Hadoop 与 Spark 应用场景研究*

冯兴杰[a,b]，王文超[a]
（中国民航大学 a. 计算机科学与技术学院；b. 信息网络中心，天津 300300）

摘　要：Spark 的崛起对作为当前最为流行的大数据问题解决方案的 Hadoop 及其生态系统形成了有力的冲击，甚至一度有人认为 Spark 有取代 Hadoop 的趋势，但是因为 Hadoop 与 Spark 有着各自不同的特点，使得两者拥有不同的应用场景，从而 Spark 无法完全取代 Hadoop。针对以上问题，对 Hadoop 与 Spark 的应用场景进行了分析。首先介绍了 Hadoop 与 Spark 的相关技术以及各自的生态系统，详细分析了两者的特性；最后针对两者特性，阐述了 Hadoop 与 Spark 各自所适应的应用场景。

关键词：Hadoop；Spark；大数据；生态系统；应用场景
中图分类号：TP391　　**文献标志码**：A　　**文章编号**：1001-3695(2018)09-2561-06
doi:10.3969/j.issn.1001-3695.2018.09.001

Survey on Hadoop and Spark application scenarios

Feng Xingjie[a,b], Wang Wenchao[a]
(*a. School of Computer Science & Technology, b. Information Network Center, Civil Aviation University of China, Tianjin* 300300, *China*)

Abstract: The rise of Spark has a strong impact on Hadoop and its ecological system as two big data problems solutions, even some people think that Spark has the trend to replace Hadoop, but because Hadoop and Spark have different characteristics, so they have different application scenarios, as a result, Spark cannot completely replace Hadoop. Based on the above problems, this paper analyzed the application scenarios of Hadoop and Spark. First it introduced the Hadoop and Spark related technologies and their ecosystems, and then detailed analysis of the characteristics of the two, finally for the two characteristics, described the Hadoop and Spark each adapted to the application scenarios.

Key words: Hadoop; Spark; big data; ecosystem; application scenarios

0　引言

在过去的 20 年里，社会的各个领域所产生的数据实现了大规模的增长[1]，并且就目前来看，未来全球的数据量将呈现出爆炸性的增长趋势，会产生海量数据。2008 年《自然》杂志刊登了名为“Big Data”的专辑[2]，正式提出大数据的概念，标志着大数据时代的到来。随之而来的研究，加深了人们对大数据时代的认识[3]。大数据正在改变着人们的生活、工作和思想[4,5]。

面对如此海量的数据，传统的数据计算和数据存储方式已经无法满足要求，因此在解决当前基于大数据的实际应用问题时，也显得捉襟见肘。为此急需一套完整的大数据问题解决方案。2004 年 Google 的三篇论文 MapReduce[6]、GFS[7] 和 Big Table[8] 介绍了 Google 内部对于大数据问题的解决方案，从此奠定了针对大数据问题的关键技术基础。受此启发的 Cutting 等人开始尝试实现开源的 MapReduce 计算框架，并与 NDFS 相结合，使之成为一套完整软件，并在 2008 年成为Apache软件基金会旗下的顶级项目，命名为 Hadoop[9]。Hadoop经过近十年的发展，期间形成了以 Hive、Hbase、ZooKeeper 等软件为核心的 Hadoop 生态系统，并且仍然处在不断更新完善的过程中。目前 Hadoop 已发展到 2.0 版本，且已经成为当前最为流行且成熟的大数据问题解决方案。

随着对 Hadoop 的广泛应用，也暴露出 Hadoop 存在的一些问题。例如作为 Hadoop 核心技术的 MapReduce 计算框架，一方面缺少对迭代的支持，另一方面在计算过程中会将中间数据输出到硬盘存储，因此会产生较高的延迟。为此，加州大学伯克利分校实验室基于 AMPLab 的集群计算平台立足于内存计算开发了 Spark[10]，并提交给 Apache 软件基金会，使 Spark 作为顶级项目实现了开源。此后衍生出 Spark 生态系统，包括 MLlib、Spark SQL 等。Spark 因其具有快速、易用、通用以及有效继承 Hadoop 等特点，从提出至今也已经实现了快速发展，且正在逐渐被广泛地应用在各种实际的应用场景当中。

由于 Spark 相比于 MapReduce 所具有的各种优势，越来越多的人认为 Spark 大有取代 Hadoop 的趋势。但是一方面 Spark 准确来说只是一款大数据的计算框架，其功能类似于 Hadoop 的 MapReduce，本身不具有文件管理功能，因此它不可能完全取代 Hadoop，为了实现数据计算，还需要依靠 Hadoop 的另一项核心技术 HDFS；另一方面 Hadoop 的设计初衷是使得用户可以在不了解分布式底层细节的情况下开发分布式程序，能够充分利用集群的威力进行高速运算和存储实现数据的大规模批量处理，是一款真正意义上的大数据处理平台。而 Spark 的开发者对于 Spark 的定位是“one stack to rule them all”，也就是全栈多计算范式的高效数据流水线[11]，使得 Spark 能够完成多种复杂的任务，加之 Spark 是基于内存计算，使得 Spark 能够更高效地处理数据。但是由于内存的限制，Spark 是一款轻量级的大数据处理平台。正因为设计思想的不同，使得两者有着不同的实际应用场景。

收稿日期：2017-07-05；**修回日期**：2017-09-04　　**基金项目**：国家自然科学基金委员会与中国民用航空局联合基金资助项目(U1233113)；国家自然科学基金青年基金资助项目(61301245,61201414)

作者简介：冯兴杰(1969-)，男，河北邢台人，教授，硕导，博士，主要研究方向为数据库及数据仓库、智能信息处理理论与技术；王文超(1991-)，男，河北沧州人，硕士研究生，主要研究方向为数据挖掘、大数据技术(chaozicauc@126.com).

1 Hadoop 核心技术及生态系统

1.1 Hadoop 核心技术

目前 Hadoop 的核心技术包括 Hadoop Common、Hadoop distributed file system (HDFS)、Hadoop MapReduce 以及 Hadoop Yarn 四大模块,共同构成了 Hadoop 的基础架构。

a) Common。Common 是 Hadoop 的基础模块,主要为生态系统中其他软件提供包括文件系统、RPC 以及串行化库等功能在内的支持,为云平台提供基本服务[12]。

b) HDFS[13]。HDFS 是一个分布式文件系统,是 Hadoop 的存储核心,它可以被部署运行于大量的廉价服务器上,可以处理超大文件,其设计是建立在"一次写入,多次读取的"思想之上。对于被上传到 HDFS 上的数据,系统会对其进行分块保存。分块概念的存在是 HDFS 可以存储大量文件的重要原因。HDFS 中有 nameNode 和 dataNode 两个重要概念。nameNode 是中心服务器,负责管理文件系统命名空间以及客户端对文件的访问; dataNode 是 Hadoop 集群的主体,一般地,一个存储节点运行一个 dataNode 进程,dataNode 的作用是管理本节点上所存储的文件。HDFS 的体系结构如图 1 所示,概括了对数据基本读写操作的过程。

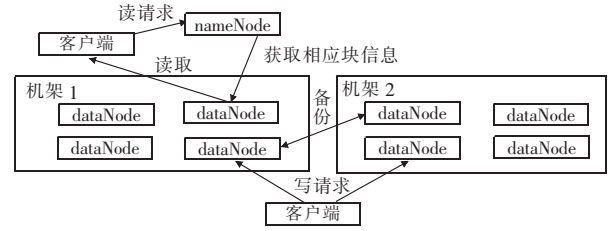

图 1 HDFS 架构

c) MapReduce。MapReduce 是一个并行计算框架,是 Hadoop 的计算核心,它通过将数据分割、并行处理等底层问题进行封装,使得用户只需要考虑自身所关注的并行计算任务的实现逻辑,从而极大地简化了分布式程序的设计。在整个计算过程中,数据始终以〈key, value〉键值对的形式存在。其核心是 map 与 reduce 函数。MapReduce 的处理流程如图 2 所示。

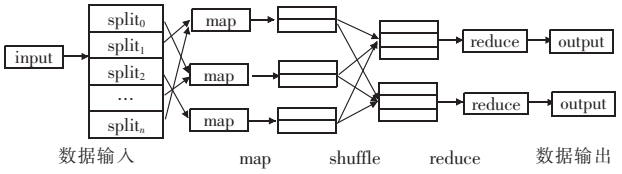

图 2 MapReduce 处理流程

对于输入数据,首先进行数据分片;然后交给 map 函数进行处理,处理之后的结果进行 shuffle 合并,合并之后的结果交由 reduce 函数处理,最终将结果输出到 HDFS 上。

d) Yarn[14]。Yarn 是针对 Hadoop 存在的 jobTracker 单点瓶颈、编程框架不够灵活等问题提出的改进方案。通过将集群资源管理和作业管理分离开来,以降低 jobTracker 的负载。其中集群资源管理由 resourceManager 负责,作业管理由 applicationMaster 负责,container 负责为集群各个节点调度资源,与所有计算节点 nodeManager 共同构成新的数据计算框架,如图 3 所示。

1.2 Hadoop 生态系统

从 Hadoop 问世至今,其核心功能在不断完善的同时,也衍生出以 Hadoop 核心技术为基础的新技术,共同构成了 Hadoop 生态系统[15],并在生态系统中发挥着不同的作用。Hadoop 生态系统主要子项目如表 1 所示。

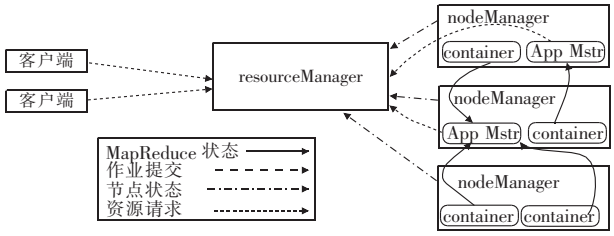

图 3 Yarn 计算框架

表 1 Hadoop 生态系统

生态系统子项目	项目功能
Hive	基于 Hadoop 文件系统的数据仓库
Hbase	基于 Hadoop 的面向列存储的数据库
Mahout	集成了经典的数据挖掘与机器学习算法的算法库
Pig	支持大规模数据分析的平台
ZooKeeper	为 Hadoop 集群提供协调服务
Avro	将数据结构或对象转换成便于存储或传输的
Chukwa	针对 Hadoop 集群的分布式数据收集和分析系统

2 Spark 核心技术及生态系统

2.1 Spark 核心技术

Spark 最主要的核心技术是 resilient distributed datasets (RDD)[16],即弹性分布式数据集,此外还包括 Spark 有向无环图(DAG)、Spark 部署模式以及 Spark 架构。

a) RDD。RDD 是对分布式内存数据的一个抽象,对数据的所有操作最终会转换成对 RDD 的操作,即 RDD 是数据操作的基本单位。对于 RDD 的操作,分为 transformation(转换)和 action(执行)。其中 transformation 又包括多种基本操作,如 map、filter、flatmap、groupByKey、reduceByKey、union 等操作; action 包括 count、collect、reduce 等操作。Spark 对于两种操作采取不同机制,对于所有的转换操作都是惰性操作,即从一个 RDD 通过转换操作生成另一个 RDD 的过程在 Spark 上并不会被马上执行,只有在 action 操作触发时,转换操作才会被真正执行。

b) DAG。在一个 Spark 应用当中,数据执行流程是以 RDD 组成的有向无环图[17]的形式存在,Spark 根据用户提交的应用逻辑,绘制 DAG,并且依据 RDD 之间的依赖关系,将 DAG 划分成不同阶段(stage);但是 DAG 绘制完成之后并不会被马上执行,只是起到一个标记数据集的作用。DAG 如图 4 所示。

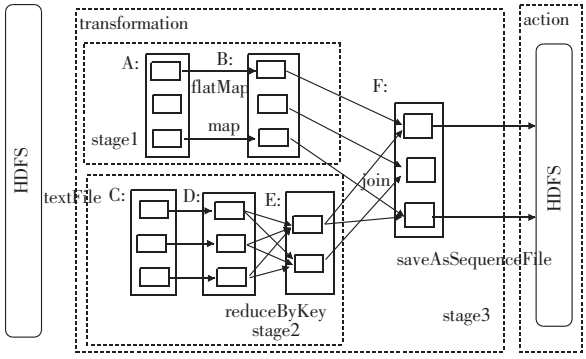

图 4 DAG

c) Spark 部署模式。当前 Spark 存在多种部署模式,包括 Local、Standalone 模式、基于 Mesos 的模式以及基于 Yarn 的部署模式。其中基于 Yarn 的部署模式是当前最为主流的部署模式,其核心思想是利用 Hadoop Yarn 实现集群资源的管理。

d) Spark 架构。尽管 Spark 有不同的部署模式,但是其基本组成部分包括控制节点 master、作业控制进程 driver、资源管理器 clusterManager、执行节点 worker、执行单元 executor 以及

客户端 client。其中 driver 进程位于 master 主控节点上,一个 worker 节点一般维护一个 executor 进程。Spark 架构如图 5 所示。

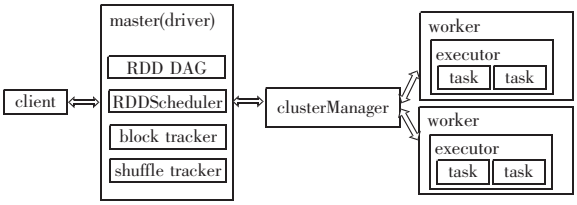

图 5　Spark 基本架构

2.2　Spark 生态系统

Spark 仅是一款高效的大数据计算框架,为了解决实际问题,还需要其他软件的支持。为了应对问题的多样化,Spark 的开发团队以 Spark 为基础开发出一套系统的数据分析软件栈(BDAS)[18],即 Spark 生态系统。

目前 Spark 生态系统主要由 Spark SQL、Spark streaming、machine learning library(MLlib)以及 GraphX 四部分组成。分别针对不同实际问题,各项目的具体功能如表 2 所示。

表 2　Spark 生态系统

Spark 生态系统子项目	项目功能
Spark SQL	基于 Spark 平台的结构化数据处理工具,Spark SQL 简化了 SQL 查询与其他复杂数据分析算法的集成,向用户提供了统一的数据源访问,此外还支持标准的 jdbc 和 odbc 连接
Spark streaming	基于 Spark 平台的流式数据计算工具,具有特定的 API,极大地方便了流式数据处理程序的开发
MLlib	基于 Spark 平台的分布式机器学习算法库,包含一些常用的分类、聚类、推荐算法
GraphX	基于 Spark 平台并行图计算框架,它提供了针对图计算的一站式解决方案

3　Hadoop 与 Spark 特性分析

3.1　Hadoop 技术优势

考虑到 MapReduce 是与 Spark 相对应的大数据编程框架,因此主要阐述 MapReduce 的技术优势。

a)MapReduce 之所以能够成为面向大数据的计算框架,原因在于其具有优良的伸缩性[19]。在 Hadoop 集群当中,每当有新的计算节点加入时,MapReduce 几乎会增加该节点所具有的全部计算能力,而如此优良的伸缩性是以往大多数分布式计算框架所不具备的,这也使得 MapReduce 成为最受欢迎的主流大数据计算框架。

b)MapReduce 为用户提供了一套极为简易的编程模型,也就是以 map 与 reduce 函数为核心的编程接口,使得在进行 MapRedcuce编程时,用户只需考虑实际应用问题的应用逻辑,而不需要考虑负载均衡、并行计算等底层细节问题。

c)MapReduce 的设计目标是为了运行批量作业,处理海量数据,在同等大小集群下所能处理的数据量上限要远大于 Spark,因此 MapReduce 是真正意义上的大数据计算框架。

3.2　Hadoop 技术劣势

a)MapReduce 是面向磁盘存储的,在 MapReduce 应用程序运行过程所有的中间数据会输出到磁盘中[20],当再需要时才会将数据从磁盘调入内存,因磁盘速度与内存速度的差距,导致了计算过程的高延迟现象。高延迟是 MapReduce 最大的技术瓶颈。

b)由于 MapReduce 主要用于处理批量数据,所以在需要对数据进行随机访问时,MapReduce 明显不能满足。

c)考虑 MapReduce 的实现方式是以 map 和 reduce 函数为基础,虽然这套编程接口极为简易,但是面对多样性的实际问题时,仅仅只有这两个算子会使得 MapRedcue 在编程时需要花费更多的时间去考虑如何能够在这两个函数中实现所需要的应用逻辑,即算子缺乏多样性和灵活性。

d)根据 MapReduce 的计算思想,在 map 过程之后,框架会对中间数据进行一个统计排序(shuffle、sort),然后才将排序之后的结果交由 reduce 过程处理。但是在某些情况下并不需要这过程,但是 MapReduce 还是会强制进行统计排序,这无疑会增加计算耗时。

3.3　Spark 技术优势

Spark 是以 MapReduce 基本思想为基础,并针对 MapReduce 现今所存在的问题而设计开发的框架。它的提出弥补了 MapReduce 诸多不足之处,因此它的优势也极为明显。

a)Spark 最明显优势在于它的内存计算[21],不同于 MapReduce面向磁盘,它的数据计算在内存中完成,产生的中间数据也大部分驻留在内存中,不需要进行 I/O 操作,使得 Spark 的计算过程要远快于 MapReduce。

b)弹性分布式数据集 RDD 作为 Spark 的核心技术,对调入内存数据实现了分布式抽象划分,使得 Spark 不仅能够进行大数据的计算,同时也可以实现数据的随机查询与分析,弥补了 MapReduce 在这方面的不足。

c)针对 MapReduce 计算算子的不足,Spark 提出的转换与执行两大类型算子来解决这个问题,使得 Spark 可以支持更为复杂的数据查询和分析任务,降低了用户开发 Spark 应用时的代码复杂度。

d)针对 MapReduce 的强制排序机制,Spark 进行了改进,改进了 shuffle 的传输方式,提升了其稳定性和速度,并利用基本算子使得 Spark 不用在所有场景中均进行排序,节省了计算耗时。

3.4　Spark 技术劣势

虽然 Spark 拥有诸多优势,且大多数情况下在性能上要优于 MapReduce,在面对大数据问题时具有更广泛的适用性,但是一方面 Spark 的开发时间较晚,远不如 MapReduce 的相关技术成熟,另一方面 Spark 在基本思想方面的一些设计使得其在适用性上也有一定局限。

a)Spark 的内存计算为其带来了速度上的优势,但是在容量上内存要远小于磁盘,MapReduce 所能处理的数据上限要远大于 Spark,因此 Spark 被定义为轻量级的大数据计算框架,而 MapReduce 是实际意义上的大数据计算框架。

b)同样由于内存计算,Spark 在计算过程中无疑会给 Java 虚拟机的垃圾回收机制带来严重压力。例如当两个 Spark 应用使用同样的数据时,那么同一份数据会被缓存两次,不但会造成较大的内存压力,同时也使得垃圾回收缓慢,从而影响 Spark 性能,导致其不稳定。

c)由于弹性分布式数据集的只读特性,使得 Spark 只适合处理粗粒度的数据并行计算,而不适合那些异步细粒度的更新计算。

4　Hadoop 与 Spark 应用场景

4.1　Hadoop 应用场景

虽然 Spark 的提出使得其在一定的应用场景中优于 Hadoop,但是 Hadoop 及生态系统技术成熟的优势仍然使得其在当前大多数基于大数据的应用场景中得到应用,并取得了明显的应用效果。其中较为广泛的应用场景包括 ETL、日志分析、数据挖掘、统计机器学习以及数据采集与处理。

4.1.1 ETL 过程场景

首先 MapReduce 作为针对处理批量数据的大数据计算框架,在面对需要批量处理的数据时表现出理想的效果,最主要的应用场景是在构建数据仓库时,针对存入数据仓库数据进行预处理,即数据的抽取、转换和装载(ETL)[22]。数据仓库主要存储的是企业大量的历史数据,原始数据在大多数情况下并不是规范的满足存储的数据,因此需要进行 ETL。但是基于传统数据仓库的 ETL 在面对大数据时的性能瓶颈使得基于 MapReduce 的并行 ETL 逐渐受到人们重视,且成为当前主流的 ETL 抽取方式。例如,Bala 等人[23]基于 MapReduce 并行编程框架,开发了一套完整的数据仓库 ETL 平台,证明了与传统 ETL 工具相比所存在的诸多优势;同时 Zhang 等人[24]也实现了基于 MapReduce 的并行 ETL 过程,并应用在 Web 页面处理之上;Li 等人[25]将基于 MapReduce 的 ETL 应用在种子的选育实际问题上,取得了比较良好的选育结果;Priya 等人[26]应用 MapReduce 的 ETL 工具进行实体识别,也同样取得了显著成果。总之,MapReduce 在针对大数据的 ETL 方面具有较强的性能优势。

4.1.2 日志处理场景

由于 MapReduce 的高延迟现象,使得其适合处理离线数据,而不适合处理实时数据,所以一些针对离线数据的应用场景,MapReduce 成为主要贡献者。日志问题是 MapReduce 在开发之初所解决的一个主要问题,离线日志是 MapReduce 所处理的主要方面。在电商交易方面,MapReduce 可以通过对用户交易数据的分析,对用户进行分类划分不同客户群,从而有益于电商企业针对不同客户开展不同的促销活动,提升企业效益,具有代表性的是阿里巴巴集团开发的基于 Hadoop 的云梯系统;在网页搜索方面,最为典型的实例是 2009 年谷歌计算机工程师发表的针对流感预测的文章,谷歌通过分析海量的用户查询日志,对冬季流感的传播趋势进行了准确预测,其中 MapReduce 发挥了重要作用;此外 Dewangan 等人[27]基于 MapReduce框架进行事务日志文件的分析,证明了 MapReduce 在事务日志分析上的性能优势;Chen 等人[28]利用 MapReduce 提出了一种名为 MaRAOS 的新框架,该框架主要针对的是离线流数据,并在 Web 日志分析证明了该框架的优势。日志的类型多种多样,包括系统、服务器、应用程序、数据库、用户活动日志等,且大部分都为离线日志。Xhafa[29]利用 MapReduce,通过对诸多类型日志的分析,揭示了 Hadoop 处理海量日志数据的潜力。总之,日志数据的分析是 MapReduce 的主要应用场景,应用 MapReduce 对离线日志进行有效的分析对企业决策有着极为重要的辅助作用。

4.1.3 数据挖掘与统计机器学习应用场景

由于 MapReduce 针对批量数据的特性,使得其在数据挖掘或者是统计机器学习领域也占有一席之地。例如谷歌著名的索引技术倒排索引,谷歌公司在研发出 MapReduce 之后,便应用 MapReduce 对其倒排索引系统进行了并行化改造,通过将海量数据的大规模操作分发给集群上的每个计算节点进行运算,同时保证每个节点会周期性地把完成的工作和状态的更新报告回主节点,使得谷歌在搜索领域占据了更大份额;此外作为 Hadoop 机器学习算法库的 Mahout 同样以 MapReduce 为计算框架,实现了基本的分类、聚类、关联规则以及推荐算法。虽然 Spark 也提出了其机器学习算法库 MLlib,但是目前在算法的多样性上仍然不如 Mahout。总之在需要进行大规模数据挖掘以及机器学习的一些应用场景,MapReduce 发挥着一定作用。

4.1.4 数据采集与处理场景

在大数据时代,只有拥有足够量的数据才能够根据数据得到正确的决策,但是当前只有少数的大型公司拥有足够的行业数据。为了解决数据不足的问题,网络爬虫技术成为重要的解决办法。但是由于 Spark 不支持异步细粒度更新,使得其不能够有效地支持爬虫技术,而 MapReduce 则能够有效地支持爬虫技术,包括增量爬虫的实现。Cafarella 等人[30]起初实现 MapReduce的主要目的是令其为 Nutch 的主要算法提供计算支持,而 Nutch 是 Apache 软件基金会下另一开源顶级项目,它是一款分布式爬虫软件,因此成为 Nutch 的标准计算引擎。例如 Li 等人[31]利用 Nutch 采集音乐页面,并利用 MapReduce 框架进行网页评分的计算,得到用户偏好音乐的推荐; Zhang 等人[32]通过对微博数据的收集,在 Nutch 框架下利用 MapRedcue 分析微博站点的特色。综上,MapReduce 适合应用在需要进行大规模数据采集的应用场景中。

4.2 Spark 应用场景

Spark 的内存计算是相比于 Hadoop MapReduce 而言最显著的优势,基于内存计算特性,使得 Spark 能够有效地处理实时数据,因此 Spark 很适合应用在对实时性要求较高的大数据场景。

4.2.1 流数据处理场景

流数据是一种顺序、快速、连续到达的数据序列,因此流数据是一种具有较强的实时性数据。但是 MapRedcue 作为一种高延迟的计算框架,针对海量流数据时处理结果的实时性无法保证,而其他针对流数据的技术如 Storm 等则无法处理海量数据,Spark 则凭借其内存计算优势有效解决了流数据的处理分析问题,特别是 Spark 生态系统中的 Spark streaming,它克服了 MapReduce 与 Strom 的缺点,在保证实时性的前提下还保证了高吞吐性与高容错性,实现了对海量流数据的处理。考虑到流数据类型的多样性,其在网络监控、航空航天、气象测控、数据通信和金融服务等应用领域广泛出现,而在当今社会环境下,这些领域所产生的数据量也急剧增长,从而使得基于 Spark 的流数据处理技术得到了广泛的关注与应用。Fang 等人[33]将 Spark 流处理技术应用在移动 Web 流量分析问题上,构建起了基于流数据的 Web 分析模型,能够帮助网络运营商有效地了解用户行为;而 Birke 等人[34]应用 Spark streaming 技术对突发事件的分析上取得了良好的预测效果;Fang 等人[35]利用 Spark streaming 进行 DDoS 网络攻击流量的检测与防御,证明了该技术能够有效地提升检测能力,保证网络安全;Maarala 等人[36]将 Spark streaming 应用在交通流量问题分析上,对交通拥堵问题进行了有效研究与解决。

综上,在当前大数据环境下,针对实时性要求较高的流数据处理问题,Spark 和 Spark straming 是目前首要的解决方案。

4.2.2 多轮迭代问题

MapReduce 在计算过程中会将计算的中间数据输出到磁盘中,因此在针对数据的重用与迭代方面拥有较低的执行效率;而 Spark 则弥补了这种不足,Spark 内存计算的特性以及核心技术 RDD 和 DAG,避免了将多次计算的中间结果写到 HDFS 的开销,同时提供 cache 机制来支持需要反复迭代计算或者多次数据共享,减少数据读取的 IO 开销,因此在需要多次操作特定数据、进行多轮迭代的应用场景中,Spark 要明显优于 MapReduce,且针对特定数据,反复操作的次数越多,所需读取的数据量越大,受益越大。多轮迭代问题在机器学习领域较为常见,众多机器学习算法在计算过程中往往需要进行多轮迭代才能够取得最优解。例如经典 K-means 算法,为了寻求最佳聚类结果,往往需要进行大量迭代,而应用 MapReduce 框架进行的并行 K-means 算法,虽然相比于串行 K-means 取得了良好的计算优势,但是因为每轮迭代 MapReduce 都会将中间聚类结

果输出,导致算法的 IO 开销较大,从而使其成为并行 K-means 的主要技术瓶颈。Wang 等人[37]则应用 Spark 框架,对 K-means 算法进行了基于 Spark 的并行化改造,且详述了并行化过程,通过实验证明了 Spark 应用在 K-means 算法上的性能优势;Kusuma 等人[38]也基于 Spark 框架对 K-means 进行了并行化,并证明了算法能够有效加快大数据问题的计算时间。此外针对其他涉及多轮迭代问题的机器学习方法,在 Spark 平台的并行化研究与应用上也取得了诸多成果。Triguero 等人[39]对 K-近邻算法进行了 Spark 的并行化;Li 等人[40]利用 Spark 平台实现了并行人工蜂群算法;Bharill 等人[41]在 Spark 平台下利用并行模糊 C-均值算法实现了对多个大数据集的处理;作为 Spark 生态系统机器学习算法库的 MLlib,目前也实现了诸多涉及多轮迭代的经典算法,如 K-means、LDA 主题模型、贝叶斯分类算法等。

综上,应用 Spark 框架进行多轮迭代问题的计算,能够在保证计算结果的前提下加快计算时间,目前在机器学习领域得到了广泛应用。

4.2.3　快速查询与实时推荐场景

在 Spark 之前,Hive 因其以 MapReduce 为计算引擎,使得在大数据查询领域得到了广泛应用;在 Spark 出现之后,凭借内存计算优势,使得其针对特定数据的快速查询优势明显,特别是在查询性能上,要优于 Hive 2 ~ 10 倍。而 Spark SQL 的提出更加增强了 Spark 这种优势。因此 Spark 以及 Spark SQL 在数据的快速查询领域逐渐得到广泛的应用。例如 Lyu 等人[42]应用 Spark 进行基于大数据的 OLTP 查询,通过更改 Spark 查询参数,在网络配置信息数据集以及 IP 通信流量数据集上进行数据的查询,实验结果表明 Spark 的响应时间短、查询速度快,满足了对快速查询的要求;Li 等人[43]通过实验对 Hive、Impala 以及 Spark SQL 进行了查询性能的对比,结果表明 Spark SQL 在多种查询场景上明显优于 Hive 与 Impala。

随着互联网技术的发展,以电商为代表的众多企业为了提升效益,纷纷制定符合相应用户群体的推荐系统,但是用户的行为偏好实时多变,唯有实时推荐系统才能够满足实时推荐需求。以 MapReduce 为基础的推荐系统因为高延迟而缺乏这种实时性,不能满足实时推荐;而使用 Spark 开发的推荐系统,能够使得推荐系统的模型训练由小时级、天级转变为分钟级,从而可以实现这种实时推荐。基于 Spark 的实时推荐系统逐渐得到商界和学术界的广泛关注与应用,如阿里、京东、雅虎、亚马逊等国内外大公司纷纷针对公司业务开发了基于 Spark 的实时推荐系统,使得推荐效果实现了极大的提升;Li 等人[44]结合 Spark 平台,利用基于内容的过滤与协同过滤技术开发了一种移动应用推荐系统,并通过实验证明了系统的实时推荐性能;Wijayanto 等人[45]提出了一种基于 Spark 的多准则协同过滤推荐系统,使得推荐结果在保证实时的前提下更为准确。

4.2.4　图计算应用场景

图计算是以“图论”为基础的对现实世界的一种“图”结构的抽象表达,以及在这种数据结构上的计算模式。图数据结构能够很好地表达数据之间的关联性,而这种关联性计算是大数据计算的核心,通过获得数据的关联性,可以从包含噪声的海量数据中抽取有用的信息。因此,为了提升计算精度,许多问题可以转换成图形式进行计算,但是 MapReduce 框架无法满足这种关联性计算。Spark 生态系统中的 GraphX 为针对图数据的计算框架,GraphX 通过丰富的图数据操作符简化了图计算的难度,因此相比于 Pregel、GraphLab 等传统计算框架具有一定的优势,并且被逐渐应用在实际的图计算问题中。例如 Ling 等人[46]将 GraphX 应用在社交网络问题中;Abu-Doleh 等人[47]使用 GraphX 进行基因组序列的检测;此外 PageRank 网页排序、能量传播模型等经典问题也在 GraphX 中得到了有效解决。

4.3　应用场景总结

以上分析了 Hadoop 与 Spark 各自所适用的应用场景,但是针对某些应用场景,Hadoop 所适用并不表示 Spark 不适用,只是说明在该特定应用场景下,Hadoop 最为适用;同理,在 Spark 所适用的应用场景中,Hadoop 也可能会存在一定适用性。针对以上应用场景,表 3 给出了 Hadoop 与 Spark 适用性总结。

表 3　Hadoop 与 Spark 场景适用性总结

	Hadoop(MapReduce)	Spark
ETL 过程场景	√	×
日志处理场景	√	√
数据挖掘与统计机器学习应用场景	√	○
数据采集与处理场景	√	×
流数据处理场景	×	√
多轮迭代问题	○	√
快速查询与实时推荐场景	○	√
图计算应用场景	×	√

注:√为适用;×为不适用;○为比较适用

其中针对日志处理,由于日志分为离线日志与实时在线日志,使得当针对离线日志时,Hadoop MapReduce 为首选计算框架;当处理实时在线日志时,Spark 具有较强的适用性。针对多轮迭代问题,MapReduce 也可以进行迭代计算,只是计算代价要高于 Spark。针对推荐问题,基于 MapRedcue 的推荐技术也已比较成熟,只是在实时性上比较差。

5　结束语

Hadoop 与 Spark 作为目前两种针对大数据问题的解决方案,因核心技术的不同而拥有不同的特性,因此在适用场景上也有所不同。本文重点分析了 Spark 与 Hadoop MapReduce 所适用的应用场景。但是在大数据环境下,实际问题往往包含多种应用场景,为了解决实际问题,需要 Hadoop 与 Spark 合作使用,这是当前大数据解决方案的发展趋势。

参考文献:

[1] Chen Min, Mao Shiwen, Liu Yunhao. Big data: a survey[J]. Mobile Networks and Applications, 2014, 19(2): 171-209.

[2] Kitchin R, McArdle G. The diverse nature of big data[J]. SSRN Electronic Journal, 2015, 25(3): 1-10.

[3] Mayer-Schonberger V, Cukier K. Big data: a revolution that will transform how we live, work, and think[M]. [S. l.]: John Murray Publishers, 2013.

[4] Walker S J. Big data: a revolution that will transform how we live, work, and think[J]. International Journal of Advertising, 2014, 33(1): 181-183.

[5] Tempini N. Book review: big data: a revolution that will transform how we live, work, and think[J]. Media Culture & Society, 2013, 37(1): 1-3.

[6] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]//Proc of Conference on Symposium on Opearting Systems Design & Implementation. Berkeley, CA: USENIX Association, 2004: 10-11.

[7] Ghemawat S, Gobioff H, Leung S T. The Google file system[C]//Proc of the 19th ACM Symposium on Operating Systems Principles. New-York: ACM Press, 2003: 29-43.

[8] Chang F, Dean J, Ghemawat S, *et al*. Bigtable: a distributed storage system for structured data[C]//Proc of USENIX Symposium on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2006: 15-15.

[9] White T, Cutting D. Hadoop: the definitive guide[M]. [S. l.]: O'Reilly Media Inc, 2009: 1-4.

[10] Zaharia M, Chowdhury M, Franklin M J, *et al*. Spark: cluster computing with working sets[C]//Proc of USENIX Conference on Hot Topics in Cloud Computing. Berkeley, CA: USENIX Association, 2010: 10.

[11] 于俊. Spark 核心技术与高级应用[M]. 北京: 机械工业出版社, 2016.

[12] Steinmacher I, Wiese I S, Chaves A P, *et al*. Newcomers withdrawal in open source software projects: analysis of Hadoop common project [C]//Proc of Brazilian Symposium on Collaborative Systems. Washington DC: IEEE Computer Society, 2012: 65-74.

[13] Wang Youwei, Zhou Jiang, Ma Can, *et al*. Clover: adistributed file system of expandable metadata service derived from HDFS[C]//Proc of IEEE International Conference on Cluster Computing. Washington DC: IEEE Computer Society, 2012: 126-134.

[14] Vavilapalli V K, Murthy A C, Douglas C, *et al*. Apache Hadoop Yarn: yet another resource negotiator[C]//Proc of the 4th Annual Symposium on Cloud Computing. New York: ACM Press, 2013: 5-7.

[15] Chowdhury B, Rabl T, Saadatpanah P, *et al*. A BigBench implementation in the Hadoop ecosystem[C]//Advancing Big Data Benchmarks. 2013: 3-18.

[16] Zaharia M, Chowdhury M, Das T, *et al*. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[C]//Proc of Conference on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2012: 2.

[17] 高彦杰. Spark 大数据处理[M]. 北京: 机械工业出版社, 2014.

[18] Stoica I. Conquering big data with spark and BDAS[J]. ACM SIGMETRICS Performance Evaluation Review, 2014, 42(1): 193.

[19] Jiang Dawei, Ooi B C, Shi Lei, *et al*. The performance of MapReduce: an in-depth study [J]. Proceedings of the VLDB Endowment, 2010, 3(2): 472-483.

[20] Gu Lei, Li Huan. Memory or time: performance evaluation for iterative operation on Hadoop and Spark[C]//Proc of International Conference on Embedded and Ubiquitous Computing. Piscataway, NJ: IEEE Press, 2013: 721 - 727.

[21] Han Zhijie, Zhang Yujie. Spark: a big data processing platform based on memory computing[C]//Proc of the 7th International Symposium on Parallel Architectures, Algorithms and Programming. Piscataway, NJ: IEEE Press, 2015: 172-176.

[22] El-Sappagh S H A, Hendawi A M A, Bastawissy A H E. A proposed model for data warehouse ETL processes[J]. Journal of King Saud University Computer & Information Sciences, 2011, 23(2): 91-104.

[23] Bala M, Boussaid O, Alimazighi Z. P-ETL: parallel-ETL based on the MapReduce paradigm[C]//Proc of the 11th ACS/IEEE International Conference on Computer Systems and Applications. Piscataway, NJ: IEEE Press, 2014: 42-49.

[24] Zhang Yan, Ma Hongtao, Xu Yunfeng. An intelligence gathering system for business based on cloud computing[C]//Proc of the 6th International Symposium on Computational Intelligence and Design. Washington DC: IEEE Computer Society, 2013: 201-204.

[25] Li Dongming, Li Yan, Yuan Chao, *et al*. Research on private cloud platform of seed tracing based on Hadoop parallel computing[C]//Proc of the 4th International Conference on Computer Science and Network Technology. Piscataway, NJ: IEEE Press, 2016: 134-137.

[26] Priya P A, Prabhakar S, Vasavi S. Entity resolution for high velocity streams using semantic measures[C]//Proc of IEEE International Advance Computing Conference. Piscataway, NJ: IEEE Press, 2015: 35-40.

[27] Dewangan S K, Pandey S, Verma T. A distributed framework for event log analysis using MapReduce[C]//Proc of International Conference on Advanced Communication Control and Computing Technologies. Piscataway, NJ: IEEE Press, 2017: 503-506.

[28] Chen Ruoyu, Zhang Yangsen, Bi Rongrong, *et al*. A MapReduce-based framework for analyzing Web logs in offline streams[C]//Proc of the 2nd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress. Piscataway, NJ: IEEE Press, 2016: 178-183.

[29] Xhafa F. Processing and analysing large log data files of a virtual campus[C]//Proc of the 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. 2015: 200-206.

[30] Cafarella M, Cutting D. Building Nutch[J]. Queue, 2004, 2(2): 100-103.

[31] Li Ying, Sha Fei, Wang Shujuan, *et al*. The improvement of page sorting algorithm for music users in Nutch[C]//Proc of the 15th International Conference on Computer and Information Science. Piscataway, NJ: IEEE Press, 2016: 1-4.

[32] Kai Zhang, Du Yuncheng, Lyu Xueqiang, *et al*. The study and implementation of micro-blog search engine based on nutch[C]//Proc of the 2nd International Conference on Future Computer and Communication. Piscataway, NJ: IEEE Press, 2010: 850-854.

[33] Fang Cheng, Liu Jun, Lei Zhenming. Fine-grained HTTP Web traffic analysis based on large-scale mobile datasets[J]. IEEE Access, 2016, 4(11): 4364-4373.

[34] Birke R, Björkqvist M, Kalyvianaki E, *et al*. Meeting latency target in transient burst: a case on spark streaming[C]//Proc of IEEE International Conference on Cloud Engineering. Piscataway, NJ: IEEE Press, 2017: 149-158.

[35] Fang Feng, Cai Zhiping, Zhao Qijia, *et al*. Adaptive technique for real-time DDoS detection and defense using Spark streaming[J]. Journal of Frontiers of Computer Science and Technology, 2016, 10(5): 601-611.

[36] Maarala A I, Rautiainen M, Salmi M, *et al*. Low latency analytics for streaming traffic data with apache Spark[C]//Proc of IEEE International Conference on Big Data. Washington DC: IEEE Computer Society, 2015: 2855-2858.

[37] Wang Bowen, Yin Jun, Hua Qi, *et al*. Parallelizing K-means-based clustering on Spark[C]//Proc of International Conference on Advanced Cloud and Big Data. Piscataway, NJ: IEEE Press, 2016: 31-36.

[38] Kusuma I, Ma'sum M A, Habibie N, *et al*. Design of intelligent K-means based on spark for big data clustering[C]// Proc of International Workshop on Big Data and Information Security. Piscataway, NJ: IEEE Press, 2017: 89-96.

[39] Triguero I, Maillo J, Luengo J, *et al*. From big data to smart data with the K-nearest neighbours algorithm[C]// Proc of IEEE International Conference on Internet of Things. Piscataway, NJ: IEEE Press, 2016: 859-864.

[40] Li Chunfeng, Wen Tingxi, Dong Huailin, *et al*. Implementation of parallel multi-objective artificial bee colony algorithm based on Spark platform[C]//Proc of the 11th International Conference on Computer Science & Education. Piscataway, NJ: IEEE Press, 2016: 592-597.

[41] Bharill N, Tiwari A, Malviya A. Fuzzy based clustering algorithms to handle big data with implementation on Apache Spark[C]//Proc of the 2nd International Conference on Big Data Computing Service and Applications. Piscataway, NJ: IEEE Press, 2016: 95-104.

[42] Lyu Yanfei, He Huihong, Zheng Yasong, *et al*. OLAP query performance tuning in Spark[C]//Proc of the 3rd International Conference on Cyberspace Technology. [S. l.]: IET Press, 2015: 1-5.

[43] Li Xiaopeng, Zhou Wenli. Performance comparison of Hive, impala and Spark SQL[C]//Proc of the 7th International Conference on Intelligent Human-Machine Systems and Cybernetics. Washington DC: IEEE Computer Society, 2015: 418-423.

[44] Li Zhengxian, Hu Jinlong, Shen Jiazhao, *et al*. A scalable recipe recommendation system for mobile application[C]//Proc of the 3rd International Conference on Information Science and Control Engineering. Piscataway, NJ: IEEE Press, 2016: 91-94.

[45] Wijayanto A, Winarko E. Implementation of multi-criteria collaborative filtering on cluster using apache Spark[C]//Proc of the 2nd International Conference on Science and Technology-Computer. Piscataway, NJ: IEEE Press, 2017: 177-181.

[46] Ling Xiao, Yang Jiahai, Wang Dan, *et al*. Fast community detection in large weighted networks using GraphX in the cloud[C]//Proc of the 18th IEEE International Conference on High Performance Computing and Communications. Piscataway, NJ: IEEE Press, 2017: 1-8.

[47] Abu-Doleh A, Catalyurek U V. Spaler: Spark and GraphX based de novo genome assembler[C]//Proc of IEEE International Conference on Big Data. Washington DC: IEEE Computer Society, 2015: 1013-1018.