文章编号:1000-5641(2014)05-0043-12

大数据环境下并行计算模型的研究进展

潘 巍[1,2], 李战怀[1]

(1. 西北工业大学 计算机学院, 西安 710072;
2. 广东省普及型高性能计算机重点实验室/深圳市服务计算与应用重点实验室, 深圳 518000)

摘要: 在大数据时代,制约并行计算发展的掣肘正在发生改变,为分布式并行计算带来了前所未有的机遇和挑战. 回顾了并行计算的发展和大数据环境下的新变化;结合硬件环境、计算模式、以及应用需求等对于并行计算模型研究的影响,综述了面向批处理、面向流处理、面向图数据以及面向内存等几类并行计算模型的相关研究;展望了其发展趋势.

关键词: 大数据; 计算模型; 并行计算; 内存计算

中图分类号: TP301 **文献标识码**: A **DOI**:10.3969/j.issn.1000-5641.2014.05.004

Development of parallel computing models in the big data era

PAN Wei[1,2], LI Zhan-huai[1]

(1. *School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an* 710072, *China*;
2. *Guangdong Key Laboratory of Popular High Performance Computers, Shenzhen Key Laboratory of Service Computing and Applications, Shenzhen* 518000, *China*)

Abstract: In the era of big data, the changing of the constraints gives the parallel computing opportunities and challenges for developing. This paper reviewed the new progress and changes of the parallel computing; combining with the effects of the hardware environments, computing pattern, application requirements on the parallel computing, the relevant research on batch-oriented parallel computing model, streaming-oriented parallel computing model, graph-oriented parallel computing model and in-memory parallel computing model are summarized; Finally, the future development trends are evaluated.

Key words: big data; computational model; parallel computing; in-memory computing

0 引 言

从计算机诞生以来,人类对于计算性能的追求从未停止,作为高性能计算和超级计算的

收稿日期:2014-07

基金项目:国家 973 重点基础研究发展计划项目基金(2012CB316203);国家自然科学基金(61033007, 61332006);国家 863 高技术研究发展计划项目基金(2009AA011902);广东省普及型高性能计算机重点实验/深圳市服务计算与应用重点实验室开放课题. 西北工业大学基础研究基金(3102014JSJ0005)

第一作者:潘巍,男,博士,讲师,研究方向为数据库理论与技术、数据密集型计算、内存计算等.

通信作者:李战怀,男,教授,博士生导师,研究方向为数据库理论与技术. E-mail: lizhh@nwpu.edu.cn.

核心技术,并行计算是充分利用资源加速计算的主要途径. 但是对于大多数程序员而言,编写分布式并行应用具有极高的编程门槛,不但需要程序员具有丰富的硬件、体系结构、操作系统等背景知识,还需要其编写大量复杂繁琐的、应用逻辑之外的分布式并行控制逻辑. 这些因素使得并行程序设计始终没有成为主流程序设计的核心. 而且,由于遵循摩尔定律的处理器的性能在过去很长一段时间内能够持续稳定的增长,因此设计和开发并行程序的必要性和紧迫性被无意的掩盖. 但是,随着大数据时代的来临,从应用需求、硬件环境、互联模式到计算技术都在发生显著的变化,人们对分布式并行计算的需求也在日益突出. 上述变化为分布式并行计算提供了新的发展契机,同时也带来了巨大的研究挑战. 其中,多样化的并行计算模型是消除分布式并行应用开发瓶颈、推动大数据发展的核心技术之一. 目前,工业界和学术界正在进行各种尝试和探索,研究和开发不同的并行计算模型,以满足大数据处理的多样化需求. 本文以并行计算模型相关研究为核心,首先回顾并行计算的发展以及大数据环境下推动其发展的主要变化,然后对现有的面向批处理模式、面向流处理模式、面向图数据以及面向内存的并行计算模型进行归纳和总结,最后阐述大数据时代并行计算模型面临的挑战和发展趋势.

1 并行计算的回顾

在计算机最初发展的十几年里,从体系结构到系统软件以及应用软件都采用串行计算作为主要的设计和开发模式. 但是很快人们就意识到,并行计算是突破串行计算效率瓶颈、提高计算性能的有力和必须的手段. 相对于串行计算,并行计算可以划分成由流水线技术为代表的时间并行,以及以多处理器并发执行为代表的空间并行,两种并行方式都可以有效地提高计算资源的利用能力,从而提高程序的执行性能,并行计算的思想也开始渗透到计算机硬件和软件技术发展的各个层面.

1954 年,阿姆达尔定律的提出者 Gene Amdahl 设计出首台内建浮点硬件的商用机器 IBM704,并行加速理念被随之带入. 1964 年第一台在技术和市场上同时获得成功的高性能大规模并行计算机 CDC6600 研制成功,多功能单元并行技术是其中领先的核心技术之一. 同时,1965 年由贝尔实验室、MIT 和美国通用电气联手开发的分时操作系统 Multics,尝试在操作系统层面通过引入并行计算思想提高运行其上的应用程序的性能和编程效率. 虽然该系统最终因为超前的设计思维、执行计划延误以及资金短缺等情况宣告失败,但是受其思想激发和推动,诞生了两个在计算机发展历史中最具影响力的软件产品:Unix 系统和 C 语言. 1966 年,Michael Flynn 根据指令流和数据流的倍数特征首次针对并行计算系统提出了著名的 Flynn 分类法[1],SISD、SIMD、MISD、MIMD 的分类至今仍然具有一定的代表性. 此外,还有 Handler 分类法、Hockney-Jesshope 等分类方法根据执行特征或者结构特征等不同条件对并行计算系统进行分类,不同的分类方法更有利于人们根据应用需求进行适合的并行系统的选择. 当然,还有大量数值并行算法和并行程序设计技术的出现也在有力推动着并行计算的发展和应用.

计算科学也成为继实验科学、理论科学之后的第三种科学研究范式,并在以生物制药、气候预测、能源勘探、高精武器设计、制造环境模拟为代表的科学计算领域获得了巨大的成功. 但是,在非科学计算领域,并行计算的普及和应用却一直未得到长足的发展. 发展滞后的主要原因有以下几点:(1)首先,在 2002 年之前,如摩尔定律所言,随着半导体技术的发

展，单个处理器的性能以平均年 50%的速度在不断提升，使得大量应用可以在不增加任何成本的情况下在新的处理器上获得指数级的性能提升. “免费的午餐”使普通程序员找不到编写复杂分布式并行应用的理由和动力；(2)其次，分布式并行编程的门槛相对较高，相对于执行路径可预测的串行程序，分布式并行程序的执行具有不确定性和异步性，因而需要大量复杂的通信、同步以及调度等并发控制操作. 虽然面向共享存储体系结构的 OpenMP 以及面向分布式存储体系结构的 MPI 等并行计算模型的出现在一定程度上解决了由程序员负责一切并发控制细节的困难局面，但是这些早期的计算模型在容错性、可扩展性、负载平衡等方面仍存在很大不足，而且抽象层次比较低，因此正确编写高性能的分布式并行应用仍具有极高的复杂度；(3)再者，并行计算的模式高度依赖于并行硬件环境和体系结构，但是并行硬件环境本身的构建也存在着成本高、可扩展性差、管理困难、能耗大等诸多问题. 因此，仅有国家实验室、航天航空、国防安全等大型机构有能力建设和维护此类高性能集群，也使得并行计算一直局限于计算密集型的科学计算领域.

上述这些问题成为广泛采用并行计算机和分布式并行计算技术的主要阻碍，也使得分布式并行计算发展遇到了一些瓶颈. 但是，在大数据时代，上述制约并行计算发展的掣肘正在发生改变，为分布式并行计算的发展带来了前所未有的机遇和挑战. (1)摩尔定律的影响力正在减弱. 因为受限于散热问题、功耗问题以及处理器材料物理特性的极限，单一地通过增加晶体管集成度从而提高单个处理器的运算能力和处理能力的可行性正在变得越来越低. 因此，应用越来越难以通过单纯地依赖处理器性能稳步提升而获得“免费”的性能提升. 同时，多\众核技术、异构多核集成技术(低功耗与高性能核心的集成、CPU 与 GPU 的组合等)以及多 CPU 的并行处理技术，已经成为目前提高处理器能力和运算速度的主流途径. 但是，现有的软件无法直接利用多核的优势提高执行性能，并行技术成为最大限度利用多\众核处理器能力的必须途径. (2)云计算为分布式并行计算提供了新的平台. 云计算从某种意义上来说，也是原始 Beowulf 集群的在商业应用模式上经过优化的衍生物，都是使用非定制的普通硬件设备、网络以及软件构建并行硬件环境. 同时，云计算技术更强调其商业运作模式，即如何通过虚拟化技术和统一的编程模型开放软硬件资源供用户使用. Pay-as-you-go 的使用模式也使得大量中小型机构甚至个人都可以根据需求构建高可定制的并行集群环境，为分布式并行计算在非科学计算领域的普及与应用带来了新的可能性和可行性. (3)大数据带来了迫切的应用需求. 在大数据时代，藉由自动感知、模拟实验、软件处理已经累积了足够多的数据，对科研人员而言如何从海量数据中发现信息、提取知识、预测趋势、制定决策成为更为紧迫的任务. 数据密集型科学研究成为工业界和学术界新的共同关注点，由于它有别于计算科学的特点、需求和环境，一种称之为“数据密集型计算”的新的研究范式成为“第四范式”. 在大数据背景下，数据密集型应用越来越多，计算系统的发展正在由高性能计算(HPC, High-Performance Computing)向高吞吐计算(HTC, High-Throughput Computing)转变. 相比于传统的 CPU 密集型计算，数据密集型计算是结合数据局部性原则采用数据并行方法实现大规模并行计算的应用技术，计算数据量级为 TB 级或 PB 级. 这种数量级是传统数据处理技术无法支撑的，分布式并行处理再次成为应对大数据应用需求的热点研究方向.

在这样的前提下，集群规模的可伸缩性和分布式并行应用的可编程性成为了主要的发展矛盾，而架构在硬件和应用之间的计算模型则成为缓解两者之间矛盾的最关键的技术之

一. 计算模型是一种涵盖存储模型、执行模型、调度模型、恢复模型的综合抽象,可以有效屏蔽大量繁杂的并行控制细节,能够在大规模廉价集群中以并行、可扩展、容错、易用、透明的方式支持各种并行算法的高效执行,为开发人员提供简洁的编程抽象,极大降低大规模集群并行编程的门槛. 在过去,分布式并行计算领域多呈现出一种应用比硬件规模小的软硬失衡的格局,关键问题就在于缺乏易用、高效、多样化的计算模型. 但是在大数据背景下,随着以 MapReduce/Hadoop 为代表的大规模数据分布式并行计算模型的出现与巨大成功,工业界和学术界对于并行计算模型的关注越来越高. 而且随着应用需求的变化,目前在不同应用领域中也涌现了多种并行计算模型. 本文在后续章节从面向批处理、面向流处理、面向图数据以及面向内存等几个层面介绍多样化的并行计算模型,以及国内外相关的研究成果.

2 并行计算模型

2.1 面向批处理的并行计算模型及优化技术

目前,具有代表性的面向批处理式分析的分布式并行计算模型有微软公司的 Dryad[2]、谷歌公司的 MapReduce[3]等. 其中,谷歌公司的 MapReduce 是最早受到关注且应用最广泛的并行编程模型,并且随着其开源实现 Hadoop 的兴起,目前已经形成了初具规模的海量数据分析新生态系统,大量的周边系统[4-8]为其成功应用提供了坚实的支撑. MapReduce 已经成为学术界和工业界事实上的海量数据并行批量处理的标准. 相比于传统的并行计算技术,MapReduce 具备了线性可扩展性、高可用性、易用性、容错性、负载平衡以及鲁棒性等大数据处理系统所必须的特性. 而且,用户在使用 MapReduce 并行编程模型时只需要关注与具体应用相关的高层处理逻辑,而将其余低层复杂的并行事务(如输入分布、任务划分与调度、任务间通信、容错处理以及负载平衡等)交与执行引擎完成. 同时,配合用户自定义的输入输出流处理、任务调度、中间数据分区和排序等可编程接口,该模型可伸缩性和可编程性上实现了极佳的平衡点. 但是,当 MapReduce 在工业界和学术界一路高歌猛进的同时,越来越多质疑和负面的声音[9,10]也随之出现. 争论主要来自于传统关系数据管理技术阵营,争论的焦点直指 MapReduce 的执行性能. 作为一个优先考虑易用性和通用性的编程模型,MapReduce 在完成同等级任务的执行性能上低于同等规模的并行数据库系统[11]. 这种性能上的差距部分来自于架构上的制约,部分来自于实现上的差异[12]. 因此,MapReduce 仍存在很大的优化改进和性能提升空间. 大量面向 MapReduce 的优化研究工作随之展开.

原始的 MapReduce 更关注工作节点间的高层次并行,并未针对多核或 GPU 等新型硬件进行专门的优化处理,为提供对多核和 GPU 的充分支持,研究人员做了大量的工作. 架构在共享内存体系结构上 Phoenix[13]系统是 MapReduce 在多核环境下的实现方案. Phoenix 的底层实现建立在 P-threads 之上,使用线程创建并行化的 Map 和 Reduce 任务,并为使用者提供了一组易用的 API 和高效的运行时. Phoenix 发布后,很多文献围绕 Phoenix 开展了有针对性的优化. 文献[14]针对 UltraSPARC 处理器实现了 Phoenix 系统多层次的优化,包括算法、实现以及 OS 接口. Mao 等人[15]通过定位多核处理中的性能瓶颈,实现了针对中间数据分组处理的数据结构级的 Phoenix 优化,并引入了一个新的 MapReduce 库 Metis. 文献[16]在 Phoenix 系统上提供了抽象程度更高更易用的 MATE 编程接口和环境,同时大幅减少了数据密集型应用所需的内存空间. 而 Phoenix++[17]则提供了一个支持模块化和可扩展流水线的 Phoenix 增强版本. 此外,文献[18]和[19]分别关注了 MapReduce 在非对

称多核集群以及异构多核集群上的研究挑战，并给出了相应的解决方案. 除了在通用多核处理器(集群)上的研究外，MapReduce 在专用的多核处理器芯片 Cell B. E(Cell Broadband Engine)以及 GPU 上也有相关的优化研究工作[20-23].

此外，为提高并行的效率，针对 MapReduce 架构性的优化也受到了广泛的关注，主要改善内容包括提高迭代性能[24]、优化调度效率[25]、增强流水线处理[26]、减小数据倾斜对性能的影响[27]、增加索引[28]以及对关系代数提供支持[29]等. 虽然 MapReduce 模型在大量优化技术的改善和支持下，在可扩展性、易用性、可用性、性价比等方面都具有较强的优势，但是作为通用的大数据分布式并行计算模型，其并没有针对某些特殊数据类型、应用领域以及新型存储器件进行专门的优化. 而且面向特定需求的优化技术既难以从根本上解决多样化数据处理需求给计算模型带来的挑战，甚至还可能会对模型本身造成不同程度的影响和损害. CCCF《2014 年大数据发展趋势预测》一文中指出："大数据处理架构的多样化模式并存"将成为未来并行计算模型发展的新趋势，MapReduce 框架一统天下的格局将被打破，而根据不同需求设计的多样化并行计算模型将成为与之并存的重要研究内容. 后面几节将介绍在不同领域获得成功应用的并行计算模型.

2.2　面向流处理的并行计算模型及优化技术

在大数据应用中，实时数据流应用是一类新型的用瞬态数据建模的数据密集型应用. 其典型特征是数据价值具有时效性，即数据蕴含的价值会随着时间的流逝而降低. 因此，低延迟是对数据流处理系统的一个基本要求. 同时，数据以大量、连续、快速、时变的方式到达系统，需要处理的数据不可能全部存储在可随机访问的磁盘或内存中. 使用面向批处理高度优化的 MapReduce 直接处理无界的数据流具有很大的应用局限性. 一种典型的 MapReduce 数据流处理策略是将无界的数据流划分成较小的有界批处理子集，然后用批处理模式对已持久化的数据流快照进行分析. 因为该处理策略会引入大量不必要的磁盘及网络 I/O 开销，所以无法满足流式应用对实时性的需求. 此外，在该策略中处理延迟与划分的长度以及 MR 作业的初始化开销高度耦合，在执行代价和响应时间上存在难以调和的矛盾. 也有部分研究者尝试改进 MapReduce 体系使其适应数据流，如 HOP[30]，但其仅解决了任务间数据传输模式导致的阻塞问题，缺少完整的面向数据流处理的系统设计.

为缓解上述问题，一些研究者尝试将 MapReduce 模型与典型的数据流系统进行融合，形成更高效的处理架构. 其中，文献[31]将 MapReduce 与分布式流处理系统 Mortar 进行了融合，实现了适合数据流处理的连续型 MapReduce. Kumar[32]等人通过在 IBM 的 System S 数据流处理中间件上进行 MapReduce 扩展实现了 DEDUCE 系统，该系统可同时支持批量处理与数据流处理. Brito[33]等人则试图将事件流处理系统(ESP)的设计概念与 MapReduce 思想进行结合，尝试构建可扩展且低延迟的数据处理系统. 布朗大学的 C-MR[34]则在 MapReduce 模型中引入滑动窗口概念以支持数据流的连续执行，但是其只适用于多核架构的单机系统. 而 M3[35]则是利用大容量内存屏蔽低效的磁盘 IO 提高流数据处理效率的 MapReduce 改进版. 对基于 MapReduce 的数据流处理而言，这些研究都是有益的尝试. 但事实上，MapReduce 在内部架构上还存在大量不利于流水处理的阻塞式设计，其中包括基于磁盘的容错、Map 阶段和 Reduce 阶段之间的障栅延迟、基于拉模式的中间数据传输方式等都是导致其高处理延迟的内在原因. 因此面向流数据的专用并行计算模型才是解决问题的根本途径.

在大数据概念出现之前,实时数据流处理领域已经出现了许多专用系统,如布朗大学与MIT 联合开发的 Aurora[36]以及后续演化的加入分布式特征的 Borealis[37]、加州大学伯克利分校的 Telegraph[38]、IBM 的 SPC[39]、威斯康星大学的 NiagaraCQ[40]等. 但是在大数据环境下,上述专用系统在可扩展性、并行度、吞吐量、可处理数据对象的多样性等方面难以满足新的挑战和需求. 近年来伴随社交网络的急速发展,大规模高扩展的流式计算模型再次成为业界研究的热点,具有代表性的有 Yahoo! 的 S4[41]、Facebook 的 Puma[42]、谷歌的 MillWheel[43]等、Twitter 的 Storm[44]等,这些系统与企业自身的具体需求紧密结合,致力于解决实际的应用问题. 相比于批处理计算模型,上述流式并行计算模型从流数据本身的特征出发,从底层架构上就与流数据处理高度耦合,虽然适用范围比较局限,但是可以有效地将系统响应时间控制在毫秒级,但是在吞吐能力、负载平衡等方面尚有待进一步提高.

2.3 面向大图数据的并行计算模型及优化技术

大数据时代背景下,数十亿顶点级别大规模图的不断涌现以及云计算基础设施的持续完善,推动着图数据处理的研究重心由单机图算法的高度优化逐渐转向分布式并行大图处理的优化. 目前,大图数据处理存在两种典型的模式:一是采用通用的海量数据分布式并行计算框架 MapReduce 进行处理;二是采用完全面向图结构设计的专用大图计算框架[45-48]. 由于 MapReduce 在应用定位与设计目标上主要是针对模式自由(schema-free)的数据对象实现高吞吐的批量处理,因此其缺乏有效处理大图的内部机制. 相比之下,专用大图并行计算模型有针对性地考虑了图计算的基本特征,因此从框架内部就已经提供了对大图处理的支持,能获得较好的性能. 其显著的特征是提供了对图计算中最常见的迭代操作的支持和实现了直观易用的以顶点为中心的编程抽象[45].

从存储架构上可以将大图并行计算模型分为:面向分布内存架构以及面向单机多核共享内存架构两类. 在分布内存架构下,目前具有代表性的大图并行计算模型有 Pregel[45]、HAMA[49]、Giraph[47]、Distributed GraphLab[50]以及 Trinity[46]等. 由于图结构的高耦合性,分布式环境下图计算的网络通信代价很高,图划分是优化分布式大图计算通信开销的有效手段. 部分大图处理系统采用经典的划分方法,如 ParMetis[51],也有其他一些系统探索了新的划分方法,如 GraphX 和 PowerGraph 均采用 vertex-cut 的划分方法[52,53]缓解自然图中高度数(high-degree)顶点通信集中的问题;Trinity 采用多层标签传递的划分方法[54],借助图的语义有效支持了十亿顶点级别大图的划分;而 GPS 和 Mizan 都利用动态的划分方法[55,56]缓解了大图计算过程中负载失衡时的顶点重分配的问题. 除了最棘手的图划分问题外,分布式内存架构仍然面临很多困难,如需要开发者具有专门的分布式编程知识在集群环境中进行复杂的调试与优化工作. 鉴于此,一些研究工作面向单机环境提出了有针对性的优化技术,如序列化随机访问[57]、利用多核以及新型存储的高并发能力[58]、引入异步机制[59]等,并设计了相应的大图计算模型. 上述研究利用外存的一些访问特性,通过有针对性的优化提升处理效率,缓解大图对内存的压力. 此外,还有一些基于单机的图计算系统[60-62],仍然假设以全内存的方式进行图数据处理. 虽然这些研究的侧重点不在存储模式,但是在大图环境下这些研究成果的实施效果可能会受到一定的影响.

此外,从计算模式上,又可以将大图并行计算模型分为同步与异步两种模式. BSP 同步计算模型对数据竞争是免疫的,无需特别的调度来保证程序执行结果的确定性(determinacy),同时超步(super step)概念天然的与图计算的迭代操作相匹配. 因此,大量图计算系

统[45-47, 49, 55]采用 BSP 同步计算模型. 虽然 BSP 同步模型具有诸多优点，但是模型本身也存在自身所固有的性能瓶颈：通信与同步. 在通信代价优化技术方面，基于邻接链表的冗余存储[55]和基于本地内存的异步访问[63]都是减少通信代价的有效手段. 在成本量化中，相比于通信，BSP 模型固有的同步开销在整体计算成本中所占的比例相对更高. 因此，异步执行机制成为解决该问题的关键所在. 基于共享内存的 GraphLab[61]及其分布式版本 Distributed GraphLab[50]是同时期具有代表性的异步大图处理系统. 相比于只能使用上轮迭代获得的输入进行计算的同步模型，异步执行模型可以使用最新的数据作为计算输入，因此其能够加快迭代的收敛速度. 此外，使用基于优先级[64]或增量[65]的优化技术还可以进一步提高迭代的收敛速度. 同时，面向顶点的执行调度也可以加快收敛，例如，动态重划分[55]和基于残差值的优先级调度[60]都是有效技术. 需要说明的是，异步虽然在图计算效率上具有优势，但对于基于异步模型的大图算法实现上，需要应用开发者管理更多的控制细节，增加了编程的难度. 因此，在大数据环境下，一方面，图在结构和计算上的高耦合性给分布式环境下大图计算模型的设计带来了难以回避的研究挑战；另一方面，新型硬件技术特别是存储级内存(storage class memory，SCM)[66]的高速发展，从硬件层面为重构分布式大图计算模型的支撑环境提供了新的机遇.

2.4　基于内存的并行计算模型及优化技术

受限于廉价 PC 构建的运行环境以及面向领域高度优化的系统架构，上述并行计算模型难以有效应对新型实时型应用对于实时、即席、交互式分析的复杂业务诉求. 同时，大数据不同维度特征所表现出的增量速度快、持续增加规模大、数据类型差异明显等客观事实，也进一步加剧了现有计算模型所面临的内存容量有限、I/O 效率低下、并发控制困难、数据处理总体性能较低等诸多问题. 然而，随着计算机硬件、体系结构的快速发展，客观上为设计以数据为中心的基于高性能服务器集群的内存数据管理新格局创造了发展契机. 大数据的海量特征一直在强调有限的存储空间无法容纳数据密集型应用的全部工作负载，但是使用基于磁盘的分布式存储环境又难以满足性能上的实时需求. 而随着 SCM 技术的快速发展，内存容量越来越大，同时价格越来越便宜，目前适用于内存计算的拥有 TB 级内存容量的服务器正在逐步普及，为内存计算提供了物理条件上的保障. 而且，相比于全部数据级，大部分应用的活跃的工作数据集明显有限，也为常驻内存处理提供了事实的依据，典型的例子如采用读写事务分离架构的 OceanBase.

在基于内存的面向大数据的分布式并行计算模型研究工作开展之前，工业界和学术界在基于内存的数据管理技术，特别是主存数据库领域已经累积的大量的研究成果和经验. 关于主存数据库研究的早期动因是内存容量的增长速度可能高于磁盘，但是当九十年代后期，新型巨磁阻技术的出现将磁盘容量年增量率由原来的 30%大幅提升到 200%后，基于内存的单一系统的发展逐渐放缓. 面向磁盘的分布式数据管理系统由于其高可扩展的显著优势成为大数据时代初期最有效的数据管理技术. 但是其性能上的不足，又促使基于分布式内存的数据管理系统的发展成为新的焦点. 近些年工业界和学术界都针对于分布式内存数据库系统进行了研究. 工业界出现了很多相关产品，Memcached[67]是最著名的全内存式数据存取系统，被 Facebook, Twitter, YouTube, Reddit 等多家世界知名企业所使用，该系统使用 DHT 实现网络拓扑的构建以及数据的布局及查询，为上层应用提供了高可用的状态存储和可伸缩的应用加速服务. 与 Memcached 类似，VMware 的 Redis[68]也提供了性能卓

越的内存存储功能,支持包括字符串、哈希表、链表、集合、有序集合等多种数据类型以及更加易于使用的 API,相比于 Memcached,Redis 提供了更灵活的缓存失效策略和持久化机制. 此外,还有如微软的 Hekaton[69],SAP 的 HANA[70]等内存数据库产品随着需求的发展仍在不断涌现. 在学术界,MIT 的 H-Store[71](VoltDB 的前身)是内存数据处理环境下的原型系统,其根据 CPU Core 进行数据分区,通过数据库多副本来获得数据的持久性. 为解决处理数据超过内存容量的问题,布朗大学的 DeBrabant 等人还针对 H-Store 系统提出了 Anti-Caching 策略[72]. 此外,慕尼黑工业大学开发的 HyPer[73]原型系统试图利用内存计算的特点基于统一的存储模型将传统上分离的 OLTP 和 OLAP 融合在一起进行处理.

近年来,在新型实时应用的驱动下,以最短响应时间为设计目标的面向内存设计的编程模型及其系统也在不断涌现,为即席实时可交互的分析提供了多样化的选择. 其中具有代表性的是 UCBerkeley 的基于内存的分布式并行处理框架 Spark[74],其利用内存计算避免了高延迟的磁盘物化,有效保证了处理的实时性并提供了交互式的迭代分析能力. Spark 提供的最主要的抽象即弹性分布式数据集 RDD(resilient distributed datasets)[75],RDD 是一个分布在一组节点之间的只读的对象集合. 这些集合是弹性的,能够在部分数据集丢失的情况下利用血统(lineage)容错机制进行重建. 为了提供操作的便捷性,Spark 框架还提供了和 Hive 类似的类 SQL 命令接口 Shark[76]. 同时,基于 Spark 的内存计算分析生态系统,如处理流数据的 Spark streaming[77],用于大图计算的 GraphX[53]等,正在不断的完善与开发之中. 需要指出的是,内存数据管理并不是一个全新的话题,但是在大数据背景下,该研究领域的外延和内涵都在发生变化,特别是随着各种新型存储技术、多/众核处理器技术、虚拟化技术、硬件图形加速技术的发展,内存数据管理特别是内存计算模型的发展将迎来新一轮的研究高峰,并将成为未来大数据实时处理领域的热点.

3 结 论

在大数据背景下,传统应用领域以及新兴应用领域都对信息系统的数据处理能力提出越来越高的需求. 从吞吐量到响应时间,从可扩展性到可容错性,从易用性到高性价比,这些指标无一不给数据处理技术带来了重大的挑战. 此外,基础硬件环境的重大技术突破以及云计算等商业模式的成功也为数据处理提供了新的发展契机. 上述原因决定了在未来很长一段时间内,分布式并行处理将是解决大数据处理问题的主要技术手段,而分布式并行计算模型在多个指标上有针对性的优化和平衡则是推动分布式并行计算在大数据环境下成功应用的关键所在. 目前,在应用需求和技术变革的双重推动下,并行计算模型已经起步,但是仍处于发展初期,尚面临着大量的研究挑战. 不过可以预见的是,未来多样化的并行计算模型百花齐放的发展格局将成为推动并行计算快速发展的有力助推器.

[参 考 文 献]

[1] FLYNN M J. Some computer organizations and their effectiveness[J]. IEEE Transactions on Computers. 1972, 21(9): 948-960.

[2] ISARD M, BUDIU M, YU Y, et al. Dryad: distributed data-parallel programs from sequential building blocks[C]//Proceedings of ACM SIGOPS/EuroSys European Conference on Computer Systems (SIGOPS 2007). Lisbon, Portugal: ACM, 2007: 59-72.

[3] DEAN J, GHEMAWAT S. MapReduce: a flexible data processing tool[J]. Commun ACM 2010, 53(1): 72-77.

[4] CAFARELLA M, CUTTING D. Building Nutch: Open Source Search[J]. Queue, 2004, 2(2): 54-61.

[5] RABKIN A, KATZ R. Chukwa: a system for reliable large-scale log collection[C] //Proceedings of the International Conference on Llarge Installation System Administration (LISA 2010). San Jose, CA: [s. n.] 2010: 1-15.

[6] HUNT P, KONAR M, JUNQUEIRA F P, et al. ZooKeeper: wait-free coordination for internet-scale systems[C] //Proceedings of the USENIX Conference on USENIX Annual Technical Conference (USENIXATC 2010). Boston, MA: [s. n.], 2010: 11-11.

[7] OLSTON C, REED B, SRIVASTAVA U, et al. Pig latin: a not-so-foreign language for data processing[C] // Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD 2008). Vancouver, Canada: [s. n.], 2008: 1099-1110.

[8] THUSOO A, SARMA J S, JAIN N, et al. Hive: a warehousing solution over a map-reduce framework[J]. Proceedings of the VLDB Endowment (VLDB 2009). 2009, 2(2): 1626-1629.

[9] DEWITT D J, STONEBRAKER M. MapReduce: A major step backwards [J]. 论坛. 2009.

[10] STONEBRAKER M, ABADI D, DEWITT D J, et al. MapReduce and parallel DBMSs: friends or foes? [J]. Communications of the ACM, 2010, 53(1): 64-71.

[11] PAVLO A, PAULSON E, RASIN A, ABADI D J, et al. A Comparision of Approaches to Large-Scale Data Analysis [J]. SIGMOD. 2009.

[12] JIANG D, OOI B C, SHI L, et al. The performance of MapReduce: An in-depth study[J]. VLDB. 2010.

[13] RANGER C, RAGHURAMAN R, PENMETSA A, et al. Evaluating MapReduce for Multi-core and Multiprocessor Systems[C] //Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture. 2007: 13-24.

[14] YOO R M, ROMANO A, KOZYRAKIS C. Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system[C] //Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC). 2009: 198-207.

[15] MAO Y, MORRIS R, KAASHOEK M F. Optimizing MapReduce for Multicore Architectures[J]. Technical Report MIT-CSAIL-TR-2010-020. 2010.

[16] JIANG W, RAVI V T, AGRAWAL G. A Map-Reduce System with an Alternate API for Multi-core Environments[C] //Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID 2010). 2010: 84-93.

[17] TALBOT J, YOO R M, KOZYRAKIS C. Phoenix + +: modular MapReduce for shared-memory systems[C] // Proceedings of the Second International Workshop on MapReduce and Its Applications (MapReduce 2011). San Jose, California: [s. n.], 2011: 9-16.

[18] RAFIQUE M M, ROSE B, BUTT A R, et al. Supporting MapReduce on large-scale asymmetric multi-core clusters[J]. ACM SIGOPS Operating Systems Review. 2009, 43(2): 25-34.

[19] LINDERMAN M D, COLLINS J D, WANG H, et al. Merge: a programming model for heterogeneous multi-core systems[J]. ACM SIGOPS Operating Systems Review, 2008, 42(2): 287-296.

[20] KRUIJF M D, SANKARALINGAM K. MapReduce for the cell broadband engine architecture[J]. IBM Journal of Research and Development, 2009, 53(5): 747-758.

[21] RAFIQUE M M, ROSE B, BUTT A R, et al. CellMR: A framework for supporting mapreduce on asymmetric cell-based clusters[C] //Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing. 2009: 1-12.

[22] CHEN L, AGRAWAL G. Optimizing MapReduce for GPUs with effective shared memory usage[C] //Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing (HDPC 2012). Delft, The Netherlands: [s. n.], 2012: 199-210.

[23] HE B, FANG W, LUO Q, et al. Mars: a MapReduce framework on graphics processors[C] //Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT 2008). Toronto, Ontario: [s. n.], 2008: 260-269.

[24] ONIZUKA M, KATO H, HIDAKA S, et al. Optimization for iterative queries on MapReduce[C]. Proceedings of the VLDB Endowment (VLDB 2014). 2014, 7(4).

[25] SUN X, HE C, LU Y. ESAMR: An Enhanced Self-Adaptive MapReduce Scheduling Algorithm[C] //Proceedings of the 2012 IEEE 18th International Conference on Parallel and Distributed Systems. 2012: 148-155.

[26] LI B, MAZUR E, DIAO Y, et al. A platform for scalable one-pass analytics using MapReduce[C] //Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD 2011). Athens, Greece, 2011: 985-996.

[27] KWON Y, BALAZINSKA M, HOWE B, et al. SkewTune: mitigating skew in mapreduce applications[C] //Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2012). Scottsdale, Arizona, USA, 2012: 25-36.

[28] DITTRICH J, QUIANERUIZ J A, RICHTER S, et al. Only aggressive elephants are fast elephants[J]. Proceedings of the VLDB Endowment (VLDB 2012). 2012, 5(11): 1591-1602.

[29] ZHANG X, CHEN L, WANG M. Efficient multi-way theta-join processing using MapReduce[C]. Proceedings of the VLDB Endowment (VLDB 2012). 2012, 5(11): 1184-1195.

[30] CONDIE T, CONWAY N, ALVARO P, et al. MapReduce online[C] //Proceedings of USENIX conference on networked systems design and implementation (NSDI 2010). San Jose, California, 2010: 21-21.

[31] LOGOTHETIS D, YOCUM K. Ad-hoc data processing in the cloud[C]. Proceedings of the VLDB Endowment (VLDB 2008). 2008, 1(2): 1472-1475.

[32] KUMAR V, ANDRADE H, B U, #287, GEDIK R, et al. DEDUCE: at the intersection of MapReduce and stream processing[C] //Proceedings of the 13th International Conference on Extending Database Technology (EDBT 2010). Lausanne, Switzerland, 2010: 657-662.

[33] BRITO A, MARTIN A, KNAUTH T, et al. Scalable and Low-Latency Data Processing with Stream MapReduce [C] //Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CLOUDCOM 2011). 2011: 48-58.

[34] BACKMAN N, PATTABIRAMAN K, FONSECA R, et al. C-MR: continuously executing MapReduce workflows on multi-core processors[C] //Proceedings of Third International Workshop on MapReduce and Its Applications Date (MapReduce 2012). Delft, The Netherlands: [s. n.], 2012: 1-8.

[35] ALY A M, SALLAM A, GNANASEKARAN B M, et al. M3: Stream processing on main-memory mapReduce [C] //Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE 2012). 2012: 1253-1256.

[36] ABADI D J, CARNEY D, CETINTEMEL U, et al. Aurora: a new model and architecture for data stream management[C]. Proceedings of the VLDB Endowment (VLDB 2003). 2003, 12(2): 120-139.

[37] AHMAD Y, BERG B, CETINTEMEL U, et al. Distributed operation in the Borealis stream processing engine [C] //Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD 2005). Baltimore, Maryland: [s. n.], 2005: 882-884.

[38] MADDEN S, SHAH M, HELLERSTEIN J M, et al. Continuously adaptive continuous queries over streams[C] //Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD 2002). Madison, Wisconsin: [s. n.], 2002: 49-60.

[39] AMINI L, ANDRADE H, BHAGWAN R, et al. SPC: a distributed, scalable platform for data mining[C] //Proceedings of the international workshop on data mining standards, services and platforms (DMSSP 2006). Philadelphia, Pennsylvania: [s. n.], 2006: 27-37.

[40] CHEN J, DEWITT D J, TIAN F, et al. NiagaraCQ: a scalable continuous query system for Internet databases [C] //Proceedings of the ACM SIGMOD international conference on Management of data (SIGMOD 2000). Dallas, Texas: [s. n.], 2000: 379-390.

[41] NEUMEYER L, ROBBINS B, NAIR A, et al. S4: Distributed Stream Computing Platform[J]. Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW 2010). 2010: 170 -177

[42] ZHENG S. Data Freeway and Puma: Realtime Data Streams and Analytics[C] //Proceedings of Hadoop in China (HiC 2011) 2011.

[43] AKIDAU T, BALIKOV A, BEKIROGLU K, et al. Whittle. MillWheel: fault-tolerant stream processing at internet scale[C]. Proceedings of the VLDB Endowment (VLDB 2013). 2013, 6(11): 1033-1044.

[44] STORM, http://storm.incubator.apache.org/.

[45] MALEWICZ G, AUSTERN M H, BIK A J C, et al. Pregel: a system for large-scale graph processing[C] //Proceedings of the ACM SIGMOD International Conference on Management of data (SIGMOD 2010). Indianapolis, Indiana: [s. n.], 2010: 135-146.

[46] SHAO B, WANG H, LI Y. Trinity: a distributed graph engine on a memory cloud[C] //Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2013). New York: [s. n.], 2013: 505-516.

[47] Giraph, https://giraph.apache.org/.

[48] VENKATARAMANI V, AMSDEN Z, BRONSON N, et al. TAO: how facebook serves the social graph[C] // Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2012) Scottsdale, Arizona: [s. n.], 2012: 791-792.

[49] HAMA, https://hama.apache.org/.

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

[51] KARYPIS G, KUMAR V. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering[J]. Journal of Parallel and Distributed Computing, 1998, 48(1): 71-95.

[52] GONZALEZ J E, LOW Y, GU H, et al. PowerGraph: distributed graph-parallel computation on natural graphs [C] //Proceedings of the USENIX Conference on Operating Systems Design and Implementation (OSDI 2012). Hollywood, CA: [s. n.], 2012: 17-30.

[53] XIN R S, GONZALEZ J E, FRANKLIN M J, et al. GraphX: a resilient distributed graph system on Spark[C] // First International Workshop on Graph Data Management Experiences and Systems (GRADES 2013). New York: [s. n.], 2013: 1-6.

[54] WANG L, XIAO Y, SHAO B, et al. How to Partition a Billion-Node Graph[C] //Proceedings of the IEEE International Conference on Data Engineering (ICDE 2014). (Accepted).

[55] SALIHOGLU S, WIDOM J. GPS: a graph processing system[C] //Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM 2013). Baltimore, Maryland, 2013: 1-12.

[56] KHAYYAT Z, AWARA K, ALONAZI A, et al. Mizan: a system for dynamic load balancing in large-scale graph processing[C] //Proceedings of the ACM European Conference on Computer Systems (EuroSys 2013). Prague: [s. n.], 2013: 169-182.

[57] KYROLA A, BLELLOCH G, GUESTRIN C. GraphChi: large-scale graph computation on just a PC[C] //Proceedings of the USENIX conference on Operating Systems Design and Implementation (OSDI 2012). Hollywood, CA: [s. n.], 2012: 31-46.

[58] HAN W S, LEE S, PARK K, et al. TurboGraph: a fast parallel graph engine handling billion-scale graphs in a single PC[C] //Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD 2013). Chicago, Illinois: [s. n.], 2013: 77-85.

[59] PEARCE R, GOKHALE M, Amato N M. Multithreaded Asynchronous Graph Traversal for In-Memory and Semi-External Memory[C] //Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2010). 2010: 1-11.

[60] WANG G, XIE W, DEMERS A, et al. Asynchronous Large-Scale Graph Processing Made Easy [C] //Proceedings of the Conference on Innovative Data Systems Research (CIDR 2013). 2013.

[61] LOW Y, GONZALEZ J, KYROLA A, et al. Graphlab: A new parallel framework for machine learning[C] // Conference on Uncertainty in Artificial Intelligence (UAI 2010). Catalina Island, CA, USA, 2010: 340-349.

[62] XIE W, WANG G, BINDEL D, et al. Fast iterative graph computation with block updates[C]. Proceedings of the VLDB Endowment (VLDB 2013). 2013, 6(14): 2014-2025.

[63] TIAN Y, BALMIN A, CORSTEN S A, et al. From "Think Like a Vertex" to "Think Like a Graph"[C]. Proceedings of the VLDB Endowment (VLDB 2013). 2013, 7(3): 193-204.

[64] ZHANG Y, GAO Q, GAO L, et al. PrIter: a distributed framework for prioritized iterative computations[C] // Proceedings of the ACM Symposium on Cloud Computing (SOCC 2011). Cascais, Portugal, 2011: 1-14.

[65] ZHANG Y, GAO Q, GAO L, et al. Maiter: An Asynchronous Graph Processing Framework for Delta-based Accumulative Iterative Computation[J]. IEEE Transactions on Parallel and Distributed Systems (TPDS), 2013: Pre-Print.

[66] JUNG J, WON Y, KIM E, et al. FRASH: Exploiting storage class memory in hybrid file system for hierarchical storage[J]. ACM Transactions on Storage (TOS), 2010, 6(1): 1-25.

[67] FITZPATRICK B. Distributed caching with memcached[J]. Linux Journal. 2004, 2004(124): 5.

[68] ZAWODNY J. Redis: Lightweight key/value Store That Goes the Extra Mile[J]. Linux Magazine. 2009.

[69] DIACONU C, FREEDMAN C, ISMERT E, et al. Hekaton: SQL server's memory-optimized OLTP engine[C] // Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2013). New York, New York: [s. n.], 2013: 1243-1254.

[70] SIKKA V, FARBER F, GOEL A, et al. SAP HANA: the evolution from a modern main-memory data platform to an enterprise application platform[C]//Proceedings of the VLDB Endowment (VLDB 2013). 2013, 6(11): 1184-1185.

[71] KALLMAN R, KIMURA H, NATKINS J, et al. H-store: a high-performance, distributed main memory transaction processing system[C]//Proceedings of the VLDB Endowment (VLDB 2008). 2008, 1(2): 1496-1499.

[72] DEBRABANT J, PAVLO A, TU S, et al. Anti-caching: a new approach to database management system architecture[J]. Proceedings of the VLDB Endowment (VLDB 2013). 2013, 6(14): 1942-1953.

[73] KEMPER A, NEUMANN T. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots[C] //Proceedings of the IEEE International Conference on Data Engineering (ICDE 2011). 2011: 195-206.

[74] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: cluster computing with working sets[C] // Proceedings of the USENIX conference on hot topics in cloud computing (HotCloud 2010). Boston, MA: [s. n.], 2010: 10.

[75] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[C] //Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI 2012). San Jose, CA: [s. n.], 2012: 2-2.

[76] XIN R S, ROSEN J, ZAHARIA M. et al. Shark: SQL and rich analytics at scale[C] //Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2013). New York: [s. n.], 2013: 13-24.

[77] ZAHARIA M, DAS T, LI H, et al. Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters[C] //Proceedings of the USENIX Conference on Hot Topics in Cloud Computing (HotCloud 2012). Boston, MA: [s. n.], 2012: 10-10.

(责任编辑 王善平)