软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(1):59−83 [doi: 10.13328/j.cnki.jos.005141]

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

新型非易失存储环境下事务型数据管理技术研究*

潘 巍, 李战怀, 杜洪涛, 周陈超, 苏 静

(西北工业大学 计算机学院,陕西 西安 710072)

通讯作者: 李战怀, E-mail: lizhh@nwpu.edu.cn

摘 要: 为适应底层存储架构的变化,上层数据库系统已经经历了多轮的演化与变革.在大数据环境下,以非易失、大容量、低延迟、按字节寻址等为特征的新型非易失存储器件(NVM)的出现,势必对数据库系统带来重大影响,相关的存储与事务处理技术是其中值得关注的重要环节.首先,概述了事务型数据库系统随存储环境发展的历史与趋势;然后,对影响上层数据管理系统设计的非易失性存储技术以及面向大数据应用领域与硬件环境优化的事务技术进行综述与分析;最后,对非易失存储环境下事务型数据库面临的挑战与研究趋势进行了展望.

关键词: 非易失存储器;存储级内存;事务处理;日志;数据库

中图法分类号: TP311

中文引用格式: 潘巍,李战怀,杜洪涛,周陈超,苏静.新型非易失存储环境下事务型数据管理技术研究.软件学报,2017,28(1):59−83. http://www.jos.org.cn/1000-9825/5141.htm

英文引用格式: Pan W, Li ZH, Du HT, Zhou CC, Su J. State-of-the-Art survey of transaction processing in non-volatile memory environments. Ruan Jian Xue Bao/Journal of Software, 2017,28(1):59−83 (in Chinese). http://www.jos.org.cn/1000-9825/5141.htm

State-of-the-Art Survey of Transaction Processing in Non-Volatile Memory Environments

PAN Wei, LI Zhan-Huai, DU Hong-Tao, ZHOU Chen-Chao, SU Jing

(School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: The design of the upper lever database has experienced several rounds of development and transformation to adapt to the changing architecture of the underlying storage system. In the big data era, the emergence of the novel non-volatile memory (NVM) technologies, which exhibit a series of non-volatile (persistent writes), high-capacity, low-latency and byte-addressable characteristics, has brought significant impact on traditional database systems, especially for techniques related to storage and transaction processing. First, in this paper, the phylogeny and development trend of the OLTP database along with the storage subsystem is introduced. Then, the non-volatile memory technology which affects the upper data management system design is reviewed along with an analysis on the domain-oriented and the NVM-oriented transaction technologies. Finally, challenges and opportunities are addressed for the NVM-oriented OLTP database.

Key words: non-volatile memory; storage-class memory; transaction processing; log; database

近年来,数据管理领域的发展非常迅猛,特别是在大数据应用的推动下,传统关系型数据库"一体适用(one size fit all)"[1]的"美好时代"正在远去,依据"应用场景定制(one size fits a bunch)"的数据管理系统正在成为大家新的共识.在此过程中,虽然数据库系统在积极地不断自我调整与优化,但是数据库系统的基础架构、核心功能、策略模式以及优化技术,特别是与事务相关的部分内容,在很大程度上都是由其当时的计算机硬件水平决定的.

* 基金项目: 国家自然科学基金(61472321, 61332006, 61672434); 国家高技术研究发展计划(863)(2015AA015307)

Foundation item: National Natural Science Foundation of China (61472321, 61332006, 61672434); National High Technology Research and Development Program (863) of China (2015AA015307)

收稿时间: 2016-10-17; 修改时间: 2016-10-26; 采用时间: 2016-11-11; jos 在线出版时间: 2016-11-24

CNKI 网络优先出版: 2016-11-24 13:41:13, http://www.cnki.net/kcms/detail/11.2560.TP.20161124.1341.002.html

其中,存储层级之间的数据 I/O 特征对事务型关系数据库系统在设计空间中的选择有着重要的影响.

为积极适应底层存储环境的变化,上层数据库系统已经经历了多轮的演化与变革.以磁盘为主要存储介质、以闪存为主要存储介质以及以内存为主要存储介质[2]的研究方向,都反映出数据库关键技术随存储环境一致变化的发展趋势.因此可以预见:以非易失、大容量、低延迟、高并发访问能力、按字节寻址等为特征的新型非易失存储(non-volatile memory,简称 NVM)的出现和快速发展,势必和大数据应用环境一起给数据库系统的发展带来重大影响,相关的存储和事务处理技术是其中值得关注的重点.

本文第 1 节分析 OLTP 数据库系统随底层存储子系统共同演化的历史.第 2 节从存储体系策略、访问接口、NVM 模拟器等层面分析对上层数据管理系统设计产生影响的非易失存储技术.第 3 节剖析面向领域和存储环境优化的事务处理技术研究现状和关键技术.最后,在第 4 节对非易失存储环境下 OLTP 数据管理系统面临的挑战和研究趋势进行展望.

1 研究背景

诞生于 20 世纪 60 年代末的数据库技术经过不到半个世纪的发展,已经成为高效组织、管理和存取海量数据的标准方法,被广泛地应用于金融、交通、制造、能源、旅游、通信等众多领域,并获得了极大的成功.其中,关系数据库理论的发展为数据库技术奠定了理论基础.随后,事务处理技术[3]的引入使得关系型数据库具备了良好的保障数据完整性、一致性、并发性和可恢复性的能力,从而确保了数据库操作的可靠性和安全性,为数据库技术在不同领域中的大规模应用创造了必要的条件.

事务是关系数据库构成单一逻辑工作单元的操作序列,满足 all-or-nothing 原则,即:一个事务的全部动作要么全部执行,要么全部不执行.严格的事务语义可以确保事务在执行过程中一旦发生错误,其产生的部分影响会被正确撤销,这是事务的原子性;其次,事务一旦执行成功,其对数据库所产生的影响必须永久保存下来,这是事务的持久性;再者,对于存在多个事务并发执行的数据库系统,当多个事务企图对共享数据同时进行更新访问时,数据库系统需要保证事务不受并发执行的任何其他事务的影响,这是事务的隔离性;最终,使得通过事务来完成操作的数据库系统始终保持一致性.严格事务语义的保证,使得传统的关系型数据库在任务关键型应用领域中具有不可替代的地位.

传统的关系型数据库系统是面向磁盘(disk-oriented)的,其主要架构在由 HDD 和 DRAM 构建的两层存储层级上.为了保证在系统发生故障时(特别是系统崩溃或断电时)的数据持久化,数据库使用非易失的、高容量、慢速、读写不对称、块可寻址(block-addressable)、廉价的 HDD 或 SSD 作为主存储介质,而将易失的、低容量、高速随机读写、字节可寻址(byte-addressable)的 DRAM 作为缓存.因此,慢速 HDD 与高速 DRAM 之间的 I/O 成为整个系统中最主要的性能瓶颈.

为了在保证严格事务语义的同时获得较好的开销性能比,面向磁盘的数据库系统采用诸如缓存池、并发控制等多种组件或策略,试图协调或隐藏存储体之间的性能差异.Harizopoulos 等人[4]通过在改造的开源数据库系统 Shore 上执行一组 TPC-C 的 New Order 事务测试,证实了这些与存储相关的模块在事务执行过程中所占的时间开销比高达近 85%.特别是在当下以数据为中心的大数据时代,面向磁盘的数据库系统固有的内存-磁盘访问模式所带来的 I/O 瓶颈愈发凸显,各种优化技术也只能在一定程度上实现缓解,采用磁盘作为主存储介质将会严重制约数据库系统的整体性能.

20 世纪 80 年代,伴随着半导体技术的发展,内存的容量不断增加,同时价格不断降低,海量的工作负载与有限的内存容量之间的矛盾正在缓解,数据无法常驻内存的场景也在发生变化.随之,面向内存(memory-oriented)的主存数据库[5]走入人们的视野.面向内存的数据库系统仍然建立在两层存储结构之上,对内存数据库而言,活跃的工作数据集(数据主版本)全部保存在内存中,而磁盘仅作为线下的备份设备.由于 DRAM 与 HDD 之间的 I/O 大幅减少,相比于面向磁盘的数据库系统,面向主存的系统在保证严格事务语义的同时,还具有很高的吞吐量.但是由于 DRAM 的易失性,很多内存数据库[6−9]还是不得不沿袭 System R[10]中的部分传统组件来对事务持久性提供保证.例如,基于磁盘的 WAL 日志[11],这部分组件的存在,严重影响了系统的整体性能.也有一些被称为

NewSQL 的新型内存数据库系统采用非 WAL 日志以减少日志所引入的磁盘 I/O,典型的如 H-Store[12]及其商用版本 VoltDB 采用的命令日志(command logging).命令日志借助快照和高速的内存重放能力,满足了系统对高可靠持久化的需求,但是相比于 WAL 日志,其恢复时间相对较长.同时,受限于 DRAM 本身的硬件特性,高容量的 DRAM 也会带来很高的功耗[13],这也不符合现今“高性能/能耗比”系统的发展趋势.

具有非易失性的闪存(flash)在存储容量和能耗上都优于 DRAM,同时,读写速度则超过磁盘百倍以上.随着闪存技术的不断成熟和规模化量产,其优异的特性引起了研究人员的关注.作为全电设备的闪存,与机械化磁盘具有完全迥异的物理特性,因此,简单地将数据库迁移到由闪存替换磁盘的存储体系中并不能充分发挥闪存的优势.基于此,研究人员在缓存区管理[14,15]、索引[16,17]、查询优化[18,19]、事务恢复[20,21]等方面开展了大量的面向闪存数据库的研究,试图寻找更适合闪存特征的数据管理技术.近期,3D NAND[22]技术通过将存储单元垂直堆叠,进一步提高了闪存的存储密度,再一次突破了平面结构闪存的发展极限.但是客观上,闪存与 DRAM 相比仍然存在不小的性能鸿沟,而且从存储体系上来看,闪存更多的是作为磁盘的替代者,而本质上并没有改变两层存储体系结构.同时,闪存介质始终存在一些难以克服的内在缺陷,例如异位更新、读写不对称、访问粒度粗、使用寿命有限等问题[23].这些问题也在很多方面制约着闪存数据库的大规模应用.此外,一种名为 NVDIMM[24]的新型存储也引发了研究人员的关注,其利用超级电容,使得 DRAM 在掉电后可以在短时间内应对易失性问题.但其不能解决存储介质高能耗、低性价比的问题.因此从长远来看,上述新硬件引发的数据库技术只能作为“权宜之计”.

由此可见,在严格事务语义的要求下,数据库不得不在易失与非易失存储设备之间,在访问延迟、存储容量、性价比等多个重要指标中进行取舍与权衡.这主要是因为传统的存储器件在这些关键指标上“鱼与熊掌不可兼得”.而且在大数据应用环境下,保证严格 ACID 的需求与系统吞吐能力之间的矛盾愈发突出.但是近年来,一些新型非易失存储器(NVM),如相变存储器(phase change memory,简称 PCM)、磁阻式存储器(magnetoresistive random-access memory,简称 MRAM)、阻变式存储器(resistive random access memory,简称 RRAM)、铁电存储器(ferroelectronic RAM,简称 FeRAM)等的出现正在打破这种桎梏.这类存储器共同的特点使它们同时拥有内存式的高速处理以及外存式的持久化双重能力.因此,一旦当 NVM 进入现有的存储体系后,它必将打破计算机传统架构中 CPU、主存、系统总线、外存之间的平衡,也会显著改变传统存储体系构筑的金字塔结构.显而易见,融入 NVM 的新型非易失存储环境将突破以往存储层级间无法逾越的 I/O 瓶颈,为大数据环境下的事务型数据管理技术的研究带来全新的机遇.

2 影响上层数据管理系统设计的非易失性存储技术研究

近年来出现的新型非易失存储(NVM)[25]是一类存储技术的概括性术语,它并不代表某个具体的存储技术或者介质,在部分研究文献中也被称为存储级内存(storage class memory,简称 SCM)[26].不同的名称只是从不同侧面突出了新型存储在非易失与高容量上的显著特征.

新型非易失存储器在持久化、随机读写能力、存储密度、扩展能力、漏电功耗等多个方面与传统的存储介质相比具有明显优势.虽然不同的 NVM 在主要特征上具有一定的相似性,但是它们的工作原理以及设计和制造工艺都存在明显差异,因此在性能指标上也有所不同.根据 NVM 的各自特点以及访问需求的差异,以合适的方式将其融入存储架构是最基础的问题.无论是高吞吐的事务处理或是数据密集型计算,都需要高性能存储环境的支撑,融入 NVM 的新型存储环境有望跨越 CPU 与外存之间的性能鸿沟,消除计算机系统中制约上层软件设计的 I/O 瓶颈.

2.1 NVM非易失存储体系的架构策略与优化技术研究

将 NVM 融入现有存储体系的主要策略有两种:替换与混合.如图 1 所示:由于不同的 NVM 在访问延迟、耐久性等方面各具特点,理论上可以出现在传统存储体系的任何层级中.

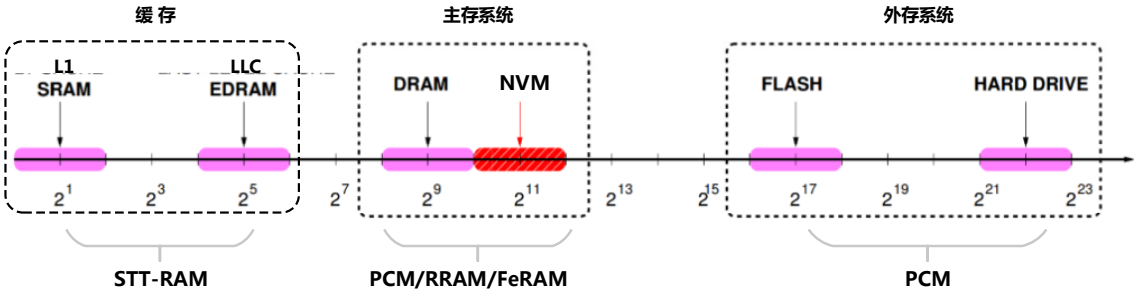


Fig.1 Typical access latency[27] and NVM-integrated hierarchy

图 1 访问延迟[27]与融入 NVM 的存储层级

2.1.1 替换片上缓存的优化技术研究

STT-RAM 的高写耐受性使其适合替换片上缓存.但是相比于 SRAM,STT-RAM 仍然存在写延迟高和写功耗高的现状,一些适用于 SRAM 的经典技术的有效性也因此受到一定的影响.如果将 STT-RAM 作为最后一级缓存(last level cache,简称 LLC),其写延迟会对传统预取技术的有效性产生明显的影响.文献[28]分析指出:为关键路径上的访问请求分配较高的优先级,可以避免较长的等待时间,有效控制写延迟对系统性能的影响.同时,系统整体性能受写请求频率、访问冲突以及缓存容量等多个因素共同影响,因此需要在局部和全局进行全面权衡.文献[28]提出了请求优先级(RP)和局部-全局预取控制(HLGPC)等技术.实验结果表明:通过对上述技术的组合应用,在四核系统中获得了 6.5%~11%的系统性能提升,并节省了 4.8%~7.3%的能耗.

相比于读操作,STT-RAM 的写操作不但具有相对较高的延迟,同时还会产生较高的隧穿电流并导致显著的功耗.电路级的位粒度写避免技术[29]是一种减少写操作、控制写功耗的优化策略.该技术通过复用现有的读电路,回避了传统的写避免技术需要设计专门的读出电路以探索待写入数据变化状态的额外开销.

除了写操作本身,在末级缓存中还存在大量的在缓存生命周期内并未被重新使用的数据,如果将此类数据旁路缓存,并不会引发显著的缓存缺失.基于上述重要的观察,文献[30]将传统的不活跃缓存块预测技术应用于 STT-RAM 的不活跃写的预测中,通过将预测的不活跃写数据旁路缓存,大幅减少了不必要或者冗余的写操作,降低了缓存缺失,并有效地减少了写能耗.

但是要想完全消除不活跃写数据是非常困难的,除非能够准确获得未来的缓存访问模式.同时,由于误判导致的旁路缓存会带来一定的缓存缺失,虽然可以利用刷新机制有效降低缓存缺失发生的频率,但当刷新操作过于频繁时又会与读写操作产生严重的冲突,并产生大量的能耗.因此,文献[31]提出了一种缓存一致的自适应更新策略,实现了刷新操作的最小化.

相比于其他新型非易失存储,虽然 STT-RAM 在理论上具有较高的写耐受性,但在实际测试中,由于各种因素的影响,其使用寿命尚无法达到理论期望的数量级[32].同时,如果频繁地在局部进行写操作,也会缩短整个器件的寿命.因此,减少写操作[33]以及设计均衡写操作的缓存管理策略[34,35]成为解决该问题的主要技术手段.此外,利用与 SRAM 的混合架构分离或重定向写请求,也可以有效地降低写损耗[36],但是只有写密集的应用才可以摊销复杂的写请求管理开销.上述面向 SST-RAM 的缓存优化技术在不同层面弥补了 SST-RAM 由于其物理特性导致的负面影响.

2.1.2 替换内存的优化技术研究

在内存替换层面,由于 PCM 具有低静态功耗、高存储密度、低制程工艺等特点,因而成为大容量非易失内存的最佳候选.但是 PCM 写操作依然存在高延时、高功耗的问题.同时,PCM 还存在明显的读写不对称现象(写延迟大约是读延迟的 20 倍),同一个 bank 上读写请求的冲突会显著增加读请求的响应延迟.因此,优先 PCM 读是一种缩减读写总体响应时间的优化思路.文献[37]设计了基于阈值的写取消策略来优先响应读请求,其核心思想是:在预定周期内,当新的读请求到来时,系统主动取消已被调度但还未被执行的在同一个 bank 上的写请求.为了减少由于写取消策略导致的写请求重新调度的开销,还针对需要多轮迭代才能完成的写操作提供了写

暂停策略.写暂停策略不会直接取消整个写操作,而是在每轮迭代结束时暂停写操作,检查并优先响应新的读请求.

PCM 除了存在读写不对称的现象外,其写入二进制 0 与 1 的速度也存在显著的差异.PCM 利用硫化物在非晶态和晶态之间存在不同的阻性这一特点来存储数据.PCM 写 1 是一个变化较慢的中等电脉冲加热结晶 SET 过程,而写 0 则是变化较快的高强度电脉冲转入非结晶态的 RESET 过程.因此准确来说,PCM 写操作较慢的本质原因是由于 PCM 存在写 1 的 SET 过程.根据 PCM 的这一特性,Qureshi 等人[38]提出了 PreSET 优化机制,通过预先对给定的内存行的所有位执行 SET 操作,使得后续基于该内存行的写操作实际只需要在特定的位置完成快速的 RESET 过程即可.相比于之前的写取消策略,该预写技术能够保证更低的读延迟,并且可以和写取消策略联合使用,进一步获得更显著的性能提升.

文献[39]也针对 PCM 的 SET 与 RESET 过程的不对称性提出了将写操作分离为优先写 0 和并发写 1 两个阶段的 two-stage-write 技术.同时,对于待写数据块中 1 的数量超过半数以上的情况,利用写 1 和写 0 反转的思想实现了进一步的优化,降低了写 1 的开销,并利用反转标志位保证了数据的正确读取.

为了缓解内存频繁的写操作需求与 PCM 有限的寿命之间的矛盾,研究者们探索了包括写前读策略[40,41]、磨损平衡算法[42,43]、数据编码压缩[44]、写敏感的缓存替换方法[45,46]等优化技术来控制写操作的执行过程,提升 PCM 的使用寿命.虽然这些方案本身不可避免地在软、硬件上引入了一些额外的开销,并对 PCM 的写性能产生了一定的影响,但是基于内存对可靠读写的要求和 PCM 读写不对称的特点,上述都是有效且可行的折中方法.

2.1.3 替换外存的优化技术研究

用 NVM 直接替换外存,也是在存储体系中使用 NVM 的主要方式之一.为低速块级外存设计的软件系统是限制与 NVM 进行高速 I/O 的关键瓶颈,同时,一些针对块级设备行之有效的假设和设计策略也需要被重新考量.

中断驱动的异步 I/O 访问模式一直以来都是一种非常有效的面向磁盘的数据读写方式,其不但缩短了宝贵的 CPU 周期,而且为提高块级设备读写性能提供了重新排列和融合多路 I/O 的机会.但是按位访问的 NVM 的高速 I/O 的能力,使得异步读写以及 I/O 调度所能获得的收益逐渐消失.针对这一趋势,文献[47]重新审视了用同步 I/O 代替异步 I/O 的可能性,分析指出:对于高速 NVM 设备,异步 I/O 中高优先级的中断不但会消耗大量的 CPU 资源、增加单请求响应延时,同时,中断引发的进程切换以及随之带来的缓存抖动等问题将极大地影响系统的整体性能,而基于轮询的同步 I/O 则能使 NVM 获得更高的收益.

面向 NVM 的同步 I/O 模式对于上层软件的设计也是值得借鉴的,虽然需要在软件层面进行一定的调整,因为诸如缓存和预取等一些经典的策略在同步 I/O 模式下将失去以往的作用.需要指出的是:即使是针对高速的 NVM 设备,当存在大规模数据读写以及较高的硬件延迟时,异步 I/O 访问模式依然是高效.因此,如何有效地混合异步和同步 I/O,是提高 NVM 存储环境下系统读写能力的有效途径,值得进一步探索.由于写延迟对于数据库事务提交的性能影响是非常显著的,因此上述研究对于上层软件系统的高效执行是非常重要的技术支撑.

2.1.4 新型混合存储结构的优化技术研究

直接替换方案是否真正可行,取决于未来量产的 NVM 能否在关键性能指标上全面地超越传统存储介质.因此,还有一些研究者提出了基于 NVM 与传统存储介质混合的存储构成技术方案.线性平行[27]和主从层级[48]是两种主要的混合方式:对于线性平行模式,DRAM 和 NVM 同时作为主存,统一编址;而对于主从层级模式,一般是 DRAM 作为 NVM 的缓存,相当于在原有的存储层级中新增了一层,进而可以扩展为多层.与混合存储组织方式相关的研究涉及数据分布[49]、数据迁移[50]、磨损均衡[51]、缓存管理[52]、元数据管理[53]等诸多技术.

这两种混合内存结构各有优劣,平行结构可以避免冗余,编程简单;而层次结构可根据不同层级的访问特征为冷热不同的数据提供更有效的访问,也可以利用缓存中的副本缩短数据访问时间.混合的技术思路为 NVM 和传统存储介质的融合在性能、使用寿命、能耗和容量上找到了平衡点.

未来会不会出现由 NVM 构筑的统一内外存存储环境,取决于 NVM 技术的最终发展趋势和产业化水平.其间,我们需要不断重新审视传统的针对数据访问行之有效的经典假设和技术,并设计全新的算法来适配新的存储环境,特别是像 I/O 密集型的数据库技术领域.

2.2 NVM非易失存储体系的访问接口研究

与如何将 NVM 融入存储体系密切相关的另一个重要问题就是:为了更好地利用非易失存储环境,系统要根据不同的存储架构方式为上层应用提供访问 NVM 的接口.

2.2.1 内存访问接口模式

在以内存访问模式暴露 NVM 的接口方式下,NVM 可以利用现有的内存总线或者专用总线直接与处理器交换数据.这种持久化内存抽象不但可以帮助程序员创建无需面向磁盘序列化的持久化内存数据对象,同时,直接访问模式也可以大幅减少存储访问延迟.虽然 NVM 具有非易失性,但是,如果没有合适机制的支持,当系统发生错误时,驻留在 NVM 中的数据结构将处于无效状态.

基于此,文献[54]设计了 Mnemosyne 编程接口来支持持久化内存的访问.Mnemosyne 提供了 3 种重要的关键服务:可动态申请或用 pstatic 关键字静态创建的持久化内存区域(persistent memory regions)、支持一致性更新的低阶持久化原语(persistence primitives)以及支持任意数据结构一致性原位更新的持久内存事务机制(durable memory transaction).利用这些服务,Mnemosyne 既提供了低阶的内存数据结构操作原语,又保证了高阶的事务语义,为上层用户提供了透明、灵活的非易失编程接口.

NV-heap[55]采用了和 Mnemosyne 非常类似的思想,实现了轻量级、鲁棒性的持久化内存抽象,支持用户级别的持久化对象的事务性更新.持久化内存抽象不但要能够应对易失内存抽象中常见的悬垂指针、重复释放、内存泄露以及加锁错误等问题,还需要面对由 NVM 带来的一些难以发现的新错误.例如,在易失内存中迁移持久化数据对象的指针,这种操作是不安全的,因为在程序结束后这些指针都将失效.为了在应用或者系统失效时仍然保证持久化数据结构的鲁棒性,NV-heap 提供了原子性垃圾回收、安全指针等一系列专门为 NVM 持久化内存抽象设计的保障机制.这些机制是 Mnemosyne 所不具备的,但因此而需付出的代价就是要求程序员必须在特定的对象框架下进行程序设计,同时需要对处理器进行一定的改进.

Mnemosyne 和 NV-heap 都利用内存映射文件实现 NVM 物理地址与用户程序逻辑地址的直接映射,但是这种持久化堆的管理方式存在几个难以回避的问题:一是同步核上和磁盘上存在的冗余元数据的开销比较大;二是需要依赖重量级的系统调用完成命名空间的管理.同时,持久化对象难以灵活地扩展.针对上述问题,HEAPO[56]专门为 NVM 设计了持久化堆布局、持久化对象格式、基于字典树的命名空间组织、对象共享和保护以及基于 undo 日志的容错等一系列机制,用本地堆的方式替换内存映射文件来管理持久化堆.

如图 2 所示,上述研究都是以持久化堆的方式来暴露 NVM 的访问,但其设计策略有所不同:有些通过增加专门的软件层来对持久化堆进行系统性的管理,有些则以库文件的形式嵌入到应用层面为持久化堆提供用户态的灵活控制.以堆使用 NVM 的技术可以成为构筑上层数据库事务处理的基本构件[57],因此,这些技术对于低级的事务性操作的支持程度对于数据库系统高层的事务语义会产生直接的影响.未来数据库软件也需要根据上述不同的策略,设计与之适配的技术手段.

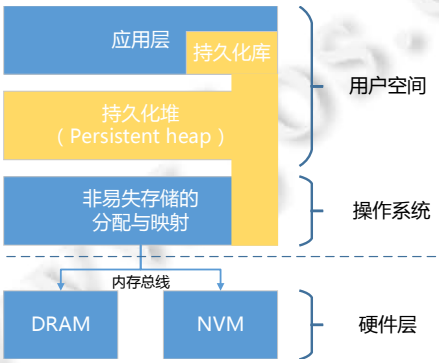


Fig.2 Architecture to exposure NVM as heap

图 2 面向持久化堆的不同系统架构

2.2.2 文件访问接口模式

以文件方式提供对 NVM 的访问支持是另一种常见的方式.在传统的块级文件系统中,数据在多个软硬层级之间移动和管理的开销(如块设备驱动、I/O 调度、多级缓存、逻辑卷管理)往往可以被慢速的磁盘读写所掩盖.但是,对于高速的 NVM 设备,这些以往占比不高的开销会被显著放大[58],因而有必要设计面向 NVM 的、以位驱动(byte-addressable)为主要访问模式的文件系统.PRAMFS[59],BPFS[60],SIMFS[61]等都是面向非易失存储的新型文件系统.

PRAMFS[59]是面向嵌入式设计的轻量级“可持续”“受保护”的文件系统,提供标准的文件接口访问 NVM 中的数据.其中,“可持续”的概念是指介质中的数据在系统重启或者断电后依然存在并且状态完整,PRAMFS 通过块级驱动直接将非易失的NVM挂载到基于磁盘的文件系统中.而所谓的“受保护”是指:文件系统只能通过只读的页表访问 NVM,该页表仅在写操作时被暂时标记为可写,而且在对文件块执行写操作时,会借助锁机制对该块进行写保护锁定.这种机制可以有效地保护文件系统不受系统内核错误的影响.此外,PRAMFS 支持类似 NOR Flash 的芯片内执行机制(execute in place,简称 XIP).利用片内执行机制,读写操作可以直接在 NVM 上执行,无需在用户空间和内核空间中进行数据传输.得益于 NVM 的高速并发 I/O 能力,在读写过程中,PRAMFS 采用的是非阻塞的同步 I/O 模式.但其读写性能(目前无法使内存带宽达到饱和)、扩展性以及一致性保证尚存在较大的优化空间.

BPFS[60]是为字节可寻址的非易失存储器设计的软硬结合的文件系统.BPFS 虽然是以文件系统的管理方式操作 NVM,但其本质上是将 NVM 直接连接在内存总线上,在访问路径上屏蔽了 I/O 控制器.利用 NVM 字节可寻址的特性支持 CPU 直接寻址,并利用 CPU 的 L1/L2 cache 为 NVM 提供缓存保证.这种方式不但节省了宝贵的内存空间,还有效提高了系统的整体读写性能.BPFS 采用了如图 3 所示的树形文件存储结构,该结构非常适合元数据 inode 的分配、回收以及存储空间管理,但是树形结构也使得系统在回溯 inode 时需要消耗较多的时间,多层级的中间节点也会浪费一定的存储空间.同时,借助 NVM 的字节可寻址以及原位更新的能力,BPFS 提出了“短路影子页(short-circuit shadow paging)”技术,支持对 NVM 细粒度、原子性的一致更新.“短路影子页”技术利用硬件支持的 8-byte 原子写以及 epoch barriers 机制,支持在文件树的任何层次上完成以较小的子树为粒度的修改提交操作,大幅减少了传统影子页所需要的开销.在具体实现上,BPFS 利用的是 PCM 与 DRAM 混合的存储架构,而原子性和顺序性的保障都依赖于特定硬件的支持,实现上具有一定的难度.面向 NVM 的文件系统在设计上除了要充分利用 NVM 的细粒度寻址以及原位更新能力外,还需要考虑频繁写操作对 NVM 寿命的影响,文献[62]通过差异更新的优化技术进一步减少了 BPFS 中“短路影子页”产生的写时拷贝数据块的数量,从文件系统层面延长了 NVM 的使用寿命.

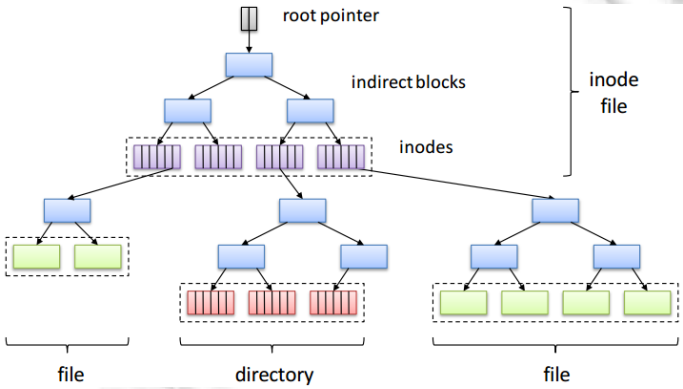


Fig.3 BPFS storage organization[60]

图 3 BPFS 的文件存储结构[60]

SIMFS[61]充分利用了虚拟内存管理机制的内存文件系统,它基于文件虚拟空间(file virtual space)的概念实

现了对 NVM 数据的高效访问.在 SIMFS 中,每个被访问的文件都拥有一个独立的连续虚拟地址空间.数据地址空间的连续性可以支持高速的顺序访问,而且独立的空间划分方式可以确保所有基于文件的访问都被隔离到当前文件的虚拟地址空间中,不存在地址冲突现象.在虚拟地址空间的概念下,SMFS 还提出了层级结构的文件页表(file page table)技术来组织文件数据.如图 4 所示:在 inode 节点中,每个文件都有一个与内存页表格式完全一致的、地址空间独立的文件页表.用相同的形式统一地组织文件地址空间与内存页表,有利于实现文件管理和内存管理的融合.同时,SIMFS 借助 MMU 实现了与文件大小无关的虚拟地址空间的高速物理地址映射.实验结果表明,SIMFS 的读写可以使内存带宽接近饱和.

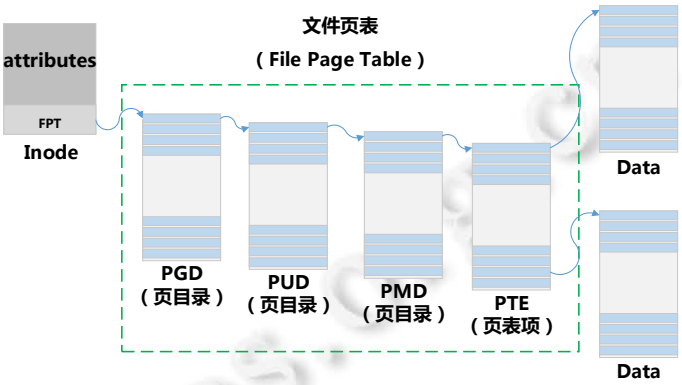


Fig.4 An illustration of file page table

图 4 文件页表组织方式

2.2.3 文件功能迁移的研究

通过文件系统暴露 NVM,更多的是从兼容性的角度来考虑如何在现有的 I/O 栈中使用 NVM.虽然可以利用现有文件系统的很多功能(如命名空间、访问控制、读写保护等)来保障 NVM 的数据读写,但是由此引入的性能损失不可忽视.文献[63]分析了支持 NVM 的 Moneta 文件系统在执行 4KB 的读写请求时,需要 8μs 的用户态与核心态的切换时间来实施文件权限检查,这部分时间占到了全部访问延迟的近 30%,并使得系统吞吐量下降了 85%.这些开销主要是因为在基于文件的 NVM 数据访问路径上,由于某些文件功能的需要,存在多次用户态与核心态的转换.能否将这些功能进行迁移,减少用户态与核心态的切换,进而减少读写开销是可行思路.目前,主要存在两种典型的技术途径:一是将部分文件功能上移到用户端,由用户进行直接的控制;另一种途径是将部分功能下移到存储器件内部,在器件的控制层完成处理.

出于安全上的考量,在文件系统中基于核心态对元数据进行集中访问,是 NVM 环境下文件开销放大的主要来源之一[64,65].基于此,文献[64,65]提出了用户态应用直接访问共享元数据(不经过文件系统)的技术框架 Aerie. Aerie 优化了与元数据相关的接口语义以及操作,降低了频繁内核访问与切换的开销.同时,为了减少互不授信的用户应用直接访问元数据可能导致的安全问题,系统采用了分散的元数据访问架构,通过未授信的库和授信的服务来支持对数据的一致、受保护以及同步的访问.用户应用可以使用未授信的 libFS 库直接访问 NVM 上的数据,只有在对元数据进行更新或者并发访问时才调用授信的服务 TFS,以日志的形式完成事务性的更新,并利用分层锁服务发放的租约对元数据的并发访问提供控制.这种分散设计的方式将一部分原本在文件系统中实现的操作迁移到了应用端,为应用提供了极大的灵活性.同时,这种将部分功能上移至应用端的方式还消减了文件系统中对于 NVM 不必要的开销,提高了系统的整体 I/O 性能.

在传统的 I/O 栈结构中,为了对存储设备提供访问保护,任何访问请求都会触发系统调用,这部分开销对于高速的 NVM 设备是难以摊销的.因此,Moneta-D[66]将访问权限控制功能下推到硬件层,消除了权限检查时频繁的系统调用需求以及出入内核的开销,操作系统和文件系统仅需完成访问保护和共享策略的设定.Moneta-D 提供了基于通道的(channel-based)I/O 能力,为每个用户端程序都提供了一条私有的虚拟化访问通道.对于大多数

不涉及修改元数据的访问,每个进程使用自己的通道在用户空间直接访问 NVM 设备,无需与操作系统和文件系统进行交互.仅有少量涉及元数据更改的访问仍然遵循传统的 I/O 栈访问层次.相比于将功能上移到应用端的实现方式,将部分功能下推到器件内部的方式具有更好的兼容性,因其对上层应用而言是透明的,应用无需任何修改就可以获取 NVM 带来的性能收益.未来,随着硬件技术的快速发展,存储器件将具备实施简单计算和逻辑控制的能力,因此,将一些简单功能下压到硬件层也符合目前智能化设备的发展趋势.

需要说明的是,上层应用,例如数据管理系统会直接受益于底层数据读写能力的提升.Moneta-D 的实验表明:对于简单的数据库,工作负载能够提升 2.6~5.7 倍的处理速度;而对于复杂的包含事务处理的数据库,工作负载也能获得 1.1~2.0 倍的性能提升.实验同时表明:如果将包含复杂事务处理的数据库不加修改地运行其上,数据库本身的缓冲机制则会降低事务处理的效率.因此,非常有必要从应用层面对 NVM 的使用进行更高层次的优化.

从目前的研究来看:无论是以内存堆的形式抽象 NVM,还是通过文件系统访问 NVM,都需要设计专门的技术才能充分发挥 NVM 的特性.而且从实现的角度来看,接入方式非常灵活,如图 5 所示,既可以通过兼容传统内存的方式利用总线和内存控制器与 CPU 相连,也可以通过兼容文件系统的方式利用 I/O 总线进行相连,甚至可以绕过 I/O 控制器直接与 CPU 进行访问.但在实现难度上存在显著差异,有的需要对既有硬件进行功能扩展或改进,有的需要新增专门的控制层,有的需要对操作系统核心进行相应的更改,有的还要求上层应用使用特定的框架和接口.这给上层数据管理软件的设计带来了机遇,也带来了挑战,未来是以透明兼容的方式使用 NVM,还是以感知与变革的方式利用 NVM,仍有待进一步研究.

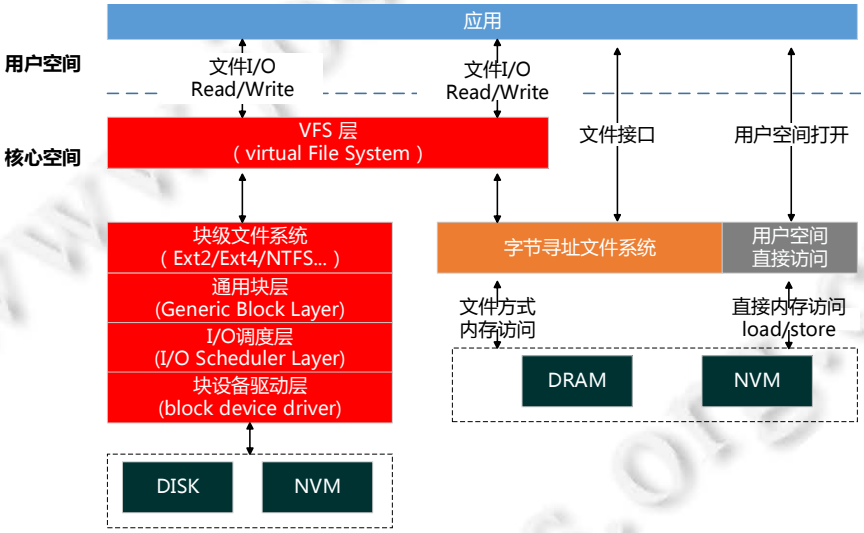


Fig.5 An organizational view of different kinds of interface[61]

图 5 不同的 NVM 接口组织形式[61]

2.3 面向NVM的模拟环境研究

虽然非易失存储器的相关技术发展非常迅速,但是多数研究都集中在设备级别,而系统级别的研究都面临着缺少真实可用的 NVM 硬件设备(或测试平台)的窘境.目前,工业级的 NVM 产品大多仅在嵌入式设备或者移动设备中得到一定的应用,因此,在 PC 或者服务器领域,大多数普通研究者不具备在真实 NVM 存储环境下验证上层软件技术有效性的条件.构建 NVM 的模拟器或模拟平台是解决现阶段“无的放矢”困难的有效技术手段,也对上层系统软件和技术的研究起着非常重要的支撑作用.

2.3.1 可改造的常规内存模拟器

通过改造现有的面向常规内存的模拟器以支持对 NVM 的模拟是一种可行的手段.具有代表性的常规内存

模拟器包括 gem5[67],DRAMsim2[68]等.gem5 是基于 M5[69]与 GEMS[70]紧耦合设计的全系统模拟器,采用模块化设计思想,除了可以对缓存、内存以及 I/O 设备进行模拟外,还可以模拟集成多种总线以及 CPU 模型的体系结构.此外,gem5 在存储模拟上还针对缓存容量、一致性协议、替换策略、互连模型等提供了非常灵活的参数配置能力.DRAMsim2 是一款具备高时钟精度的内存模拟器,可以准确刻画多种不同类型的内存硬件的复杂行为,包括 SDRAM,DDR,FBDIMM 等.同时,DRAMsim2 还具有易于集成的优点,适合作为全系列模拟器的基础构件.

主流的内存模拟器具有良好的可扩展和可配置能力,经过适当的改造,可使其在一定程度上具有模拟 NVM 特征、构筑 NVM-only 存储环境的能力.但是这种改进方式存在两个主要问题:一是此类模拟器很难通过改造获得构筑混合存储环境的能力;其次,此类针对 DRAM 设计的内存模拟器难以有效刻画 NVM 设备的一些重要特征,如读写不对称、有限的耐久性等.

2.3.2 NVM 模拟器

完全面向 NVM 设计的模拟器可以实现对 NVM-only 以及混合非易失存储环境的全面支持,是测试与验证 NVM 感知的上层技术的有效手段.PMEP[71],NVSim\NVSim-VX^S[72,73],NVMain2.0[74]等都是面向 NVM 设计的新型模拟器.

文献[71]指出:现有的部分 NVM 模拟器功能尚不完全,无法有效地对缓存替换、推测执行、内存级的并行以及预取等复杂的功能进行全面的模拟.为解决上述问题,该文献提出了利用 DRAM 模拟 NVM 的模拟器 PMEP. PMEP 将普通内存划分为常规的易失性内存以及模拟的非易失性存储,通过对带宽以及延迟等重要参数的设置,PMEP 可以模拟 NVM 的高并发、读写不对称等读写特征.由于 CPU 具有预取、推测执行等复杂的特性,因此,准确地刻画 NVM 的动态访问延迟是非常困难的事情.PMEP 基于内存访问延迟与末级缓存缺失(LLC miss)成比例的客观事实,借助 CPU 微码,通过监测和统计 LLC miss 实现对内存访问延迟的模拟.模拟的准确性依赖于监测的频率,但是,过高的监测频率又会显著地增加模拟开销,因此在实际使用中,需要根据需求进行合适的设置.

NVSim 是由宾夕法尼亚大学开发的一款电路级别的 NVM 模拟器,可以在性能、动态能耗、漏电功耗、硅面积等多个对于存储芯片设计非常重要的方面进行模拟,为 NVM 芯片的设计选型提供重要的参考和依据,目前可以对 PCM,ReRAM,FBDRAM,STT-RAM 以及单层的 NAND flash 提供支持.NVSim-VX^S 是 NVSim 的扩展,增强了对 STT-RAM 动态性的模拟.NVSim-VX^S 通过将显著影响 STT-RAM 读写性能的 COMS 电路和磁性隧道结构的参数可变性以及芯片的操作温度纳入建模指标,解决了以往确定性的模拟方式无法动态反映 STT-RAM 由于制程变化和热扰动引起的非确定性行为,并利用蒙特卡洛模拟方法进行了校准.

NVMain 2.0 特别关注了与能耗、耐久性、容错、多层单元(multi-level cells,简称 MLC)等参数指标密切相关的 NVM 建模问题.在能耗估计上,面向常规内存的模拟器依据操作电流 IDD 参数估算内存电源消耗,但在 NVM 模拟中,IDD 参数无从获取,因此,NVMain 2.0 提出一种新的能耗建模方式,允许从电路级别的模拟器,如 NVSim 中获取相应的能耗数值进行 NVM 能耗估算.同时,从 NVM 耐久性与容错的角度分析了数据编码形式对于拥有多层单元的 NVM 的重要性,建模了面向 MLC 的编码器.此外,NVMain 2.0 还实现了细粒度的刷新机制,支持 all-bank,bank-group,per-bank 这 3 种刷新模式.

3 面向大数据应用领域与硬件环境优化的事务技术

在数据管理技术的发展过程中,应用领域与硬件环境都在持续发生变化,数据管理系统面临的操作环境、数据类型以及技术需求随之不断更新.大数据应用与 NVM 的出现,是近年来推动数据管理系统发展的重要动力来源,而且两者产生的作用并不是正交的.应用需求的多样性和硬件条件的局限性,使得数据管理系统需要设计与之相应的技术来缓解两者之间的矛盾.但是随着 NVM 的出现,传统的存储体系结构和 I/O 模式与特征都在发生显著的变化,因此,我们既需要了解以往设计策略的演化背景,同时也需要审视在新硬件环境下满足大数据应用需求的技术的有效性.其中,事务处理技术是两者共同作用下最集中的体现点,既存在面向策略选择的独立性,又存在面向技术设计的相关性.

3.1 领域驱动的新型数据管理系统事务策略研究

在大数据背景下,数据产生的速度已经远远超越传统计算机系统的数据处理能力.确保强事务语义的集中式关系型数据库系统成为整个应用体系的性能瓶颈,在可扩展性、可伸缩性、容错性、实时处理、深度分析、数据类型表达能力等诸多方面逐渐难以满足大数据应用的多样化需求[75].围绕着在不同领域的大数据业务场景下是否放松或者提供更灵活的一致性要求,一些新的数据管理系统设计准则、概念和技术应运而生.数据管理的范式也由 SQL 转向 NoSQL[75],再变迁到 NewSQL[76,77].随之而来的大量新系统、新概念和新技术正在形成完整的数据管理新生态.在这场由大数据应用推动的数据管理研究的发展浪潮中,提供什么形态的事务处理、保持何种程度的一致性,始终是数据管理研究领域中的焦点所在.

3.1.1 NoSQL 系统的事务支持策略

众所周知:在分布式环境下,由于受两阶段提交协议的制约,传统的关系数据库系统为保证事务特性而难以对可扩展性和高效性提供良好的支持.但在 Web 2.0 时代,对高并发负载提供有效的支持成为底层数据库的主要设计目标,因此,牺牲对强一致性的保证,追求高可扩展性和高可用性的需求愈发强烈.随之,放松事务 ACID 属性、遵循 CAP 理论[78]和 BASE 原则[79]的 NoSQL 数据处理模式出现在研究人员的视野中.NoSQL 泛指那些非关系型、模式自由(schema-free)、不完全提供 ACID 事务语义、横向扩展(scale out)、分布式的数据管理系统.

键值型(key value)数据管理系统是最朴素的一类 NoSQL 系统,其共同的特点是采用哈希方式维护 key 到 value 的快速映射,具有高效的点查能力,但是对事务的支持比较有限.以 Redis[80]为例,其单线程的执行方式以及在回滚能力上的缺失,使其只能保证事务的一致性和隔离性,而不能满足原子性和持久性的要求.Google 公司的 BigTable[81]是具有代表性的基于列簇(column family)的 NoSQL 系统,它维护的是由行和带时间戳的列构成的多维稀疏排序表,该类系统仅能提供行级的事务支持,不支持跨行、跨表级别的事务.类似的 Cassandra[82]系统也只能保证写入操作在行级别的原子性,但是可以借助 ZooKeeper[83]等外部库提供的事务支持,在更高级别实现更大范围的读写操作并发控制.基于文档(document)的 NoSQL 系统仍然以 key-value 存储模型为基础,作为 value 存在的文档具有自述性,并利用循环嵌套的结构为数据建模提供了极大的灵活性.但其也只能在单文档级别上提供原子事务.

上述几类主流的 NoSQL 系统本质上提供的都是面向单 key 的有限事务.为了增强面向 key 的事务处理能力,G-Store[84]利用 key 群组(key group)的概念,借助一致性保证增强协议提供了面向多 key(multi-key)的事务处理语义,但是仍然存在不支持跨分组事务以及分组开销过高等局限性.Megastore[85]的优化思路则是将数据切分成不同的实体组(entity group),实体组的副本跨越数据中心存放,并在实体组内部提供完整的 ACID 保证,确保写操作可以在不同数据中心被同步复制.但是实体组之间只能支持有限的 ACID,无法支持数据的强一致性.此外,Percolator[86]借助快照隔离(snapshot isolation)语义实现了支持完全 ACID 的跨行及跨表事务,但是这种乐观锁的方式在事务语义上并不严谨,牺牲了一些可用性.

这些优化技术虽然扩大了事务的支持范围,从单 key 到多 key、从单行到多行,但其仍然存在很多的限制,无法实现跨表、跨组等任意范围的事务支持.因此,一些更复杂的事务需求只能依赖应用层实现跨域数据访问的原子性与一致性保证.随着事务支持范围的扩大,NoSQL 主要需要避免的是高昂的网络信息传输开销,磁盘 I/O 开销在其中并不是关键的瓶颈所在,即使在 NVM 非易失存储环境下,NoSQL 系统所能获得的性能收益也是有限的.因此,NoSQL 事务技术的研究重点应集中在设计灵活、弹性、多粒度的轻量级事务处理模型上.

3.1.2 NewSQL 系统的事务支持策略

NoSQL 虽然提供了良好的可扩展性,但其较弱的一致性使其应用范围受到了一定的限制.此外,由于其缺少对完整严格事务的支持,给上层应用的开发者也带来了棘手的问题,因为他们不得不在应用逻辑层对其所需要的一致性提供谨慎的保证.再者,对于某些领域,特别是处理关键数据的企业级 OLTP 核心应用,如金融领域的交易系统,其不但对可扩展性具有实际的需求,而且更不可能放弃对严格一致性提供保证的事务机制[87].如何构建可扩展、高性能且保持事务 ACID 属性的数据库,成为新的数据处理需求和发展的方向.随着对 NoSQL 运动和应用需求的反思,衍生出一种新的既提供高可扩展性又支持 ACID 事务的新数据库类型,这类数据库被归类为

NewSQL[77].VoltDB[88](源于 H-Store[12]),Spanner[89],CockroachDB[90],Rubato DB[91]等都是遵循 NewSQL 概念的新型数据管理系统.

H-Store 是最具代表性的 NewSQL 类学术原型系统,它使用存储过程模拟事务,将所有事务都转为基于存储过程的序列操作.这种执行方式不但杜绝了人为中断事务执行的可能性,还为服务器根据存储过程的实现逻辑进行事务执行优化提供了可能性.H-Store 中的表被划分成分布在多个站点上满足 K-safety 需求的分区(partition),每个分区都采取单线程的方式执行存储过程,控制事务以一定顺序执行,有效消除了传统数据管理系统在多线程环境下执行并发事务时所需要的 lock 和 latch 控制开销,同时保证了严格的 ACID 语义.H-Store 实现了可扩展与强事务语义的和谐一致,具有超越磁盘数据库近 60 倍的事务处理性能.但因受限于 procedure-based 的事务提交方式,事务执行的灵活性受到影响,系统无法支持 Adc-Hoc 的事务请求.同时,该方式还限制了并发事务的执行效率,在分布式事务增多的场景下,系统的吞吐量会显著下降.

Google 公司设计的 Spanner 是一个支持全球地理分布的、高可扩展的新型数据库,采用去中心化(decentralized)的事务协调协议.在去中心化的系统中,每个节点都需要负责维护本节点数据的事务访问状态,并与其他节点进行必要的交互以确定并发事务之间是否存在冲突,但是节点之间的协作需要依赖于某种时钟同步机制.Spanner 最具创新的关键技术是实现了以 TrueTime API 方式暴露的、基于硬件保障的时钟同步机制.借助 GPS 天线和高精度的原子钟,Spanner 可以为事务分配全球范围内有意义的提交时间戳.同时,基于时间戳的事务序列化次序可以满足系统对于外部一致性的要求,确保多个数据中心的状态不必经过复杂的通信就可以始终保持一致. Spanner 也是首个可以在全球范围内提供这种保证的数据库系统.

CockroachDB 是受 Spanner 启发而设计的一个支持高可伸缩、跨域复制以及完整 ACID 保障的开源分布式数据库系统.基于现实应用的考虑,CockroachDB 没有使用较为昂贵的原子钟技术(除非在服务器中添加专用的硬件,否则无法使用该技术),而是采用一种混合的时钟协议.该时钟协议主要依赖宽松的硬件同步时钟和逻辑计数器来保证线性一致化.但是目前项目还处于 beta 开发阶段,距离实际应用还有很多工作亟待完成.Rubato DB 则通过支持 ACID,BASE 以及 BASIC[92]这 3 种不同的一致性级别提供了更为灵活的事务语义,它利用基于 formula 的并发控制协议减少了系统维护多版本数据的内存开销.同时,它还支持在不违反串行化的情况下对事务的时间戳排序进行修改,提高了系统的并行度.

客观来说,上述 NewSQL 类系统在事务处理机制上,特别是在并发控制的处理上并没有实质性的创新[77],大多是对经典并发控制机制进行不同程度的更新,如,Rubato DB 本质上采用的是基于时间戳的多版本并发控制协议的一种变体;其次,现阶段 NewSQL 类系统更多的是将研究焦点放在 memory-oriented 的存储架构上,重点研究如何利用内存计算技术对 NewSQL 系统提供保障,例如解决内存容量无法满足数据库存储需求等 memory- oriented 的关键问题[93,94].但是,当数据管理系统迁移到 NVM 存储环境时,上述问题发生的可能性也许会随之降低甚至消失.然而从另一个角度来看,全新架构的 NewSQL 系统,如 H-Store,充分证明了面向新存储环境重新架构与设计系统的正确性和必要性.一方面,在完全 memory-oriented 的架构下,由于事务可能访问到不在内存中的数据而被迫暂停的假设已经不再适用,因此可以消除大量 disk-oriented 的缓存和并发控制等事务组件[4];另一方面,系统可以实现对底层存储的完全控制,这意味着系统可以利用自定义的引擎实现复杂的副本策略以及在处理节点中分发资源,特别是用分发计算代替分发数据,满足了大数据处理的实际需求.因此,这种面向存储重新架构的技术思路对于 NVM-oriented 的数据管理系统的设计也同样适用[95].

3.1.3 新环境下分布式事务理论与技术

虽然 NewSQL 类系统同时对可扩展性与严格事务语义提供了支持,但当系统涉及大量的分布式事务时,其性能还是会受到显著的影响.未来,即使在引入 NVM 构筑的新型非易失存储环境下,分布式事务依然是严峻的挑战.

在分布式事务处理的研究中,CAP 理论一直是能否构筑一致(consistency)、可用(availability)且分区容忍(network partition tolerance)系统的“魔咒”.其重要的假设是:网络分区在分布式环境下是难以避免的,因此在分区容忍必须被支持的前提下,系统只能在另外两个特性中根据应用需求选择其一.但是近年来,一些知名学者也

发出了不同的声音,Stonebraker 根据数据库系统的实际应用经验指出,牺牲“C”来换取“AP”并不可取.其原因在于:导致“P”问题发生的场景在实际中并不常见,而且在某些情况下(如硬件失效)牺牲“C”也难以保证“AP”[96].同时,随着硬件可靠性的提高,考虑到未来网络分区减少的情况,Abadi 也给出了一种新的权衡原理 PACELC[97](如图 6 所示),PACELC 将延迟因素“L”也作为系统设计时需要考量的关键指标.目前,也出现了一些运行在大规模集群中、在可接受延迟内提供良好分布式事务处理能力的系统,典型的如 Google 公司的 Spanner.在这些理论的支持下,系统根据应用的需求,需要在不同的隔离级别和高可用、高性能收益之间加以折中.客观来说,现有理论模型的研究尚不成熟,亟待进一步探索.但是近年来,符合新应用和硬件环境需求的分布式处理技术却取得了一定的进展,研究主要分为两类:分布式事务避免和分布式事务比例控制与优化.

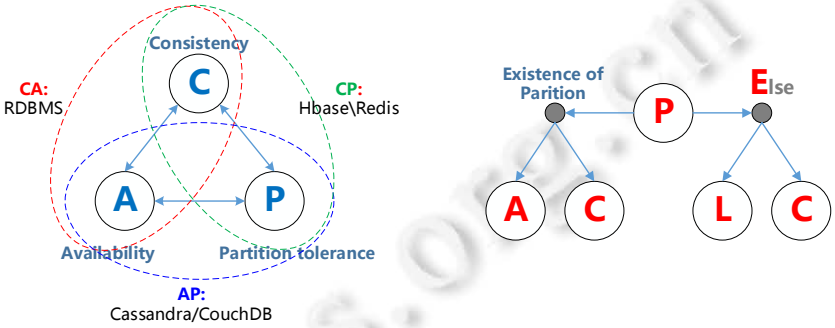


Fig.6 CAP and PACELC[96]

图 6 CAP 与 PACELC[96]

集中式事务处理与读写分离是两种常见的分布式事务避免手段.Oceanbase[98]系统根据电子商务的业务特点与数据访问模式,将处理事务的更新模块与非事务性的查询模块解耦,将事务处理完全迁移到集中的高性能节点中,有效地避免了分布式事务.文献[99]表明:对于很多实际应用而言,可以将应用在逻辑上进行分区隔离,用局限于分区内的强一致性事务来避免分布式事务.也有研究[100]将读写操作分离,利用不同的混合存储架构实现事务处理.文献[101]中思考与探索了一种更为激进的、完全不需要分布式事务支持的高伸缩性应用设计模式,其前提在于不允许在单个序列化范围中跨越一系列机器进行更新.但该设计仅存在于探讨设想之中,尚无实际的理论论证和原型系统支持.需要指出的是:对于集中式事务处理方式,往往会存在单点性能瓶颈,因为事务处理能力完全取决于集中处理事务节点的性能,缺乏 scale-out 的能力.而读写分离的架构也在数据同步、线上线下融合的复杂查询、可靠性等方面面临挑战.

分布式事务比例控制与优化方面的研究主要集中在负载划分、架构优化、事务协调与调度等几个方面.合理的数据分区能够有效地减少跨节点的分布式事务,在最理想的情况下,甚至可能在每个节点上都是 single-partition 事务[102].此外,通过分析并发访问的事务逻辑,有机会将同时被访问的数据划分在一起进而实现分布式事务最小化.但这类技术没有考虑被划分数据本身的特征,特别是当数据倾斜时,即使划分策略只产生最少的分布式事务,数据倾斜也可能会对系统性能产生显著影响.文献[103]设计了一种倾斜感知的数据划分模式 Horticulture,在最小化分布式事务的同时,减少了负载随时间变化的暂时性倾斜(temporal skew)对划分和访问的不利影响,实现了两者的平衡.但该方法主要针对的是 procedure-based 的数据库系统,适用性有限.

从数据库架构层面对分布式事务的比例进行控制,是更高层面的技术手段.NuoDB[104]采用异构的系统架构将节点分为事务引擎(transaction engine,简称 TE)和存储管理器(storage manager,简称 SM).在事务执行时,TE 会从其他 TE 以及 SM 中获取并缓存事务需要访问的所有数据,从而有效控制分布式事务的比例.类似地,MemSQL[105]也被设计成由聚集节点(aggregator nodes)和叶子节点(leaf nodes)构成的异构架构,它借助叶子节点的部分事务执行能力,有效地减少了分布式事务执行时数据在不同节点传输的开销,提高了事务执行效率.

在事务调度与协调方面,确定性事务技术[106−108]是内存环境下事务调度执行的重要研究方向.但是,确定性

事务技术虽然在强一致性保证、避免死锁和提交协议、高可扩展性和吞吐量等方面具有优点,同时也带来了事务执行灵活性差、并发能力低以及事务延迟高等副作用[109].此外,还有一些异步两阶段提交协议优化[110]、无锁的分布式事务处理[111]、支持事务的动态负载迁移[112]等技术用来提高在不同环境下分布式事务执行的灵活性与性能.

无论是新型的NoSQL或NewSQL数据管理系统,还是各种分布式事务机制,其发展本质上都离不开应用对事务的多样化需求以及系统自身架构性的制约.事务在一致性语义上强大的优点与在执行效率上的高消耗是难以取舍的权衡.但是,非易失存储环境的引入为提高事务执行效率带来的新的研究机遇,在非易失存储环境下是否会有新的范式出现,有待于我们进一步的探索和研究.

3.2 面向NVM优化的事务型数据管理技术研究

从计算机系统的角度来看,新型非易失性存储器只是在器件层对存储在其上的数据提供了原子性和持久性的保证,并不能满足数据库所需的高层事务语义.如何利用 NVM 的特性来优化高层软件系统,特别是数据库系统所需的严格事务语义还有待探索,相关研究工作刚刚展开,面向新型存储的 OLTP 数据库系统的架构设计以及事务的恢复与并发控制都是值得关注的研究重点.

3.2.1 面向非易失存储环境的事务型数据库架构研究

不论以何种方式将NVM融入现有的存储体系,其优异的读写能力和非易失性都会对数据库系统的架构设计带来直接的影响.长期以来,为了便于软件系统的开发,底层存储系统与上层数据管理系统大多采用相互透明的标准化设计模式,因此,disk-oritented 和 memory-oriented 数据库系统有可能在新的 NVM 存储环境下直接运行.与内存数据库发展的思路一样,研究人员首先希望探索上述两种数据架构中哪种更适合新型的非易失存储环境,以及是否有必要对数据库进行完全的重构.

文献[113]在这方面开展了具有前瞻性的研究探索.新型非易失存储环境的构成对于上层数据管理软件的设计具有决定性的影响,因此,其首先确定了两种未来可能会大规模应用的融入NVM的存储架构:NVM-only存储架构和 NVM+DRAM 混合存储架构.在 NVM-only 架构中,数据库系统的内外存环境都是基于 NVM 的,用drop-in的方式替换了传统的DRAM和磁盘.而在NVM+DRAM混合存储架构中,内存还是使用传统的DRAM,外存用 NVM 进行替换.在 NVM+DRAM 混合存储架构中,文献[113]的作者还采用了与传统缓存策略不同的anti-caching[93]技术.该技术将内存视为数据管理系统的主要存储用来保存“热”数据,而将磁盘作为辅助存储保存“冷”数据,并且相同的数据不会同时出现在两个存储介质上.该技术不但能够缓解有限内存空间与大容量数据之间的矛盾,还可以降低维护数据一致性的开销.因为缺少真实的硬件环境,该技术采用前文介绍的 PMEP[71]模拟器来进行存储环境仿真以及性能测试.为了使模拟环境更接近于现实应用场景,还特别针对模拟的 NVM 内存设计了支持多核环境的 NUMA 接口.同时,De Brabant 等人[113]选择 MySQL 和 H-Store 分别作为 disk-oritented 和 memory-oriented 数据库系统的代表.

通过在Read-Only,Read-Heavy以及Write-Heavy这3种不同的工作负载模式下进行实验,结果表明:无论是在NVM-Only架构还是在NVM+DRAM混合架构上,memory-oriented数据库的性能表现都优于disk-oritented数据库.特别是当负载的访问倾斜程度较高时,差异尤为明显.因为高负载倾斜会导致读写冲突增多,进而会在disk-oritented数据库中引发大量的锁开销,而 H-Stroe 由于采用基于存储过程的事务处理机制所以不存在锁开销.而当负载的访问倾斜程度降低时,memory-oriented数据库的访问性能开始下降,因为聚焦热点数据的访存局部性优势将不复存在,缓存缺失也随之上升,最终导致memory-oriented数据库与disk-oritented数据库在吞吐能力上趋于一致.

同时,实验还发现,NVM的延迟对两类系统的性能影响都比较小.这种现象说明,NVM的I/O已经不再是系统的主要性能瓶颈,基于 NVM 的数据管理,例如数据的组织、更新、置换等操作成为新的性能限制.系统恢复性能的实验结果也表明:无论是disk-oritented数据库采用的物理日志,还是memory-oriented数据库采用的逻辑日志,因为没有有效利用NVM的非易失特点,都存在大量不必要的操作.上述实验结果充分表明:目前没有任何一个数据库架构是适合所有不同负载访问模式的,也无法获得一致的性能优势.因此,自底向上设计全新的

NVM-oriented 数据库架构是非常必要的.目前,这方面的研究刚刚起步,CMU 的 N-Store(http://db.cs.cmu.edu/projects/n-store/)是该方向的探索性原型系统.

3.2.2 面向非易失存储环境的恢复技术研究

事务的恢复机制是保证事务原子性和持久性最重要的组件,也与底层存储环境的关系最为紧密,是目前的研究焦点.预写日志(write-ahead logging,简称 WAL)[11]是传统 disk-oriented 的数据库中主要的事务恢复方法,其中,ARIES[11]是最经典的 WAL 日志算法.ARIES-style 日志利用了磁盘原位更新的特点,借助日志缓冲区缓解了频繁随机 I/O 导致的高昂访问代价.借助持久化在磁盘上的日志,系统可以在发生故障时,根据日志中记录的事务更新状态来执行相应的 Redo 以及 Undo 操作.但在非易失存储环境下,由于 NVM 高速的随机读写能力,缓冲的优势变成了劣势.极端情况下,事务更新数据还会在不同位置(日志缓冲区、交换区、磁盘)出现大量冗余[114].此外,基于 NVM 的日志在写入之后就是持久化状态,因此,事务在提交时不再需要强制性地将日志持久化到磁盘中.这些改变都需要有面向 NVM 的日志技术进行积极应对.

在 NVM 的非易失存储环境下,读写特征的显著改变不但使以往的恢复机制难以发挥效率,同时也给开发面向 NVM-oriented 数据库的恢复技术提供了新的机遇.如前所述,NVM 融入存储体系的方式对上层数据管理系统的功能设计具有直接的影响.Pelley 等人[115]探索了在以 RAMDisk 的方式用 NVM 直接替换二级存储的存储环境下,事务恢复所面临的问题.文献[115]指出:虽然用低延迟的 NVM 替换磁盘可以使系统获得接近实时的恢复能力,但是复杂的 ARIES-style 日志带来的软件副作用却无法消除.基于此,Pelley 等人提出了一种新的面向 NVM 的组提交(NVRAM group commit)恢复机制.该机制消除了传统管理日志所需要的缓存区,利用批量事务处理模式减少了确保正确次序所需的写障栅次数,提供了较高的事务吞吐能力.MARS[114]则利用硬件原语实现了面向 NVM 作为磁盘时高度优化的 WAL 日志,将传统日志所需的日志序列号和检查点等要素下移到硬件层面维护,有效提高了事务吞吐量.

Fang 等人[57]探索了将 NVM 作为主存时,设计面向 NVM 日志技术的一些新思路.在内存层次,借助 NVM 的持久化能力,提出了一种新的面向 NVM 的日志架构,将 NVM 作为唯一的日志存储设备(磁盘仅作为存档设备).利用基于 NVM 的单一日志空间替换了传统日志所需的内存日志缓冲和磁盘日志文件 two-layer 结构,简化了恢复系统的设计,并有效解决了 NVM 环境下系统崩溃时特有的如空块检测、部分写等问题.这种通过合并日志缓存区和存储区的方式缩短了事务恢复所需的执行路径,减少了恢复延迟.PCMLogging[116]在混合 DRAM 和 NVM 的主存架构下提出了一种新的日志计划,该计划消除了显式的 Undo 和 Redo 日志,将隐式的日志合并入缓存的更新中,并同时保存在 NVM 上.利用这种融合缓存数据和日志记录的方式,有效消除了用 NVM 分离维护缓存数据与日志带来的数据冗余、写负载大、恢复延迟高等问题.同时,PCMLogging 特别关注了 NVM 的耐受性问题,利用异位更新(out-of-place)和冷数据迁移等技术,从上层软件层面设计了磨损平衡算法以延长 NVM 的使用寿命.此外,文献[117]基于主外存统一的 NVM-only 的存储架构设计了 3 种不同的存储引擎和恢复技术,并在基于硬件实现的模拟器上进行了全面的比较,为深刻理解 NVM 环境下不同存储引擎和恢复方法的特征提供了宝贵的经验.文献[118]从性价比的角度验证了:相比于在某个存储层级全面使用 NVM,仅在日志子系统中使用 NVM 能够获得更高的性价比.同时,如图 7 所示,也给出了在恢复技术中使用 NVM 的可能候选方式,不同的方式在易用性、效率、实现难易程度、兼容性、性价比等多个方面各有不同.

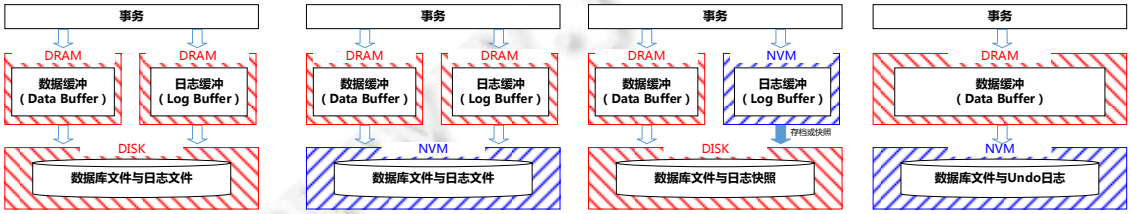


Fig.7 Candidate ways to use NVM in a transaction logging[118]

图 7 日志技术使用 NVM 的方式[118]

除了利用 NVM 的高速读写能力以及架构性优化提高事务恢复性能以外,部分研究还探索了在 NVM 环境下如何消除集中式日志产生的竞争开销,以进一步提高事务恢复性能.文献[119]分析了利用分布式日志技术缓解 NVM 环境下集中式日志瓶颈的可能性,提出了轻量级的被动式组提交协议(passive group commit),利用 NVM 以及 CPU 的合并写(write combining)与内存屏障(memory barriers)功能实现了在多核多通道环境下的面向 NVM 的分布式日志.该技术确保了在事务提交前,所有必要的分布日志记录的持久化,并保持了系统近乎线性的扩展能力.此外,文献[118]也提出了基于 per-transaction 的日志技术,实现了去中心化的处理.

从某种程度上来说,现有的面向 NVM 的日志技术基本上都是"权宜之计"[119],因为仅考虑了在存储体系的内存和外存中使用 NVM,但是数据在 CPU 的 cache 中还是易失的.因此,只有当整个存储体系,包括 CPU 的多级缓存都实现了基于 NVM 的非易失化,基于全非易失存储体系设计的日志可能才是"最终"的解决方案.因为 CPU 缓存的非易失化不但对上层系统是透明的,而且还能有效地提高整个存储层级中下层存储介质的寿命(其可以直接吸收大量的对于 NVM 而言非常敏感的频繁写操作).

还有一些诸如面向 NVM 的基于 MVCC 的无日志快速恢复机制[120]、最小化 NVM 写次序依赖的事务优化[121]等技术可以作为面向 NVM 的事务恢复技术的重要补充.因此,当 NVM 进入数据库的存储环境后,以往经典的各种技术有必要被重新审视,针对硬件特征设计恢复技术,是未来研究的必要条件.

3.2.3 面向新型存储环境的并发控制技术研究

数据库系统使用并发控制方法来避免同时执行的多个事务间的冲突,进而保障事务执行的隔离性.从不同并发控制机制表面的执行逻辑来看,似乎与底层存储是隔离的或者说是透明的.但在本质上,并发控制的具体实现以及其执行开销在系统开销中的比重,都与底层存储环境存在密切的关系.

以内存数据库为例,由于 DRAM 成为系统的主要存储介质,在此背景下,大多数事务的执行都不再需要磁盘的 I/O 操作[5],传统的磁盘与内存两层存储层级之间的最严重的 I/O 瓶颈被克服了,但同时也给锁机制带来了新的冲击.锁机制是数据库最重要的并发控制方法之一,其采用锁管理器[10]实现对锁表的管理,而锁管理器的操作都是内存操作,对传统 disk-oriented 数据库系统而言,锁管理器所需的内存开销比例很少.但在主存储环境是内存的背景下,锁管理器的开销比例显著增加,需要频繁操作内存的锁管理器成为新的瓶颈.文献[4]针对在单核机器上的操作分析得出:在内存数据库中,16%~25%的事务时间花费在锁管理器上.还有一些研究[122−124]表明:在多核处理器中,锁管理器所消耗的事务时间将会更长.因为多个 CPU Core 访问共享的锁管理器会带来一些冲突,进而使得锁管理器的开销更大.而一些从锁操作逻辑入手的优化技术[125,126]也无法消减锁管理器加解锁操作所需要的内存开销.因此,面向内存环境的并发控制方法成为一个研究热点.文献[127]在这种发展背景下提出了轻量级锁机制 VLL,通过把锁信息与对应的数据元组存储在一起,有效地减少了锁管理器操作内存的次数,降低了锁开销.文献[128]还进一步提出了选择性冲突分析的优化策略,保证了在冲突率较高的场景下,系统也具有良好的吞吐能力.

此外,基于 MVCC 的乐观锁也是并发控制的经典方法之一,它回避了悲观锁可能产生的死锁以及检测和处理死锁产生的不良影响.MVCC 利用每次更新产生新的数据版本解决了读写冲突的问题,同时可以保证不阻塞地读到一致的数据.例如,HANA[110]和微软的 Hekaton[9]使用的都是 MVCC 并发控制方法.但是 MVCC 机制下记录的更新操作会增大索引维护的开销,这种开销的主要来源就是低效的磁盘 I/O 以及索引与多版本记录物理存储位置的高度耦合关系.文献[129]利用低延迟的 SSD 引入了一个间接层,改变了在混合存储层次中的数据结构,利用间接层解耦了多版本记录在物理表示与逻辑表示之间的关系.当记录更新时,除了建立在更新属性上的索引需要涉及磁盘 I/O 外,其他属性上的索引都只会涉及低延迟的 SSD I/O.这种利用新型存储介质的读写优势、通过构筑新的存储层次缓解读写瓶颈的设计思路值得借鉴.

对于 NVM 构筑的新型非易失存储环境而言,其瓶颈在随着架构的读写特征而发生迁移时,以往并发控制机制的开销比例也在发生变化.相对于 NVM 读操作而言,NVM 写操作成为新的需要被关注的瓶颈,如何设计 write-limited 的 NVM friendly 的面向 MVCC 等机制的优化技术,以及重新评估确定性事务处理技术,都是值得探索的课题.

4 面临挑战与研究展望

作为新型的存储技术,NVM 虽然具有很多令人鼓舞的优良特性,但是并非简单地将面向磁盘或者面向内存的数据库系统直接迁移到 NVM 非易失存储环境下就能自动获得“免费的午餐”[113].主要有以下两个原因.

1) 在 NVM 非易失存储环境下,基于传统存储层级架构的数据库系统存在低效或无用的组件与技术.

此前,在数据库存储环境由磁盘向内存进行迁移时已有类似的结论.研究人员发现[4]:将面向磁盘的数据库系统裁剪成内存可完全驻留的版本后,在全内存执行环境下系统并没有获得预想的性能收益.其原因在于:除了约 12%的 CPU 时间花费在有用的工作上以外,其他的时间大量被用于缓冲、并发控制、恢复等与事务处理密切相关的辅助性工作上.类似的研究表明[113]:目前没有任何一种现有的 OLTP 数据库架构能在不同的负载特征下(如只读、读多写少、写多读少等),在 NVM 非易失存储环境下获得一致最佳的性能.

2) 在 NVM 非易失存储环境下,基于传统存储层级架构的数据库系统缺少有针对性的组件与技术.

如果数据库管理系统能够充分利用 NVM 优异的硬件特性以及新存储架构的特点,理论上,数据库在关键路径上操作的吞吐量和响应时间上都能获得极大的提升[130].当然,前提是数据库管理系统必须根据 NVM 非易失存储环境的特点重新设计合适的架构、组件、策略以及技术,以使 NVM 被充分而高效地利用.其中,与事务相关的技术又成为突破的关键所在,例如 NVM 环境下持久化的数据结构、恢复策略、原子操作原语等.而目前不论是 disk-oriented 亦或是 memory-oriented 数据库系统,都缺少相应的组件和优化技术来有效地发挥 NVM 非易失存储环境中的硬件优势[131].

未来,面向 NVM 的事务型数据管理技术的研究有可能从以下几个方面展开.

(1) 面向事务优化的非易失存储环境构成技术.

存储环境的构成方式对上层事务处理技术存在决定性的影响,在缺少真实 NVM 器件的情况下,研究面向 NVM 非易失存储环境的事务技术离不开 NVM 模拟器的支持.但是需要指出的是,即使是完全面向 NVM 进行设计的模拟器也存在很大的优化空间.目前,大多数模拟器仅能对带宽、延迟、能耗等有限的读写指标以及部分 NVM 介质的特征,如 STT-RAM 的非确定性行为进行调整,无法覆盖完整的 NVM 参数空间和全系列的 NVM 介质.此外,各种参数之间互相影响的关联特征以及混合存储的架构特征也没有被充分考量.这些问题对上层事务处理的设计决策都有重要的影响,因此有必要探索更有效的软硬共筑的 NVM 模拟环境.未来的研究可能包括:支持非易失性存储的高通用、高可定制的 NVM 模拟器;支持 NVM-only 以及 NVM-mix 存储架构的多模式访问接口;面向事务处理定制优化的 NVM 模拟器优化以及用户友好的参数可调的 NVM 模拟器策略库等技术.

(2) NVM 感知的负载划分与事务调度技术.

分布式 NVM 集群环境下,划分质量的好坏直接决定系统分布式事务的数量,并对系统的执行性能产生显著的影响.目前,很多研究从最大化系统并行度的角度探索了面向内存环境的划分技术,利用执行逻辑对并发访问的数据进行了访问模式的预测,并关注了运行时访问倾斜的问题,但是面向混合 NVM 存储环境的研究目前还鲜有文献涉及.同时,事务调度技术也是缓解分布式事务开销的有效手段,随着内存数据库的发展,确定性事务技术再次显现了一定的优势.这种在新环境下重新检视以往“不适宜”的技术的研究思路值得借鉴.未来的研究可能包括:NVM-mix 的混合非易失环境下的划分技术;NVM-only 单纯非易失环境下的划分技术;NVM 感知的事务控制模块的功能分割和解耦合技术;NVM 感知的冲突与非冲突事务的并行执行以及乐观调度策略等.

(3) NVM 非易失存储环境下的数据布局与一致性保证技术.

NVM 非易失存储环境下良好的布局策略可以有效地减少事务被暂停或终止的概率,现有的技术大多都是在面向磁盘或者面向主存环境进行设计和优化的,并不能完全适用于 NVM 环境,但是很多设计思想均可被借鉴.此外,系统不同层面的缓存机制可能会引入不一致的问题.目前,大量的研究已经从最底层的器件层面解决了 NVM 的一致性更新问题.同时,在较高的软件层面也通过面向 NVM 的文件系统(如 SCMFS,PMFS 等)或者 NVM 堆管理(NV-heaps 等)实现了一致性保证,但是从数据库事务语义层面的一致性维护技术还有待进一步展开.未来的研究可能涉及:在 two-tier 或者 three-tier 存储层级以及 NVM-only 或者 NVM-mix 的介质构成的新非易失性存储空间中的数据布局策略;研究保证高层事务语义的面向多级缓存结构的一致性保证技术以及面向

NVM 的缓存方法等.

(4) 面向 NVM 存储环境的恢复技术.

NVM 非易失存储环境中存储层次的架构与不同层次间存储介质的读写特征,对事务恢复技术而言都有最直接的影响.现有的 NVM-aware 的日志技术主要探索了 NVM 的持久化能力和读写能力对于事务恢复技术的影响,从减少 NVM 写延迟、重新架构面向 NVM 的日志 I/O 栈、存储层面的事务保证、性价比等方面进行了前期研究,但是这些研究对 NVM 的耐久性问题关注不足,仅有部分研究从磨损平衡的角度关注了写密集型日志技术对 NVM 寿命的影响.目前,虽然在 NVM 器件的控制层面以及操作系统层面实现了很多磨损平衡技术来减少不利于 NVM 寿命的频繁局部写操作,但是由于高层的数据管理系统能够获取更多的应用读写特征,因此有机会在更高的软件层在 NVM 的高速读写、非易失和写不耐受之间寻找更合适的设计平衡点.研究内容可能包括:面向 NVM 内存环境的关注写不耐受性的物理日志优化技术;面向 NVM 内存环境的逻辑日志优化技术;面向 NVM 非易失存储环境下的影子页技术;NVM 非易失存储环境下的日志数据存储技术等.

值得关注的是:NVM 的引入除了对事务产生直接影响外,还会给数据库的查询操作以及相关的算法和数据结构等带来显著的影响[132,133],探索和研究 NVM-aware 或者 Write-limited 的相关技术也是 NVM-oriented 数据库将要面临的挑战.此外,近年来随着现代处理器技术的高速发展和不断成熟,在多核环境下,事务技术受到了更深刻的影响[134],丰富的硬件上下文和维护一致性的同步开销成为突出的矛盾,因此也必须研究相应的事务优化技术[135]以适应硬件环境的变化.

5 结束语

伴随着存储介质的发展,数据库经历了面向磁盘、面向内存、面向闪存等多轮进化.在不远的未来,数据库运行的存储环境可能会逐渐转移到以 NVM 为代表的、具有非易失、低延迟、高随机读写能力的新型存储平台上.目前,学术界对 NVM 非易失存储器件本身的研究以及对基于磁盘和基于内存的数据库系统的事务处理技术的研究都相当充分,但却受限于缺少实际的 NVM 硬件环境,导致面向 NVM 的上层软件系统和技术,特别是数据库事务技术的研究非常有限.随着 NVM 在器件层面的不断发展和成熟,由 NVM 构筑的新型非易失存储体系将彻底改变传统计算机的存储支撑.可以预见:以此为契机,在 NVM 非易失存储环境下探索并研究 NVM 感知的事务处理技术与优化策略,将成为新型数据库系统研究的新浪潮.

References:

[1] Stonebraker M, Cetintemel U. “One size fits all”: An idea whose time has come and gone. In: Proc. of the 21st Int’l Conf. on Data Engineering (ICDE 2005). IEEE Computer Society, 2005. 2−11. [doi: 10.1109/icde.2005.1]

[2] 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 with English abstract). http://www.jos.org.cn/1000-9825/5103.htm [doi: 10.13328/j.cnki.jos.005103]

[3] Gray J. The transaction concept: Virtues and limitations (invited paper). In: Proc. of the 7th Int’l Conf. on Very Large Data Bases (VLDB’81). Cannes: VLDB Endowment, 1981. 144−154. http://www.eecs.harvard.edu/~margo/cs165/papers/gray-1981.pdf

[4] Harizopoulos S, Abadi DJ, Madden S, Stonebraker M. OLTP through the looking glass, and what we found there. In: Proc. of the 2008 ACM SIGMOD Int’l Conf. on Management of Data (SIGMOD 2008). Vancouver: ACM Press, 2008. 981−992. [doi: 10.1145/1376616.1376713]

[5] De Witt DJ, Katz RH, Olken F, Shapiro LD, Stonebraker MR, Wood DA. Implementation techniques for main memory database systems. In: Proc. of the 1984 ACM SIGMOD Int’l Conf. on Management of Data (SIGMOD’84). Boston: ACM Press, 1984. 1−8. [doi: 10.1145/602259.602261]

[6] Sikka V, Farber F, Goel A, Lehner W. SAP HANA: The evolution from a modern main-memory data platform to an enterprise application platform. Proc. of the VLDB Endowment, 2013,6(11):1184−1185. [doi: 10.14778/2536222.2536251]

[7] Kemper A, Neumann T. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In: Proc. of the IEEE Int’l Conf. on Data Engineering (ICDE 2011). IEEE Computer Society, 2011. 195−206. [doi: 10.1109/icde.2011.5767867]

[8] Team CT. In-Memory data management for consumer transactions the timesten approach. In: Proc. of the '99 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'99). Philadelphia: ACM Press, 1999. 528−529. [doi: 10.1145/304182.304244]

[9] 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 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2013). New York: ACM Press, 2013. 1243−1254. [doi: 10.1145/2463676.2463710]

[10] Astrahan MM, Blasgen MW, Chamberlin DD, Eswaran KP, Gray JN, Griffiths PP, King WF, Lorie RA, McJones PR, Mehl JW, Putzolu GR, Traiger IL, Wade BW, Watson V. System R: Relational approach to database management. ACM Trans. on Database Systems, 1976,1(2):97−137. [doi: 10.1145/320455.320457]

[11] Mohan C, Haderle D, Lindsay B, Pirahesh H, Schwarz P. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Trans. on Database Systems, 1992,17(1):94−162. [doi: 10.1145/128765.128770]

[12] Kallman R, Kimura H, Natkins J, Pavlo A, Rasin A, Zdonik S, Jones EPC, 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]

[13] Pisharath J, Choudhary A, Kandemir M. Energy management schemes for memory-resident database systems. In: Proc. of the 13th ACM Int'l Conf. on Information and Knowledge Management (CIKM 2004). Washington: ACM Press, 2004. 218−227. [doi: 10.1145/1031171.1031214]

[14] Lv Y, Cui B, He B, Chen X. Operation-Aware buffer management in flash-based systems. In: Proc. of the 2011 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2011). Athens: ACM Press, 2011. 13−24. [doi: 10.1145/1989323.1989326]

[15] Kang WH, Lee SW, Moon B, Kee YS, Oh M. Durable write cache in flash memory SSD for relational and NoSQL databases. In: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2014). Snowbird: ACM Press, 2014. 529−540. [doi: 10.1145/2588555.2595632]

[16] Agrawal D, Ganesan D, Sitaraman R, Diao Y, Singh S. Lazy-Adaptive tree: An optimized index structure for flash devices. Proc. of the VLDB Endowment, 2009,2(1):361−372. [doi: 10.14778/1687627.1687669]

[17] Athanassoulis M, Ailamaki A. BF-Tree: Approximate tree indexing. Proc. of the VLDB Endowment, 2014,7(14):1881−1892. [doi: 10.14778/2733085.2733094]

[18] Shah MA, Harizopoulos S, Wiener JL, Graefe G. Fast scans and joins using flash drives. In: Proc. of the 4th Int'l Workshop on Data Management on New Hardware (DaMoN 2008). Vancouver: ACM Press, 2008. 17−24. [doi: 10.1145/1457150.1457154]

[19] Tsirogiannis D, Harizopoulos S, Shah MA, Wiener JL, Graefe G. Query processing techniques for solid state drives. In: Proc. of the 2009 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2009). Providence: ACM Press, 2009. 59−72. [doi: 10.1145/1559845.1559854]

[20] Fang HW, Yeh MY, Kuo TW. MLC-Flash-Friendly logging and recovery for databases. In: Proc. of the 28th Annual ACM Symp. on Applied Computing (SAC 2013). Coimbra: ACM Press, 2013. 1541−1546. [doi: 10.1145/2480362.2480648]

[21] On ST, Xu J, Choi B, Hu H, He B. Flag commit: Supporting efficient transaction recovery in flash-based DBMSs. IEEE Trans. on Knowledge and Data Engineering, 2012,24(9):1624−1639. [doi: 10.1109/TKDE.2011.122]

[22] Chen TY, Chang YH, Ho CC, Chen SH. Enabling sub-blocks erase management to boost the performance of 3D NAND flash memory. In: Proc. of the 53rd Annual Design Automation Conf. (DAC 2016). Austin: ACM Press, 2016. 1−6. [doi: 10.1145/2897937.2898018]

[23] Lee SW, Moon B, Park C, Kim JM, Kim SW. A case for flash memory ssd in enterprise database applications. In: Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2008). Vancouver: ACM Press, 2008. 1075−1086. [doi: 10.1145/1376616.1376723]

[24] Wei Q, Chen J, Chen C. Accelerating file system metadata access with byte-addressable nonvolatile memory. Trans. on Storage, 2015,11(3):1−28. [doi: 10.1145/2766453]

[25] Xue CJ, Zhang Y, Chen Y, Sun G, Yang JJ, Li H. Emerging non-volatile memories: Opportunities and challenges. In: Proc. of the 7th IEEE/ACM/IFIP Int'l Conf. on Hardware/software Codesign and System Synthesis (CODES+ISSS 2011). Taipei: ACM Press, 2011. 325−334. [doi: 10.1145/2039370.2039420]

[26] Freitas RF. Storage class memory: Technology, systems and applications. In: Proc. of the 2009 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2009). Providence: ACM Press, 2009. 985−986. [doi: 10.1145/1559845.1559961]

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

[28] Mengjie M, Guangyu S, Yong L, Jones AK, Yiran C. Prefetching techniques for STT-RAM based last-level cache in CMP systems. In: Proc. of the 19th Asia and South Pacific Design Automation Conf. (ASP-DAC 2014). 2014. 67−72. [doi: 10.1109/ASPDAC.2014.6742868]

[29] Bishnoi R, Oboril F, Ebrahimi M, Tahoori MB. Avoiding unnecessary write operations in STT-MRAM for low power implementation. In: Proc. of the 15th Int'l Symp. on Quality Electronic Design (ISQED 2014). 2014. 548−553. [doi: 10.1109/ISQED.2014.6783375]

[30] Junwhan A, Sungjoo Y, Kiyoung C. DASCA: Dead write prediction assisted STT-RAM cache architecture. In: Proc. of the IEEE 20th Int'l Symp. on High Performance Computer Architecture (HPCA 2014). 2014. 25−36. [doi: 10.1109/HPCA.2014.6835944]

[31] Li J, Shi L, Li Q, Xue CJ, Chen Y, Xu Y. Cache coherence enabled adaptive refresh for volatile STT-RAM. In: Proc. of Design, Automation & Test in Europe Conf. & Exhibition (DATE 2013). 2013. 1247−1250. [doi: 10.7873/DATE.2013.258]

[32] Mittal S. A survey of architectural techniques for managing process variation. ACM Computing Surveys, 2016,48(4):1−29. [doi: 10.1145/2871167]

[33] Mittal S, Vetter J. A technique for improving lifetime of non-volatile caches using write-minimization. Journal of Low Power Electronics & Applications, 2016,6(1). [doi: 10.3390/jlpea6010001]

[34] Wang J, Dong X, Xie Y, Jouppi NP. Endurance-Aware cache line management for non-volatile caches. ACM Trans. on Architecture and Code Optimization, 2014,11(1):1−25. [doi: 10.1145/2579671]

[35] Mittal S, Vetter JS. EqualWrites: Reducing intra-set write variations for enhancing lifetime of non-volatile caches. IEEE Trans. on Very Large Scale Integration Systems, 2015. [doi: 10.1109/TVLSI.2015.2389113]

[36] Ahn J, Yoo S, Choi K. Write intensity prediction for energy-efficient non-volatile caches. In: Proc. of the IEEE Int'l Symp. on Low Power Electronics and Design (ISLPED 2013). 2013. 223−228. [doi: 10.1109/ISLPED.2013.6629298]

[37] Qureshi MK, Franceschini MM, Lastras-Monta LA. Improving read performance of phase change memories via write cancellation and write pausing. In: Proc. of the 2010 IEEE 16th Int'l Symp. on High Performance Computer Architecture (HPCA 2010). 2010. 1−11. [doi: 10.1109/HPCA.2010.5416645]

[38] Qureshi MK, Franceschini MM, Jagmohan A, Lastras LA. PreSET: Improving performance of phase change memories by exploiting asymmetry in write times. SIGARCH Computer Architecture News, 2012,40(3):380−391. [doi: 10.1145/2366231.2337203]

[39] Yue J, Zhu Y. Accelerating write by exploiting PCM asymmetries. In: Proc. of the IEEE 19th Int'l Symp. on High Performance Computer Architecture (HPCA 2013). 2013. 282−293. [doi: 10.1109/HPCA.2013.6522326]

[40] Jacobvitz AN, Calderbank R, Sorin DJ. Coset coding to extend the lifetime of memory. In: Proc. of the 2013 IEEE 19th Int'l Symp. on High Performance Computer Architecture (HPCA 2013). 2013. 222−233. [doi: 10.1109/HPCA.2013.6522321]

[41] Cho S, Lee H. Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance. In: Proc. of the 42nd Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO 2009). New York: ACM Press, 2009. 347−357. [doi: 10.1145/1669112.1669157]

[42] Yun J, Lee S, Yoo S. Bloom filter-based dynamic wear leveling for phase-change RAM. In: Proc. of the Conf. on Design, Automation and Test in Europe (DATE 2012). Dresden: EDA Consortium, 2012. 1513−1518. [doi: 10.1109/DATE.2012.6176713]

[43] Chen CH, Hsiu PC, Kuo TW, Yang CL, Wang CYM. Age-Based PCM wear leveling with nearly zero search cost. In: Proc. of the 49th Annual Design Automation Conf. (DAC 2012). San Francisco: ACM Press, 2012. 453−458. [doi: 10.1145/2228360.2228439]

[44] Sun G, Niu D, Ouyang J, Xie Y. A frequent-value based PRAM memory architecture. In: Proc. of the 16th Asia and South Pacific Design Automation Conf. (ASPDAC 2011). Yokohama: IEEE Press, 2011. 211−216. [doi: 10.1109/ASPDAC.2011.5722186]

[45] Rodriguez-Rodriguez R, Castro F, Chaver D, Pinuel L, Tirado F. Reducing writes in phase-change memory environments by using efficient cache replacement policies. In: Proc. of the Design, Automation & Test in Europe Conf. & Exhibition (DATE 2013). 2013. 93−96. [doi: 10.7873/DATE.2013.033]

[46] Awad A, Blagodurov S, Solihin Y. Write-Aware management of NVM-based memory extensions. In: Proc. of the 2016 Int'l Conf. on Supercomputing (ICS 2016). Istanbul: ACM Press, 2016. 1−12. [doi: 10.1145/2925426.2926284]

[47] Yang J, Minturn DB, Hady F. When poll is better than interrupt. In: Proc. of the 10th USENIX Conf. on File and Storage Technologies (FAST 2012). San Jose: USENIX Association, 2012. 3. https://www.usenix.org/legacy/events/fast12/tech/full_papers/Yang.pdf

[48] Dhiman G, Ayoub R, Rosing T. PDRAM: A hybrid PRAM and DRAM main memory system. In: Proc. of the Annual Design Automation Conf. (DAC 2009). San Francisco: ACM Press, 2009. 664−469. [doi: 10.1145/1629911.1630086]

[49] Kim D, Lee S, Chung J, Kim DH, Woo DH, Yoo S, Lee S. Hybrid DRAM/PRAM-based main memory for single-chip CPU/GPU. In: Proc. of the 49th Annual Design Automation Conf. (DAC 2012). San Francisco: ACM Press, 2012. 888−896. [doi: 10.1145/2228360.2228519]

[50] Yoon H, Meza J, Ausavarungnirun R, Harding RA, Mutlu O. Row buffer locality aware caching policies for hybrid memories. In: Proc. of the 2012 IEEE 30th Int'l Conf. on Computer Design (ICCD 2012). 2012. 337−344. [doi: 10.1109/ICCD.2012.6378661]

[51] Hu J, Zhuge Q, Xue CJ, Tseng WC, Sha EHM. Software enabled wear-leveling for hybrid PCM main memory on embedded systems. In: Proc. of the Design, Automation & Test in Europe Conf. & Exhibition (DATE 2013). 2013. 599−602. [doi: 10.7873/DATE.2013.131]

[52] 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 2011 IEEE 29th Int'l Conf. on Computer Design (ICCD 2011). 2011. 381−387. [doi: 10.1109/ICCD.2011.6081427]

[53] Meza J, Chang J, Yoon H, Mutlu O, Ranganathan P. Enabling efficient and scalable hybrid memories using fine-granularity DRAM cache management. Computer Architecture Letters, 2012,11(2):61−64. [doi: 10.1109/L-CA.2012.2]

[54] Volos H, Tack AJ, Swift MM. Mnemosyne: Lightweight persistent memory. ACM SIGPLAN Notices, 2011,47(4):91−104. [doi: 10.1145/ 2248487.1950379]

[55] 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. SIGARCH Computer Architecture News, 2011,39(1):105−118. [doi: 10.1145/1961295.1950380]

[56] Hwang T, Jung J, Won Y. HEAPO: Heap-Based persistent object store. Trans. on Storage, 2014,11(1):1−21. [doi: 10.1145/2629619]

[57] Fang R, Hsiao HI, He B, Mohan C, Wang Y. High performance database logging using storage class memory. In: Proc. of the IEEE 27th Int'l Conf. on Data Engineering (ICDE 2011). 2011. 1221−1231. [doi: 10.1109/ICDE.2011.5767918]

[58] Swanson S, Caulfield AM. Refactor, reduce, recycle: Restructuring the I/O stack for the future of storage. Computer, 2013,46(8):52−59. [doi: 10.1109/MC.2013.222]

[59] PRAMFS Team. Protected and persistent RAM file system. 2016. http://pRamfs.SourceForge.net

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

[61] Sha EHM, Chen X, Zhuge Q, Shi L, Jiang W. A new design of in-memory file system based on file virtual address framework. IEEE Trans. on Computers, 2016,65(10):2959−2972. [doi: 10.1109/TC.2016.2516019]

[62] Lee S, Kim J, Lee M, Lee H, Eom YI. Last block logging mechanism for improving performance and lifetime on SCM-based file system. In: Proc. of the Int'l Conf. on Ubiquitous Information Management and Communication (ICUIMC 2014). 2014. 1−4. [doi: 10.1145/2557977.2558014]

[63] Caulfield AM, De A, Coburn J, Mollow TI, Gupta RK, Swanson S. Moneta: A high-performance storage array architecture for next-generation, non-volatile memories. In: Proc. of the 2010 43rd Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO 2010). IEEE Computer Society, 2010. 385−395. [doi: 10.1109/micro.2010.33]

[64] Volos H, Panneerselvam S, Nalli S, Swift MM. Storage-Class memory needs flexible interfaces. In: Proc. of the 4th Asia-Pacific Workshop on Systems (APSys 2013). Singapore: ACM Press, 2013. 1−7. [doi: 10.1145/2500727.2500732]

[65] Volos H, Nalli S, Panneerselvam S, Varadarajan V, Saxena P, Swift MM. Aerie: Flexible file-system interfaces to storage-class memory. In: Proc. of the 9th European Conf. on Computer Systems (Eurosys 2014). Amsterdam: ACM Press, 2014. 1−14. [doi: 10.1145/2592798.2592810]

[66] Caulfield AM, Mollov TI, Eisner LA, De A, Coburn J, Swanson S. Providing safe, user space access to fast, solid state disks. SIGARCH Computer Architecture News, 2012,40(1):387−400. [doi: 10.1145/2189750.2151017]

[67] Binkert N, Beckmann B, Black G, Reinhardt SK, Saidi A, Basu A, Hestness J, Hower DR, Krishna T, Sardashti S, Sen R, Sewell K, Shoaib M, Vaish N, Hill MD, Wood DA. The gem5 simulator. SIGARCH Computer Architecture News, 2011,39(2):1−7. [doi: 10.1145/2024716.2024718]

[68] Rosenfeld P, Cooper-Balis E, Jacob B. DRAMSim2: A cycle accurate memory system simulator. IEEE Computer Architecture Letters, 2011,10(1):16−19. [doi: 10.1109/L-CA.2011.4]

[69] Binkert NL, Dreslinski RG, Hsu LR, Lim KT, Saidi AG, Reinhardt SK. The M5 simulator: modeling networked systems. IEEE Micro, 2006,26(4):52−60. [doi: 10.1109/MM.2006.82]

[70] Martin MMK, Sorin DJ, Beckmann BM, Marty MR, Xu M, Alameldeen AR, Moore KE, Hill MD, Wood DA. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. SIGARCH Computer Architecture News, 2005,33(4):92−99. [doi: 10.1145/1105734.1105747]

[71] Dulloor SR, Kumar S, Keshavamurthy A, Lantz P, Reddy D, Sankaran R, Jackson J. System software for persistent memory. In: Proc. of the 9th European Conf. on Computer Systems (Eurosys 2014). Amsterdam: ACM Press, 2014. 1−15. [doi: 10.1145/2592798.2592814]

[72] Dong X, Xu C, Xie Y, Jouppi NP. NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2012,31(7):994−1007. [doi: 10.1109/TCAD.2012.2185930]

[73] Eken E, Song L, Bayram I, Xu C, Wen W, Xie Y, Chen Y. NVSim-VXs: An improved NVSim for variation aware STT-RAM simulation. In: Proc. of the 53rd Annual Design Automation Conf. (DAC 2016). Austin: ACM Press, 2016. 1−6. [doi: 10.1145/2897937.2898053]

[74] Poremba M, Zhang T, Xie Y. NVMain 2.0: A user-friendly memory simulator to model (non-) volatile memory systems. IEEE Computer Architecture Letters, 2015,14(2):140−143. [doi: 10.1109/LCA.2015.2402435]

[75] Stonebraker M. SQL databases v. NoSQL databases. Communications of the ACM, 2010,53(4):10−11. [doi: 10.1145/1721654.1721659]

[76] Ozcan F, Tatbul N, Abadi DJ, Kornacker M, Mohan C, Ramasamy K, Wiener J. Are we experiencing a big data bubble? In: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2014). Snowbird: ACM Press, 2014. 1407−1408. [doi: 10.1145/2588555.2618215]

[77] Pavlo A, Aslett M. What's really new with NewSQL? SIGMOD Record, 2016,45(2):45−55. [doi: 10.1145/3003665.3003674]

[78] Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant Web services. SIGACT News, 2002,33(2):51−59. [doi: 10.1145/564585.564601]

[79] Pritchett D. BASE: An acid alternative. Queue, 2008,6(3):48−55. [doi: 10.1145/1394127.1394128]

[80] Ji Z, Ganchev I, Droma MO, Ding T. A distributed redis framework for use in the UCWW. In: Proc. of the 2014 Int'l Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC 2014). 2014. 241−244. [doi: 10.1109/CyberC.2014.50]

[81] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. ACM Trans. on Computer Systems, 2008,26(2):1−26. [doi: 10.1145/1365815.1365816]

[82] Lakshman A, Malik P. Cassandra: A decentralized structured storage system. ACM SIGOPS Operating Systems Review, 2010, 44(2):35−40. [doi: 10.1145/1773912.1773922]

[83] Hunt P, Konar M, Junqueira FP, Reed B. ZooKeeper: Wait-Free coordination for internet-scale systems. In: Proc. of the USENIX Conf. on USENIX Annual Technical Conf. (USENIXATC 2010). Boston: USENIX Association, 2010. 11−11. https://www.usenix.org/legacy/event/atc10/tech/full_papers/Hunt.pdf

[84] Das S, Agrawal D, Abbadi AE. G-Store: A scalable data store for transactional multi key access in the cloud. In: Proc. of the 1st ACM Symp. on Cloud Computing (SoCC 2010). Indianapolis: ACM Press, 2010. 163−174. [doi: 10.1145/1807128.1807157]

[85] Baker J, Bond C, Corbett JC, Furman J, Khorlin A, Larson J, Leon JM, Li Y, Lloyd A, Yushprakh V. Megastore: Providing scalable, highly available storage for interactive services. In: Proc. of the Conf. on Innovative Data Systems Research (CIDR 2013). 2011. 223−234. http://cidrdb.org/cidr2011/Papers/CIDR11_Paper32.pdf

[86] Peng D, Dabek F. Large-Scale incremental processing using distributed transactions and notifications. In: Proc. of the USENIX Conf. on Operating Systems Design and Implementation (OSDI 2010). Vancouver: USENIX Association, 2010. 1−15. https://www.usenix.org/legacy/event/osdi10/tech/full_papers/Peng.pdf

[87] Krueger J, Kim C, Grund M, Satish N, Schwalb D, Chhugani J, Plattner H, Dubey P, Zeier A. Fast updates on read-optimized databases using multi-core CPUs. Proc. of the VLDB Endowment, 2011,5(1):61−72. [doi: 10.14778/2047485.2047491]

[88] VoltDB Team. VoltDB. 2016. https://www.voltdb.com/

[89] Corbett JC, Dean J, Epstein M, Fikes A, Frost C, Furman JJ, Ghemawat S, Gubarev A, Heiser C, Hochschild P, Hsieh W, Kanthak S, Kogan E, Li H, Lloyd A, Melnik S, Mwaura D, Nagle D, Quinlan S, Rao R, Rolig L, Saito Y, Szymaniak M, Taylor C, Wang R, Woodford D. Spanner: Google's globally-distributed database. In: Proc. of the 10th USENIX Conf. on Operating Systems Design and Implementation (OSDI 2012). Hollywood: USENIX Association, 2012. 251−264. https://www.usenix.org/system/files/conference/osdi12/osdi12-final-16.pdf

[90] CockroachDB Team. CockroachDB. 2016. https://www.cockroachlabs.com/

[91] Yuan LY, Wu L, You JH, Chi Y. A demonstration of rubato DB: A highly scalable newSQL database system for OLTP and big data applications. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2015). Melbourne: ACM Press, 2015. 907−912. [doi: 10.1145/2723372.2735380]

[92] Wu L, Yuan LY, You JH. BASIC: An alternative to BASE for large-scale data management system. In: Proc. of the IEEE Int'l Conf. on Big Data (IEEE Big Data 2014). 2014. 5−14. [doi: 10.1109/BigData.2014.7004206]

[93] DeBrabant J, Pavlo A, Tu S, Stonebraker M, Zdonik S. Anti-Caching: A new approach to database management system architecture. Proc. of the VLDB Endowment (VLDB 2013), 2013,6(14):1942−1953. [doi: 10.14778/2556549.2556575]

[94] Stoica R, Ailamaki A. Enabling efficient OS paging for main-memory OLTP databases. In: Proc. of the 9th Int'l Workshop on Data Management on New Hardware (DaMoN 2013). New York: ACM Press, 2013. 1−7. [doi: 10.1145/2485278.2485285]

[95] Pavlo A. Emerging hardware trends in large-scale transaction processing. IEEE Internet Computing, 2015,19(3):68−71. [doi: 10.1109/MIC.2015.59]

[96] Stonebraker M. Errors in database systems, eventual consistency, and the CAP theorem. 2010. http://cacm.acm.org/blogs/blog-cacm/83396-errors-in-database-systems-eventual-consistency-and-the-cap-theorem/fulltext.

[97] Abadi D. Consistency tradeoffs in modern distributed database system design: CAP is only part of the story. Computer, 2012,45(2):37−42. [doi: 10.1109/mc.2012.33]

[98] Zhenkun Y, Chuanhui Y, Zhen L. OceanBase—A massive structured data storage management system. E-science Technology & Application, 2013,4(1):41−48.

[99] Bailis P, Davidson A, Fekete A, Ghodsi A, Hellerstein JM, Stoica I. Highly available transactions: Virtues and limitations. Proc. of the VLDB Endowment, 2013,7(3):181−192. [doi: 10.14778/2732232.2732237]

[100] Campos AF, Esteves S, Veiga L. HBase++: Extending HBase with client-centric consistency guarantees for geo-replication. 2013. http://www.gsd.inesc-id.pt/~sesteves/papers/inforum14-hbase-plus-plus.pdf

[101] Helland P. Life beyond distributed transactions: An apostate's opinion. In: Proc.of the Biennial Conf. on Innovative DataSystems Research (CIDR 2007). Asilomar, 2007. 132−141. http://cidrdb.org/cidr2007/papers/cidr07p15.pdf

[102] Stonebraker M, Madden S, Abadi DJ, Harizopoulos S, Hachem N, Helland P. The end of an architectural era (it's time for a complete rewrite). In: Proc. of the 33rd Int'l Conf. on Very Large Data Bases (VLDB 2007). Vienna: VLDB Endowment, 2007. 1150−1160. http://www.cs.yale.edu/homes/dna/vldb07hstore.pdf

[103] Pavlo A, Curino C, Zdonik S. Skew-Aware automatic database partitioning in shared-nothing, parallel OLTP systems. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2012). Scottsdale: ACM Press, 2012. 61−72. [doi: 10.1145/2213836.2213844]

[104] NuoDB Team. NuoDB. 2016. http://www.nuodb.com/

[105] MemSQL Team. MemSQL. 2016. http://www.memsql.com/

[106] Kemme B, Alonso G. Don't be lazy, be consistent: Postgres-R, a new way to implement database replication. In: Proc. of the 26th Int'l Conf. on Very Large Data Bases (VLDB 2000). Morgan Kaufmann Publishers, 2000. 134−143. https://static.aminer.org/pdf/PDF/000/642/954/don_t_be_lazy_be_consistent_postgres_r_a_new.pdf

[107] Jiminez-Peris R, Patino-Martinez M, Arevalo S. Deterministic scheduling for transactional multithreaded replicas. In: Proc. of the 19th IEEE Symp. on Reliable Distributed Systems (SRDS 2000). 2000. 164−173. [doi: 10.1109/RELDI.2000.885404]

[108] Thomson A, Diamond T, Weng SC, Ren K, Shao P, Abadi DJ. Calvin: Fast distributed transactions for partitioned database systems. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2012). Scottsdale: ACM Press, 2012. 1−12. [doi: 10.1145/2213836.2213838]

[109] Ren K, Thomson A, Abadi DJ. An evaluation of the advantages and disadvantages of deterministic database systems. Proc. of the VLDB Endowment, 2014,7(10):821−832. [doi: 10.14778/2732951.2732955]

[110] Sikka V, Farber F, Goel A, Lehner W. SAP HANA: The evolution from a modern main-memory data platform to an enterprise application platform. Proc. of the VLDB Endowment, 2013,6(11):1184−1185. [doi: 10.14778/2536222.2536251]

[111] Ferro DG, Junqueira F, Kelly I, Reed B, Yabandeh M. Omid: Lock-Free transactional support for distributed data stores. In: Proc. of the 2014 IEEE 30th Int'l Conf. on Data Engineering (ICDE 2014). 2014. 676−687. [doi: 10.1109/ICDE.2014.6816691]

[112] Elmore AJ, Arora V, Taft R, Pavlo A, Agrawal D, Abbadi AE. Squall: Fine-Grained live reconfiguration for partitioned main memory databases. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2015). Melbourne: ACM Press, 2015. 299−313. [doi: 10.1145/2723372.2723726]

[113] DeBrabant J, Arulraj J, Pavlo A, Stonebraker M, Zdonik SB, Dulloor S. A prolegomenon on OLTP database systems for non-volatile memory. In: Proc.of the 5th Int'l Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures(ADMS). Hangzhou, 2014. 57−63. http://adms-conf.org/2014/adms14_debrabant.pdf

[114] Coburn J, Bunker T, Schwarz M, Gupta R, Swanson S. From ARIES to MARS: Transaction support for next-generation, solid-state drives. In: Proc. of the 24th ACM Symp. on Operating Systems Principles (SOSP 2013). Farminton: ACM Press, 2013. 197−212. [doi: 10.1145/2517349.2522724]

[115] Pelley S, Wenisch TF, Gold BT, Bridge B. Storage management in the NVRAM era. Proc. of the VLDB Endowment, 2013,7(2): 121−132. [doi: 10.14778/2732228.2732231]

[116] Gao S, Xu J, Haerder T, He B, Choi B, Hu H. PCMLogging: Optimizing transaction logging and recovery performance with PCM. IEEE Trans. on Knowledge & Data Engineering, 2015,27(12):3332−3346. [doi: 10.1109/TKDE.2015.2453154]

[117] Arulraj J, Pavlo A, Dulloor SR. Let's talk about storage & recovery methods for non-volatile memory database systems. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2015). Melbourne: ACM Press, 2015. 707−722. [doi: 10.1145/2723372.2749441]

[118] Huang J, Schwan K, Qureshi MK. NVRAM-Aware logging in transaction systems. Proc. of the VLDB Endowment, 2014,8(4): 389−400. [doi: 10.14778/2735496.2735502]

[119] Wang T, Johnson R. Scalable logging through emerging non-volatile memory. Proc. of the VLDB Endowment, 2014,7(10): 865−876. [doi: 10.14778/2732951.2732960]

[120] Oukid I, Booss D, Lehner W, Bumbulis P, Willhalm T. SOFORT: A hybrid SCM-DRAM storage engine for fast data recovery. In: Proc. of the 10th Int'l Workshop on Data Management on New Hardware (DaMoN 2014). Snowbird: ACM Press, 2014. 1−7. [doi: 10.1145/2619228.2619236]

[121] Kolli A, Pelley S, Saidi A, Chen PM, Wenisch TF. High-Performance transactions for persistent memories. SIGARCH Computer Architecture News, 2016,44(2):399−411. [doi: 10.1145/2980024.2872381]

[122] Larson PK, Blanas S, Diaconu C, Freedman C, Patel JM, Zwilling M. High-Performance concurrency control mechanisms for main-memory databases. Proc. of the VLDB Endowment, 2011,5(4):298−309. [doi: 10.14778/2095686.2095689]

[123] Johnson R, Pandis I, Ailamaki A. Improving OLTP scalability using speculative lock inheritance. Proc. of the VLDB Endowment, 2009,2(1):479−489. [doi: 10.14778/1687627.1687682]

[124] Pandis I, Johnson R, Hardavellas N, Ailamaki A. Data-Oriented transaction execution. Proc. of the VLDB Endowment, 2010,3(1-2): 928−939. [doi: 10.14778/1920841.1920959]

[125] Atkins MS, Coady MY. Adaptable concurrency control for atomic data types. ACM Trans. on Computer Systems, 1992,10(3): 190−225. [doi: 10.1145/146937.146939]

[126] Joshi AM. Adaptive locking strategies in a multi-node data sharing environment. In: Proc. of the 17th Int'l Conf. on Very Large Data Bases (VLDB'91). Morgan Kaufmann Publishers, 1991. 181−191. http://vldb.org/conf/1991/P181.PDF

[127] Ren K, Thomson A, Abadi DJ. Lightweight locking for main memory database systems. Proc. of the VLDB Endowment, 2012,6(2): 145−156. [doi: 10.14778/2535568.2448947]

[128] Ren K, Thomson A, Abadi DJ. VLL: A lock manager redesign for main memory database systems. The VLDB Journal, 2015,24(5): 681−705. [doi: 10.1007/s00778-014-0377-7]

[129] Sadoghi M, Ross KA, Canim M, Bhattacharjee B. Making updates disk-I/O friendly using SSDs. Proc. of the VLDB Endowment, 2013,6(11):997−1008. [doi: 10.14778/2536222.2536226]

[130] Ailamaki A. Databases and hardware: The beginning and sequel of a beautiful friendship. Proc. of the VLDB Endowment, 2015, 8(12):2058−2061. [doi: 10.14778/2824032.2824142]

[131] Viglas SD. Data management in non-volatile memory. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2015). Melbourne: ACM Press, 2015. 1707−1711. [doi: 10.1145/2723372.2731082]

[132] Viglas SD. Write-Limited sorts and joins for persistent memory. Proc. of the VLDB Endowment, 2014,7(5):413−424. [doi: 10.14778/2732269.2732277]

[133] Chen S, Gibbons PB, Nath S. Rethinking database algorithms for phase change memory. In: Proc. of the Conf. on Innovative Data Sysytems Research (CIDR 2011). 2011. 21−31. http://cidrdb.org/cidr2011/Papers/CIDR11_Paper3.pdf

[134] Ren K, Faleiro JM, Abadi DJ. Design principles for scaling multi-core OLTP under high contention. In: Proc. of the 2016 Int'l Conf. on Management of Data (SIGMOD 2016). San Francisco: ACM Press, 2016. 1583−1598. [doi: 10.1145/2882903.2882958]

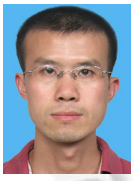
[135] Zhu YA, Zhou X, Zjang YS. A survey of optimization methods for transactional database in multi-core era. Chinese Journal of Computers, 2015,38(9):1865−1879 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2015.01865]

附中文参考文献:

[2] 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 [doi: 10.13328/j.cnki.jos.005103]

[135] 朱阅岸,周烜,张延松,周明,牛嘉,王珊.多核处理器下事务型数据库性能优化技术综述.计算机学报,2015,38(9):1865−1879. [doi: 10.11897/SP.J.1016.2015.01865]

潘巍(1977－),男,安徽蚌埠人,博士,副教授,CCF 专业会员,主要研究领域为数据库理论与技术,海量数据管理,内存计算.



周陈超(1982－),男,博士,工程师,CCF 学生会员,主要研究领域为知识发现,知识管理.

李战怀(1961－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论与技术,海量数据存储与管理.

苏静(1987－),女,博士,主要研究领域为大数据处理技术,图数据处理.

杜洪涛(1978－),男,博士,讲师,主要研究领域为海量数据管理,分布式数据库.