical machine learning platforms for big data

JIAO Jiafeng, LI Yun*
(*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing Jiangsu* 210003, *China*)

Abstract: Due to the volume, complex and fast-changing characteristics of big data, traditional machine learning platforms are not applicable. Therefore, designing an efficient and general machine learning platform for big data has become an important research issue. By introducing and analyzing the characteristics of machine learning algorithms and the data and model parallelization for large-scale machine learning, some common parallel computing models were presented. Bulk Synchronous Parallel (BSP), Stale Synchronous Parallel (SSP) computing models and the differences between BSP, SSP, and Asynchronous Parallel model (AP) were introduced. Then the typical machine learning platforms based on these parallel models and the advantages and disadvantages of these platforms were mainly introduced, and what kind of big data each typical machine learning platform was best suited for was pointed out. Finally, the typical machine learning platforms were summarized from the aspects of abstract data structure, parallel computing model and fault tolerance mechanism. Some suggestions and prospects were put forward.

Key words: big data; machine learning platform; parallel computing model; parameter server

0　引言

近年来,随着数据收集手段的丰富及数据存储能力的提升,公司、企业存储的以及科学研究(如:脑电信号分析等)产生的数据量急剧增加。对大数据进行科学的分析来获取更加有价值的信息是一项具有挑战性的任务,大数据机器学习则是完成这项任务的关键技术。目前机器学习应用广泛,但是机器学习处理大数据的效率不高,主要面临两大类挑战:大数据和大模型。当需要处理的数据量达到 PB、EB 级别时,单台高性能计算机已经无法在较短的时间内给出计算结果,因此学术界提出了许多并行计算模型,为提高计算效率提供了新的方案。常见的并行计算模型[1-2]有并行随机存取器(Parallel Random Access Machine, PRAM)、LogP[3]、块分布模型(Block Distributed Model,BDM)、整体同步并行模型(Bulk Synchr-onous Parallel model, BSP)[4-5]、AP(Asynchronous Parallel)[4]和 SSP(Stale Synchronous Parallel)模型[6]等。由于并行计算模型众多,因此如何设计出一个高效的大数据机器学习平台[7]或者框架成为了目前的一个研究热点,并取得了一些成果。图 1 列举了常见的机器学习平台,这些机器学习平台是基于 BSP、AP、SSP 并行模型的,并且采用各自的编程范式。

图 1 中的机器学习平台借助高级编程语言实现各自的编程模型,并基于这些编程模型实现各类机器学习算法[8],编程模型能够对相应的并行计算模型进行仿真。图 1 中还列举了部分商业机器学习平台:AmazonMachineLearning(https://aws.amazon.com/cn/machine-learning/)、微软 Azure(https://azure.microsoft.com/zh-cn/services/machine-learning/)、百度 BML(https://cloud.baidu.com/product/bml.html)、GoogleCloudPlatform(https://cloud.google.com/products/machine-learning/)、阿里云 DTPAI(https://data.aliyun.com/product/learn?spm=5176.8142029.388261.122.CnBRoG)、IBMSystemML(http://systemml.incubator.apache.org/)、腾讯 TML(https://cloud.tencent.com/product/tml)等。

收稿日期:2017-05-16;**修回日期**:2017-07-21。
基金项目:国家自然科学基金资助项目(61603197);江苏省自然科学基金资助项目(BK20140885)。
作者简介:焦嘉烽(1991—),男,江苏江阴人,硕士研究生,主要研究方向:大规模机器学习;　李云(1974—),男,安徽望江人,教授,博士生导师,博士,主要研究方向:机器学习、特征工程。

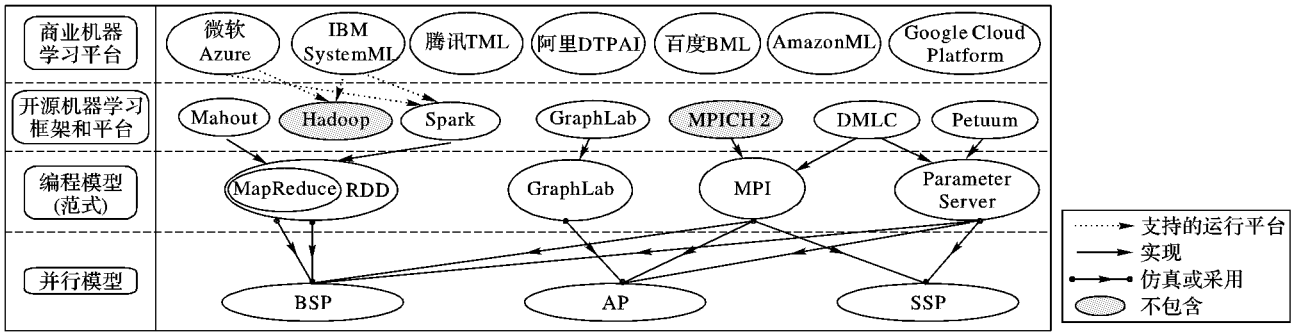

图 1 典型的机器学习平台架构

Fig. 1 Architecture of typical machine learning platforms

1 机器学习算法的特点及并行化

1.1 机器学习算法的数学表达

在完成数据预处理(包括数据清理、数据集成、数据变换等)以及特征工程之后，机器学习算法通常是解决一些最优化问题，其目标是训练模型的参数 $\boldsymbol{A}$，使得该模型能够拟合 N 个数据样本 $D=\{\boldsymbol{x}_i,y_i\}_{i=1}^N$($y_i$ 表示样本 $\boldsymbol{x}_i$ 的标记，y_i 的有无根据实际应用而定)。在这个最优化问题中，常常使用一种损失函数来描述模型在给定样本 $\boldsymbol{x}_i$ 的预测值与其真实值的不一致程度，这种损失函数一般采用最小二乘损失。但是训练好的模型通常存在过拟合问题，因此在损失函数之后加上正则化项来防止模型过拟合。最后这类机器学习算法可表示为如下的数学表达式：

$$\max_{\boldsymbol{A}} L(D,\boldsymbol{A}) \text{ OR } \min_{\boldsymbol{A}} L(D,\boldsymbol{A}) \tag{1}$$

$$\text{where } L(D,\boldsymbol{A}) = f(\boldsymbol{x}_i,y_i)_{i=1}^N + r(\boldsymbol{A})$$

其中：$f(\boldsymbol{x}_i,y_i)_{i=1}^N$ 是损失函数，$r(\boldsymbol{A})$ 是正则化项。对于式(1)，常常借助随机梯度下降法[9]、坐标下降法[10]以及变分推理[11]等算法来求解，求解过程中会得到一系列迭代收敛的等式，可归结为如下形式：

$$\boldsymbol{A}^{(t)} = F(\boldsymbol{A}^{(t-1)},\Delta_L(\boldsymbol{A}^{(t-1)},D)) \tag{2}$$

其中 t 表示迭代轮数，该等式表示从模型参数 $\boldsymbol{A}^{(t-1)}$ 和数据 D 通过更新函数 $\Delta_L()$ 和聚合函数 $F()$ 生成下一次迭代的模型参数 $\boldsymbol{A}^{(t)}$，通过迭代上述等式至 L 达到最优值或者达到收敛条件。综上所述，这类迭代式机器学习算法由如下部分组成：

1) 数据 D 和模型参数 $\boldsymbol{A}$；

2) 损失函数 $f(D,\boldsymbol{A})$；

3) 正则化项 $r(\boldsymbol{A})$；

4) 最优化算法(例如随机梯度下降法、坐标下降法)。

1.2 机器学习算法的特点

由于机器学习问题通常采用梯度下降类算法求解，因此机器学习算法有如下特点[12]：

1) 容错(Error Tolerance)。在使用随机梯度下降等最优化算法求解机器学习问题时通过迭代的方式来求解，因此只需保证每一步计算得到的梯度是在收敛的方向上，而不需要保证每一步的梯度都完全计算正确。该特点为大数据机器学习系统的设计提供了折中的空间。

2) 依赖结构(Dependency Structure)。在更新函数 $\Delta_L()$ 中有些变量依赖于其他算式的值而无法实现并行化。

3) 非一致收敛(Non-uniform Convergence)。指不是所有模型参数 $\boldsymbol{A}$ 都能在相同的迭代次数之后收敛到最优值 $\boldsymbol{A}^*$。

在大规模机器学习中，数据 D 和模型参数 $\boldsymbol{A}$ 将会非常巨大，因此需要采用并行化方式来提高效率，即数据并行化和模型并行化。

1.3 数据并行化

在数据并行化中，数据 D 被分片后分配给 P 个计算节点(用 $p=1,2,\cdots,P$ 进行编号)，这里假设数据是独立同分布的，更新函数 $\Delta_L()$ 在每一个数据分片计算得出的结果之后，采用求和的方式进行聚合(随机梯度下降法和基于采样的算法使用的就是这种聚合方式)。梯度下降优化算法有三种变型形式：1) 批梯度下降法(batch gradient descent)；2) 随机梯度下降法(Stochastic Gradient Descent，SGD)；3) 小批量梯度下降法(mini-batch gradient descent)，详细内容见文献[13]。例如在采用随机梯度下降法进行最优化的机器学习问题中，更新函数在每个 D_p 上计算出中间结果(次梯度)，将这些次梯度求和之后乘以学习率 η 再更新模型参数 $\boldsymbol{A}^{(t)}$，针对随机梯度下降法的学习率用 AdaGrad[14] 和 AdaDelta[15] 算法进行计算，则可得到如下数据并行的更新等式(省略了学习率)：

$$\boldsymbol{A}^{(t)} = F\left(\boldsymbol{A}^{(t-1)},\sum_{p=1}^{P}\Delta_L(\boldsymbol{A}^{(t-1)},D_p)\right) \tag{3}$$

1.4 模型并行化

在模型并行化中，模型参数 $\boldsymbol{A}$ 分片之后，分配给 P 个计算节点($p=1,2,\cdots,P$)，每个计算节点并行的计算更新函数 $\Delta_L()$ 并更新模型参数 $\boldsymbol{A}$。与数据并行化不同的是模型并行化中的更新函数 $\Delta_L()$ 的参数中包含调度函数 $S_p^{(t-1)}()$。模型并行更新等式如下：

$$\boldsymbol{A}^{(t)} = F(\boldsymbol{A}^{(t-1)},\{\Delta_L(\boldsymbol{A}^{(t-1)},S_p^{(t-1)}(\boldsymbol{A}^{(t-1)})\}_{p=1}^P) \tag{4}$$

因为数据 D 没有被分片，这里为了简洁明了将其省略。调度函数的作用是限制 $\Delta_L()$ 只在指定的模型参数子集进行更新操作，因为模型参数子集之间不一定是相互独立的，但是只能在相互独立的模型参数子集或者弱相关的模型参数子集上进行并行更新。调度函数 $S_p^{(t-1)}()$ 通过输出下标集合 $j1,j2,\cdots$ 来限制更新函数 $\Delta_L()$，只更新模型子集 $\boldsymbol{A}_{j1},\boldsymbol{A}_{j2},\cdots$。上文给出了理论上的数据和模型并行化方式，但实际应用时还需要借助底层的并行计算模型来实现真正的并行化，下面就介绍大数据机器学习平台下常用的并行计算模型。

2 并行计算模型

并行计算模型通常指从并行算法的设计和分析出发，抽象出各类并行计算机(至少某一类并行计算机)的基本特征，

形成一个抽象的计算模型。从广义上来说,并行计算模型为并行计算提供了硬件和软件界面,在该界面的约定下,并行系统硬件设计者和软件设计者可以开发对并行性的支持机制,从而提高系统的性能。常见的并行计算模型有 PRAM、AP、BSP、LogP、SSP 模型等。因为 BSP、SSP 模型常被大数据机器学习平台所采用,所以本文主要介绍 BSP、SSP 模型以及与 AP 模型的差异。由于 AP 模型较为简单,所以只介绍一下 AP 模型的基本思想,即将计算和通信重叠,从而提高了计算效率。另外需要说明的是 MapReduce[16] 模型也常被大数据平台所采用,本文将 MapReduce 看作一种编程范式。

2.1　BSP 并行计算模型

BSP[5] 模型的创始人是英国著名的计算机科学家 Valiant,BSP 模型的目标是为现有和未来可能出现的各种并行体系结构提供一个独立于具体体系结构的理论模型基础,故又称之为桥接模型(Bridging Model)。

BSP 模型是包含如下 3 个部分的并行计算模型:

1) 计算组件(至少由处理器和存储器组成);

2) 路由器,为各个计算组件提供一个可通信的网络,实现各计算组件之间点对点的消息传递;

3) 执行间隔为 T 的栅栏同步器(Barrier Synchronisation)。

在 BSP 模型中,整个计算过程是由栅栏同步器分开的一系列计算部分组成(如图 2 所示),这些计算部分称为超步(Super Step)(如图 3 所示)。BSP 模型的独特之处就在于"超步"的引入,一个超步包括以下 3 个阶段:

1) 本地计算阶段。计算节点对本地数据进行计算,将计算的结果存储在本地的存储器,将需要发送到其他计算节点的消息数据存储到本地消息队列,等待发送。

2) 全局通信阶段。计算节点之间以点对点的方式进行通信。

3) 栅栏同步阶段。超步以栅栏同步为结束点,本次超步的数据通信在栅栏同步结束之后才生效,供下一超步使用。在确保通信过程中交换的数据被传送到目的计算节点上,并且每个计算节点完成当前超步中执行的计算和通信之后,才可以进入下一超步;否则停止等待其他节点完成计算和通信。

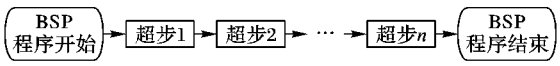

图 2　BSP 模型结构

Fig. 2　Structure of BSP model

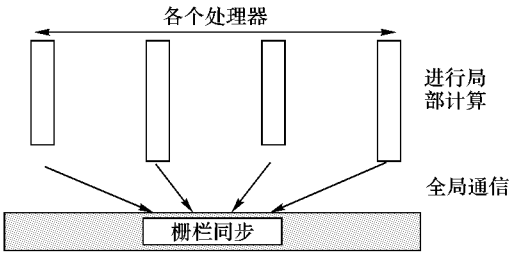

图 3　超步的结构

Fig. 3　Structure of super step

综上所述,BSP 模型可以理解为是水平方向的并行(各个计算节点并行计算)和垂直方向的串行(超步之间串行执行),有如下特点:

1)将整个计算过程细分为多个串行的超步,超步内并行计算,将通信和同步解耦,从而有效避免死锁;

2)强调计算任务和通信任务分离,而通信任务仅仅完成点对点的消息传递,不提供组合、复制和广播等功能,简化了通信协议;

3)超步之内各计算节点之间无需同步,超步之间各计算节点之间需要同步,因此 BSP 模型是介于严格同步的并行计算模型和异步的并行计算模型之间的模型。

2.2　SSP 并行计算模型

BSP 模型在实际应用中存在慢机(Straggler)现象,由于计算节点的实际性能有差异,整个计算进度由最慢的计算节点决定。于是 Cipar 等[6] 提出了 SSP 模型来解决这个问题。

SSP 模型是一个有界异步的桥接模型,该模型不仅可以解决 BSP 模型中的慢机问题,同时保留了 BSP 模型中同步机制的优点。假设应用 SSP 模型时采用主从式架构,其基本原理如下:假设有 P 个从计算节点,以迭代的方式来优化机器学习问题(Δ、F),每一个计算节点维护迭代计数器 t 和模型参数 $\boldsymbol{A}$ 的本地视图(相当于一份拷贝),在完成一轮迭代计算之后,每个计算节点提交本次运算得出的参数更新 Δ,然后执行如下操作:

1) 调用 clock()函数,表示完成了该次迭代计算;

2) 迭代计数器 t 增加 1;

3) 通知主节点将该节点的参数更新 Δ 传播给其他的计算节点,以便其他计算节点更新其模型参数的本地视图 $\boldsymbol{A}_{\text{loc}}$;

这里的 clock()函数的作用类似于 BSP 模型中的栅栏同步,不同的是在 SSP 模型中,由于每个节点是异步计算,所以一个计算节点提交的更新 Δ 不会立刻传递给其他节点,这就导致其他节点进行下一轮迭代时可能只接收到了部分参数更新,那么这些节点的模型参数 $\boldsymbol{A}_{pt}$(在第 t 轮第 p 个节点保存的模型参数)的本地视图就变得陈旧(Stale)。对于一个给定的陈旧阈值 $s \geqslant 0$(s 表示节点之间的迭代轮数差),基于 SSP 模型的并行系统必须满足以下有界陈旧条件(如图 4 所示):

1) 迭代轮数之差不大于给定的阈值。运行最快的和最慢的计算节点的迭代轮数之差必须不大于 s,否则最快的计算节点将被强制等待最慢的计算节点。

2) 提交的更新带有标签。在第 t 轮迭代结束,调用 clock()函数之前,提交的更新 Δ 需要带有迭代轮数 t。

3) 能够保证模型状态。一个计算节点第 t 轮计算更新 Δ 时,需保证该节点接收到了在 $[0, t-s-1]$ 轮内的所有模型参数更新。

4) 能够读取本地缓存。每一个计算节点直接用自己提交的更新 Δ 来更新模型参数 $\boldsymbol{A}$ 的本地视图。

SSP 模型的理论分析和收敛性分析参见文献[12,17]。

2.3　并行计算模型小结

作 BSP、SSP、AP 模型各有优缺点,适用于不同的应用场景,下面对 BSP、SSP 和 AP 模型作对比。这些模型之间的主要区别就在于运行原理不同(如图 4~6)。其中 AP 模型是异步并行模型,其优点是由于没有栅栏同步,因此不存在慢机问题,并且计算节点在运算的同时可以与其他节点进行通信,传递更新;但是难以维护数据一致性。BSP 模型中的同步机制保证了数据一致性,但是存在慢机问题。SSP 模型放松了一致性要求,解决了 BSP 模型中的慢机问题,并借鉴了 AP 模型的优点,在指定迭代轮数范围内允许各计算节点异步执行计

算任务。因此这些模型之间不存在孰优孰劣，只是各个模型适用于解决不同的大数据问题。AP 模型适合于解决实时的大数据问题；BSP 模型和 MapReduce 模型类似，适合于解决离线的大数据问题以及图计算；因为有界异步的执行方式与机器学习算法的容错特点契合，所以 SSP 模型相比 BSP 模型更适合于解决大数据机器学习问题。

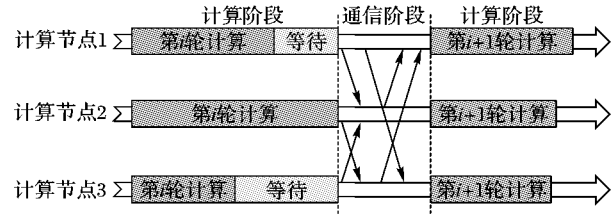

图 4 BSP 模型的运行原理示意图

Fig. 4 Execution of BSP model

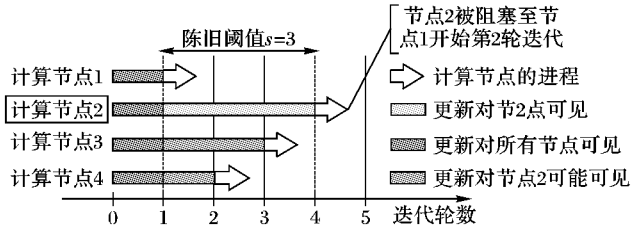

图 5 SSP 模型的运行原理示意图

Fig. 5 Execution of SSP model

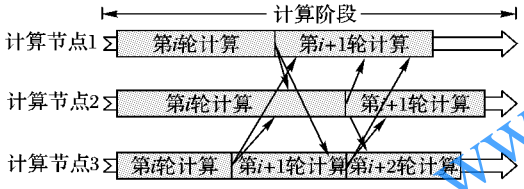

图 6 AP 模型的运行原理示意图

Fig. 6 Execution of AP model

3 大数据机器学习平台及相关编程模型

介绍完机器学习的相关背景和并行计算模型之后，接下来重点介绍大数据机器学习平台（包括框架）。GraphLab[18-19] 由卡内基·梅隆大学（Carnegie Mellon University，CMU）的 Select 实验室于 2010 年提出，是面向机器学习的流处理并行框架。GraphLab 适用于图计算，其基本思想是将数据抽象成 Graph 结构，将算法的执行过程抽象成 Gather、Apply、Scatter 三个步骤，其并行化的核心思想是对顶点的切分。还有 DMLC（http://dmlc.ml/）是一个开源分布式机器学习项目，集成了 XGBoost、MXNet、Minerva 等机器学习库与 dmlc-core、Rabit、mshadow 和参数服务器 ps-lite 等系统组件。因为 Apache Mahout 框架、Apache Spark 平台以及 Petuum 较为流行，所以本文主要介绍这三种平台（框架）。

3.1 Apache Mahout 框架

基于 Chu 等[20] 的文章，Apache Lucene（开源搜索）社区中对机器学习感兴趣的成员于 2008 年发起了 Mahout，2010 年 Mahout 成为了 Apache 顶级项目，Apache Mahout 的目标是为快速创建可扩展的高性能的机器学习应用提供运行和开发环境。在 Apache Mahout 0.10.0 版本之前，Mahout 主要是基于 MapReduce，运行于 Hadoop 平台。从 Apache Mahout 0.10.0版本开始支持运行于 Apache Spark 平台。Apache Mahout 的主要特征有：

1）是一个开源的机器学习软件库；

2）是为创建可扩展算法的一个既简单又扩展性好的编程环境和框架；

3）为 Hadoop、Scala + Apache Spark、H2O 等平台提供多种预制的机器学习算法；

4）提供类似于 R 语言语法的矢量数学实验环境 Samsara。

Apache Mahout 有 3 个核心主题：推荐引擎（协同过滤、频繁项集挖掘），聚类和分类。下面列出一些 Mahout 中的算法：随机矩阵的奇异值分解算法（SSVD（Stochastic Singular Value Decomposition）、DSSVD（Distributed Stochastic Singular Value Decomposition））、随机主成分分析算法（SPCA（Stochastic Principal Component Analysis）、DSPCA（Distributed Stochastic Principal Component Analysis））、朴素贝叶斯分类算法、协同过滤算法（Item 和 Row 的相似性）、分布式正则化交替最小二乘法（Distributed Alternating Least Squares，DALS）等。在结合开源的协同过滤项目 Taste 后，协同过滤算法成为 Apache Mahout 中最受欢迎的算法。下面简单介绍 Apache Mahout 0.10.0版本之前所采用的 MapReduce 编程模型。

3.1.1 MapReduce 编程模型

MapReduce[21] 是一种面向大规模数据处理的并行程序设计模式（parallel programming paradigm），由两名 Google 的研究员 Jeffrey Dean 和 Sanjay Ghemawat 在 2004 年时提出。Google 公司设计 MapReduce 的初衷主要是为了实现在搜索引擎中大规模网页数据的并行化处理，MapReduce 的推出给大数据并行处理带来了革命性的影响，一度成为大数据处理的工业标准。图 7 描述了典型的大数据处理的过程，MapReduce 将以上的处理过程抽象为两个基本操作。把前两步（a，b）操作抽象为 Map 操作，把后两步（d，e）操作抽象为 Reduce 操作。Map 操作主要负责对一组数据记录进行某种重复处理，而 Reduce 操作主要负责对 Map 的中间结果进行某种进一步的结果整理和输出。

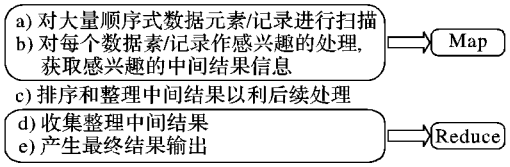

图 7 典型的顺序式大数据处理的过程

Fig. 7 Typical sequential process to deal with big data

由于 MapReduce 模型在完成一次同步通信需要消耗大量时间（每一次迭代后的中间结果需要写入磁盘，下一次迭代又从磁盘读取数据），以及粗粒度的容错机制（主要采用检查点 checkpoint 方式，阶段任务有一个失败需要全部重新计算），已经不能够满足用户对实时性、高效运算等方面的要求。随着内存计算 Apache Spark 的流行，Mahout 因其强大的可扩展能力，已经支持运行于 Apache Spark 平台。接下来介绍继承了 MapReduce 的可扩展性以及容错能力并且克服了部分 MapReduce 缺陷的 Apache Spark 平台，Apache Spark 涉及图处理、机器学习、流处理等多个大数据处理领域。

3.1.2 Apache Mahout 小结

Apache Mahout 简化了编程人员的工作。通常，在 Hadoop

云平台下编程不仅要求编程人员对 Hadoop 云平台框架比较熟悉,还要熟悉 Hadoop 云平台下底层数据流、Map 和 Reduce 原理,这是基本的编程要求。此外,在编写某一个算法时需要对算法原理有透彻的理解。因此,编写云平台下的算法程序是高难度的开发工作。但是,如果使用 Apache Mahout,那么编程人员就不用自己编写复杂的算法,不需要非常了解云平台的框架和数据流程的理论知识。只需要了解算法的大概原理、算法实际应用环境和如何调用 Apache Mahout 相关算法的程序接口。当然,在具体的项目中,编程人员需要根据实际需求在 Apache Mahout 源代码基础上进行二次开发以满足具体的实际应用情况。

3.2　Apache Spark 平台

Spark[22] 诞生于伯克利大学的 AMPLab(Algorithms Machines and People Lab),于 2010 年正式开源,于 2014 年成为 Apache 的顶级项目。Spark 在 2014 年 11 月的 Daytona Gray Sort 100 TB Benchmark 竞赛中打破了由 Hadoop MapReduce 保持的排序纪录。Spark 利用 1/10 的节点数,把 100 TB 数据的排序时间从 72 min 降低到了 23 min。在介绍 Apache Spark 之前,先介绍一下 AMPLab 的伯克利数据分析栈(BDAS(https://amplab.cs.berkeley.edu/software/))。如图 8 所示,BDAS 包括 5 层:资源虚拟化(Resource Virtualization)、存储系统(Storage)、处理引擎(Processing Engine)、访问接口(Access and Interfaces)以及内部应用(In-house Apps)。资源虚拟化主要指集群的管理和计算资源的调度;存储系统主要指分布式存储系统,具有高度容错性,提供高吞吐量的数据访问;处理引擎指通用的大数据处理引擎;访问接口指 BDAS 为各类大数据问题提供的解决方案的 APIs。图 8 中粗框中的就是 Apache Spark 平台。Spark 主要具有如下特点:

1)速度极快。使用有向无环图(Directed Acyclic Graph,DAG)执行引擎以支持循环数据流,并将中间数据放到内存中,迭代运算效率高。

2)易于使用。支持使用 Scala、Java、Python 和 R 语言进行编程,可以通过 Spark Shell 进行交互式编程。

3)通用性强。Spark 提供了完整而强大的组件库,包括 SQL 查询、流式计算、机器学习和图算法组件。

4)模式多样。可运行于独立的集群模式中,可运行于 Hadoop 平台,也可运行于 Amazon EC2 等云环境中,并且可以访问 HDFS、Cassandra、HBase、Hive 等多种数据源。

Spark 具有上述诸多优点主要是由于其采用了不同于 MapReduce 的数据结构,即弹性分布式数据集(Resilient Distributed Dataset,RDD)[23],下面就介绍一下 RDD。

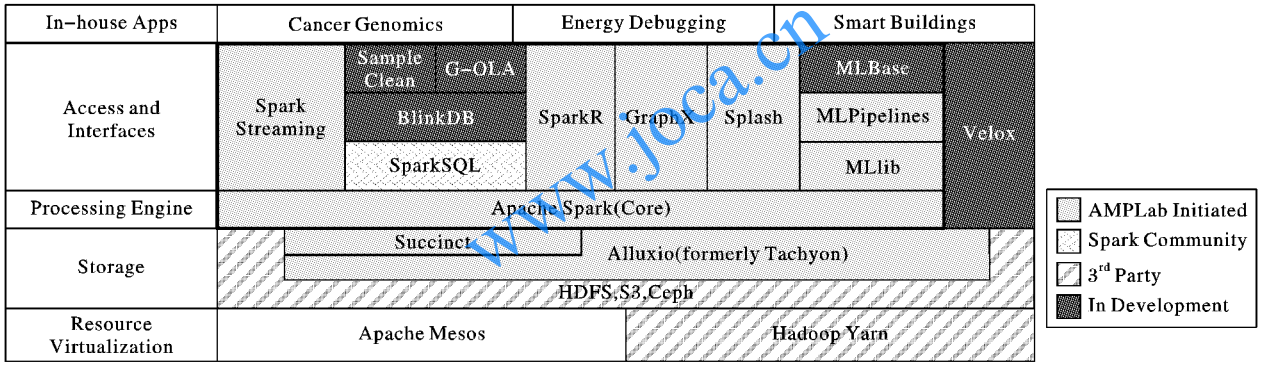

图 8　伯克利数据分析栈

Fig. 8　Berkeley data analytics stack

3.2.1　RDD

RDD 是 Apache Spark 平台的基础,RDD 有两层含义:

1)数据结构 RDD 是一个只读的、可分区的记录集合,一个 RDD 可以包含多个分区,每个分区就是一个数据集片段。RDD 本质上是一个内存数据集,解决了 MapReduce 中将中间结果写入磁盘或者 HDFS 带来的密集的磁盘读写和大量的网络通信开销问题。

2)编程模型 Spark 在 RDD 上定义了两类操作(如图 9 所示):转换(Transformation)和动作(Action),使得开发人员不必关心数据的分布式特性,只需将具体的应用逻辑编写为一系列 RDD 的操作。转换操作返回新的 RDD,而动作操作的结果是一个值或是将数据导入到存储系统。

Apache Spark 平台为了节约计算资源,在 RDD 上定义的采用惰性调用机制,只有在调用动作操作时才会真正触发所有定义的操作。由转换操作得到的父子 RDD 之间存在依赖关系(如图 10 所示),如果一个父 RDD 中的一个分区只被一个子 RDD 的一个分区使用,则称之为窄依赖(Narrow Dependency);若被一个子 RDD 的多个分区使用,则称之为宽依赖(Wide Dependency)。如果 RDD 之间是窄依赖,则可以在同一个计算节点上以管道形式(pipeline)执行这些 RDD 上的操作,从而避免了多个操作之间的数据同步。

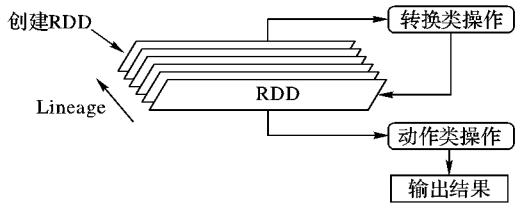

图 9　RDD 上的两大类操作

Fig. 9　Two kinds of operations on RDD

Spark 在容错性能方面有较大的提升,增加了 Lineage 容错机制。实际编程时,在一个 RDD 上调用的各种转换操作构成了计算链(Compute Chain),把这个 Compute Chain 看作是 RDD 之间演化的 Lineage(如图 9 所示)。因此在部分计算结果丢失时,只需要根据这个 Lineage 重算即可。因为当 RDD 之间是窄依赖时,重新计算中间丢失的 RDD 不需要依赖其他计算节点参与,所以此时采用 Lineage 容错的效率才比采用数据检查点(Check point)的高。当 RDD 之间是宽依赖或者 Lineage 过长时,设置数据检查点比采用 Lineage 的效率更高。

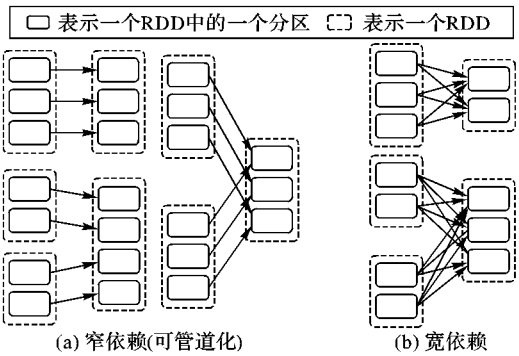

图 10 RDD 之间的依赖关系

Fig. 10 Dependency between RDDs

下面介绍一下基于 RDD 构建的 Spark 生态系统(如图 11 所示),由如下部分组成。

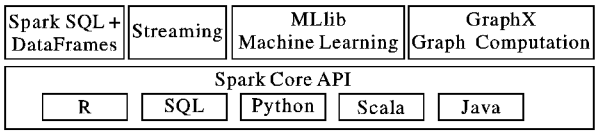

图 11 Spark 生态系统

Fig. 11 Spark ecosystem

1) Spark Core。定义了 Spark 的基本功能,包含任务调度、内存管理、容错恢复、与存储系统交互等。Spark Core 为上层组件提供了 Scala、Java、Python 和 R 语言的编程接口即 Spark 核心应用编程接口(Spark Core API)。

2) Spark Streaming。是构建在 Spark 上的流数据处理框架,基本原理是将流数据分成小的时间片断(例如:每 2 秒分割为一个片段),以类似批量处理的方式来处理这小部分数据。这种小批量处理的方式使得 Spark Streaming 可以同时兼容批量和实时数据处理的逻辑和算法。

3) Spark SQL。用于处理结构化数据,可使用类似于 Hive 查询语言(Hive Query Language, HQL)的 SQL 语句来查询数据。

4) Spark MLlib。是一个可扩展的 Spark 机器学习库,由通用的机器学习算法和工具组成,包括二元分类、线性回归、聚类、协同过滤、梯度下降等。

5) Spark GraphX。用于操作图(如社交网络的好友图)和执行基于图的并行计算。通过引入弹性分布式属性图(Resilient Distributed Property Graph),一种顶点和边都带有属性的有向多重图,可以使用与每个节点和边绑定的任意属性来创建一个有向图。GraphX 还提供了各种各样的操作图的操作符,以及通用的图算法。

其中 Spark Core 是其他 4 个部分的基础,Apache Spark 通过 Spark SQL、Spark Streaming、MLLib、GraphX 这些组件近乎完美地解决了大数据处理中批处理(Batch Processing)、流处理(Streaming Processing)和即席查询(Ad-Hoc Query)三大核心问题。下面详细介绍一下 Spark 机器学习库 MLLib。

3.2.2 Spark 机器学习库

Spark MLlib 是 Spark 的分布式机器学习库,包含丰富的机器学习算法实现。需要注意的是,Spark 机器学习库从 1.2 版本之后被分为两个包:

1) spark.mllib。是 Spark 最原始的机器学习库,其中的机器学习算法都是基于 RDD 实现。但是使用 spark.mllib 构建完整的机器学习工作流比较困难,需要用到 spark.ml。

2) spark.ml。在 Spark 1.2 版本中加入 spark.ml,spark.ml 引入了 ML Pipeline。ML Pipeline 提供了基于 DataFrame 的一套统一的高级 APIs,帮助用户创建和调整机器学习工作流。

使用机器学习来解决实际问题时通常包含一系列基本的步骤:数据预处理、特征提取、训练模型、模型验证等,可以将其看作是一个包含多个步骤的工作流。许多机器学习库不提供工作流中所需要的全部功能,大多数机器学习库只专注于一项功能,比如数据预处理或者特征提取;因此往往需要使用各种库来拼凑出一条机器学习工作流,这样做既费时又费力。现在只需要借助 spark.ml 便可完成这样的机器学习工作流的构建,而且从 Spark2.0 开始,spark.mllib 进入维护模式(即不增加任何新的特性),并预期于 3.0 版本的时候被移除出 spark.mllib,spark.ml 将成为 Spark 主要的机器学习库。Spark 机器学习库提供了通用机器学习算法的实现,目前支持 4 种常见的机器学习问题:分类、回归、聚类和协同过滤。表 1 列出了 Spark 机器学习库中主要的机器学习算法。

表 1 MLlib 中主要的机器学习算法

Tab. 1 Main machine learning algorithms in MLlib

机器学习类型	数据类型	
	离散	连续
监督学习	Logistics Regression, Support Vector Machine, Decision Tree, Random Forest, Naive Bayes, Gradient-Boosted Tree	Linear Regression, Decision Tree, Gradient-Boosted Tree, Random Forest
无监督学习	*K*-means, Gaussian Mixture Model, Bisecting *K*-means, Latent Dirichlet Allocation	Dimensionality Reduction, Matrix Factorization, Principal Component Analysis, Singular Value Decomposition

Spark 机器学习库还提供了一系列低级原语和基础工具用于凸优化、分布式线性代数、统计分析、特征提取等,还支持对各种输入输出格式化,并针对分布式机器学习有许多优化措施。例如在交替最小二乘法中,使用阻塞来减少 Java 虚拟机(Java Virtual Machine, JVM)垃圾回收的开销;在决策树算法中,离散化依赖数据的特征来降低通信开销。

3.2.3 Apache Spark 小结

Spark 提供了丰富的编程接口,支持多种编程语言,为开发人员带来了极大的便利。Spark 支持多种部署方式,包括单机模式、分布式模式。采用分布式模式部署时,通常采用主从式架构,虽然在高效的通信原语的基础上,Spark 机器学习算法能够高效运行,并且 Spark 能够快速将规模较大的模型分配给计算节点。但是主节点通常只有一个,这就导致 Spark 在训练大规模的机器学习模型,尤其是迭代式机器学习模型时,就有点力不从心。因为 Apache Mahout 和 Spark 都是基于 BSP 模型,在通信上的开销占据了整个计算过程开销的较大部分,而且每个计算节点都需要和主节点通信,因此训练大规模模型将十分缓慢。如果需要解决这个问题,可以采用 SSP 模型作为底层的并行计算模型。下面就介绍基于 SSP 模型的机器学习平台 Petuum。

3.3　Petuum

Petuum[24,29]是一个分布式机器学习平台。为了应对机器学习在大规模场景下的挑战:大数据(数据样本数目庞大)以及大模型(模型参数众多),Petuum 为数据并行和模型并行提供了通用的 APIs,从而简化了相关机器学习算法的实现。与通用的大数据处理平台(如 Hadoop、Spark)不同,Petuum 专门为处理机器学习问题而设计。Petuum 通过充分利用机器学习算法的特征(如迭代性、容错性、参数收敛的不均匀性等)来提高训练机器学习模型的效率。Petuum 系统由调度器(Scheduler)、参数服务器、计算节点(Workers)和 Petuum 机器学习算法组成,如图 12 所示。接下来介绍一下各部分的功能:

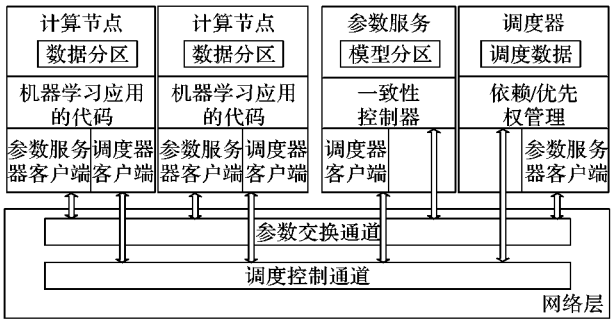

图 12　Petuum 系统的结构

Fig. 12　Structure of Petuum system

调度器　该模块负责模型并行化,允许开发人员通过自定义函数 schedule()(类似于式(4)中的)来控制计算节点在本轮迭代中仅对指定的模型参数子集进行更新。调度器通过调度控制通道将这些参数的标识(函数 schedule()的输出值)传递给计算节点,而实际的参数值由参数服务器通过参数交换通道分发。

计算节点　计算节点 p 在接收到由参数服务器分配的模型参数之后,在数据 D 或者数据分片上并行地执行更新函数 push()(类似于式(3)中的)。当计算节点执行 push()时,模型参数的本地视图 $\boldsymbol{A}_{\text{loc}}$ 会自动地通过参数交换通道与参数服务器进行同步。在 push()执行完毕之后,调度器将使用新的模型参数来决策下一轮迭代的调度。在 Petuum 系统中没有定义固定的数据抽象,所以计算节点可以读取内存中的数据、磁盘中的数据或者分布式文件系统中的数据;而且开发人员可以控制计算节点读取数据的顺序,在随机优化算法中计算节点可以一次仅采样一个数据点,在批处理算法中计算节点可以在一轮迭代中读遍所有的数据。

参数服务器　通过一个类似于键值存储的分布式共享内存 API 为开发人员提供了对模型参数 $\boldsymbol{A}$ 的全局访问。为了利用机器学习算法的特点,参数服务器是基于 SSP 并行计算模型实现的。机器学习算法大多是基于优化的算法,而且用递归来实现,有一定的容错能力,这些特点恰好与 SSP 模型中的有界陈旧相契合。

上述 Petuum 系统的功能主要由以下 3 个组件实现:

1) Bösen。针对数据并行机器学习编程是具备高效通信机制的分布式键值存储系统。

2) Strads。一个高速、高精度的可编程参数调度器,为了模型并行化的机器学习问题而设计。

3) Poseidon。一个基于 Caffe 的分布式多 GPU 深度学习编程框架。

Petuum 能够高效地运行在集群和云计算平台(比如 Amazon EC2 和 Google GCE)上。其中的参数服务器被誉为“MapReduce 的替代者”,下一节将详细阐述参数服务器。

3.3.1　参数服务器

参数服务器到目前为止可分为三代,Petuum 采用的是第二代参数服务器。第一代的原型可以追溯到文献[25],如图 13 所示。其中每个 sampler 处理与之相关的数据子集,同时有一个用于同步的线程,使本地模型参数和全局模型参数保持一致,采用一个分布式的 Memcached 存放全局模型参数,这样就提供了有效的机制用于在分布式系统不同的 Worker 节点之间同步模型参数,而每个 Worker 只需要保存它计算时所依赖的一小部分参数即可。当然,这里存放模型参数的不是普通的 Key-Value 抽象,因为以 Key-Value 为单元进行频繁的模型参数同步会导致过高的通信开销,因此参数服务器通常采用数学封装来进行参数同步,比如向量、张量、矩阵的行列等。但是这个原型在可扩展性方面有所欠缺,并且通信延迟高。Ahmed 等[26]提出的框架扩展了该原型,并引入了新的特性:为异步更新全局变量提供一个更高效的通信机制,为本地变量提供一个基于磁盘的有计划的缓存机制以及一个全新的容错机制等,并且开源了该系统的实现 YahooLDA(https://github.com/sudar/Yahoo_LDA)。

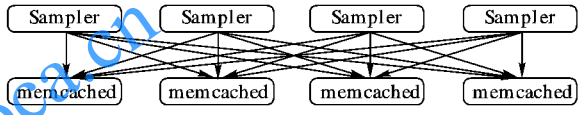

图 13　第一代参数服务器

Fig. 13　The 1st generation parameter server architecture

第二代参数服务器[27]可以在 Dean 等[28]提出的 DistBelief 框架中找到原型,基本架构如图 14 所示。Dai 等[29]提出的 Petuum-PS 结合 SSP 模型降低了第一代参数服务器的同步和通信开销。

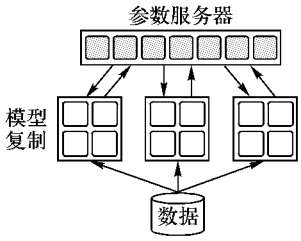

图 14　第二代参数服务器架构

Fig. 14　The 2nd generation parameter server architecture

Li 等[30-31]提出了第三代参数服务器,克服了第两代的缺点(只能采用单一的一致性模型等),其基本架构如图 15 所示,参数服务器中的节点分为服务组(Server Group)和多个工作组(Worker Group):1)服务组中包含一个服务管理节点(Server Manager)和多个服务节点(Server Node)。服务管理节点负责维护元数据的一致性,比如服务节点的状态、参数的分配等;服务节点负责各自的参数,服务节点之间可以互相通信,复制或者迁移参数;服务组共同维护所有参数的更新。2)工作组包含一个任务调度器(Task Scheduler)和多个计算节点(Worker Node)。一个工作组运行一个应用,任务调度器负责向计算节点分配任务,并且监控计算节点的运行状态,当有计算节点被加入或者移除,任务调度器则重新分配任务;计算节点之间不进行通信,只和对应的服务节点进行通信。第三代参

数服务器有如下 5 个特点：

1）高效的通信(Efficient Communication)。采用异步通信模型，完成一轮计算的节点不必等待(除非被请求)其他节点完成计算，这样的机制能够减少延时，并且优化机器学习任务的调度，提高效率。

2）灵活的一致性模型(Flexible Consistency models)。宽松的一致性要求允许算法设计者根据自身的情况(数据量、硬件等)权衡算法收敛速度和系统性能。

3）灵活的可扩展性(Elastic Scalability)。采用一致性哈希算法[32-33]进行节点管理，在添加删除节点时无需重新运行系统。

4）容错和稳定性(Fault Tolerance and Durability)。从非灾难性机器故障中恢复只需 1 s，不中断计算；Vector clocks[34-35]保证在经历网络分区和故障之后系统能够良好运行。

5）易用(Ease of Use)。为了简化机器学习应用开发，全局共享的参数被表示成(稀疏的)向量和矩阵，并且提供的线性代数的数据类型支持多线程。

综上所述，参数服务器由服务节点和计算节点组成，基本结构如图 16 所示，其中机器学习算法的参数由服务节点管理，机器学习算法的训练由计算节点完成；计算节点只能与对应的服务节点通信，服务节点之间相互复制或迁移参数。

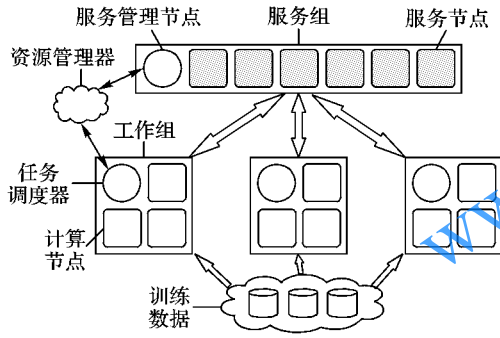

图 15 第三代参数服务器架构

Fig. 15 The 3rd generation parameter server architecture

3.3.2 Petuum 小结

Petuum 创新之处是将参数服务器和 SSP 并行模型结合，较好地实现了数据和模型同时并行化。因此，与 Spark 相比，Petuum 能够处理更大规模的数据和训练更大规模的模型。由于 Petuum 采用第二代参数服务器，第三代参数服务器在第二代的基础上进行了很多优化，比如采用更灵活的一致性模型，支持 ASP、BSP、SSP 多种并行计算模型；更强的可扩展性，支持动态添加和删除计算节点，因此 Petuum 系统还有较大的提升空间。由于只提供了简单的编程接口，而且目前提供的机器学习算法实现较少，所以相对于 Spark，在 Petuum 系统上实现自定义的机器学习算法的难度较大。

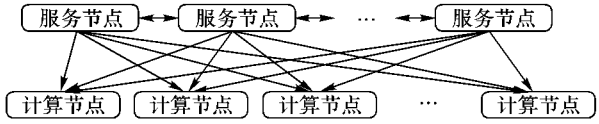

图 16 参数服务器基本架构

Fig. 16 Basic structure of parameter server

4 结语

本文从归纳大部分机器学习算法的数学表达式开始，分析介绍了机器学习算法的特点及其理论上的数据并行化和模型并行化，进而介绍相关的并行计算模型以及基于这些模型的大数据机器学习框架和平台。下面从采用的抽象数据结构、并行计算模型、容错机制等方面总结一下上述典型的机器学习平台，如表 2 所示。在机器学习平台(框架)设计上，没有普适的最好平台，只有最适合实际计算问题的平台。在训练机器学习模型时有一个共享的中间状态：模型参数，计算过程中会不断地读写中间状态。在基于 ASP 并行模型的系统中，因为计算节点之间完全异步执行，所以这种一致性模型的计算效率很高，但是模型参数没有一致性保证，不同节点获取到的是不同版本的模型参数，这样的训练过程是不稳定的，将会影响算法收敛性；在基于 BSP 并行模型的系统中，所有的计算节点在计算过程中都获取一致的模型参数，因此能保证算法的收敛性，但是代价是同步造成的计算资源和时间的浪费；CMU 的 Xing 教授提出了介于 BSP 和 ASP 两者之间的 SSP，通过限制模型参数的最大不一致版本数来控制整体的同步节奏，这样既能缓解慢机问题，又使得算法相对于 ASP 在收敛性上有更好的保证。基于不同的并行模型可以很好地在运行速度和算法效果上进行权衡。因此：1) Spark 是一个通用的大数据平台，适用于进行大规模数据处理和小规模的机器学习，不适合于大规模机器学习；另外由于 RDD 的不可变性，不适合参数反复更新的需求。这也是 Spark 不适合大规模机器学习的另一个原因。2) 基于 AP 模型的 GraphLab 适合于进行大规模图计算的平台，但是 AP 模型缺乏一致性保证，不适合于大规模机器学习。3) Petuum采用了 SSP 模型结合参数服务器，适合于大规模机器学习。

表 2 典型机器学习平台对比

Tab. 2 Comparison of typical machine learning platforms

平台	数据结构	并行模型	容错机制	可扩展性	通信效率	稳定性	适用场景
Mahout	—	BSP	Checkpoint	不高	不高	较高	推荐系统 + 小规模机器学习
Spark	RDD	BSP	Checkpoint + Lineage	较高	较高	高	大规模数据处理 + 小规模机器学习
GraphLab	Graph + Table	AP	Checkpoint	较高	高	较高	大规模图计算
Petuum	None	SSP	—	高	高	不高	大规模机器学习

参数服务器有较高的可扩展性和灵活性，能够仿真 AP、BSP、SSP 等并行计算模型。参数服务器将模型表示成(Key, Value)向量，所以其在处理稀疏矩阵上有明显的优势。而 Spark 的 RDD 数据抽象在处理低维稠密矩阵时有明显的优势。针对参数服务器各类优化将会成为未来主要的研究方向之一，比如服务节点和计算节点解耦来进一步提高灵活性，提供更好的编程接口，方便编程人员开发复杂的机器学习算法；合并同一台机器中多个线程的请求，因为同一次机器学习训练过程中，不同线程之间大概率会有很多重复的模型参数请求；使用或开发更好的缓存机制，来提升查询效率。对于基于 BSP 模型的平台，可以借鉴文献[31]中提出的 User-defined Filters 过滤掉一些不重要的更新来提高通信效率，另外可以

加入小任务调度控制,将慢机上未完成的小任务调度到处于等待状态的节点上运行等等。另外,基于 BSP 模型的 Spark 平台亦可尝试更参数服务器结合,来提高性能和灵活性。对于基于 SSP 模型的平台(如 Petuum),可以结合 Li 等提出的第三代参数服务器来取得更好的容错性和稳定性。总而言之,未来大规模机器学习平台将会主要围绕参数服务器来设计。

参考文献(References)

[1] 王庆先, 孙世新, 尚明生, 等. 并行计算模型研究[J]. 计算机科学, 2004, 31(9):128 – 131.(WANG Q X, SUN S X, SHANG M S, et al. Research on parallel computing model[J]. Computer Science, 2004, 21(9):128 – 131.)

[2] 王欢, 都志辉. 并行计算模型对比分析[J]. 计算机科学, 2005, 32(12):142 – 145.(WANG H, DU Z H. Contrastive analysis of parallel computation model[J]. Computer Science, 2005, 32(12):142 – 145.)

[3] 涂碧波, 邹铭, 詹剑锋, 等. 多核处理器机群 Memory 层次化并行计算模型研究[J]. 计算机学报, 2008, 31(11):1948 – 1955.(TU B B, ZOU M, ZHAN J F, et al. Research on parallel computation model with memory hierarchy on multi-core cluster[J]. Chinese Journal of Computers, 2008, 31(11):1948 – 1955.)

[4] 刘方爱, 刘志勇, 乔香珍. 一种异步 BSP 模型及其程序优化技术[J]. 计算机学报, 2002, 25(4):373 – 380.(LIU F A, LIU Z Y, QIAO X Z. An asynchronous BSP model and optimization techniques[J]. Chinese Journal of Computers, 2002, 25(4):373 – 380.)

[5] VALIANT L G. A bridging model for parallel computation[J]. Communications of the ACM, 1990, 33(8):103 – 111.

[6] CIPAR J, HO Q, KIM J K, et al. Solving the straggler problem with bounded staleness[C]// Proceedings of the 14th USENIX Conference on Hot Topics in Operating Systems. Berkeley, CA: USENIX Association, 2013: Article No. 22.

[7] 黄宜华. 大数据机器学习系统研究进展[J]. 大数据, 2015, 1(1):28 – 47.(HUANG Y H. Research progress on big data machine learning system[J]. Big Data Research, 2015, 1(1):28 – 47.)

[8] 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.(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.)

[9] BOTTOU L. Large-scale machine learning with stochastic gradient descent[C]// Proceedings of the 19th International Conference on Computational Statistics Paris France. Berlin: Springer, 2010: 177 – 186.

[10] FERCOQ O, RICHTÁRIK P. Accelerated, parallel and proximal coordinate descent[J]. SIAM Journal on Optimization, 2014, 25(4): 1997 – 2023.

[11] BLEI D M, KUCUKELBIR A, MCAULIFFE J D. Variational inference: a review for statisticians[EB/OL].[2016-11-20]. https://www.cse.iitk.ac.in/users/piyush/courses/pml_winter16/VI_Review.pdf.

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

[13] RUDER S. An overview of gradient descent optimization algorithms [EB/OL].[2016-11-20]. http://128.84.21.199/pdf/1609.04747.pdf.

[14] DUCHI J, HAZAN E, SINGER Y. Adaptive subgradient methods for online learning and stochastic optimization[J]. Journal of Machine Learning Research, 2011, 12(7):2121 – 2159.

[15] ZEILER M D. ADADELTA: an adaptive learning rate method [EB/OL].[2016-11-20]. http://www.matthewzeiler.com/wp-content/uploads/2017/07/googleTR2012.pdf.

[16] 郝树魁. Hadoop HDFS 和 MapReduce 架构浅析[J]. 邮电设计技术, 2012(7):37 – 42.(HAO S K. Brief analysis of the architecture of Hadoop HDFS and MapReduce[J]. Designing Techniques of Posts and Telecommunications, 2012(7):37 – 42.)

[17] HO Q, CIPAR J, CUI H, et al. More effective distributed ML via a stale synchronous parallel parameter server[C]// Proceedings of the 26th International Conference on Neural Information Processing Systems. Lake Tahoe, Nevada: Curran Associates Inc., 2013: 1223 – 1231.

[18] LOW Y, GONZALEZ J E, KYROLA A, et al. GraphLab: a new framework for parallel machine learning[EB/OL].[2016-11-20]. http://wwwdb.inf.tu-dresden.de/misc/SS15/PSHS/paper/GraphLab/Low_2010.pdf.

[19] LOW Y, BICKSON D, GONZALEZ J, et al. Distributed GraphLab: a framework for machine learning and data mining in the cloud[J]. Proceedings of the VLDB Endowment, 2012, 5(8): 716 – 727.

[20] CHU C T, KIM S K, LIN Y A, et al. Map-Reduce for machine learning on multicore[C]// Proceedings of the 19th International Conference on Neural Information Processing Systems. Cambridge, MA: MIT Press, 2006: 281 – 288.

[21] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters[C]// Proceedings of the 6th USENIX Conference on Symposium on Opearting Systems Design and Implementation. Berkeley, CA: USENIX Association, 2004: Article No. 10.

[22] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: cluster computing with working sets[C]// Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. Berkeley, CA: USENIX Association, 2010: Article No. 10.

[23] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[C]// Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2012: Article No. 2.

[24] 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.

[25] SMOLA A, NARAYANAMURTHY S. An architecture for parallel topic models[J]. Proceedings of the VLDB Endowment, 2010, 3(1/2):703 – 710.

[26] AHMED A, ALY M, GONZALEZ J, et al. Scalable inference in latent variable models[C]// Proceedings of the 5th ACM International Conference on Web Search and Data Mining. New York: ACM, 2012: 123 – 132.

[27] DAI W, KUMAR A, WEI J, et al. High-performance distributed ML at scale through parameter server consistency models[C]// Proceedings of the 29th AAAI Conference on Artificial Intelligence. Menlo Park, CA: AAAI Press, 2015: 79 – 87.

(下转第 3052 页)

[4] LEISTRITZ F L. Use of dummy variables in regression analysis[J]. Agricultural Economic Miscellaneous Report Technical, Agricultural Experiment Station, North Dakota State University, 1973, 4(43): 434 - 442.
[5] HARDY M A. Regression with Dummy Variables[M]. Thousand Oaks, CA: SAGE Publications, 1993: 96.
[6] GROTENHUIS M T, THIJS P. Dummy variables and their interactions in regression analysis: examples from research on body mass index[EB/OL]. [2016-11-20]. http://www.ru.nl/publish/pages/780171/table1-4.pdf.
[7] USMAN A U, ABDULKADIR H S, TUKUR K. Application of dummy variables in multiple regression analysis[J]. Recent Scientific Research, 2015, 7(11): 7440 - 7442.
[8] GÜRTLER M, HIBBELN M, WINKELVOS C. The impact of the financial crisis and natural catastrophes on CAT bonds[J]. Journal of Risk and Insurance, 2016, 83(3): 579 - 612.
[9] SKRIVANEK S. The use of dummy variables in regression analysis [EB/OL]. [2016-11-20]. https://www.moresteam.com/WhitePapers/download/dummy-variables.pdf.
[10] SUITS D B. Use of dummy variables in regression equations[J]. Journal of the American Statistical Association, 1957, 52(280): 548 - 551.
[11] HELLMANN T F, SCHURE P, VO D. Angels and venture capitalists: substitutes or complements?[J]. Social Science Electronic Publishing, 2015, 11(7): 1301 - 1307.
[12] SEARLE S R, UDELL J R. The use of regression on dummy variables in management research[J]. Management Science, 1970, 16(6): 397 - 409.
[13] 杨希, 王苏生. 政府背景风险投资对创业企业经营绩效的影响[J]. 大连海事大学学报(社会科学版), 2016, 15(5): 52 - 58. (YANG X, WANG S S. Influence of government background venture capital on the performance of startup enterprises[J]. Journal of Dalian Maritime University (Social Science Edition), 2016, 15(5): 52 - 58.)
[14] 徐卫华, 何宜庆, 钟慧安. 金融深化、科技创新与产业结构优化升级——基于我国 30 个省市 1997 ~ 2014 年面板数据分析[J]. 金融与经济, 2017, 15(3): 54 - 64. (XU W H, HE Y Q, ZHONG H A. Financial deepening, technological innovation and industrial structure optimization and upgrading-based on panel data analysis of 30 provinces in China from 1997 to 2014[J]. Finance ans Economy, 2017, 15(3): 54 - 64.)
[15] POLISSAR L, DIEHR P. Regression analysis in health services research: the use of dummy variables[J]. Medical Care, 1982, 20(9): 959 - 966.
[16] 庞浩. 计量经济学[M]. 北京: 科学出版社, 2015: 190 - 199. (PANG H. Econometric Analysis[M]. Beijing: Science Press, 2015: 190 - 199.)
[17] 高铁梅. 计量经济分析方法与建模[M]. 北京: 清华大学出版社, 2009: 76 - 79. (GAO T M. Econometric Analysis Method and Modeling[M]. Beijing: Tsinghua University Press, 2009: 76 - 79.)
[18] TIBSHIRANI R. Regression shrinkage and selection via the Lasso: a retrospective[J]. Journal of the Royal Statistical Society, 2011, 73(3): 273 - 282.
[19] MALLOWS C L. Some comments on CP[J]. Technometrics, 2000, 42(1): 87 - 94.

This work is partially supported by the National Natural Science Foundation of China (61672157), the Project of Network and Information Security Key Theory and Technological Innovation Team in Fujian Normal University (IRTL1207).

LI Haichao, born in 1990, M. S. candidate. His research interests include machine learning, financial data mining.

WANG Kaijun, born in 1965, Ph. D., associate professor. His research interests include machine learning, intelligent learning and reasoning, data mining, pattern recognition.

HU Miao, born in 1994, M. S. candidate. His research interests include machine learning, data mining.

CHEN Lifei, born in 1972, Ph. D., professor. His research interests include statistical machine learning, data mining, pattern recognition.

(上接第 3047 页)

[28] DEAN J, CORRADO G S, MONGA R, et al. Large scale distributed deep networks [C]// Proceedings of the 25th International Conference on Neural Information Processing Systems. Lake Tahoe, Nevada: Curran Associates Inc., 2012: 1223 - 1231.
[29] DAI W, WEI J, ZHENG X, et al. Petuum : a framework for iterative-convergent distributed ML[EB/OL]. [2016-11-20]. http://www.u.arizona.edu/ ~junmingy/papers/Dai-etal-NIPS13.pdf.
[30] LI M, ZHOU L, YANG Z, et al. Parameter server for distributed machine learning[EB/OL]. [2016-11-20]. http://www-cgi.cs.cmu.edu/ ~muli/file/ps.pdf.
[31] LI M, ANDERSEN D G, PARK J W, et al. Scaling distributed machine learning with the parameter server[C]// Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2014: 583 - 598.
[32] KARGER D, LEHMAN E, LEIGHTON T, et al. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide Web[C]// Proceedings of the 29th ACM Symposium on Theory of Computing. New York: ACM, 1997: 654 - 663.
[33] BYERS J, CONSIDINE J, MITZENMACHER M. Simple load balancing for distributed hash tables[C]// Proceedings of the 2nd International Workshop Peer-to-Peer Systems Ⅱ. Berlin: Springer, 2003: 80 - 87.
[34] CHOUDHARI R, JAGADISH D. Paxos made simple[J]. ACM SIGACT News, 2001, 32(4): 51 - 58.
[35] DECANDIA G, HASTORUN D, JAMPANI M, et al. Dynamo: Amazon's highly available key-value store[C]// Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles. New York: ACM, 2007: 205 - 220.

This work is partially supported by the National Natural Science Foundation of China (61603197), the Natural Science Foundation of Jiangsu Province (BK20140885).

JIAO Jiafeng, born in 1991, M. S. candidate. His research interests include large scale machine learning.

LI Yun, born in 1974, Ph. D., professor. His research interests include machine learning, feature engineering.