软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(8):2147−2167 [doi: 10.13328/j.cnki.jos.005103]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

内存计算技术研究综述*

罗 乐, 刘 轶, 钱德沛

(北京航空航天大学 计算机学院,北京 100191)

通讯作者: 罗乐, E-mail: luole@buaa.edu.cn

摘 要: 在大数据时代,如何高效地处理海量数据以满足性能需求,是一个需要解决的重要问题.内存计算充分利用大容量内存进行数据处理,减少甚至避免 I/O 操作,因而极大地提高了海量数据处理的性能,同时也面临一系列有待解决的问题.首先,在分析内存计算技术特点的基础上对其进行了分类,并分别介绍了各类技术及系统的原理、研究现状及热点问题;其次,对内存计算的典型应用进行了分析;最后,从总体层面和应用层面对内存计算面临的挑战予以分析,并且对其发展前景做了展望.

关键词: 内存计算;新型混合内存;分布式集群;图计算;大数据处理

中图法分类号: TP316

中文引用格式: 罗乐,刘轶,钱德沛.内存计算技术研究综述.软件学报,2016,27(8):2147−2167. http://www.jos.org.cn/1000-9825/5103.htm

英文引用格式: Luo L, Liu Y, Qian DP. Survey on in-memory computing technology. Ruan Jian Xue Bao/Journal of Software, 2016,27(8):2147−2167 (in Chinese). http://www.jos.org.cn/1000-9825/5103.htm

Survey on In-Memory Computing Technology

LUO Le, LIU Yi, QIAN De-Pei

(School of Computer Science and Engineering, BeiHang University, Beijing 100191, China)

Abstract: In the era of big data, systems need to process massive data efficiently to meet performance requirements of applications. In-memory computing technology can improve performance of massive data processing significantly by utilizing memory and avoid I/O operations. However, the technology faces a series of challenges that need to be solved. This paper analyzes characteristics of in-memory computing technology, lays out its classification, and introduces principles, related works and hot topics of every category. In addition, several typical applications of in-memory technology are introduced. Finally, challenges and opportunities for in-memory computing are elaborated.

Key words: in-memory computing; hybrid memory; distributed system; graph processing; big data processing

随着大数据时代的来临,数据规模越来越大,从万亿字节(TB)到千万亿字节(PB)级;数据种类繁多,包括传统的结构化数据,又包括文字、图片、音频和视频等非结构化数据,且非结构化数据比重在快速增长;而数据快速增长,引发的数据处理时效性难以保障.大数据所带来的大规模及需要实时处理等特点与传统的以计算为中心的模式产生巨大矛盾,使得传统计算模型难以适应当今大数据环境下的数据处理.总的来说,数据处理从以计算为中心转变成了以数据为中心,这样,通过使用传统的内存—磁盘访问模式来处理大数据总会存在 I/O 瓶颈,处理的速度问题愈发突出,且时效性难以保证.现有的方案都只能一定程度上缓解这个瓶颈,而不能彻底解决这个

* 基金项目: 国家自然科学基金(91530324); 国家高科技研究发展计划(863)(2015AA01A301)

Foundation item: National Natural Science Foundation of China (91530324); National High-Tech R&D Program of China (863) (2015AA01A301)

收稿时间: 2015-07-22; 修改时间: 2016-03-18; 采用时间: 2016-05-23; jos 在线出版时间: 2016-07-30

CNKI 网络优先出版: 2016-08-01 09:38:56, http://www.cnki.net/kcms/detail/11.2560.TP.20160801.0938.005.html

问题.而大数据所表现出的增量速度快、时间局部性低等特点,客观上使得以计算为中心的传统模式面临着内存容量有限、输入/输出(I/O)压力大、缓存命中率低、数据处理的总体性能低等诸多挑战.

对此,在体系结构方面,工业界和学术界通过使用众核处理器及分布式集群,增加协处理器和 GPU,使用大内存,增加 I/O 通道等方式来应对.然而,大内存、众核处理器等能耗超高.文献[1,2]指出,如今大部分的 DRAM 内存能耗多达 40%,这对于如今的大数据处理是一个非常重要的考虑因素.在编程模型方面,研究者提出了以 MapReduce[3],Hadoop[4]为代表的编程框架来解决大数据问题.MapReduce 在分布式系统上具有很好的可扩展性和容错性,但是它需要从磁盘获取数据,再将中间结果数据写回磁盘,导致系统的 I/O 开销极大,不适用于具有实时性需求的应用.为了解决 I/O 开销大的问题,近些年又衍生出许多 In-Memory MapReduce 系统[5−14],即把 Map 结果不再写入磁盘,而是将其写入内存,这些就避免了过多的 I/O 操作,减少了开销.然而,虽然分布式的编程框架在一定程度上解决了大数据处理的问题,但是分布式系统带来的一致性问题、节点间通信及容错数据复制等,在一定程度上限制了大数据处理的并行性.

近年来,随着多核 CPU 的快速发展,内存价格的不断下降,以及系统架构的不断演进下,为大数据处理在硬件方面提供了有利的条件.SAP 公司在 2012 年推出的 HANA 内存计算[15]及加州大学伯克利分校开发的 Apache Spark[16]使得内存计算再次得到学术界和工业界的广泛关注.同时,IBM 的 solidDB[17]、Oracle 的 Exadata X3、微软的 SQLServer 2012 也引入了内存计算.内存计算不是最新提出的概念,但是近年来它却成为业界和研究领域的一个热点,解决了前面提到的大数据时代数据处理速度以及时效性的问题,其原因在于,在内存计算模式下,所有的数据在初始化阶段全部加载到内存中,数据及查询的操作都在高速内存中执行,CPU 直接从内存读取数据,进行实时地计算和分析,减少了磁盘数据访问,降低了网络与磁盘 I/O 的影响,大幅提升了计算处理的数据吞吐量与处理的速度,减少了原本占大量计算资源的 I/O 开销.通过内存计算的应用,避免了 I/O 瓶颈,以前在数小时、数天时间内计算的结果,在内存计算环境中,可以在数秒内完成.因此,在高性能的计算背景下,内存计算能再次成为工业界和学界研究关注的热点,成为海量数据分析的利器则不足为奇.另外,关于大内存引起的能耗问题,近些年出现了大量新型非易失性随机存储介质,比如电阻存器、铁电存储器、相变存储器等,其容量大、价格低、读写速度与 DRAM 相当,最重要的是,其能耗远远低于 DRAM,因此,在一定程度可以替换 DRAM 成为新型内存.为此,出现了大量 DRAM 与非易失性的混合内存的研究.

本文主要对内存计算的技术特点、分类、研究现状、热点问题和典型应用进行介绍分析,展望内容计算的发展前景.首先,介绍和分析内存计算的概念和技术特点;其次,给出内存计算技术及系统的分类,介绍 3 种类别的原理、现状和问题;再次,介绍内存计算的几种典型应用;最后,从总体层面和应用层面对内存计算面临的挑战予以分析,并且对其发展前景进行展望.

1 内存计算概念

内存计算不是一个新的概念,早在 20 世纪 90 年代就有关于内存计算雏形的论述[18,19],只是当时硬件发展有限,没有得到进一步地研究.关于内存计算的概念,至今没有统一的定义.Gartner 对其定义为:一种应用平台中间件,实现分布式、可靠及可扩展性、强一致或最终一致性的内存 NoSQL 数据存储,可供多个应用共享[20].IBM 给出的解释是:内存计算主要是将数据存放在服务器的内存中,以此作为数据处理加速的一个手段,其主要适用于数据访问密集型的应用[21].GridGrain 关于内存计算给出这样的解释:通过使用一种中间件的软件将数据存储于分布式集群中的内存当中,并且进行并行处理[22].TIBCO 认为,内存计算是对处理大数据所遇到瓶颈的一种突破[23].Techopedia 认为,随着内存价格大幅下迭,内存容量增长,这样就更好地将信息存入专用服务器内存,而不是存储速度更慢的磁盘.它能帮助商务用户快速地进行模式识别,及时分析大数据,即,所谓的内存计算[24].内存计算不仅仅是把数据驻留内存,还需要对软件体系、计算模型等进行专门的设计[25].因此可以看出,内存计算主要有以下特性:

(1) 硬件方面拥有大容量内存,可将待处理数据尽量全部存放于内存当中,内存可以是单机内存或者分布式内存,且单机内存要足够大;

(2) 具有良好的编程模型和编程接口;

(3) 主要面向数据密集型应用,数据规模大,处理实时性要求高;

(4) 大多支持并行处理数据.

综上所述,内存计算是以大数据为中心、依托计算机硬件的发展、依靠新型的软件体系结构,即,通过对体系结构及编程模型等进行重大革新,将数据装入内存中处理,而尽量避免 I/O 操作的一种新型的以数据为中心的并行计算模式.在应用层面,内存计算主要用于数据密集型计算的处理,尤其是数据量极大且需要实时分析处理的计算.这类应用以数据为中心,需要极高的数据传输及处理速率.因此在内存计算模式中,数据的存储与传输取代了计算任务成为新的核心.

内存计算与传统的内存缓存有着较大的区别,主要体现在数据在内存中的存储和访问方式上.在内存计算中,数据长久地存储于内存中,由应用程序直接访问.即使当数据量过大导致其不能完全存放于内存中时,从应用程序视角看,待处理数据仍是存储于内存当中的,用户程序同样只是直接操作内存,而由操作系统、运行时环境完成数据在内存和磁盘间的交换.而内存缓存,利用部分内存缓存磁盘/文件数据,应用程序通过文件系统接口访问缓存中的数据,而不是像内存计算那样直接访问.因此,内存计算和传统的内存缓存都可以通过减少 I/O 操作提升系统性能,内存计算支持数据直接访问,效率更高,也更适合大数据应用,缺点是不像传统内存缓存那样对应用程序透明,通常需要专门的编程模型和接口支持.

2 内存计算分类

内存计算系统结构和实现方法在很大程度上取决于底层硬件架构,更准确地说,取决于底层内存架构.根据内存计算所依托硬件架构的不同,可将内存计算分为 3 类:(1) 基于单节点的内存计算;(2) 基于分布集群的内存计算;(3) 基于新型混合结构内存的内存计算.

2.1 基于单节点的内存计算

单节点内存计算系统运行于单个物理节点上,节点拥有一个或多个处理器以及共享内存,内存结构可以是集中式共享内存,或者非一致性共享内存(non-uniform memory access,简称 NUMA).单节点上的内存计算利用多核 CPU,采用大内存和多线程并行,以充分发挥单机的计算效能,并且采取充分利用内存和 CPU 的 cache、优化磁盘读取等措施.

在软件层面,单节点内存计算主要分为内存数据处理系统和内存存储系统两类:

- 在内存数据处理系统方面,主要利用如今发展起来的众核 CPU 和大内存,使得单节点系统有一定的大数据处理能力;且其易于编程,CPU 和内存资源能被充分利用,因此具有良好的使用价值.文献[27]中指出,拥有 100G 到 1TB 内存的高端服务器足够处理现实中的图数据.近几年出现了众多基于众核内存计算的数据处理框架,比如 Grace[26],Ligra[27],GRACE[28]和 GraphLab[29].其中,Grace,Ligra 和 GRACE 都利用多核 CPU 和大内存,并采用了多线程并行技术,充分利用内存和 CPU,只是三者处理机制及侧重点不同.Grace 提出一种图更新聚集策略,针对图划分间(线程之间)进行了通信及负载均衡优化,提高图处理的效率;Ligra 以图为中心的计算方式,提出一种轻量级图处理框架,其重点在于使图遍历算法更容易实现;GRACE 则重点针对同步模式,提出一种可以根据应用处理需求,由用户决定切换同步执行还是异步执行的并行图处理框架.而 GraphLab 是一种基于图模型的处理机器学习算法的异步并行框架.
- 在内存存储系统方面,单节点内存计算在数据库方面应用较多.比如早在上世纪 80 年代,全内存数据库 MMDB[30,31],其思路是,将整个数据库存入内存,即可加快数据处理.然而在当时,由于内存价值高,这种思路只是理论论述,直到近些年内存数据库对于企业级用户来说才可实现.比如:Hyper[32]是一种混合式 OLTP 和 OLAP 高性能内存数据库;Hekaton[33]是一个针对事务处理(TP)的基于行的内存数据管理系统,其为遗留应用程序提升 10 倍的 TP 速度、为新优化的应用提升 50 倍的速度,且完全集成进 SQL Server.又如,图数据库 WhiteDB[34]是建立在共享内存上的一个轻量级的 NoSQL 内存数据库,它没有服务器进程,可直接对其共享内存进行读写.Neo4j[35]同样是建立在单节点众核上的一种广泛使用的内存

图数据库.

相对分布式内存计算而言,单节点内存计算资源利用率高、处理效率高,不需要管理集群及考虑容错,也不存在节点间通信的巨大开销,系统性能也具有较强的可预估性;从编程者的角度来看,调试及优化算法比分布式更容易.缺点是单节点 CPU、内存等资源有限,在单节点计算机上处理现实世界的大数据,则很可能面临内存不足的情况(out-of-core).在此情况下,内存数据处理的解决方案有 3 种趋势.

(1) 内存压缩技术.Ligra+[36]系统就是在 Ligra 系统之上添加内存压缩技术,达到和 Ligra 相当的性能.

(2) 提高 I/O 访问效率.Graphchi[37]正是针对内存不足的情况,采用内存并行滑动窗口机制的一种单节点众核并行图处理框架.PrefEdge[38]通过预取减少了固态硬盘 I/O 随机访问次数.X-Stream[39]利用了流数据的顺序访问特性,对其与随机访问进行折中.实验表明:X-Stream 在拥有 3TB 磁盘的单节点系统上,可处理含 640 万条边的图数据.

(3) 使用高速 I/O 设备,比如固态硬盘阵列.FlashGraph[40]即为基于固态硬盘阵列的异步图处理框架,它采用消息传递机制,以顶点为中心,在处理数据过程中,图顶点和算法状态存于内存,边存于外存当中,其处理性超出 X-Stream 和 Graphchi 几个数量级.

2.2 基于分布式系统的内存计算

单节点内存计算受硬件资源限制,在处理更大规模数据时面临硬件可扩展方面的问题.在以 MapReduce 为代表的大规模分布式数据处理技术快速发展的背景下,人们也开始在分布式系统上实现内存计算.这种内存计算利用多台计算机构成的集群构建分布式大内存,通过统一的资源调度,使待处理数据存储于分布式内存中,实现大规模数据的快速访问和处理.

根据内存计算的主要功用,可将基于分布式系统的内存计算进一步分为 3 种类型.

2.2.1 内存存储系统

近年来,磁盘容量快速增长,在过去的 25 年中增长 10 000 多倍,且还有继续增长的态势,但磁盘访问性能增速远低于容量增速,同时期数据传输率仅提高了 50 倍,寻道时间及转动延迟仅降低为以前的 1/2.因此,磁盘 I/O 成了主要的数据访问瓶颈,难以满足在线海量数据处理需求.为此,就有了这样的技术思路:将内存作为存储设备,数据全部存入内存,而将磁盘仅作为一种备份或存档工具.

RAMCloud[41]是一种典型的分布式内存数据存储模型,这种分布内存计算是通过上百台甚至上千台服务器互联,形成分布式内存存储系统,且易于扩展.其思路是,所有数据一直缓存于内存中.与基于磁盘的数据处理相比,该系统的延迟小 100~1000 倍,吞吐量大 100~1000 倍.考虑到内存易失性问题,RAMCloud 采用复制和备份技术,以保证它具有像磁盘存储系统一样的持续性和有效性,这也保证了内存数据和磁盘数据的同步性、一致性以及容错.但是,这样内存系统存在一定的缺点,即:其能耗很高,是磁盘系统的 50~100 倍.因此它适合有大吞吐量需求的应用,而对一些不需频繁访问且占大量存储空间的数据的应用不太适合.另一种典型内存存储系统是分布式内存数据库,包括分布式内存关系数据库,如 H-store[42],VoltDB[43],ScyPer[44]等,分布式 NoSQL 数据库,如 MemepiC[45],Redis cluster[46]等以及分布式内存图数据库 ArangoDB[47],Graph Engine[48],Sqrrl Enterprise[49],Titan[50]等.

大规模数据处理应用常常要处理 PB 级的数据[51],将这些数据全部存储于内存中一方面会占用大量内存资源,增加成本和功耗;另一方面,过大的内存空间也使得数据检索和访问效率降低.鉴于此,在提供分布式大内存的同时,也出现了另外一项重要的技术——内存压缩[52−54].内存压缩技术根据压缩率的不同分为轻量级压缩[55−57]和重量级压缩[58],根据基于软硬件不同分为软件压缩[42,59]、硬件压缩[60,61]和软硬协作压缩[62,63],但其最终目的有 3 个:(1) 提高访问效率,有利于查询性能的提高;(2) 减少内存数据量,从而降低内存消耗;(3) 提高内存使用效率,减少 CPU 等待时间,增加片间互连网络带宽利用率.

2.2.2 内存缓存系统

利用内存作为缓存这种计算方式[64−67]已经提出 20 多年,但直到近些年,随着硬件技术的发展,内存缓存系统才出现大量新的研究,比如 Memcached[68],BigTable[69],PACMan[70]和 GridGrain[71]等等.内存缓存产生的技术

背景是:磁盘容量往往比内存大两个数量级,这就说明,想通过内存缓存加速数据批处理,将内存作为数据永久存放处不太现实,在很多情况下,只能将内存当作缓存.因此,这类内存计算把数据分为访问频率高和低两类,将访问频率高的部分数据长久存放于内存当中,或者将一些重要数据长期缓存于内存当中.比如,谷歌和雅虎将其检索索引全部存于内存当中,Memcached 都是将其所有通用"键-值"对存于内存中,BigTable 存储系统把它所有列族缓存于内存.这样可以高效利用内存计算.

然而,内存局部性在很大程度上影响着作业的完成时间.只有当作业的所有任务都从内存中读取数据时,作业的完成才会被加速.相反,即使只有小部分任务不是从内存读取,那么这小部分任务(outliers)会拖慢整个作业的完成时间[72].研究表明,大数据应用具有不同于传统应用的数据局部性特征.一个突出表现是,大量的数据仅被访问一次.文献[73]根据 Facebook 的 hadoop 日志指出:保守地看,64%的作业具有任务局部性,有 75%的数据块只被访问一次,这将大大降低内存缓存的效率.

因此,对于内存缓存,仍然有很大的提升空间和面临的挑战.体现在如下两个方面.

(1) 内存替换策略

很多替换策略只是为了单纯的提高命中率,但是提高命中率并不能一定加快任务完成时间.所以对于内存缓存来说,缩短任务完成时间才是最主要的目标[73].在众多内存替换策略中,比如 LFU,LRU 等,虽然达到了一定的效果,但仍然有提升空间.其原因在于:替换策略主要目标在于使整个作业的所有任务缓存于内存当中,以便于并行处理,消除 outliers,提升整个作业执行效率.如,文献[70]针对这种状况提出两种替换策略,其原理是:或将作业的所有并行任务缓存于内存中,或者所有任务都不缓存于内存中,即,所谓的 all-or-nothing 原则.

(2) 预取

对于一些只被任务访问一次的数据块,不具有内存局部性.如前所述,某些应用中这种数据占到 75%左右.对于这种数据,只能通过预取的方式提高内存命中率,加快处理效率.例如,一种选择就是将最新产生的数据预取进内存.另外,如果作业有多个任务,我们可以根据第一个任务作为线索装载进其他任务的数据块.文献[74]提出一种内存管理框架,通过前几次从磁盘上读取数据的行为预测以后要读取的数据,进而来设计出一个预取策略.

将数据缓存到内存中可以大幅提高数据处理效率,但在可靠性及其带来的问题方面存在不足.如果集群中一个节点出现故障,那么将会产生集群内节点间的大量数据移动和复制,这将带来较大的存储和通信开销.所以,如何避免或减少容错带来的节点间数据移动和复制开销,是这类内存计算面临的一个挑战.

2.2.3 内存数据处理系统

与前两类系统不同,此类系统从支持大数据应用角度出发,主要面向迭代式数据处理、实时数据查询等应用,通过提供编程模型/接口以及运行环境,支持这些应用在内存中进行大规模数据的分析处理和检索查询.其处理机制是:首先将待处理数据从磁盘读入内存,此后,在这些数据上进行反复的迭代运算,即,除了第一次需要涉及 I/O 操作,此后便一直从内存读写数据.此类内存计算不涉及预取数据,而且在内存管理方面使用内存替换策略也非常高效,因此在很大程度上提高了处理效率.

近些年出现了众多此类内存计算的框架/系统,最具影响力的是加州大学伯克利分校开发的 Spark,它适用于迭代式及交互式的数据批处理应用,其原理即:将数据第一次从磁盘读入内存,生成一种抽象的内存对象,即,弹性分布式数据集(resilient distributed datasets,简称 RDD)[75],此后,用户程序只操作在内存当中的 RDD,计算过程只涉及内存读写,因此大幅提升了数据处理效率.Piccolo[76]同样面向需要迭代处理的应用,比如机器学习、图算法、科学计算等问题,它主要将分布式的共享中间状态以 key-value 表格存于内存当中,以便高效解决分布式多线程之间共享变量的问题.Pregel[77]和 HaLoop[78]都是基于分布式内存计算的图数据处理框架,前者将中间结果存于内存用于迭代计算,后者提供一种迭代式的 MapReduce 接口.Twister 同样提供了一种迭代式的 MapReduce 模型,用户可创建 MapReduce 作业让其迭代计算,其中,数据在迭代时被保存在内存当中.另一种典型的内存批处理应用为 M3R[79],它是 MapReduce 的内存实现框架,适用于反复分析大量数据的应用,并且提供了向下兼容 MapReduce 的接口,相对 MapReduce 大幅提升处理效率.另外,研究发现[80−83]:在数据密集型环境下,存在大量程序以相同或者略微不同的输入反复运行多次.文献[84]在基于 M3R 系统之上提出了 MapReuse 系统,

使得计算重使用发生在内存数据结构中,而不是文件系统,这样极大地加快了 In-Memory MapReduce 的处理速度.另外,内存实时数据处理近年来也发展迅猛,包括 Spark streaming[85],Storm[86],Yahoo!S4[87],MapReduce Online[88]等等.

此类内存计算研究包括以下几个关键点.

(1) 容错机制

内存数据的易失性,使得内存计算环境下数据恢复和容错至关重要.MapReduce 的容错机制主要通过定期检查节点,对于出现故障的 work 节点,其上的任务要重新执行;对于出现故障的 master 节点,通过从集群其他 master 上复制数据并传输到本地的方法来解决.因此,对于 MapReduce 来说,容错涉及到从磁盘到内存,从集群中的其他节点到本地的数据移动.这对于对实时性要求很高的内存计算来说是非常耗时的,因为网络数据移动的延迟远远大于本地的数据移动,吞吐量也远远小于本地数据移动.目前的一些集群内存存储系统,比如分布式共享内存[89]、键/值存储[41]、内存数据库等,都提供了基于细粒度更新可变状态的接口,这些接口的容错方式就是通过集群中节点间的数据移动和复制,或日志更新集群间各个节点.而这两种方法对于数据密集型负载来说开销太大,原因在于:这些方法需要在集群网点节点间复制大量数据,而网络带宽又远远小于内存带宽,这将导致大量存储开销.

Spark 采用了基于粗粒度的转化的接口,这种转化操作在计算过程中形成一种有向无环图(DAG),称为“血统(lineage)”,“血统”实质上是建立一种数据间转换的关系,而不是数据本身,因此当出现系统故障时,便可通过这个“血统”提供的信息,计算丢失的数据,即以计算换取数据移动和复制,这样即可大量节省容错所带来的开销.同样,在 Nectar[82]系统中,同样采用了“血统”容错,只是在特定编程框架下才采用“血统”容错,在传统的情形下,还是通过复制数据来解决容错问题.Tachyon[90]也采用了“血统”,使得在运用内存缓存容错时,数据恢复得到 Memory 访问速度级别的加速.另外,虽然“血统”通过计算解决了不用复制和移动数据的问题,但是当“血统”的有向无环图很大时,则需要较长的时间来计算恢复数据.在这种情况下,Spark 和 Tachyon 都采用了检查点机制来解决“血统”链很长所带的计算开销问题.

目前,容错处理主要有数据移动和复制、日志机制、分布式锁、快照、检查点机制等等.但是正如文献[80]所述,由于带宽限制,数据复制等数据恢复机制带来巨大开销,而基于“血统”容错的方法能很好地和内存计算在处理速度上相协调,因此,这可能将成为内存计算未来主要的容错方法.

(2) 同步

分布式内存处理系统主要处理机器学习、图算法、科学计算等问题,此类问题在并行计算的同时,往往涉及到结构或逻辑上的依赖关系,而并行处理的各个步骤在到达稳定点的时间不同,因此需要在并行进行的计算步之间进行同步控制,以保证结果的正确性.

常见的同步方式有同步计算、异步计算和混合方式.目前,大多数内存计算系统采用 BSP[91]同步,部分系统采用异步机制.Spark 和 Pregel 都采用 BSP 同步机制,而基于内存计算的分布式内存共享图处理系统 PowerGraph[92]则采用 BSP 同步和异步两种方式.Trinity[93]同样采用 BSP 同步和异步两种方式.

同步方式可确保计算的确定性,易于设计、编程和测试.但是由于每个计算步的处理时间不同,最慢的计算将严重影响整体收敛速度.异步处理模式的优势在于可加速计算的收敛速度,但其编程复杂,不便于调试和测试,不能保证更新一致性,且结果不确定.两者各有优缺点,如何根据不同应用系统类型,取两者优点设计出高效且易于编程的同步机制越来越重要.

(3) 内存分配与管理

内存计算的核心资源为内存,因而内存分配与管理显得尤为重要.如何在内存中安排数据存储方式,使数据访问更为高效且内存资源充分利用,这是首要解决的问题.在实际的应用实例中,应用类型不同,内存数据安排也不尽相同,如 Spark 系统采用“键-值”存储方式,Piccolo 采用内存表格方式及“键-值”两种方式,Pregel 采用了 BigTable 的内存组织方式.

另外,内存容量总是有限的,如何高效利用有限的内存资源处理海量文件又是一个挑战.Spark 将内存数据

抽象成 RDD,然后在内存不足时,利用“最近最少使用”(LRU)内存替换策略协调内存资源.同时,为了更灵活地分配内存资源,Spark 可以通过所谓的 RDD“持续存储优先权”给用户一定的管理权限.总之,在此类内存计算当中,同样存在内存替换策略的选择.因此,如何根据具体的编程框架的优缺点选择高效的内存替换策略,有待进一步研究.

(4) 网络瓶颈

谷歌的一份报告[94]显示:从内存读取数据比从本地磁盘或集群网络中其他主机读取数据快两个数量级,而磁盘和集群网络读取速度又处于同一数量级.因此,在本地处理数据时,数据存在于内存当中,CPU 很少停下来等待磁盘 I/O,这样,内存计算解决了磁盘 I/O 瓶颈;但是,如果涉及到节点间通信或数据传输,则要比直接在内存读写慢两个数量级,内存计算必然面临一定的效率损失.因此,在分布式集群内存计算当中,网络瓶颈成为主要瓶颈.如何减少节点间的通信和数据传输开销,提高内存计算效率,这又是一个挑战.

2.3 新型混合内存结构的内存计算

近几年,新兴的非易失性随机存储介质(non-volatile memory,简称 NVM)快速发展,如铁电存储器(ferroelectric random accessmemory,简称 FeRAM)[95]、相变存储器(phase change memory,简称 PCM)[96−98]、电阻存储器(resistive randomaccess memory,简称 RRAM)[99]等,其性能接近 DRAM,但容量远远大于 DRAM,同时,能耗和价格远远低于 DRAM.这为新型的内存体系结构提供了良好的硬件保障.因此,基于新型存储器件和传统 DRAM 的新型混合内存体系在大幅提升内存容量,降低成本的同时,其访问速度与 DRAM 相当.

在众多的非易失性随机存储介质中,PCM 凭借其非易失性、非破坏性读、读完无须回写、写操作无须先擦除、存储密度高等特性,逐渐成为大规模内存系统中颇具潜力的 DRAM 替代品[98].在硬件体系结构方面,研究者围绕 PCM 和 DRAM 的混合方案开展了很多研究,国内外学术界出现的混合内存结构大概分为 3 类.

2.3.1 线性统一编址混合内存

这种混合内存结构由 PCM 和 DRAM 构成[100,101].通过软件和硬件技术,避免了 PCM 较短的写寿命和较高的写能耗的缺点,充分发挥了 PCM 在读数据和存储数据方面低功耗、非易失性和 DRAM 在写数据时低功耗及超长的写寿命的特性.实验表明:相比 DRAM,PDRAM 的能耗节省达到 37%.在 PDRAM 中,PCM 和 DRAM 处于同等地位,无主次之分,对两者进行统一的线性编址,如图 1 所示.

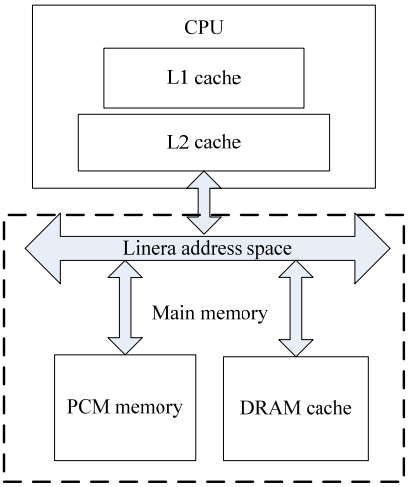


Fig.1 Hybrid memory structure sharing a single physical address space

图 1 线性统一编址混合内存结构

2.3.2 以 DRAM 作为 PCM 缓存的混合内存

这种混合型内存体系结构是由 DRAM 和 PCM 共同组成,其中,PCM 作为主存,DRAM 作为 PCM 的缓存,

其主要目的是利用 PCM 高容量以及 DRAM 快速访问的特点,同时,避免了 PCM 访问速度慢及 DRAM 容量小能耗高所带来的问题[102−104].另外,文献[104]中通过 3 种技术来减少 PCM 上的写操作,以达到延长 PCM 寿命的目的.实验结果表明,这种内存系统较传统的内存容量提高了 4 倍,页错误平均降低 5 倍,PCM 写操作次数降低了 3 倍,可延长 PCM 寿命 3~9.7 年;且当 DRAM 容量为 PCM 的 3%时,PCM 较 DRAM 的访问延迟得到了很多的弥补.图 2 描述了这种内存系统与传统内存系统的区别.

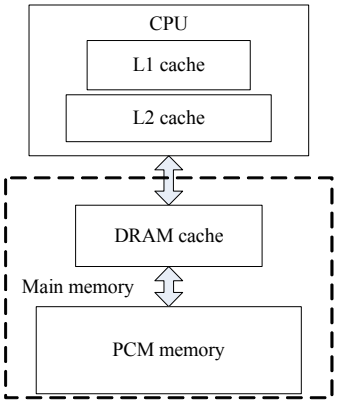


Fig.2 Hybrid memory structure using DRAM as cache

图 2 DRAM 缓存混合内存结构

2.3.3 分层混合内存(hierarchical hybrid memory)

文献[105]中提出一种名叫 MN-MATE 的混合内存.这种内存是由 PCM 和 DRAM 构成,具有层次结构.这种层次内存分为片上和片下两部分:片上内存由单独的 DRAM 构成,因其内置于处理器因此具有较小的延迟;片下部分则由 PCM 和 DRAM 混合构成,两者共用同一个内存控制器且采用统一的线性编址.如图 3 所示,M1 为上片,M2 为下片,分别由 PCM 和 DRAM 通过线性编址组成.同样,这种内存结构通过 3 种技术发挥了 PCM 的大容量低能耗以及 DRAM 访问速度快的优点,提高内存性能和减少能耗.

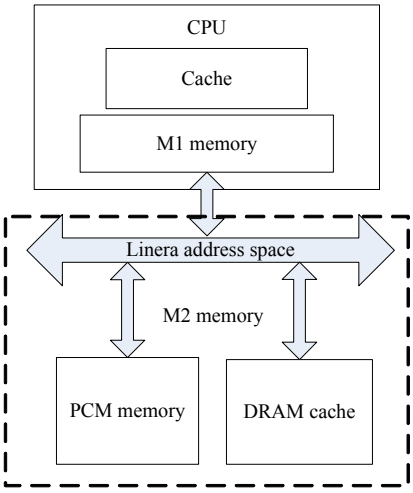


Fig.3 Hierarchical hybrid memory structure

图 3 分层混合内存结构

另外,虽然诸如 PCM 等新型存储介质和 DRAM 构成的混合内存在硬件技术方向的研究已经进展不小,但

是其相应的软件平台研究仍相对滞后.现阶段想要利用新型混合内存处理实时大数据,仍然有很多工作做要.

虽然各种新型 NVM 在容量、能耗方面比当前的 DRAM 更具有优势,但是在访问速度、写寿命等方面仍然不及 DRAM[96].比如:对于 PCM 存在的缺点,文献[106]通过使用消除冗余位写入、行替换和段交换等方法降低损耗,可延长 PCM 的使用寿命 13 到 22 年;为了解决混合内存中 PCM 写数据速度慢且写寿命较短的问题,文献[107,108]分别提出不同的内存管理技术,通过不同的预测写操作算法,让大多数写操作发生在 DRAM 而非 PCM,从而延长了 PCM 的写寿命,并且可以隐藏 PCM 写数据慢的缺点;文献[109]不仅采用减少 PCM 写次数的方法延长 PCM 寿命,还采用了提高 DRAM 缓存命中率的方法,从两个方面来优化混合内存结构的能耗和性能.此外,针对 PCM 等新型随机存储介质的缺点,还有众多研究从软硬件层面提出了很多的改进措施,也取得了不错的效果[98,110,111].

在硬件体系结构方面,这种内存计算的主要革新表现在:在原来的 DRAM 基础上,通过加入新型 SCM 扩展成了不同类型的混合结构大内存,访问速度接近于 DRAM,而容量远远大于 DRAM.在这种内存计算模式中,SCM 不仅可作为内存,而且相当于传统磁盘,计算可以直接发生在 SCM 上,避免了传统的内存—磁盘数据访问,CPU 直接从内存读取数据,进行计算分析.同时,这种新型内存架构也带来了一系列问题:

(1) 各种新型内存与 DRAM 存在访存速度、能耗、读写寿命等方面差异,如何协调这些差异,使整个内存处理效率较高,能耗较少?

(2) 随着众核处理器性能越来越高,异构内存容量增大,如何处理大内存与众核处理器间不断加大的带宽鸿沟?

(3) 内存较原来更大,如何应对大内存带来的高能耗问题?

相对于硬件体系结构,在软件方面,这种内存计算的相关研究滞后很多.因为作为一种新兴的混合内存结构,它涉及到体系结构、操作系统、编程模型方面的诸多问题.

(1) 体系结构

由于新型内存替代了磁盘可以长期存储数据,那么传统的磁盘数据访问局部性将被内存访问局部性取代而成为性能优化的主要目标;并且,传统的磁盘数据预取、缓存以及替换策略无法直接迁移到非一致缓存访问以及非一致内存访问环境中.

(2) 操作系统

由于新型混合结构的出现,操作系统需要统一管理多种异构资源,实现高效的、透明的、可靠的内存访问与管理策略.另外,由于新型内存容量增大,可用地址空间随之增大,致使操作系统内存管理开销增大.所以,为了提高操作系统对内存管理效率,需要研究新的编址策略和访问方式.

(3) 编程模型

新型混合内存解决了磁盘 I/O 瓶颈问题,因此在内存计算当中,传统的编程模型对数据传输不再占大量处理时间,而数据处理将占据大量处理时间.此外,由于新型非易失性内存的 I/O 延迟远小于磁盘,传统模型中磁盘与内存的数据一致性等问题不复存在,编程模型也因此有相应改变.

3 典型内存计算应用

当前,内存计算的应用类型众多,本文以 4 类典型的内存计算应用为例,深入剖析当今内存计算的发展和其存在的诸多挑战.

3.1 内存数据库

内存计算技术的再次兴起,始于其在内存数据库方面的广泛应用.早在 1994 年,内存墙[112]问题提出之后,为了减少内存墙的影响,针对缓存层次结构[113],出现了大量以缓存为中心的内存数据库系统研究,如 MonetDB[114],EaseDB[115],FastDB[116],TimesTen[117]等.而随着内存的容量大幅增长,特别是分布式集群环境下内存容量已经能够满足对海量数据处理的要求时,2006 年之后,出现了面向事务处理和数据分析的两大类内存数据管理系统,这类系统包括 H-store,VoltDB,HANA,DryadINQ[118],Pig[119],FlumeJava[120],Ciel[121],Twister[122]等等.

表 1 列出近几年较为典型的一些内存数据库系统.

Table 1 Comparison of some in-memory databases

表 1 部分内存数据库系统对比

系统	系统结构	索引	内存数据结构	并发控制	容错	内存不足	类型
SAP HANA	分布式	时间线索引/CSB+-树/倒排索引	关系/图/文本	多版本并发控制协议	日志/检测点/备份服务器	Partition-level swapping/内存压缩	关系数据库
H-Store	分布式	哈希/B+-树/二叉树	关系	Partition/顺序执行	Command logging/检测点/数据复制	Anti-caching	关系数据库
Hyper	单节点	哈希/平衡树	关系	虚拟快照/严格时间戳	日志/检测点/数据复制	内存压缩	关系数据库
MemepiC	分布式	哈希/跳表	键-值	原语	日志/数据复制	用户空间虚拟内存管理	NoSQL数据库
Trinity	分布式	N/A	图	细粒度自旋锁	数据复制	N/A	图数据库
Bitsy	单节点嵌入式	N/A	图	乐观并发控制	日志/备份	N/A	图数据库
MongoDB	单节点	B-树	文档	数据库级锁	内存映射文件	N/A	NoSQL数据库

不难看出,内存数据库研究热点以及面临的挑战主要存在以下几方面:

(1) 索引.内存数据库中索引的建立与传统数据库不同,其主要目的是为了减少内存密集的扫描.怎样建立一个合理的索引,以达到优化内存数据库时间和空间效率的目的?

(2) 内存数据组织.在内存数据库里,内存中的数据组织对内存和缓存的使用有很大的影响.比如,基于列的关系表可以提高 cache 命中率,同样可以获得良好的内存压缩率,节省可观内存空间;但是这种数据组织对于 OLTP 查询来说不是一个好的选择,因为基于行的安排更能提高 OLTP 查询的效率.所以,如何根据情况设计合理的内存数据组织,这对进一步优化内存及缓存的使用显得非常重要.

(3) 并发控制.随着众核系统的不断发展,在内存数据库中,并发控制显得更加重要.基于锁的重量级机制控制效果好,但是其开销大,影响系统总体性能.如何在内存系统下,根据内存计算自身特点设计合理的并发控制策略,既可以达到并发控制的效果,又不太影响系统性能,这同样是一个挑战.

(4) 容错.与传统数据库相比,由于内存的易失性,导致内存数据库对于 ACID 中的持久性不能很好支持.因此,当突发性故障,比如断电、软硬故障发生时,如何设计出合理的策略使内存数据不丢失?

(5) 内存不足.尽管如今内存容量越大越大,但是其增长速率无法赶上数据量的增长,这势必导致总内存不足的情况出现.在硬件方面,如何利用新型大内存设计出有效的内存系统以应对快速增长的数据,以及在软件方面,如何设计出有效率的策略来解决内存不足的现象,这些都是科研工作者将要面临的挑战.

3.2 内存图计算

过去 10 年内,基于图的应用快速增多.在网络中寻找最短路径、计算网页的 PageRank 以及更新社交网络等等,都需要进行大量的计算存储资源.为此,图计算变得越来越重要,而由于图数据规模的快速增长,内存计算快速成为当今图计算的热点问题.

由于图的计算中涉及到边与边的依赖关系,以及图计算缺少数据并行性,所以大规模图处理使得 MapReduce 显得力不从心.为此,近几年出现了大量内存图计算的模型,比如 Pregel、GraphLab、Piccolo、基于 Spark 的 GraphX[123]等,其中,Pregel 和 GraphLab 是以顶点为中心的计算模型.表 2 为近些年的部分图模式对比.

内存图计算为内存计算的一个应用,同样,它涉及到内存分配管理、同步和容错机制.但是图计算有其自身的关键研究点,主要包括以下两方面:

(1) 图划分.图划分是图处理的首要步骤.一个好的划分,可以使负担均衡,减小节点间通信开销以及提高计算效率.换言之,如何合理划分大规模图数据,对图处理效率至关重要.

(2) 计算模式.图的计算模式分为以顶点为中心、以边为中心及以图为中心这 3 种计算模式.不同计算模式对不同应用的图的处理效率有极大的影响,比如,以顶点为中心的计算模式适用于 PageRank、图挖掘等应用,以图为中心的计算模式适用于聚集计算.3 种计算模式各有优缺点:以顶为中心的模式编程简单,但是收敛慢;以边为中心的模式避免了边数据随机访问,但涉及到随机访问顶点,计算模型复杂;以图为中心的连通计算性能较好,但划分图负担过重.所以,没有完美的计算模式,我们需要根据不同应用设计合理的计算模式.另外,对于通用性的应用,计算模式的设计就更为困难.

Table 2 Comparison of some in-memory graph processing frameworks

表 2 部分内存图处理框架对比

图计算系统	同步	划分方式	内存组织	容错	体系结构
GraphX	BSP 同步	哈希	键-值	血统/检测点	分布式
PowerGraph	BSP 同步/异步	边切割/启发式顶点切割	N/A	快照	分布式
Pregel	BSP 同步	哈希	BigTable	检测点	分布式
GraphLab	异步/同步	哈希	N/A	检测点	单节点众核/分布式
Graphchi	异步	Interval	滑动窗口	N/A	单节点众核
Grace	BSP 同步	哈希/启发式划分	数组	N/A	单节点众核
Piccolo	同步	用户控制表划分	内存表/键-值	检测点	分布式

3.3 其他内存计算应用

由于众多机器学习算法涉及到大规模数据的反复迭代运算,比如分类算法需要通过大量数据多次迭代计算,进而产生训练模型中的参数,因而,将需反复运算的数据存放内存中将大幅加快此类计算速度.因此,内存计算非常适合此类应用.近几年出现多种基于内存计算的机器学习框架,如建立在 Spark 之上的 MLlib[124]是最为知名的基于内存计算的机器学习算法库,它支持分类、聚类以及矩阵分解(如主成分分析(PCA))等算法.另外,还有基于异构分布式系统的大规模机器学习系统 TensorFlow[125]、训练深度学习网络的框架 SparkNet[126]等.

内存计算的另一个典型应用为流数据处理,即实时计算.这类应用常见于大型网站的访问数据处理、搜索引擎的响应处理等,一般涉及海量数据处理,响应时间为秒级.而内存计算将海量数据存于内存,为实时数据处理提供了保障.当今主流的基于内存的流处理系统有 Spark Streaming[85],Yahoo!S4[87],Storm[86]等.

4 内存计算面临挑战及展望

尽管内存计算为当前工业界和学术界高度关注的热点问题,然而内存计算涉及硬件体系结构、软件体系结构、操作系统、编程模型、大数据处理等诸多方面内容,使得内存计算从底层的硬件架构到高层的编程模型都存在许多问题,面临一系列挑战.因此,本节对这些问题及挑战从总体层面到应用层面予以梳理.

4.1 总体层面的问题与挑战

内存计算的兴起,主要源于硬件的快速发展以及大数据的处理需求.大数据处理复杂,主要原因有 3 方面:数据量大、增长速度快、数据种类多[127,128].因此,无论单节点内存计算还是分布式内存计算,都将面临诸多问题.

单节点内存计算面临的主要问题是系统性能容易受限于硬件资源的不足,比如内存不足时,如何高效地访问 I/O,提高单节点内存计算的性能?而分布式内存计算面临的主要问题是通信问题,网络通信在分布式内存计算中起着十分重要的作用:容错时的数据复制,节点间相互协调时的数据信息交换,数据共享或负载平衡所需的数据传输等等.另外,相比与如今越来越大的数据量,单一节点内存大小有限,进一步增加了网络通信的需求.然而,内存与网络间的数据访问延迟差距巨大,使得网络通信效率对整个内存计算中的性能有极大的影响.因此,如何处理随着通信量大幅提升及通信多样化所带来的内存与网络间的访问延迟鸿沟,是当前分布式内存计算面临的主要问题.

不管是单节点还是分布式内存计算,都将面临着大规模并行带来的并发性控制问题.一个高效的并发控制协议既要保证原子性和隔离性,又不能影响并行执行所带来效率提高.随着集群规模的扩大和单节点 CPU 核数的增加,大量进程/线程平行执行,从而显着地增加了并发控制的复杂性,对高效的并发控制带来了巨大挑战.研

究[129]显示,目前并发控制算法还无法扩展到 1 024 个核以上.因此,如何提高并发控制的可扩展性越来越重要.

集群调度是过去几年来最为广泛研究热点问题之一[130−135],但如今,随着数据量激增,集群规模扩大,任务调度将成为分布式处理性能提高的瓶颈[136].原因在于:当前流行的数据处理系统,比如 Dryad[118,137],Spark,GraphX 等采用中心控制器(master)调度任务,当集群中工作节点(workers)数和任务数过大时,这种调度方式必然成为整体性能提升的瓶颈.由此,针对数据规模的激增以及集群规模的扩大,研究者们开始作类似图计算框架,如 GraphLab[29],PowerGraph[92]和 Galois[138]中的调度策略——控制器负责分配数据,工作节点负责在本地生成任务[139].因此,随着大数据时代来临,集群调度又将成为分布式内存计算的热点问题.

随着新型非易失性存储器的快速发展, 如何用其替换或补充原有的DRAM作为主存,成为工业界和学术界的研究热点[98,104,106,140−147].而内存计算同样为当今工业界和学术界关于大数据处理的热点,比如数据库系统方面,IBM 的 Blink 项目[148]中,Microsoft SQL Server[33]和 SAP HANA[15]等都采用内存存储及处理数据.此外,在更广泛的数据密集型计算领域,有很多学术界的研究和工业界的产品都是基于内存计算,如 Memcached,Pregel, Ramcloud 和 Spark 等等.因此,大数据时代的一个必然趋势是主存容量的快速增长及其在内存计算中的作用愈发的重要.那么,随之产生众多新的研究问题.总体上讲,如何应对新型大内存在体系结构、操作系统、编程模型方面的诸多挑战:在细节的方面,如何将现有的内存数据库、内存数据处理框架迁移到新型非易失性内存上,并且充分利用新型非易失性内存的大容量和非易失性;如何应对大容量内存与处理器之间的带宽鸿沟.这些都是内存计算将要面临的挑战.

4.2 应用层面的问题与挑战

内存计算从硬件架构来看,不管是单节点、分布式,还是基于新型混合内存系统,最终表现为软件层次的应用,比如内存存储系统表现为内存数据库,内存数据处理系统表现为图计算系框架、机器学习算法处理框架、流处理框架等.

内存数据库将整个数据库装载进内存,避免了传统数据库 I/O 操作所带来的开销.这样,内存数据库性能提升的瓶颈从传统的 I/O 访问便转为如何提高计算时间及减少访存延迟,具体表现在前文所述几方面的挑战,即索引建立、内存数据组织以及并发控制策略等,以达到优化内存数据库时间空间效率,优化内存及缓存的使用,保证系统性能的目的.另外,硬件的快速发展正在迅速改变商用处理器的场景,如 NUMA 架构[149]、SIMD 指令[150]、RDMA 网络,硬件事务内存(HTM)[151]、非易失性存储器(NVM)以及片上 GPU[152],FPGA 和其他硬件加速器,可以以较低的开销提供更高的性能[151,153−155].综上,内存数据库性能的提升不仅面临软件方面的挑战与机遇,同样,硬件加速为内存数据库提出了新的挑战以及提供了新的发展前景.

图计算是当今内存计算的一个热点问题,它在不同的硬件架构平台上存在内存计算所面临的不同挑战,比如:基于分布式的内存图计算面临数据通信时的 I/O 瓶颈问题;单节点内存图计算则要面临内存不足时系统性能降低的问题以及线程数增加所引起的内存墙问题.然而,相比其他形式的数据,图数据本身有其自身特点及复杂性,因而存在特有的问题需要解决:

- 首先,现实世界中的图数据(比如社交网络)是不断演化的,并且存在一些图算法需要在运行过程中不断改变图结构,而目前关于动态图处理的相关研究较少.
- 其次,部分图算法,如 PageRank、强连通分量、最小生成树等,在计算过程中小部分节点收敛速度慢[156],在同步时容易造成过多的超步(superstep),导致大分部收敛完毕的节点又需重新计算,降低处理效率,即所谓的不对称收敛问题.目前,此类问题存在算法层面的优化[157],然而面对大规模图数据亟待处理,系统级的优化有待进一步研究.
- 最后,图划分是图计算的基础,然而现实中的图数据基本呈偏斜分布,即,顶点的度分布不均匀.现在通用的图划分方法容易导致显著的负载不均衡,造成节点间通信开销增加.因此,如何针对偏斜图(skewed graph)设计合理的图划分方法,对内存图处理效率有着非常重要的意义.
- 另外,同内存数据库相似,如何用 GPU,MIC 等加速部件加速处理,也是当前内存图计算的一个热点问题[158−160].

深度学习是当前机器学习和模式识别领域的热点问题.深度学习需要训练的数据量庞大(通常有几百 GB),导致训练过程非常耗时.因此,为了加速训练过程,如何充分利用大规模集群的硬件资源,使数据自始至终保存在内存中,成为内存计算的研究热点[125,126,161−167].其中,文献[161]采用 InfiniBand 架构,在 GPU 集群上建立了一个训练深度网络的并行模型;文献[163]在几百台节点的集群上建立了一个使用异步随机下降梯度训练深度网络的系统;FireCaffe 系统[167]关注系统的可扩展性;SparkNet[126]重在使包括预处理在内的整个训练过程自动化完成.然而,针对深度学习算法方面的研究仍处于发展阶段,基于内存计算的深度学习框架更是面临更多的问题与挑战.不难看出,现有深度学习框架缺乏通用性.随着深度学习研究进一步发展,越来越多领域的研究人员需要运用深度学习算法高效地处理大规模数据.因此,如何为相关人员提供通用的编程框架,开发高效的深度学习应用处理框架,是当前面临的挑战之一.

5 总 结

本文在分析内存计算技术特点的基础上,根据硬件组织的不同对内存计算技术和系统进行了分类,分别从体系结构、编程模型方面介绍了各种内存计算技术的原理、关键技术以及存在的问题.首先,介绍了基于单节点的内存计算技术;其次,重点介绍了基于分布式系统的内存计算,它是近年来发展最为迅速的内存计算技术,解决了传统计算模式在处理大数据时的 I/O 性能瓶颈问题;随后,对尚处于发展阶段的基于新型混合内存结构的内存计算进行了初步介绍;再次,对若干典型的内存计算应用进行了分析和介绍;最后,对内存计算面临的挑战及发展前景作了分析.

内存计算的兴起,源于大数据处理的要求以及内存容量不断加大且价格不断下降.内存成为新型的磁盘,而磁盘成为新型的磁带.换言之,内存计算使得以计算为中心的计算模式转变为以数据为中心的计算模式,迎合了大数据时代数据处理的要求.这个转变使我们需要重新思考内存计算所产生的一系列从操作系统到编程模型出现的问题和面临的挑战,如容错、内存管理等.另外,随着新型混合内存结构的内存计算的发展,原始的单节点及分布式内存计算将面临体系结构、操作系统、编程模型方面的诸多挑战.这种架构的重大变化,也必然为内存计算带来更多的发展机遇.这样,内存计算也将会进入新一轮的研究和发展高峰,并成为大数据实时处理领域的热点.

References:

[1] Lefurgy C, Rajamani K, Rawson F, Felter W, Kistler M, Keller TW. Energy management for commercial servers. IEEE Computer, 2003,36(12):39−48. [doi: 10.1109/MC.2003.1250880]

[2] Udipi AN, Muralimanohar N, Chatterjee N, Balasubramonian R, Davis A, Jouppi NP. Rethinking DRAM design and organization for energy-constrained multi-cores. ACM SIGARCH Computer Architecture News, 2010,38(3):175−186. [doi: 10.1145/1816038.1815983]

[3] Jeffrey D, Sanjay G. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008,51(1):107−113. [doi: 10.1145/1327452.1327492]

[4] Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: Proc. of 2010 IEEE the 26th Symp. on Mass Storage Systems and Technologies. 2010. 1−10. [doi: 10.1109/MSST.2010.5496972]

[5] Pierre J. Big data: In-memory MapReduce. 2011. http://blogs.oracle.com/datawarehousing/entry/big_data_in_memory_mapreduce

[6] Chen R, Chen H, Zang B. Tiled-MapReduce: Optimizing resource usages of data-parallel applications on multicore with tiling. In: Proc. of the 19th Int'l Conf. on Parallel Architectures and Compilation Techniques. ACM Press, 2010. 523−534. [doi: 10.1145/1854273.1854337]

[7] Jiang W, Ravi VT, Agrawal G. A map-reduce system with an alternate API for multi-core environments. In: Proc. of 2010 the 10th IEEE/ACM Int'l Conf. on Cluster, Cloud and Grid Computing. IEEE Computer Society, 2010. 84−93. [doi: 10.1109/CCGRID.2010.10]

[8] Ranger C, Raghuraman R, Penmetsa A, Bradski G, Kozyrakis C. Evaluating MapReduce for multi-core and multiprocessor systems. In: Proc. of IEEE the 20th Int'l Symp. on High Performance Computer Architecture (HPCA). IEEE, 2007. 13−24. [doi: 10.1109/HPCA.2007.346181]

[9] Yoo RM, Romano A, Kozyrakis C. Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system. In: Proc. of the IEEE Int'l Symp. on Workload Characterization (IISWC 2009). IEEE, 2009. 198−207. [doi: 10.1109/IISWC.2009.5306783]

[10] Talbot J, Yoo RM, Kozyrakis C. Phoenix++: Modular MapReduce for shared-memory systems. In: Proc. of the 2nd Int'l Workshop on MapReduce and Its Applications. ACM Press, 2011. 9−16. [doi: 10.1145/1996092.1996095]

[11] Boyd-Wickizer S, Chen H, Chen R, Mao Y, Kaashoek F, Morris R, Pesterev A, Stein L, Wu M, Dai Y, Zhang Y, Zhang Z. Corey: An operating system for many cores. In: Proc. of the Usenix Symp. on Operating Systems Design and Implementation. San Diego: ACM Press, 2008. 43−57.

[12] Liu T, Curtsinger C, Berger ED. DTHREADS: Efficient deterministic multithreading. In: Proc. of the 23rd ACM Symp. on Operating Systems Principles. ACM Press, 2011. 327−336. [doi: 10.1145/2043556.2043587]

[13] Liu T, Berger ED. SHERIFF: Precise detection and automatic mitigation of false sharing. ACM Sigplan Notices, 2011,46(10):3−18. [doi: 10.1145/2076021.2048070]

[14] Mao Y, Morris R, Kaashoek MF. Optimizing MapReduce for multicore architectures. Technical Report, MIT CSAIL, 2010.

[15] Farber F, Cha SK, Primsch J, Bornhövd C, Sigg S, Lehner W. SAP HANA database: Data management for modern business applications. ACM Sigmod Record, 2012,40(4):45−51. [doi: 10.1145/2094114.2094126]

[16] Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: Cluster computing with working sets. In: Proc. of the 2nd USENIX Conf. on Hot Topics in Cloud Computing (HotCloud 2010). Berkeley: USENIX Association, 2010. 10−10.

[17] Lindstrom J, Raatikka V, Ruuth J, Soini P, Vakkila K. IBM solidDB: In-memory database optimized for ExtremeSpeed and availability. IEEE Data Engineering Bulletin, 2013,36(2):14−20.

[18] Franklin MJ, Carey MJ, Livny M. Global memory management in client-server database architectures. In: Proc. of the Int'l Conf. on Very Large Data Bases. Vancouver: ACM Press, 1992. 596−609.

[19] Garcia-Molina H, Salem K. Main memory database systems: An overview. IEEE Trans. on Knowledge & Data Engineering, 1992, 4(6):509−516. [doi: 10.1109/69.180602]

[20] GARTNER Team. GARTNER says in-memory computing is racing towards mainstream adoption. Egham, 2013. http://www.gartner.com/newsroom/id/2405315

[21] IBM Team. In-Memory computing. 2015. http://www-01.ibm.com/software/data/what-is-in-memory-computing.html

[22] Ivanov N. In-Memory computing: In plain English. 2014. http://www.gridgain.com/in-memory-computing-in-plain-english/

[23] Tibco Team. In-Memory computing. 2015. http://www.tibco.com/products/automation/in-memory-computing

[24] Techopedia Team. In-Memory computing. 2015. http://www.techopedia.com/definition/28539/in-memory-computing

[25] In memory database: HANA, exadata X3 and flash memory. 2012. http://flashdba.com/2012/10/10/in-memory-databases-part2/

[26] Shun J, Blelloch GE. Ligra: A lightweight graph processing framework for shared memory. In: Proc. of the 18th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming. ACM Press, 2013. 135−146. [doi: 10.1145/2442516.2442530]

[27] Prabhakaran V, Wu M, Weng X, McSherry F, Zhou L, Haridasan M. Managing large graphs on multi-cores with graph awareness. In: Proc. of the Presented as Part of the 2012 USENIX Annual Technical Conf. (USENIX ATC 2012). 2012. 41−52.

[28] Wang G, Xie W, Demers A, Gehrke J. Asynchronous large-scale graph processing made easy. In: Proc. of the 6th Biennial Conf. on Innovative Data Systems Research (CIDR 2013). Asilomar, 2013.

[29] Low Y, Gonzalez J, Kyrola A, Bickson D, Guestrin CE, Hellerstein J. Graphlab: A new parallel framework for machine learning. arXiv preprint arXiv:1006.4990, 2010.

[30] Garcia-Molina H, Salem K. Main memory database systems: An overview. IEEE Trans. on Knowledge and Data Engineering, 1992, 4(6):509−516. [doi: 10.1109/69.180602]

[31] DeWitt D, Katz RH, Olken F, Shapiro LD, Stonebraker MR, Wood DA. Implementation techniques for main memory database systems. In: Proc. of the SIGMOD. 1984. 1−8. [doi: 10.1145/971697.602261]

[32] Kemper A, Neumann T. HyPer: A hybrid OLTP & OLAP main memory database system based on virtual memory snapshots. In: Proc. of IEEE the 27th Int'l Conf. on Data Engineering. Hannover, 2011. 195−206. [doi: 10.1109/ICDE.2011.5767867]

[33] Diaconu C, Freedman C, Ismert E, Larson PA, Mittal P, Stonecipher R, Verma N, Zwilling M. Hekaton: SQL server's memory-optimized OLTP engine. In: Proc. of the 2013 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2013. 1243−1254. [doi: 10.1145/2463676.2463710]

[34] WhiteDB Team. Whitedb. Retrieved 2015. http://whitedb.org/

[35] DB-Engines Team. DB-Engines ranking of graph DBMS. Retrieved 2015. http://db-engines.com/en/ranking/graph+dbms

[36] Shun J, Dhulipala L, Blelloch G. Smaller and faster: Parallel processing of compressed graphs with Ligra+. In: Proc. of the IEEE Data Compression Conf. (DCC). Snowbird, 2015. 403−412. [doi: 10.1109/DCC.2015.8]

[37] Kyrola A, Blei, Loch G, Guestrin C. Graphchi: Large-scale graph computation on just a PC. In: Proc. of the Presented as Part of the 10th USENIX Symp. on Operating Systems Design and Implementation (OSDI 2012). Hollywood, 2012. 3l−46.

[38] Nilakant K, Dalibard V, Roy A, Yoneki E. PrefEdge: SSD prefetcher for large-scale graph traversal. In: Proc. of the Int'l Conf. on Systems and Storage (SYSTOR 2014). New York: ACM Press, 2014. 1−12. [doi: 10.1145/2611354.2611365]

[39] Roy A, Mihailovic I, Zwaenepoel W. X-Stream: Edge-centric graph processing using streaming partitions. In: Proc. of the 24th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2013. 472−488. [doi: 10.1145/2517349.2522740]

[40] Zheng D, Mhembere D, Burns R, Vogelstein J, Priebe CE, Szalay AS. FlashGraph: Processing billionnode graphs on an array of commodity SSDs. In: Proc. of the 13th USENIX Conf. on File and Storage Technologies (FAST 2015). Berkeley: USENIX Association, 2015. 45−58.

[41] Ousterhout J, Agrawal P, Erickson D, Kozyrakis C, Leverich J, Mazières D, Mitra S, Narayanan A, Rosenblum M, Rumble SM, Stratmann E, Stutsman R. The case for RAMClouds: Scalable high-performance storage entirely in DRAM. ACM SIGOPS Operating Systems Review, 2010,43(4):92−105. [doi: 10.1145/1713254.1713276]

[42] Kallman R, Kimura H, Natkins J, Pavlo A, Rasin A, Zdonik S, Jones E, Madden S, Stonebraker M, Zhang Y, Hugg J, Abadi DJ. H-Store: A high-Performance, distributed main memory transaction processing system. Proc. of the VLDB Endowment, 2008,1(2): 1496−1499. [doi: 10.14778/1454159.1454211]

[43] Stonebraker M, Weisberg A. The VoltDB main memory DBMS. IEEE Data Engineering Bulletin, 2013,36(2):21−27.

[44] Muhlbauer T, Rodiger W, Reiser A, Kemper A, Neumann T. ScyPer: Elastic OLAP throughput on transactional data. In: Proc. of the 2nd Workshop on Data Analytics in the Cloud. ACM Press, 2013. 11−15. [doi: 10.1145/2486767.2486770]

[45] Cai Q, Zhang H, Chen G, Ooi BC, Tan KL. Memepic: Towards a database system architecture without system calls. Technical Report, National University of Singapore, 2014.

[46] Sanfilippo S, Noordhuis P. Redis. 2015. http://redis.io

[47] ArangoDB Team. ArangoDB. 2015. https://www.arangodb.com/

[48] Graph Engine Team. Graph engine. 2015. http://www.graphengine.io

[49] Sqrrl Enterprise Team. Sqrrl enterprise. 2015. http://sqrrl.com/product/sqrrl-enterprise/

[50] Titan Team. Titan. 2015. http://thinkaurelius.github.io/titan/

[51] Jiang B. A short note on data-intensive geospatial computing. In: Proc. of the Information Fusion and Geographic Information Systems. Berlin, Heidelberg: Springer-Verlag, 2011. 13−17. [doi: 10.1007/978-3-642-19766-6_2]

[52] Sikka V, Färber F, Lehner W, Sikka V, Färber F, Lehner W, Cha SK, Peh T, Bornhövd C. Efficient transaction processing in SAP HANA database: The end of a column store myth. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2012. 731−742. [doi: 10.1145/2213836.2213946]

[53] Lemke C, Sattler KU, Faerber F, Zeier A. Speeding up queries in column stores. In: Data Warehousing and Knowledge Discovery. Berlin, Heidelberg: Springer-Verlag, 2010. 117−129. [doi: 10.1007/978-3-642-15105-7_10]

[54] Funke F, Kemper A, Neumann T. Compacting transactional data in hybrid OLTP & OLAP databases. Proc. of the VLDB Endowment, 2012,5(11):1424−1435. [doi: 10.14778/2350229.2350258]

[55] Abadi DJ. Query execution in column-oriented database systems. Massachusetts Institute of Technology, 2008,2(2):671−682.

[56] Legler T, Lehner W, Ross A. Data mining with the SAP NetWeaver BI accelerator. In: Proc. of the 32nd Int'l Conf. on Very Large Data Bases. VLDB Endowment, 2006. 1059−1068.

[57] Binnig C, Hildenbrand S, Färber F. Dictionary-Based order-preserving string compression for main memory column stores. In: Proc. of the 2009 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2009. 283−296. [doi: 10.1145/1559845.1559877]

[58] Westmann T, Kossmann D, Helmer S, Moerkotte G. The implementation and performance of compressed databases. Sigmod Record, 1998,29(3):55−67.

[59] Zukowski M, Heman A, Nes N, Boncz P. Super-Scalar RAM-CPU cache compression. In: Proc. of the 22nd Int'l Conf. on Data Engineering (ICDE 2006). IEEE, 2006. 59−70. [doi: 10.1109/ICDE.2006.150]

[60] Zhao J, Li S, Chang J, Byrne JL, Ramirez LL, Lim K, Faraboschi P. Buri: Scaling big-memory computing with hardware-based memory expansion. ACM Trans. on Architecture and Code Optimization, 2015,12(3):1−24. [doi: 10.1145/2808233]

[61] Ferdman M, Adileh A, Kocberber O, Volos S, Alisafaee M, Jevdjic D, Falsafi B. Clearing the clouds: A study of emerging scale-out workloads on modern hardware. ACM SIGPLAN Notices, 2012,47(4):37−48. [doi: 10.1145/2189750.2150982]

[62] Pekhimenko G, Seshadri V, Kim Y, Xin H, Mutlu O, Gibbons PB, Kozuch MA, Mowry TC. Linearly compressed pages: A low-complexity, low-latency main memory compression framework. In: Proc. of the 46th Annual IEEE/ACM Int'l Symp. on Microarchitecture. ACM Press, 2013. 172−184. [doi: 10.1145/2540708.2540724]

[63] Basu B, Gandhi J, Chang J, Hill MD, Swift MM. Efficient virtual memory for big memory servers. ACM SIGARCH Computer Architecture News, 2013,41(3):237−248. [doi: 10.1145/2508148.2485943]

[64] Gokhale M, Holmes B, Iobst K. Processing in memory: The terasys massively parallel PIM array. Computer, 1995,28(4):23−31. [doi: 10.1109/2.375174]

[65] Patternson D, Anderson T, Cardwell N, Fromm R, Keeton K, Kozyrakis C, Thomas R, Yelick K. A case for intelligent RAM: IRAM. In: Proc. of the IEEE Micro. IEEE, 1997. 34−44. [doi: 10.1109/40.592312]

[66] Elliott DG, Snelgrove WM, Stumm M. Computational RAM: A memory-SIMD hybrid and its application to DSP. In: Proc. of the Custom Integrated Circuits Conf. 1992. 1−30. [doi: 10.1109/CICC.1992.591879]

[67] Murakami K, Shirakawa S, Miyajima H. Parallel processing RAM chip with 256Mb DRAM and quad processors. In: Proc. of the 43rd IEEE Int'l Solid-State Circuits Conf. on Digest of Technical Papers (ISSCC). IEEE, 1997. 228−229. [doi: 10.1109/ISSCC.1997.585356]

[68] Memcached Team. Memcached: A distributed memory object caching system. Retrieved 2015. http://memcached.org/

[69] Chang, F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Gruber RE. Bigtable: A distributed storage system for structured data. ACM Trans. on Computer Systems, 2008,26(2):205−218. [doi: 10.1145/1365815.1365816]

[70] Ananthanarayanan G, Ghodsi A, Wang A, Borthakur D, Kandula S, Shenker S, Stoica I. PACMan: Coordinated memory caching for parallel jobs. In: Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation. USENIX Association, 2012. 20−20.

[71] GridGain Team. Gridgain: In-memory computing platform. 2015. http://gridgain.com/

[72] Ananthanarayanan G, Kandula S, Greenberg AG, Stoica I, Lu Y, Saha B, Harris E. Reining in the outliers in map-reduce clusters using mantri. In: Proc. of the 9th USENIX Conf. on Operating Systems Design and Implementation. USENIX Association, 2010. 265−278.

[73] Ananthanarayanan G, Ghodsi A, Shenker S, Stoica I. Disk-Locality in datacenter computing considered irrelevant. In: Proc. of the 13th USENIX Conf. on Hot topics in Operating Systems. USENIX Association, 2011. 12−12.

[74] Wei L, Lian W, Liu K, Wang Y. Hippo: An enhancement of pipeline-aware in-memory caching for HDFS. In: Proc. of the IEEE 23rd Int'l Conf. on Computer Communication and Networks. IEEE, 2014. 1−5. [doi: 10.1109/ICCCN.2014.6911847]

[75] Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proc. of the 9th USENIX Conf. on Networked Systems Design and Implementation (NSDI 2012). Berkeley: USENIX Association, 2012. 2−2.

[76] Power R, Li J. Piccolo: Building fast, distributed programs with partitioned tables. In: Proc. of the 9th USENIX Conf. on Operating Systems Design and Implementation. USENIX Association, 2010. 1−14.

[77] Malewicz G, Austern MH, Bik AJC, Dehnert JC, Horn I, Leiser N, Czajkowski G. Pregel: A system for large-scale graph processing. In: Proc. of the 2010 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2010. 135−146. [doi: 10.1145/1807167.1807184]

[78] Bu Y, Howe B, Balazinska M, Ernst MD. HaLoop: Efficient iterative data processing on large clusters. Proc. of the Vldb Endowment, 2010,3(12):1−2. [doi: 10.14778/1920841.1920881]

[79] Shinnar A, Cunningham D, Herta B, Saraswat V. M3R: Increased performance for in-memory hadoop jobs. Proc. of the VLDB Endowment, 2012,5(12):1736−1747. [doi: 10.14778/2367502.2367513]

[80] Kim SG, Han H, Jung H, Eom H, Yeom HY. Harnessing input redundancy in a MapReduce framework. In: Proc. of the 2010 ACM Symp. on Applied Computing. ACM Press, 2010. 362−366. [doi: 10.1145/1774088.1774167]

[81] He B, Yang M, Guo Z, Chen R, Lin W, Su B, Zhou L. Wave Computing in the Cloud. In: Proc. of the Hotos. 2009.

[82] Gunda PK, Ravindranath L, Thekkath CA, Yu Y, Zhuang L. Nectar: Automatic management of data and computation in datacenters. In: Proc. of the 9th USENIX Conf. on Operating Systems Design and Implementation. USENIX Association, 2010. 1−8.

[83] Chen Y, Alspaugh S, Katz R. Interactive analytical processing in gig data systems: A cross industry study of MapReduce workloads. Proc. of the VLDB Endowment, 2012,5(12):1802−1813. [doi: 10.14778/2367502.2367519]

[84] Tiwari D, Solihin Y. MapReuse: Reusing computation in an in-memory mapreduce system. In: Proc. of the IEEE 28th Int'l Parallel and Distributed Processing Symp. IEEE, 2014. 61−71. [doi: 10.1109/IPDPS.2014.18]

[85] Zaharia M, Das T, Li H, Hunter T, Shenker S, Stoica I. Discretized streams: Fault-Tolerant streaming computation at scale. In: Proc. of the 24th ACM Symp. on Operating Systems Principles. ACM Press, 2013. 423−438. [doi: 10.1145/2517349.2522737]

[86] Marz N. Storm: Distributed and fault-tolerant real-time computation. 2013. https://storm.incubator.apache.org/

[87] Neumeyer L, Robbins B, Nair A, Kesari A. S4: Distributed stream computing platform. In: Proc. of the 2010 IEEE Int'l Conf. on Data Mining Workshops (ICDMW). IEEE, 2010. 170−177. [doi: 10.1109/ICDMW.2010.172]

[88] Condie T, Conway N, Alvaro P, Hellerstein JM, Elmeleegy K, Sears R. MapReduce online. In: Proc. of the 7th USENIX Conf. on Networked Systems Design and Implementation. USENIX Association, 2010. 21.

[89] Nitzberg B, Lo V. Distributed shared memory: A survey of issues and algorithms. Computer, 1991,24(8):52−60. [doi: 10.1109/2.84877]

[90] Li H, Ghodsi A, Zaharia M, Shenker S, Stoica I. Tachyon: Memory throughput I/O for cluster computing frameworks. In: Proc. of the LADIS. 2013.

[91] Valant LG. A bridging model for parallel computation. Communications of the ACM, 1990,33(8):103−111. [doi: 10.1145/79173.79181]

[92] Gonzalez JE, Low Y, Gu H, Bickson D, Guestrin C. PowerGraph: Distributed graph-parallel computation on natural graphs. In: Proc. of the 10th USENIX Conf. on Operating Systems Design and Implementation. Hollywood, 2012. 17−30.

[93] Shao B, Wang H, Li Y. Trinity: A distributed graph engine on a memory cloud. In: Proc. of the 2013 Int'l Conf. on Management of Data. ACM Press, 2013. 505−516. [doi: 10.1145/2463676.2467799]

[94] Barroso LA, Clidaras J, Hölzle U. The datacenter as a computer: An introduction to the design of warehouse-scale machines. Synthesis Lectures on Computer Architecture, 2013,8(3):1−154. [doi: 10.2200/S00516ED2V01Y201306CAC024]

[95] Park BH, Kang BS, Bu SD, Noh TW, Lee J, Jo W. Lanthanum-Substituted bismuth titanate for use in non-volatile memories. Nature, 1999,401(6754):682−684. [doi: 10.1038/44352]

[96] Lee BC, Zhou P, Yang J, Zhang Y, Zhao B, Lpek E, Mutlu O, Burger D. Phase-Change technology and the future of main memory. IEEE Micro, 2010,30(1):143−143. [doi: 10.1109/MM.2010.24]

[97] Raoux S, Burr GW, Breitwisch MJ, Rettner CT. Phase-Change random access memory: A scalable technology. IBM Journal of Research & Development, 2008,52(4):465−479. [doi: 10.1147/rd.524.0465]

[98] Lee BC, Ipek E, Mutlu O, Burger D. Architecting phase change memory as a scalable dram alternative. ACM Sigarch Computer Architecture News, 2009,37(3):2−13. [doi: 10.1145/1555815.1555758]

[99] Govoreanu B, Kar GS, Chen YY, Paraschiv V, Kubicek S, Fantini A, Jossart N. 10×10nm 2 Hf/HfO *x* crossbar resistive RAM with excellent performance, reliability and low-energy operation. In: Proc. of the 2011 IEEE Int'l Electron Devices Meeting (IEDM). IEEE, 2011. 31.6.1−31.6.4. [doi: 10.1109/IEDM.2011.6131652]

[100] Park Y, Shin DJ, Park SK, Park KH. Power-Aware memory management for hybrid main memory. In: Proc. of the 2nd Int'l Conf. on Next Generation Information Technology (ICNIT). IEEE, 2011. 82−85.

[101] Dhiman G, Ayoub R, Rosing T. PDRAM: A hybrid pram and dram main memory system. In: Proc. of the 46th Annual Design Automation Conf. IEEE, 2009. 664−669. [doi: 10.1145/1629911.1630086]

[102] Lee HG, Baek S, Nicopoulos C, Kim J. An energy-and performance-aware dram cache architecture for hybrid DRAM/PCM main memory systems. In: Proc. of the 29th Int'l Conf. on Computer Design (ICCD). IEEE, 2011. 381−387. [doi: 10.1109/ICCD.2011.6081427]

[103] Ham TJ, Chelepalli BK, Xue N, Lee BC. Disintegrated control for energy-efficient and heterogeneous memory systems. In: Proc. of the 19th Int'l Symp. on High Performance Computer Architecture (HPCA 2013). IEEE, 2013. 424−435. [doi: 10.1109/HPCA.2013.6522338]

[104] Qureshi MK, Srinivasan V, Rivers JA. Scalable high performance main memory system using phase-change memory technology. ACM SIGARCH Computer Architecture News, 2009,37(3):24−33. [doi: 10.1145/1555815.1555760]

[105] Park KH, Park SK, Seok H, Hwang W, Shin DJ, Choi JH, Park KW. Efficient memory management of a hierarchical and a hybrid main memory for MNMATE platform. In: Proc. of the 2012 Int'l Workshop on Programming Models and Applications for Multicores and Manycores. ACM Press, 2012. 83−92. [doi: 10.1145/2141702.2141712]

[106] Zhou P, Zhao B, Yang J, Zhang Y. A durable and energy efficient main memory using phase change memory technology. ACM Sigarch Computer Architecture News, 2009,37(3):14−23. [doi: 10.1145/1555815.1555759]

[107] Lee S, Bahn H, Noh SH. Characterizing memory write references for efficient management of hybrid PCM and DRAM memory. In: Proc. of the 19th Annual IEEE Int'l Symp. on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems. IEEE Computer Society, 2011. 168−175. [doi: 10.1109/MASCOTS.2011.68]

[108] Lee S, Bahn H, Noh SH. CLOCK-DWF: A write-history-aware page replacement algorithm for hybrid PCM and DRAM memory architectures. IEEE Trans. on Computers, 2014,63(9):2187−2200. [doi: 10.1109/TC.2013.98]

[109] Khouzani HA, Yang C, Hu J. Improving performance and lifetime of DRAM-PCM hybrid main memory through a proactive page allocation strategy. In: Proc. of the 20th Asia and South Pacific Design Automation Conf. (ASP-DAC). IEEE, 2015. 508−513. [doi: 10.1109/ASPDAC.2015.7059057]

[110] Qureshi MK, Franceschini MM, Lastras-Monta LA, Karidis JP. Morphable memory system: A robust architecture for exploiting multi-level phase change memories. ACM Sigarch Computer Architecture News, 2010,38(3):153−162. [doi: 10.1145/1816038.1815981]

[111] Ferreira A, Zhou M, Bock S, Childers B, Melhem R, Mosse D. Increasing PCM main memory lifetime. In: Proc. of the Design, Automation Test in Europe Conf. (DATE). 2010. 914−919. [doi: 10.1109/DATE.2010.5456923]

[112] Wuif WA, Mckee SA. Hitting the memory wall: Implications of the obvious. ACM SIGARCH Computer Architecture News, 1994, 23(1):20−24. [doi: 10.1145/216585.216588]

[113] Hennessy JL. Patterson DA. Computer Architecture: A Quantitative Approach. 5th ed., Morgan Kaufmann Publishers, 2011. 71−105.

[114] Boncz P, Grust T, Van Keulen M, Manegold S, Rittinger J, Teubner J. MonetDB/XQuery: A fast xquery processor powered by a relational engine. In: Proc. of the 2006 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2006. 479−490. [doi: 10.1145/1142473.1142527]

[115] He B, Li Y, Luo Q, Yang D. EaseDB: A cache-oblivious in-memory query processor. In: Proc. of the 2007 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2007. 1064−1066. [doi: 10.1145/1247480.1247607]

[116] Knizhnik K. FastDB main memory relational database management system. 1999. http://www.garret.ru/fastdb.html

[117] TIMESTEN Team. TIMESTEN. 2015. http://www.oracle.com/timesten/index.html

[118] Yu Y, Isard M, Fetterly D, Budiu M, Erlingsson U, Gunda PK, Currey J. DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language. In: Proc. of the 8th USENIX Symp. on Operating Systems Design and Implementation (OSDI 2008). San Diego, 2008. 1−14.

[119] Olston C, Reed B, Srivastava U, Kumar R, Tomkins A. Pig latin: A not-so-foreign language for data processing. In: Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2008. 1099−1110. [doi: 10.1145/1376616.1376726]

[120] Chambers C, Raniwala A, Perry F, Adams S, Henry RR. Flumejava: Easy, efficient data-parallel pipelines. ACM Sigplan Notices, 2010,45(6):363−375. [doi: 10.1145/1809028.1806638]

[121] Murray DG, Schwarzkope M, Smowton C, Smith S, Madhavapeddy A, Hand S. Ciel: A universal execution engine for distributed data-flow computing. In: Proc. of the 8th ACM/USENIX Symp. on Networked Systems Design and Implementation. Boston, 2011. 113−126.

[122] Ekanayake J, Li H, Zhang B, Gunarathne T, Bae S H, Qiu J, Fox G. Twister: A runtime for iterative MapReduce. In: Proc. of the 19th ACM Int'l Symp. on High Performance Distributed Computing. ACM Press, 2010. 810−818. [doi: 10.1145/1851476.1851593]

[123] Gonzalez JE, Xin RS, Dave A, Crankshaw D, Franklin MJ, Stoica I. GraphX: Graph processing in a distributed dataflow framework. In: Proc. of the 11th USENIX Conf. on Operating Systems Design and Implementation. USENIX Association, 2014. 599−613.

[124] Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Xin D. Mllib: Machine learning in apache spark. arXiv preprint arXiv:1505.06807, 2015.

[125] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Ghemawat S. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467, 2016.

[126] Moritz P, Nishihara R, Stoica I, Jordan MI. SparkNet: Training deep networks in spark. arXiv preprint arXiv:1511.06051, 2015.

[127] Singh S, Singh N. Big data analytics. In: Proc. of the Int'l Conf. on Communication, Information and Computing Technology. IEEE Press, 2012. 1−4. [doi: 10.1109/ICCICT.2012.6398180]

[128] Zheng N, Zhang J, Wang C. Special issue on big data research in China. Knowledge and Information Systems, 2014,41(2):247−249. [doi: 10.1007/s10115-014-0792-5]

[129] Yu X, Bezerra G, Pavlo A, Devadas S, Stonebraker M. Staring into the abyss: An evaluation of concurrency control with one thousand cores. Proc. of the VLDB Endowment, 2014,8(3):209−220. [doi: 10.14778/2735508.2735511]

[130] Xu Y, Musgrave Z, Noble B, Bailey M. Bobtail: Avoiding long tails in the cloud. In: Proc. of the 10th USENIX Conf. on Networked Systems Design and Implementation (NSDI 2013). USENIX Association, 2013. 329−342.

[131] Ren X, Anajthanarayanan G, Wierman A, Yu M. Hopper: Decentralized speculation-aware cluster scheduling at scale. In: Proc. of the 2015 ACM Conf. on Special Interest Group on Data Communication (SIGCOMM 2015). ACM Press, 2015. 379−392. [doi: 10.1145/2785956.2787481]

[132] Ahmad F, Chakradhar ST, Raghunathan A, Vijaykumar TN. ShuffleWatcher: Shuffle-aware scheduling in multi-tenant MapReduce clusters. In: Proc. of the 2014 USENIX Conf. on USENIX Annual Technical Conf. (USENIX ATC 2014). USENIX Association, 2014. 1−12.

[133] Chen Y, Alspaugh S, Katz R. Interactive analytical processing in big data systems: A cross-industry study of MapReduce workloads. Proc. of the VLDB Endowment, 2012,5(12):1802−1813. [doi: 10.14778/2367502.2367519]

[134] Gandhi R, Hu YC, Kou CK, Liu H, Zhang M. Rubik: Unlocking the power of locality and end-point flexibility in cloud scale load balancing. In: Proc. of the 2015 USENIX Conf. on Usenix Annual Technical Conf. (USENIX ATC 2015). USENIX Association, 2015. 473−484.

[135] Mace J, Bodik P, Fonseca R, Musuvath M. Retro: Targeted resource management in multi-tenant distributed systems. In: Proc. of the 12th USENIX Conf. on Networked Systems Design and Implementation (NSDI 2015). USENIX Association, 2015. 589−603.

[136] Ousterhout K, Wendell P, Zaharia M, Stoica I. Sparrow: Distributed, low latency scheduling. In: Proc. of the 24th ACM Symp. on Operating Systems Principles (SOSP 2013). ACM Press, 2013. 69−84. [doi: 10.1145/2517349.2522716]

[137] Isard M, Budiu M, Yu Y, Birrell A, Fetterly D. Dryad: Distributed data-parallel programs from sequential building blocks. In: Proc. of the 2nd ACM SIGOPS/EuroSys European Conf. on Computer Systems (EuroSys 2007). ACM Press, 2007. 59−72. [doi: 10.1145/1272998.1273005]

[138] Nguyen D, Lenharth A, Pingali K. A lightweight infrastructure for graph analytics. In: Proc. of the 24th ACM Symp. on Operating Systems Principles (SOSP 2013). ACM Press, 2013. 456−471. [doi: 10.1145/2517349.2522739]

[139] Qu H, Mashayekhi O, Terei D, Levis P. Canary: A scheduling architecture for high performance cloud computing. arXiv preprint arXiv:1602.01412, 2016.

[140] Burr GW, Kurdi BN, Scott JC, Lam CH, Gopalakrishnan K, Shenoy RS. Overview of candidate device technologies for storage-cass memory. IBM Journal of Research and Development, 2008,52(4-5):449−464. [doi: 10.1147/rd.524.0449]

[141] Condit J, Nightingale EB, Frost C, Ipek E, Lee BC, Burger D, Coetzee D. Better I/O through byte-addressable, persistent memory. In: Proc. of ACM SIGOPS the 22nd Symp. on Operating Systems Principles. ACM Press, 2009. 133−146. [doi: 10.1145/1629575.1629589]

[142] Chen S, Gibbons PB, Nath S. Rethinking database algorithms for phase change memory. In: Proc. of the CIDR. 2011. 21−31.

[143] Volos H, Tack AJ, Swift MM. Mnemosyne: Lightweight persistent memory. ACM SIGPLAN Notices, 2011,46(3):91−104. [doi: 10.1145/1961296.1950379]

[144] Coburn J, Caulfield AM, Akel A, Grupp LM, Gupta RK, Jhala R, Swanson S. NV-Heaps: Making persistent objects fast and safe with next-generation, non-volatile memories. ACM SIGARCH Computer Architecture News, 2011,39(1):105−118. [doi: 10.1145/1961295.1950380]

[145] Wu X, Reddy AL. SCMFS: A file system for storage class memory. In: Proc. of the 2011 Int'l Conf. for High Performance Computing, Networking, Storage and Analysis. ACM Press, 2011. 39−49. [doi: 10.1145/2063384.2063436]

[146] Venkataraman S, Tolia N, Ranganathan P, Campbell RH. Consistent and durable data structures for non-volatile byte-addressable memory. In: Proc. of the 9th USENIX Conf. on File and Storge Technologies. San Jose, 2011. 61−75.

[147] Narayanan D, Hodson O. Whole-System persistence. ACM SIGARCH Computer Architecture News, 2012,40(1):401−410. [doi: 10.1145/2189750.2151018]

[148] Barber R, Bendel P, Czech M, Draese O, Ho F, Hrle N, Idreos S, Kim MS, Koeth O, Lee JG, Li TT, Lohman GM, Morfonios K, Muller R, Murthy K, Pandis I, Qiao L, Raman V, Szabo S, Sidle R, Stolze K. Blink: Not your father's database! In: Proc. of the Enabling Real-Time Business Intelligence. Berlin, Heidelberg: Springer-Verlag, 2011. 1−22. [doi: 10.1007/978-3-642-33500-6_1]

[149] Maas LM, Kissinger T, Habich D, Lehner W. Buzzard: A numa-aware in-memory indexing system. In: Proc. of the 2013 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2013. 1285−1286. [doi: 10.1145/2463676.2465342]

[150] Willhalm T, Popovici N, Boshmaf Y, Plattner H, Zeier A, Schaffner J. SIMD-Scan: Ultra fast in-memory table scan using on-chip vector processing units. Proc. of the Vldb Endowment, 2009,2(1):385−394. [doi: 10.14778/1687627.1687671]

[151] Leis V, Kemper A, Neumann T. Exploiting hardware transactional memory in main-memory databases. In: Proc. of IEEE the 30th Int'l Conf. on Data Engineering (ICDE). IEEE, 2014. 580−591. [doi: 10.1109/ICDE.2014.6816683]

[152] Damaraju S, George V, Jahagirdar S, Khondker T, Milstrey R, Sarkar S, Siers S, Stolero I, Subbiah A. A 22nm IA multi-CPU and GPU system-on-chip. In: Proc. of the 2012 IEEE Int'l Solid-State Circuits Conf. on Digest of Technical Papers (ISSCC). IEEE, 2012. 56−57. [doi: 10.1109/ISSCC.2012.6176876doi: 10.1109/IPEC.2012.6522610]

[153] Kalia A, Kaminsky M, Andersen DG. Using RDMA efficiently for key-value services. ACM SIGCOMM Computer Communication Review, 2014,44(4):295−306. [doi: 10.1145/2740070.2626299]

[154] Jha S, He B, Lu M, Cheng X, Huynh HP. Improving main memory Hash joins on Intel Xeon Phi processors: An experimental approach. Proc. of the VLDB Endowment, 2015,8(6):642−653. [doi: 10.14778/2735703.2735704]

[155] Loghin D, Tudor BM, Zhang H, Ooi BC, Teo YM. A performance study of big data on small nodes. Proc. of the VLDB Endowment, 2015,8(7):762−773. [doi: 10.14778/2752939.2752945]

[156] Low Y, Bickson D, Gonzalez J, Guestrin C, Kyrola A, Hellerstein JM. Distributed graphlab: A framework for machine learning and data mining in the cloud. Proc. of the VLDB Endowment, 2012,5(8):716−727. [doi: 10.14778/2212351.2212354]

[157] Salihoglu S, Widom J. Optimizing graph algorithms on pregel-like systems. Proc. of the VLDB Endowment, 2014,7(7):577−588. [doi: 10.14778/2732286.2732294]

[158] Zhong J, He B. Medusa: Simplified graph processing on GPUs. IEEE Trans. on Parallel and Distributed Systems, 2014,25(6):1543−1552. [doi: 10.1109/TPDS.2013.111]

[159] Chen L, Huo X, Ren B, Jain S, Agrawal G. Efficient and simplified parallel graph processing over CPU and MIC. In: Proc. of the 2015 IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS). IEEE, 2015. 819−828. [doi: 10.1109/IPDPS.2015.88]

[160] Fu Z, Personick M, Thompson B. Mapgraph: A high level API for fast development of high performance graph analytics on GPUs. In: Proc. of the Workshop on Grpph Data Management Experiences and Systems. ACM Press, 2014. 1−6. [doi: 10.1145/2621934.2621936]

[161] Coates A, Huval B, Wang T, Wu D, Catanzaro B, Andrew N. Deep learning with cots HPC systems. In: Proc. of the 30th Int'l Conf. on Machine Learning. Atlanta, 2013. 1337−1345.

[162] Dean J, Corrado G, Monga R, Chen K, Devin M, Mao M, Ranzato M, Aurelio S, Andrew T, Paul Y, Ke L, Quoc V, Ng A Y. Large scale distributed deep networks. In: Proc. of the Advances in Neural Information Processing Systems. 2012. 1223−1231.

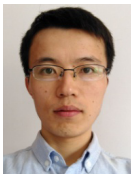
[163] Chilimbi T, Suzue Y, Apacible J, Kalyanaraman K. Project ADAM: Building an efficient and scalable deep learning training system. In: Proc. of the 11th USENIX Symp. on Operating Systems Design and Implementation. Broomfield, 2014. 571−582.

[164] Li M, Andersen DG, Park JW, Smola AJ, Ahmed A, Josifovski V, Long J, Shekita EJ, Su BY. Scaling distributed machine learning with the parameter server. In: Proc. of the 11th USENIX Symp. on Operating Systems Design and Implementation. Broomfield, 2014. 583−598.

[165] Ho Q, Cipar J, Cui H, Lee S, Kim JK, Gibbons PB, Gibson GA, Ganger G, Xing EP. More effective distributed ML via a stale synchronous parallel parameter server. In: Proc. of the Advances in Neural Information Processing Systems, 2013. 1223−1231.

[166] Hadjis S, Abuzaid F, Zhang C. Caffe con troll: Shallow ideas to speed up deep learning. In: Proc. of the Computer Science. 2015. 1−4. [doi: 10.1145/2799562.2799641]

[167] Iandola FN, Ashraf K, Moskewicz MW, Keutzer K. FireCaffe: Near-linear acceleration of deep neural network training on compute clusters. arXiv preprint arXiv:1511.00175, 2015.



罗乐(1987－),男,陕西旬邑人,博士生,主要研究领域为并行计算,内存计算.

钱德沛(1952－),男,教授,博士生导师,CCF会士,主要研究领域为计算机体系结构,高性能计算,计算机网络.

刘轶(1968－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为计算机系统结构,高性能计算,并行计算,计算机网络.