doi:10.3772/j.issn.1002-0470.2015.08-09.006

大数据处理系统的研究进展与展望①

王 鹏②* 张 利**

(*中国科学院信息工程研究所 北京 100093)

(**防灾科技学院 三河 065201)

摘 要 论述了大数据处理技术的最新研究进展。首先,为了便于从整体上了解研究现状,从负载类型和数据类型两个角度对目前的大数据处理系统进行了分类;其次,深入介绍了批处理编程框架的研究进展,重点讨论了面向大规模图分析、分布式机器学习等应用领域的编程框架,包括计算特点、面临的挑战和设计原理等;最后,对大数据的研究热点和趋势进行了展望,指出异构硬件平台的并行训练、串行代码的自动化并行以及混合编程是未来大数据处理技术的研究热点。

关键词 大数据, MapReduce, Spark, 并行计算, 大规模图计算, 分布式机器学习, 深度学习

0 引 言

互联网的迅速发展导致数据量爆炸式增长,根据国际数据公司的统计, 2011 年全球产生的数据总量为 1.8ZB(1ZB = 10^{21}B),最近 3 年人类产生的信息量已经超过了之前产生的信息量总和。IBM 用 4 个 V,即 Volume(数据体量巨大)、Velocity(数据以高速率产生)、Variety(数据类型多种多样)和 Value(数据的价值密度低),来描述大数据的特点,对低价值数据进行深度地分析,提炼出高价值的知识,已经成为目前普遍的共识。由于数据体量大、产生速度快,以关系型数据库为代表的传统数据管理与分析工具无法应对大数据时代的挑战,新型的海量数据处理技术应运而生。2012 年,美国政府提出了“大数据研究发展计划”,将大数据的研究与应用上升到战略层面。毋庸置疑,我们已经处于大数据时代,并经历这场技术变革。

Google 是大数据技术潮流的引领者。从 2003 年开始,Google 陆续公开了其内部基础设施的核心技术,包括 GFS[1]、MapReduce[2]、BigTable[3]、Chubby[4]、Tenzing[5]、Dremel[6]、FlumeJava[7]、Pregel[8]、MillWheel[9]、MegaStore[10] 和 Spanner[11] 等。HDFS[12] 和 Hadoop[12] 分别是 Google 的 GFS[1] 和 MapReduce[2] 技术的开源实现。从 2008 年开始,Hadoop 广泛流行起来,掀起了大数据处理的研究与应用热潮。学术界围绕 Hadoop 开展了大量的研究工作,衍生出大量 Hadoop 相关的技术与系统。互联网公司普遍使用 Hadoop 技术,这使得 Hadoop 迅速发展成为一个活跃、庞大的社区和生态系统,这个热潮持续至今。近几年,美国加州大学伯克利分校研发的 Spark[13] 系统从单纯的批处理系统延伸到流处理和交互式查询等领域,这种完善的分布式计算软件栈展示出多方面的优势,包括更加简洁的代码、高效的执行速度和灵活多样的数据处理和融合等。目前基于 Spark 的软件栈已经在国内外的互联网公司得到了广泛应用。Hadoop 和 Spark 软件栈包括资源管理、数据存储与管理、数据处理和应用等四个层

① 国家青年科学基金(61303060),863 计划(2012AA01A401)和中国科学院先导专项(XDA06030200)资助项目。

② 男,1980 年生,博士,助理研究员;研究方向:大数据处理;联系人,E-mail: wangpeng@iie.ac.cn

(收稿日期:2014-12-03)

次。资源管理负责管理集群的硬件资源,提供统一的接口,允许上层的计算任务能够按需向其申请资源,使用完毕后释放资源。目前的一个发展趋势是构建统一的资源管理与调度系统,能够支持多种类型的计算框架,允许批处理、实时计算和 Web 服务等异构计算任务复用集群资源。数据存储与管理主要解决如何在数以千计的普通服务器组成的集群中存储 PB 量级数据,以及如何在 PB 量级下提供一种数据模型,允许快速地读写数据,传统的关系型数据库无法胜任,以 BigTable[3]、MegaStore[10] 和 Spanner[11] 代表的列式数据库是目前的主流技术。数据处理主要研究各种编程框架与编程语言,方便程序员在大规模集群上开发分布式程序。这类系统的共同设计目标包括灵活的水平伸缩性、透明的容错性和简洁的编程接口等。在大数据时代,学术界和工业界已经研发了各种各样的编程框架,这极大地扩展了数据处理的方式。本文讨论该领域的研究进展与发展趋势。

1 大数据处理系统的分类

大数据处理系统的种类繁多,目前还没有公认的分类方法。为了清晰掌握技术的发展现状,本文从负载类型和数据类型两个角度对代表性的系统进行分类,一方面明确了各种系统的适用范围;另一方面可看到尚未解决的空白领域。

(1)从负载类型上分类

从负载类型的角度,目前的系统可以划分为批处理、流式计算和交互式查询等三类。批处理系统强调系统的高吞吐量,例如 Hadoop 系统就定位于批处理。流式计算是近几年兴起的一个研究领域,很多应用场景对计算的时效性要求高,希望能够在尽可能小的延迟内完成,例如在线广告推荐、入侵检测和作弊识别等。目前代表性的流式计算系统包括雅虎的 S4[14]、Twitter 的 Storm[15]、Google 的 MillWheel[9]、微软的 TimeStream[16] 和美国加州大学伯克利分校的 DStream[17] 等。交互式查询分析主要应用于大规模的数据仓库,从早期的 Hive[18] 到 Dremel[6]、Impala[19] 和 Shark[20] 等,各种 SQL-On-Hadoop 与大规模并行处理(massive parallel processing, MPP)系统不断出现,关于这方面的研究工作可参考综述文献[21]。

(2)从数据类型上分类

从数据类型的角度,目前的系统提供了集合、表、图和矩阵等多种数据抽象。通常一个编程框架只适合解决某类问题,无法对各种问题领域都适用。例如,MapReduce[2] 适合解决记录之间相互独立的集合类数据;Piccolo[22] 利用分布式内存存储表格数据,能够加快迭代计算的运行效率;MadLINQ[23] 提供大规模的矩阵运算,可简化一大类机器学习和数据挖掘算法的开发。

图 1 对目前代表性的编程框架进行分类。从图中可以看出,批处理领域的研究最为全面,针对各种数据类型都有相应的系统。交互式查询目前主要针对关系型数据,对于大规模图数据库的研究相对较少,目前代表性工作有 Facebook 的 Tao[24]。实时计算目前主要针对集合类数据的处理。

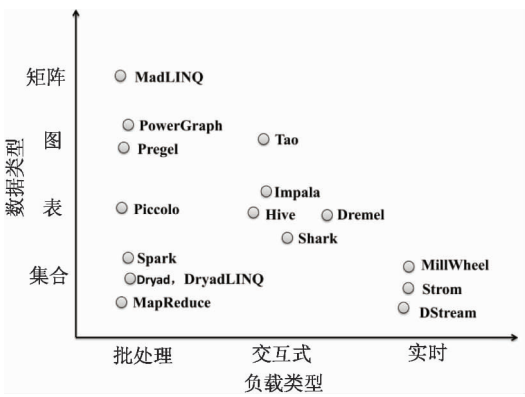

图 1 大数据处理系统的分类

2 研究进展

批处理编程框架种类繁多,但大部分编程框架都与数据流模型有密切关系,对数据流模型进行概括和总结可厘清一大类系统之间的差异。大规模图计算和分布式机器学习是目前两个非常活跃的研究方向,由于其计算特点的特殊性,数据流模型并不擅长解决此类问题,需要新式的计算模型,2.2 和 2.3 小节分别介绍这两部分的最新研究进展。

2.1　基于数据流模型的编程框架

许多的编程框架都可以归结为数据流模型,如批处理中的 MapReduce[2]、Dryad[25]、Spark[13]、FlumeJava[7],流处理中的 Storm[15]等。数据流模型有相当长的研究历史,关于这方面的工作可参考综述文献[26]。简单而言,数据流模型使用有向无环图(directed acyclic graph,DAG)表达一个计算,图中的顶点表示计算任务,而边表示数据依赖。

(1) MapReduce[2]: MapReduce 计算为程序员提供了两个简捷的编程接口,即 Map 函数和 Reduce 函数。任务的输入和输出数据按照 Key-Value 的方式组织,框架会自动将中间结果中相同 Key 值的记录聚合在一起,作为 Reduce 函数的输入值。MapReduce 具有非常突出的优点。首先是具有极高的可伸缩性,可以在数千台机器上并发执行;其次具有容错性,即使集群发生了故障,一般情况下也不会影响任务的正常执行;三是简单性,用户只需要完成 map 和 reduce 两个函数就可以完成大规模数据的并行处理。但 MapReduce 也存在一些局限性,比如 map 和 reduce 任务的启动时间长,不适合对时效性要求高的应用场景。MapReduce 模型存在多处的磁盘读写和网络传输,对于迭代机器学习应用,往往需要同一个 mapreduce 任务反复执行,这就带来了大量的磁盘读写和网络传输开销,使运行效率非常低。MapReduce 包括 Map 和 Reduce 两个阶段的任务,Map 阶段的任务执行完后才能执行 Reduce 阶段的任务,无法表达任务之间复杂的拓扑结构,本质上 MapReduce 模型可以看作是 DAG 计算的一种特例。

(2) Dryad[25]: Dryad 采用了通用的 DAG 计算模型,能够灵活地表达任务之间的拓扑结构,由于需要程序员显式地构建拓扑结构,也带来了一定的编程负担。Dryad 在数据传输方面提供了更多的机制,包括共享内存、TCP 管道、临时文件和分布式文件系统等,而 MapReduce 只提供了临时文件的方式传输 map 和 reduce 阶段之间的中间结果。

(3) Spark[13]: Spark 的核心思想是使用数据集的转换图(DAG 结构)来表达一个完整的数据处理过程,DAG 中的顶点表示弹性分布式数据集(resilient distributed dataset, RDD),边表示转换操作。RDD 是对迭代计算中反复使用的中间数据集的一种抽象,表示一个只读记录组成的集合。RDD 创建后其内容不能修改,但可以对 RDD 进行各种转换操作,原先 RDD 的内容经过转换后形成了新的 RDD,Spark 会维护 RDD 之间的依赖关系,称为血统(Lineage)。这种链式操作不仅简化了应用程序的开发,而且也让故障恢复变得简单了。如果某个 RDD 发生了故障需要恢复,可以根据血统信息从数据源逐步重放转换操作来重新生成该 RDD。实验结果表明,Spark 的性能比 Hadoop 要快 1 ~2 个数量级,主要利用了三种技术来达到如此高的执行效率:数据并行、流水线和缓存。Spark 将 RDD 分片(partition)到多台机器上,同时执行粗粒度的转换操作,即数据并行。一条记录进入 Spark 的任务后会经历多个转换操作再输出,即流水线。Spark 提供RDD. Cache()的应用程序编程接口(API),允许程序员将一个 RDD 显式地加载并驻留在内存中,这样可极大地加快后续的处理速度,对于迭代计算特别有效。随着内存越来越便宜,基于内存计算会越来越成为系统设计上的一种选择。目前,Spark 软件栈提供了一个相对完整的大数据解决方案,以批处理的 Spark 为基础,在其上构建了流式计算系统 DStream[17]、图计算系统 GraphX[27]和机器学习算法库 MLlib[28]等。

(4)高级的库与语言:随着 Hadoop 的广泛应用,直接采用 MapReduce 编程面临开发效率和性能等几方面的问题。目前的发展趋势是将 Hadoop 封装到下层作为执行引擎,在上层系统中提供更加简单易用的高级库或面向领域的编程语言(domain specific language, DSL)等,由转换层负责将更高级的抽象自动翻译为大量的 Hadoop 作业。例如,Pig[29]是一种高级的数据流语言,采用 MapReduce 作为其执行引擎。DryadLINQ[30]是构建在 Dryad 之上的高级语言,简化了复杂程序的开发。FlumeJava[7]是一个 Java 库,其本质是在 MapReduce 基础上的 DAG 系统,能够将多个 MapReduce 程序以一定语义拼接成 DAG 任务,从而完成更复杂的任务。Hive[18]支持结构化查询语言(structured query language, SQL),采用 MapReduce 作为执行引擎,由于 MapReduce 的表达能力有限,Stinger 项目[31]是 Hive

的改进版本,其底层执行引擎采用基于 DAG 模型的 Tez 系统[32]。

北京大学肖臻领导的研究团队针对 MapReduce 的预测执行机制进行了深入研究[33],提出了一种高效、快速的拖后腿任务识别方法,对 Hadoop 备份任务的执行策略进行了改进,从而缩短了程序的运行时间。数据偏斜是导致任务负载不均衡和出现拖后腿现象的主要原因之一,LIBRA[34]提出了一种均衡的数据分区方法,保证 reduce 任务的负载均衡,适用于一大类应用并对 Hadoop 程序保持透明。

2.2 图计算

大规模的图数据分析在实际应用中有广泛的需求。例如互联网的网页之间形成了一个超大的图,其顶点规模达到千亿级,必须借助于大规模的集群才能完成巨型图的分析与挖掘。图数据具有非规则的结构,导致图数据访问的局部性差。很多现实世界中的图符合幂律分布,即图顶点的分布极不均匀,极少的顶点通过大量的边与其他大量顶点发生关联,这意味着图数据很难切分均匀,从而带来机器负载不均衡和大量的网络通信开销,这会严重影响图计算系统的整体运行效率。

大图的分割是图计算中基础性的问题之一。目前,图数据的切分主要有两种方法:切边和切点,如图 2 所示。切边法的切割线只能通过图的边,图 2 左端的切边法表示将图顶点切分到 2 台机器上,被切开的边在计算时意味着机器之间的网络通信;切点法的切割线只能通过图顶点,图 2 中右端的切点法将中心顶点 V 切割成 2 份,意味着顶点 V 会同时出现在2台机器上,机器之间的网络通信量明显减少,但由于图算法迭代过程中需要不断更新图顶点的值,维护顶点 V 的数据一致性仍会带来一定的通信开销。衡量图数据的切分主要考虑两个因素:机器的负载均衡性和网络通信量。理想的情况是机器负载尽量均衡,并且网络通信量最小。现实世界中的大多数图的边分布遵守幂律分布,理论和实践均已证明,对于遵守这一特征的图数据,切点法比切边法的计算效率要高一个数量级[35]。

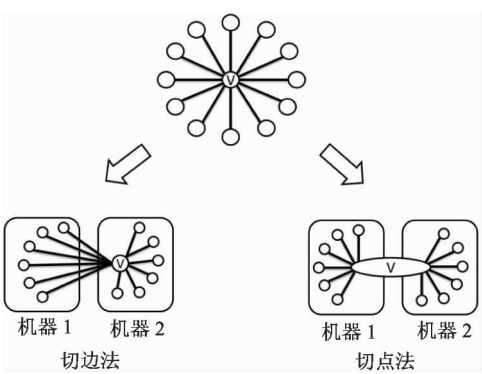

图 2 图数据的切分方法

目前图数据领域的相关系统可划分为两类:在线查询的图数据库和离线图数据分析系统。Facebook 的 Tao[24]是目前代表性的大规模图数据库查询系统,离线图分析系统包括 Pregel[8]、Giraph[36]、GraphLab[37]和 PowerGraph[35]等,适合解决大规模图数据的挖掘与分析类应用,如计算网页的 PageRank 值、计算图中两个顶点之间的最短路径等。

Pregel[8]提出了以顶点为中心的编程模型,后续大部分的图计算系统都采用这种编程模型或者其变体。框架提供了一个针对图顶点的编程接口,程序员编写针对顶点的计算函数,不仅可以访问并更新顶点与边的状态,而且还可以通过增加或删除边来修改图的拓扑结构。图计算通常需要经过多轮迭代,在每次迭代过程中底层的运行时系统会调用用户自定义的顶点更新函数,该函数会更新顶点及边的状态,如果所有更新后的状态在下一轮迭代中才可见并允许使用,称为同步执行模式;反之,如果更新后的状态在本轮迭代过程中就对其他顶点可见,称为异步执行模式。Pregel 采用块同步并行(bulk synchronous parallel,BSP)模型,整个计算由若干顺序执行的超级步组成,以同步的方式执行顶点的计算函数,即在一轮迭代中执行完所有顶点的计算函数后,在下一轮迭代中才能访问并使用上一轮迭代的状态变化。Giraph[36]是 Pregel 的开源实现。

GraphLab[37]对 Pregel 进行了多方面的改进,特别适合一些复杂的机器学习类应用,如随机梯度下降算法和马尔科夫随机场等。GraphLab 将完整的顶点计算函数进一步划分为三个连续的子处理阶段:Gather、Apply 和 Scatter,简称 GAS 模型,各个子阶段可以细粒度地并发地执行,从而显著地加快系统的运行效率。此外,GraphLab 采用了异步执行来

提供系统的并发性能,由于不需要全局的同步,更新的数据在本轮迭代中即可使用,使得算法的收敛速度很快,吞吐量和执行效率都要明显高于同步方式。异步执行也存在相应的缺点,主要是很难判断程序输出结果的正确性,掌握起来困难,学习成本高。PowerGraph[35]是 GraphLab 的进一步改进,针对幂律分布的图,提出了切点法来分割图以及对中间结果采用增量缓存来减少计算量。

在图计算领域,国内的研究团队也取得了一系列的研究进展。上海交通大学陈海波领导的研究团队针对非均匀一致存储访问服务器研发了高效的图计算系统——Polymer[38],主要通过两种手段大幅度的提高性能:(1)根据图数据的访问特点,重新分布数据来减少远程访问;(2)将一部分随机的远程访问转换为顺序的远程访问。PowerSwitch[39]提出了一种自适应、混合的图计算执行模式,通过收集并分析运行时的状态,自动切换到同步或异步的执行模式,从而缩短程序的运行时间。清华大学陈文光领导的研究团队针对核外(out-of-core)计算场景提出了基于二维划分的图存储方法,研发的单机图计算系统 GridGraph[40]比现有图计算系统性能可提高 2~3倍。

2.3　分布式机器学习系统

在大数据时代,传统的单机版机器学习算法因为处理器和内存受限制,无法处理海量数据,分布式机器学习成为备受关注的研究领域。通常,机器学习算法使用迭代计算在巨大的参数空间中求最优解,其计算特点对分布式机器学习带来了严重挑战,主要表现在以下几方面:

(1)单机版机器学习算法通过共享内存可存取全局参数,在分布式环境下需要通过网络来存取全局参数,效率比内存低很多,如何提高通信效率或者减少通信量,对提高分布式机器学习的运行效率至关重要。

(2)大量的并发任务由于各种原因(如机器负载或者软硬件故障等)会导致其执行速度不统一,快任务需要等待慢任务执行完,负载的不均衡会降低任务的整体完成效率,如何避免"快等慢"现象也是需要解决的问题。

(3)当集群中的机器发生故障后,系统应当保证程序能够容错并运行正确。

最近几年,大规模的深度学习在实际应用中获得了巨大的成功,特别是在语音识别和图像识别等领域取得了突破性进展。2012 年 Google Brain 项目使用包含 16000 个 CPU 核的并行计算平台训练超过 10 亿个神经元的深度神经网络,该系统通过分析 YouTube 上选取的视频,采用无监督的方式训练深度神经网络,能将图像自动聚类。在系统中输入"cat"后,能够在没有外界干涉的条件下自动识别出猫脸[41]。Google 的 DistBelief[42]是运行在 CPU 集群上深度学习系统,使用上万个 CPU 核来训练多达 10 亿参数的深度神经网络模型。DistBelief 应用的主要算法有 Downpour SGD 和 L-BFGS,支持的目标应用有语音识别和 2.1 万类目的图像分类。Google 的 COTS HPC[43]系统采用图形处理单元(GPU)来实现深度学习,GPU 服务器之间使用 Infiniband 连接,并采用消息传递接口(MPI)负责通信。COTS 使用 3 台 GPU 服务器能够在数天内完成对 10 亿参数的深度神经网络训练。

深度学习使用深层的神经网络来模拟人类大脑的工作原理,如图 3 所示,深层神经网络由一个输入层、多个隐含层和一个输出层组成,通过特征组合的方式,逐层将原始输入转化为浅层特征、中层特征、高层特征直至最终的任务目标。传统的机器学习算法由于只含有一层隐层节点(如支持向量机和 Boosting 等)或者没有隐层节点(如 Logistic Regression),可以看成是浅层模型。在深层的神经网络中,每层有大量的神经元,执行一个激活函数(如 Sigmoid 函数),神经元之间的一条连线对应一个连接参数。每个神经元模拟人脑的神经细胞,而节点之间的连接模拟神经细胞之间的连接。研究表明,使用大的训练样本,增加模型的参数可显著提高分类器的准确度,但训练数据集大、模型复杂、参数量巨大、计算量大等因素叠加在一起,导致训练一个模型极度耗时,如何通过大规模的集群来加速训练,提供并行框架来简化算法的开发,给大规模机器学习带来了挑战和机遇。从并行性的角度看,深度学习系统通常采用数据并行和模型并行混合的方式来加

快模型的学习过程。数据并行将训练数据拆分为多个分片(Partition),每份数据有一个模型实例进行训练,再将多个模型实例产生的结果进行合并。模型并行[41]对模型参数进行了拆分,每台机器负责其中一部分模型参数的训练,多个局部模型协同对一份训练数据进行训练。在图 3 中,一个五层的神经网络模型划分到 4 台机器上联合完成,每台机器负责其中一部分模型参数的训练过程,跨机器之间的神经元连线表示这些参数需要通过网络进行传输。

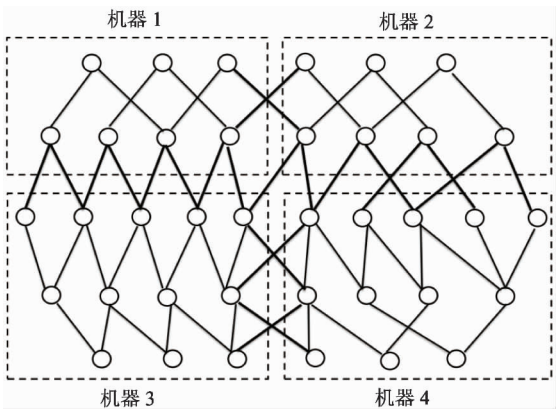

图 3 深度神经网络与模型并行

根据技术路线的差异,目前分布式机器学习系统可以划分为三类:基于 Hadoop、基于 Spark 和参数服务器架构。

Mahout[44]是在 Hadoop 上构建的机器学习算法库,提供了常见的聚类、分类和推荐等算法,由于底层机制受限于 MapReduce 提供的计算接口,算法的实现要复杂一些。此外,运算的中间结果都需要持久化到磁盘上,在迭代计算中需要频繁地读写磁盘上的数据,导致运行效率低。

MLlib[28]是在 Spark 上构建的机器学习系统,内置了大量成熟的机器学习算法库,系统的优化器会根据参数的组合、数据特征等信息自动地优化算法,降低了机器学习算法的使用门槛。由于 Spark 比 Hadoop 运行速度快几十倍甚至上百倍,因此 MLlib 的运行效率要高许多。

参数服务器[45]是目前分布式深度学习系统的典型结构,如图 4 所示。参数服务器包括大量并发执行的客户端和多台参数服务器组成的参数服务器集群,不同的系统基本遵循上述结构,但在具体实现上有差异。从概念上讲,参数服务器可以看作是分布式共享内存的 Key-Value 存储,用于存储客户端共享的全局参数,参数服务器对全局参数进行数据分片,每个数据分片维护一部分全局参数的存储与更新;在训练过程中,客户端可以读取或更新全局参数,不同应用的全局参数可以存储在参数服务器的不同表格中。从结构上讲,参数服务器集群通常采用传统的 Master-slave 结构,其中一台参数服务器充当主控节点的角色,负责数据路由以及在不同的服务器之间进行数据分片等工作。

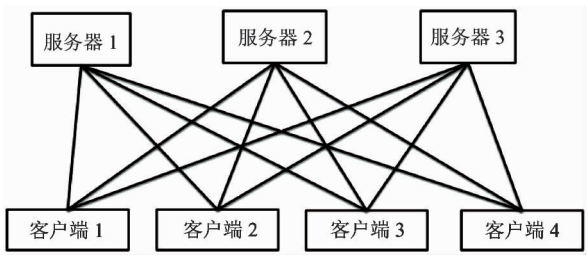

图 4 参数服务器的典型结构

3 研究热点与趋势

虽然大数据处理系统已取得大量的研究成果,各种工具与系统的研究方兴未艾,但许多技术仍处于摸索阶段,远未达到成熟。本节归纳了三个方面的研究热点与趋势,并对国内的大数据处理系统的重点突破方向进行了展望。

(1)异构硬件平台的并行训练

深度学习的计算特点是参数量巨大、更新频繁,传统的大数据处理技术如 Hadoop,由于数据处理延迟高,并不适合需要频繁迭代的深度学习,需要针对深度学习的模型结构和计算特征来设计相应的专用系统。目前大规模的并行训练是一个热点问题,特别是如何利用 CPU 和 GPU 组成的异构并行硬件平台来加速计算值得深入研究。

(2)串行代码的自动化并行

目前的编程框架提供了标准的数据操作接口,程序员编写接口的实现函数,由底层的系统负责并行地执行用户代码,这种编程方式与传统的串行编程差别大。最近,基于程序分析的自动化成为一个

研究热点,其基本思想是程序员编写传统的串行代码,通过程序分析技术来自动获取代码中的并行性。这方面的代表性工作包括 pydron[46] 和 SDG[47],二者均通过分析代码中的标注(annotation)来获取并行性。串行代码的自动化并行是一个具有吸引力的方向,许多问题有待进一步解决,包括优化的方法、减少对串行程序的限制等。

(3)混合编程

大数据处理技术的多样化丰富了数据处理的手段,但也给实际应用带来的困难。由于不存在全能通用的计算框架,多种编程框架需要协同工作才能完成复杂的数据处理任务。例如,典型的机器学习可以组织成数据预处理、特征提取、训练和评价等工作流。数据预处理和特征提取阶段可采用 MapReduce 来完成,而训练阶段采用 PowerGraph 来完成。目前实现上述处理流程需要多个独立的作业,作业之间通过 Hadoop 分布式文件系统(Hadoop distributed file system, HDFS)共享数据,造成大量的 HDFS 读写开销,并且大量的作业也增加了资源调度的开销。根本原因在于目前的编程框架相互独立,计算之间只能以粗粒度的方式组合,如何打破目前编程框架之间的壁垒,让多种计算能够自由、高效地组合在一起完成更为复杂的处理流程,这个问题目前还没有很好的解决方案。Transformer[48] 提出了一种混合计算的编程系统,能够将多种计算融合在一个程序中,并且通过分布式内存来加快中间数据集的访问。

本文从"深度"和"广度"两个层面展望国内大数据处理系统的重点突破方向。从"深度"角度讲,围绕目前广泛应用的编程框架(如 Hadoop、Spark 和 PowerGraph 等)开展改进、扩展与优化等方面的工作。例如,目前 Spark 已经在大数据的许多领域取得了广泛的应用和成功,基于 Spark 的分布式矩阵计算是非常值得探索的研究课题。从"广度"角度讲,传统的编程框架主要针对大规模的廉价集群设计,目前的底层硬件体系结构和计算平台呈现多样化(例如众核处理器、GPU 和 Amazon EC2 云计算平台等),为了充分发挥硬件和平台的计算能力,需要重新考虑系统的设计原理与优化方法,才能取得最佳的性价比,这类工作有具有广阔的研究空间。

4　结　论

近年来随着数据规模的爆炸式增长,从学术界和工业界都掀起了大数据研究和应用的浪潮。大数据研究涵盖面广,各种技术与系统不断涌现。从 2004 年 Google 公开 MapReduce 编程框架开始,10 年时间大数据处理技术进入了一个多元化的时代。虽然目前 Hadoop 系统为大数据处理提供了一种解决方案,但 MapReduce 模型相对简单,无法表达更复杂的算法逻辑,大数据处理的需求已经从深度和广度上提出了更高的要求,目前的 Hadoop 已经无法应对这些挑战。工业界及学术界都在寻找下一代的解决方案,各种解决方案的竞争异常激烈。我们期待在未来十年或更长时间内,这场大数据技术革命会推动数据科学持续的发展,将来会出现更简单易用、更强大、更通用的数据处理技术。

参考文献

[1] Sanjay G., Howard G, Shun-Tak L. The google file system. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles, Lake George, USA, 2003. 29-43

[2] Jeffrey D, Sanjay G. MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th USENIX Conference on Operating Systems Design and Implementation, San Francisco, USA, 2004. 10-23

[3] Fay C, Jeffrey D, Sanjay G, et al. Bigtable: a distributed storage system for structured data. In: Proceedings of the 7th USENIX Conference on Operating Systems Design and Implementation, Seattle, USA, 2006. 10-23

[4] Mike B. The chubby lock service for loosely-coupled distributed systems. In: Proceedings of the 7th symposium on Operating Systems Design and Implementation, Seattle, USA, 2006. 335-350

[5] Biswapesh C, Liang L, Weiran L, et al. Tenzing: a SQL implementation on the MapReduce framework. In: Proceedings of the International conference on Very Large Data Bases, Seattle, USA, 2011. 1318-1327

[6] Sergey M, Andrey G, Jing J, et al. Dremel: interactive analysis of web-scale datasets. In: Proceedings of the International conference on Very Large Data Bases, Singapore, 2010. 330-339

[7] Craig C, Ashish R, Frances P, et al. FlumeJava: easy, efficient data-parallel pipelines. In: Proceedings of the 2010 ACM SIGPLAN conference on Programming language Design and Implementation, Toronto , Canada, 2010. 363-375

[8] Grzegorz M, Matthew H, Aart J. Pregel: a system for large scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, Indianapolis, USA, 2010. 135-146

[9] Tyler A, Alex B, Kaya B, et al. MillWheel: fault-tolerant stream processing at Internet scale. In: Proceedings of the International conference on Very Large Data Bases, Riva del Garda, Trento, 2013. 1033-1044

[10] Jason B, Chris B, James C, et al. Megastore: providing scalable, highly available storage for interactive services. In: the 5th Biennial Conference on Innovative Data Systems Research, Asilomar, USA, 2011. 223-234

[11] James C, Jeffrey D, Michael E, et al. Spanner: google's globally-distributed database. In: Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation, Hollywood, USA, 2012. 251-264

[12] Hadoop. http://hadoop.apache.org/, 2015

[13] Matei Z, Mosharaf C, Tathagata D, et al. Resilient Distributed Datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, Fairmont San Jose, USA, 2012. 2-16

[14] S4. http://incubator.apache.org/s4/, 2015

[15] Storm. https://storm.apache.org/, 2015

[16] Zhengping Q, Yong H, Chunzhi S, et al. TimeStream: reliable stream computation in the cloud. In: Proceedings of the 8th ACM European Conference on Computer Systems, Prague, Czech Republic,2013. 1-14

[17] Matei Z, Tathagata D, Haoyun L, et al. Discretized streams: fault-tolerant streaming computation at scale. In: Proceedings of the 24th ACM Symposium on Operating Systems Principles, Arminton, USA, 2013. 423-438

[18] Ashish T, Joydeep S, Namit J, et al. Hive - a petabyte scale data warehouse using Hadoop. In: the IEEE 26th International Conference on Data Engineering, Long Beach, USA, 2010. 996 - 1005

[19] Impala. http://impala.io/, 2015

[20] Reynold S, Josh R, Matei Z, et al. Shark: SQL and rich analytics at scale. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, USA, 2013. 13-24

[21] Christos D, Kjetil N. A survey of large-scale analytical query processing in MapReduce. *Journal on Very Large Data Bases*, 2014, 23(3):355-380

[22] Russell P, Jinyang L. Piccolo: building fast, distributed programs with partitioned tables. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, Vancouver, Canada, 2010. 1-14

[23] Zhengping Q, Xiuwei C, Nanxi K, et al. MadLINQ: large-scale distributed matrix computation for the cloud. In: Proceedings of the 7th ACM European Conference on Computer Systems, Bern, Switzerland, 2012. 197-210

[24] Nathan B, Zach A, George C, et al. TAO: facebook's distributed data store for the social graph. In: Proceedings of the 2013 USENIX conference on Annual Technical Conference, San Jose, USA, 2013. 49-60

[25] Michael I, Mihai B, Yuan Y, et al. Dryad: distributed data-parallel programs from sequential building blocks. In: Proceedings of the 2nd ACM European Conference on Computer Systems, Lisbon, Portugal, 2007. 59-72

[26] Wesley M, Paul H, Richard M. Advances in Dataflow Programming Languages. *ACM Computing Surveys*, 2004, 36(1):1-34

[27] Joseph E, Reynold S, Ankur D, et al. GraphX: graph processing in a distributed dataflow framework. In: the 11the USENIX conference on Operating Systems Design and Implementation, Broomfield, USA, 2014. 599-613

[28] MLLib. https://spark.apache.org/mllib/, 2015

[29] Christopher O, Benjamin R, Utkarsh S, et al. Pig Latin: a not-so-foreign language for data processing. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, Canada, 2008. 1099-1110

[30] Yuan Y, Michael I, Dennis F, et al. DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language. In: Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation, San Diego, USA, 2008. 1-14

[31] Stinger. http://hortonworks.com/labs/stinger/, 2015

[32] Tez. http://tez.apache.org/, 2015

[33] Chen Qi, Liu Cheng, Xiao Zhen. Improving MapReduce performance using smart speculative execution strategy. *IEEE Transactions on Computers*, 2014, 63(4) 954-967

[34] Chen Qi, Yao Jinyu Yao, Xiao Zhen. LIBRA: lightweight data skew mitigation in MapReduce. *IEEE Transactions on Parallel and Distributed Systems*, 2015, 26(9): 2520-2533

[35] Joseph E, Yucheng L, Haijie G, et al. PowerGraph: distributed graph-Parallel computation on natural graphs. In: Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation. Holleywood,

USA, 2012. 17-30

[36] Giraph. http://giraph.apache.org/, 2015

[37] Yucheng L, Joseph G, Aapo K, et al. Distributed graphLab: a framework for machine learning. In: Proceedings of the International Conference on Very Large Data Bases, Istanbul, Turkey, 2012. 716-727

[38] Zhang Kaiyuan, Chen Rong, Chen Haibo. NUMA-aware Graph-structured Analytics. In: Proceedings of the 2015 ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Bay Area, USA, 2015. 183-193

[39] Xie C N, Chen R, Guan H B, et al. SYNC or ASYNC: time to fuse for distributed graph-parallel computation. In: Proceedings of the 2015 ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Bay Area, USA, 2015. 194-204

[40] Zhu X W, Han W T, Chen W G. GridGraph: large scale graph processing on a single machine using 2-level hierarchical partitioning. In: Proceedings of the 2015 USENIX Annual Technical Conference, Santa Clara, California, USA, 2015. 375-386

[41] Quoc L, Marc'Aurelio R, Rajat M, et al. Building high-level features using large scale unsupervised learning. In: Proceedings of the 29th International Conference on Machine Learning, Edinburgh, Scotland, UK, 2012. 8595-8598

[42] Jeffrey D, Greg S, Rajat M, et al. Large scale distributed deep networks. In: Proceeding of the Neural Information Processing Systems, Lake Tahoe, USA, 2012. 1223-1231

[43] Adam C, Brody H, Tao W, et al. Deep learning with COTS HPC systems. In: Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. 1337-1345

[44] Mahout. http://mahout.apache.org/, 2015

[45] Petuum. http://petuum.github.io/, 2015

[46] Stefan C, Gustavo A, Adam A, et al. Pydron: semi-automatic parallelization for multi-core and the cloud. In: Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation, Broomfield, USA, 2014. 645-659

[47] Raul c, Matteo M, Evangelia K, et al. Making state explicit for imperative big data processing. In: Proceedings of the 2014 USENIX conference on Annual Technical Conference, Philadelphia, USA, 2014. 49-60

[48] Wang Peng, Jiang Hong, Liu Xu, et al. Towards hybrid programming in big data. In: Proceedings of the 2015 USENIX Workshop on Hot Topics in Cloud Computing, Santa Clara, USA, 2015. 12-17

Review and outlook of large scale data processing system for big data

Wang Peng*, Zhang Li**

(* Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093)

(** Institute of Disaster Prevention, Sanhe 065201)

Abstract

The lated developments in the studies of big data processing are reviewed. Firstly, the existing big data processing systems are classified from the angles of workload type and data type. Then, the advances in research on programming frameworks for batch processing are described in detail, focusing especially on the programming frameworks for the application fields of large-scale graph computing, distributed machine learning, etc., with the characteristics, challenges and design principles being discussed. Finally, the future research on big data processing is forecasted, and the conclusion that parallel training of heterogeneous hardware platform, automatic paralleling of serial codes and hybrid programming will become the focal points in this field is given.

Key words: big data, MapReduce, Spark, parallel computing, large-scale graph computing, distributed machine learning, deep learning