面向新型非易失存储的文件系统研究综述

王继昂 [1)]

[1)](华中科技大学 计算机科学与技术学院 2205 班 M202273733)

摘 要 新型非易失存储 (NVM) 可字节寻址，具有近似内存的低延迟特性以及外存的非易失性，受限于软硬件技术成熟度，目前首先被用于外存。讨论了 NVM 用于持久性外存所面临的一系列问题，以及管理上的一些挑战; 对现有的典型 NVM 文件系统及其主要特性进行了梳理。归纳起来，这些特性主要围绕降低一致性开销、降低软件栈开销、内存与外存的融合、分布式文件系统、NVM 文件系统安全、容错、空间管理几个方面展开。最后，展望了 NVM 文件系统仍然有待探讨的几个研究方向，包括扩展性问题、虚拟内存与文件系统的有机融合以及分布式文件系统等。

关键词 非易失存储; 文件系统; 虚拟内存; 持久性外存; 远程直接内存访问

中图法分类号 TP DOI **号**: * 投稿时不提供 DOI 号

1 引言

存储和计算是计算机系统的两大子系统。在搜索引擎、社交网络、电子商务等主流应用中，I/O 和访存已经占据了很大比重。一些数据密集型应用中，无论是内存容量还是外存带宽和延迟，已经难以满足计算需求，成为制约计算机系统性能提升的主要瓶颈。近年来，各种新型非易失存储 (non-volatile memory，NVM) 器件相继出现，如相变存储器 (PCM) 、自旋转移力矩随机存储器 (STT-RAM) 、电阻随机存储器 (RRAM) 等，给存储系统的设计带来前所未有的机遇。它们可字节寻址，且同时具有与现有内存相当的性能和外存的非易失性，因此不仅可用作为工作内存 (working memory) 以缓解现有内存容量扩展能力不足的问题，也可用做持久性外存 (persistent storage) 。因为 NVM 同时具有内存的字节可寻址和外存的非易失等两类存储介质的双重特性，所以也称之为持久性内存 (persistent memory) 或新型 NVM。所谓新型 NVM，是为了与早期文献中提到的闪存区别开来，后者也归为 NVM，但属于块设备。理论上，新型 NVM 可用于计算机系统的所有存储层次，如图 1 所示。但从产品而言，目前已量产的只有 Intel 的傲腾持久性内存和固态硬盘 (SSD) 两种，由于 NVM 存在写延迟和功耗高、写入次数有限以及价格因素等问题，短期内还很难作为商用的工作内存使用，而固态硬盘与传统硬盘相比优势明显，所以在电商、金融等延迟敏感的 I/O 密集型应用中有很大的市场潜力。本文重点讨论 NVM 用做持久性外存 (即外存) 时面临的诸多挑战、研究现状及未来的方向。众所周知，现有的操作系统都是针对二级存储模型设计的，即工作内存和持久性外存，通过虚拟内存和文件系统两个子模块分别对它们进行管理。与传统的硬盘相比，NVM 有两个根本的变化，一是可字节寻址，二是延迟有几个数量级的下降。因此，尽管可以沿用现有的面向块设备的文件系统，但软件栈开销所占比重太大，难以发挥 NVM 的低延迟优势。围绕如何使用 NVM 的持久性，学术界和工业界已进行了大量探索，大致有两个方向，即使用持久性堆 [1~3] 的内存管理模式和使用文件系统 [4~10] 的外存管理模式。持久性堆的管理模式涉及到编程模型、编译器、操作系统等整个软件生态的重构，工程浩大。短期来看，在兼容传统文件系统编程接口的基础上构建轻量级的 NVM 文件系统是较为现实的选择，因此最近几年，面向 NVM 的文件系统研究得到了极大关注 [11, 12]。随着器件技术的不断成熟，应用单一 NVM 构建统一的单级存储结构，实现内存与外存的最终融合已理论上可行。单级存储结构彻底消除了数据在内存与外存之间的流动，从而显著降低了传输延迟和功耗 [13] 。内/外存融合后，现在的操作系统中分别管理内存和外存的内存管理

和文件系统两个功能模块将合二为一，现在的内存和外存两个完全不同物理存储介质也合二为一。然而要实现内存与外存的真正融合，需要重构整个软件栈，这是一个系统性工程，但由于缺少工业界的参与，进展缓慢，但毫无疑问内外存融合是终极目标。因此，本文着重探讨 NVM 出现以后文件系统如何设计，重点聚焦 NVM 文件系统中最核心部分———如何降低一致性开销和如何降低软件栈开销等内容上。

2 NVM 文件系统的几个关键问题

1) **降低一致性保证开销**

数据一致性是文件系统的基本功能，也是核心功能，没有数据一致性的保证，程序的正确性也将得不到保证。持久性数据的一致性有两层含义: a) 持久性，即保证在掉电或系统失效前已经将数据从易失性存储区域（如高速缓存）写回到持久性外存中; b) 顺序性，即保证数据是按照程序语义确定的依赖关系先后有序写入 NVM 中，不能乱序。传统的文件系统主要采取写前日志（WAL）或写时拷贝（COW）等技术保证数据的一致性，但这些技术相对 NVM 而言同样存在开销过大的问题。此外，由于 NVM 直接挂在内存总线上，CPU 可以直接通过访存指令读写持久性外存，无须像传统的二级存储系统那样必须先经过内存。为了保证一致性，需要及时将高速缓存中的数据刷回（flush）到 NVM 中，频繁的 flush 操作对性能影响很大，是需要研究的新问题。

2) **降低软件栈开销**

前文已多次提到，对 NVM 而言传统文件系统的软件栈开销过大，因此，降低软件栈开销也就成为了 NVM 文件系统研究的主要方向。比如，传统文件系统中的块设备层和驱动层对 NVM 而言就是多余的 [9, 19]，如图 1 所示。此外，减少持久性外存与工作内存、内核空间与用户空间之间的拷贝也形成共识 [8]。NVM 可字节寻址，因此 NVM 文件系统可以不需要页缓存（page caching）而使用 DAX 技术直接访问 NVM，从而减少了存储栈中 NVM 与 D R AM 之间不必要的数据拷贝。DAX 的核心是零拷贝的内存映射机制（I/O memory mapping，mmap），与传统文件系统中的 mmap 不同，DAX 中的 mmap 是将持久化的数据直接映射到进程的虚拟地址空间，完全不需要页缓存。而传统文件系统中的 mmap 在将文件映射到用户空间时，虽然减少了一次数据拷贝，但依然需要页缓存。DAX 已引入到 Linux 内核中，可有效挖掘 NVM 的低延迟特性 [23]。DAX 技术利用了 NVM 的字节寻址能力，但频繁的映射迫使应用程序静态预留部分存储区域且自行管理，影响了存储空间的利用率 [24]。此外，由于 NVM 的写性能差，如何扬长避短也是研究的方向之一。

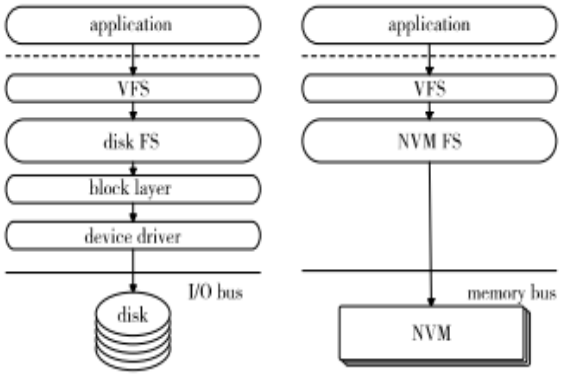

图 1 磁盘文件系统与持久性内存文件系统软件栈

3) **内存管理与文件系统的融合**

文件系统的访问是基于文件索引的，纯软件方式，效率低。NVM 可字节寻址，借助与工作内存相关的软硬件特性（如 MMU）来加速文件系统的访问成为研究方向之一; 此外，通过扩展内存管理的功能，对某些持久性外存对象进行高效管理也是努力的方向，也为内外存融合奠定了一定的基础。

4) **对分布式文件系统的影响**

NVM 的出现不仅带给本地文件系统的设计一系列思考，同时也影响了分布式文件系统的设计，尤其是随着R DMA 网络技术的应用，更使得分布式文件系统的设计发生了根本变化，主要体现在两个方面: a) 传统分布式文件系统面向的是普通硬盘和 TCP/IP 网络，延迟相对较大，对软件的开销不敏感; 而 NVM 和R DMA 的延迟大幅降低，对软件开销十分敏感，并且基于R DMA 的远程数据访问甚至低于 NVM 的本地访问; b) 通过R DMA 技术可以直接读写远程 NVM，无须远程 CPU 的干预，且 NVM 不仅字节可寻址而且具有非易失性，这些特性都深刻影响了分布式文件系统可靠性、一致性等功能的设计。

5) **对文件系统安全、可靠机制的影响**

NVM 支持 XIP（execute in place）特性，该特性会增加不经意写操作的风险。NVM 的写次数有限，也要防止恶意的磨损攻击，同时还要能够抵御内存错误和软件错误，具有一定的鲁棒性。

3 降低一致性开销的主要方法

在降低一致性方面，学术界主要提出了如下一些方法: 尽可能利用 NVM 的字节寻址特性和原子

性操作原语进行原位原子更新（in-place update）；尽量减少一致性保证中写的数据量；优化传统的日志技术；优化元数据的分配结构；延迟关键路径写操作。从本质上看，前四种都是通过减少写操作降低一致性保证开销，最后一种则是通过放松写顺序约束降低一致性保证开销。

3.1 原位原子更新

传统的一致性保证技术，如 WAL 和 COW，要么在等新数据写成功后再改写原数据，要么在改写原数据前先备份原数据，实际上都写了两次，原位原子更新是直接对数据进行改写，并保证原子性。Condit 等人 [4] 在 2009 年最早设计了第一个面向 NVM 的文件系统 BPFS，使用短路影子分页法来保证元数据和一般数据的一致性，其核心思想是对细粒度数据提供原子写操作。具体地，对小于或等于 8 Byte 的数据利用硬件原语操作指令直接进行原位更新；对于超过 8 Byte 的数据则使用 COW 操作，但与完全拷贝不同，它只拷贝那些未被更新的数据，因为即将被改写的数据是没有必要拷贝的。在基于树结构的文件系统中，一个数据的更新可能涉及到父节点的递归更新，直到根节点，如图 2 所示，所有的数据更新均采取以上规则。BPFS 要求硬件提供原子性（atomic write）和顺序性（epoch barrier）两个原语的支持，且原位更新只适用于小粒度数据，如果某些操作涉及的数据更新较大，仍然需要开销较大的 COW 保证一致性。尽管如此，BPFS 仍在 NVM 一致性保证方面进行了积极探索，其中的 epoch 顺序性保证机制影响深远。

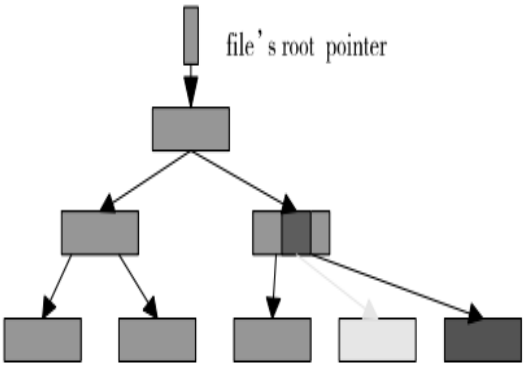

图 2 短路影子分页法示例

Cai 等人 [25] 提出了一种透明高效的硬件支持的非原地更新机制来支持原子性的数据持久化，通过在内存控制器中引入一个轻量级的间接层来实现高效的地址转换和自适应的垃圾回收，确保旧数据一直保留至新数据持久化后。

3.2 减少写操作

减少写操作是降低一致性开销最直接也是最主要的方法，BPFS 实际上也有一些减少写操作的举措 [4]，比如它只拷贝未被更新的数据。Lee 等人 [26，27] 于 2012 年提出了第一个面向 NVM 的日志文件系统 shortcut-JFS，它仍然以块为管理单位但考虑了 NVM 的字节寻址特性，shortcut-JFS 采取了两种机制：差分日志和原地检查。差分日志是指只将与原始数据不同的字节写入日志区，当写入的数据量大于 1 /2 块大小时就写整个块，小于 1 /2 块大小时就只写实际改动的数据，如图 3 所示；原地检查是指在检查点到来时通过修改指针，将日志区中的数据块直接转换为数据区中的数据块，从而避免了数据块在日志区与数据区之间的拷贝过程，这种修改指针的方式充分利用了 NVM 的字节寻址特性，十分巧妙，被广泛应用 [22，28]。通过差分日志和原地检查两种技术，写操作的数量下降了一半。

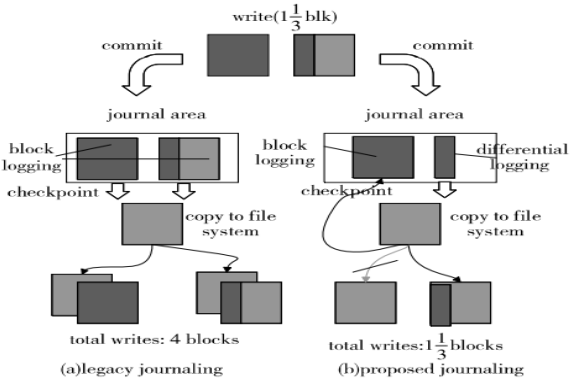

图 3 日志文件系统的比较

Lee 等人 [28] 也提出了类似的思想，他们观察到针对文件的大部分写操作都是在前一版本基础上进行一些细微改动，通过挖掘不同版本之间的相似性将对 NVM 的写操作降到最少，提出了 DTFS 文件系统。DTFS 中有两棵文件系统树，分别为 current 和 shadow 版本，current 版本代表处于一致性状态的最新版本，shadow 版本代表当前文件系统的工作版本，当数据更新时，DTFS 将新的数据块写到 shadow 版本；当系统崩溃时，使用 current 版本将文件系统恢复到之前的一致性状态；当 shadow 版本更新成功后，通过交换指针将 shadow 版本转换为 current 版本，如图 4 所示。通过以上机制，尤其是指针操作大大减少了写操作，保证了一致性，也降低了开销。

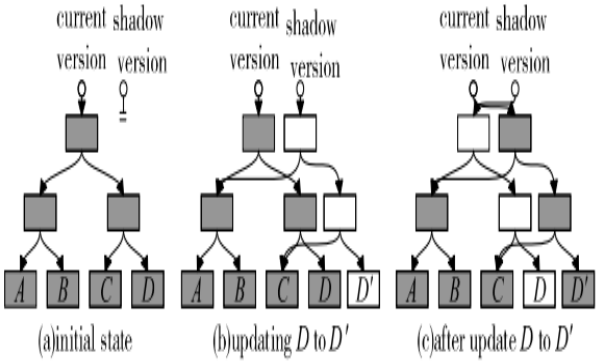

图 4 一系列写操作时 DTFS 树结构的变化

3.3 减少写操作

由于原位更新需要硬件支持且只能支持小粒度数据，对于大粒度的数据一致性保证仍然需要通过传统的日志技术 WAL、COW 或者日志结构 (log-structured) 技术 [29] 等来实现，所以，对这些传统的日志技术进行优化也是降低一致性保证的研究方向之一。Ou 等人 [22] 通过优化 WAL 日志技术来降低一致性开销，提出混合细粒度日志技术 (HFL) 和并发选择性检查点技术 (CSC) 。HFL 的核心思想是将文件系统的元数据日志和文件系统用户数据日志分开，分别采取不同的日志方法，具体地，元数据采用字节粒度的 undo 日志方法，而用户数据使用 cache 行粒度的 redo 日志方法，实现了拷贝开销和日志跟踪开销之间的平衡; CSC 使用并发技术尽可能让不同的数据块同时被更新，如果对一个数据块更新时存在有多个版本，则从该数据块的所有版本 (包括待更新版本和永久数据块版本) 中选择新缓存行数最多 (即更新最多) 的版本作为永久数据块，并将其他数据块中的新缓存行也拷贝到该数据块中，最后修改文件系统的数据块指针，从而减少了不必要的数据拷贝。针对现有的日志技术在 NVM 上表现不佳等问题 [23]，包括写前日志需要往存储设备写入两次、影子页更新时叶子节点到根节点都要受到影响、日志结构要求空闲空间连续且持续可用，Xu 等人 [8] 提出了一种高效的日志文件系统 NOVA，在很多细节上都进行了改进，包括为每一个索引节点分配单独的日志，使得不同文件的更新可以并行; 有效的日志记录很小，保证了快速扫描; 使用 4 KB 的页来保存日志，以链表形式存放，因此不需要连续空间; 原子更新日志的尾指针，实现了日志的原子追加; 日志清理以页为粒度，粒度变得更小; 将日志数据和文件数据存在 NVM 中，而将 radix tree 存放在 D R AM 中，使得 NVM 中的数据结构变得简单高效; NOVA 对文件数据使用的是 COW 技术，恢复时只需要扫描 NVM 中的一小部分，能够及时回收旧页，减少了垃圾回收开销。Kim 等人 [30] 提出 pNOVA，主要是使用细粒度锁解决了 NOVA 存在的扩展性问题。

3.4 优化元数据的分配结构

文件系统由常规数据 (normal data) 和元数据 (meta data) 两部分组成，即便常规数据写入成功而元数据写入失败仍然是不一致的 [31]，因此通常是先更新常规数据，成功后才去更新元数据 [4, 5]。由于针对元数据的更新频繁而 NVM 的写延迟偏高，通常是将元数据的分配结构 (allocation structure) 存放在 D R AM 中。一些经典的文件系统 (如 BPFS [4] 和 PMFS [5]) 要求分配结构必须在 D R AM 中，不允许在 NVM 中。然而，元数据的分配结构存放在 D R AM 中有如下几个缺点: a) 每次挂载文件系统时需要在 D R AM 中重建分配结构，这种重建不能简单地顺序扫描，而是需要遍历所有文件的索引结构，随着 NVM 容量增大，重建分配结构会产生延迟; b) 为了节省 D R AM 的空间，分配结构通常采用链表结构，便于快速地插入和删除操作，因此导致不能随机访问任意节点，影响了文件操作的并发性; c) 需要占用很大的 D R AM 空间，以 PMFS 为例，每 4 KB 的数据块需要 32 Byte 的 D R AM 空间存放索引，那么一片 512 GB 的 NVM 就需要 4 GB 的 D R AM 空间专门用于存放分配结构。为了缓解该问题，Zha 等人 [32] 基于 NVM/D R AM 混合内存架构的文件系统 (HMFS) 提出了一种新的一致性保证机制，首先在物理布局上，将存储区域分为一个用于随机写的原位更新区域和多个用于顺序写的 COW 更新区域; 在逻辑结构上，将存储区域分为多个不同的层，每层存放不同的信息和采取不同的策略。每次挂载文件系统时，根据最新快照中的系统状态检查文件系统并修复不一致性; 通过懒惰验证技术保持主区域中的块和 NVM 中分配结构的一致性，即将新块分配的一致性验证推迟到快照获取时进行，文件删除或截断的失效延迟到快照删除时进行。

3.5 推迟关键路径上的写操作

NVM 的一致性开销很大一部分源于缓存刷出 (cache flush) 。与块设备不同，为了防止数据的无序逐出而出现数据的不一致，必须立即将关键的元数据刷出到 NVM 中，为此，不得不在文件系统的关键路径上使用刷出 (CLFLUSH) 和同步 (SFENCE) 等高延迟高指令，使得关键路径延迟变长。为了缓解该问题，Dong 等人 [33] 重新讨论了软更新 (soft update) 技术，设计了一个新的文件系统 SoupFS，它通过推迟写操作和相关性跟踪相结合来消除大部分元数据同步更新。传统软更新中相

关性跟踪非常复杂，Dong 等人利用了 NVM 的字节寻址特性，通过更好的目录组织和指针匹配简化了相关性跟踪，同时提出了基于指针的新双视图技术。双视图是指始终有两个视图，分别为最新的视图和一致性视图，前者指向的数据总是最新的但可能不一致，后者指向的数据永远是一致的且是持久化的。基于指针的双视图共享大部分数据结构，但使用不同的指针，从而允许延迟持久化，并消除了崩溃后的文件系统检查。SoupFS 通过延迟绝大部分的同步缓存刷出操作显著缩短了关键路径延迟。以上阐述了降低一致性开销的主要相关工作，包括尽可能原位原子更新、尽量减少写操作、对传统日志技术进行优化、优化元数据的分配结构、推迟关键路径上的写操作等。其中用到的主要思想包括通过原子写、增量更新、修改指针来减少更新的数据量，通过并发来提高更新效率，通过延迟更新来放松更新的顺序约束

4 结束语

存储与计算、网络共同组成计算机系统的三要素，数据密集型应用的大量涌现，使存储面临前所未有的压力。NVM 的出现给存储系统带来巨大机遇，同时也带来诸多挑战，简单的器件替换无法挖掘 NVM 的性能优势，软件栈的重构也需要同步跟上。采取虚拟内存的方式管理持久性外存涉及到整个软件生态系统的革新，是一项庞大且耗时的工程。短期来看，随着 3D XPoint 芯片推向市场，仍然采取文件系统管理持久性外存的方式是现实可行的，过去十余年学术界开展了大量研究，一些技术已趋于成熟。本文梳理了近年来针对 NVM 文件系统的相关研究成果，通过梳理发现这些工作主要集中在以下几个方面: 降低一致性开销; 降低软件栈开销; 内存管理与文件系统融合; 分布式文件系统; 文件系统的安全、容错和空间管理等。

参 考 文 献

[1] wang T，Jung J，Won Y. HEAPO: Heap-based persistent object store [J]. ACM Trans on Storage，2014，11(1) : article No. 3.

[2] olos H，Tack A J，Swift M M. Mnemosyne: Lightweight persistent memory [J]. ACM SIGPLAN Notices，2011，46(3) : 91-104.

[3] oburn J，Caulfield A M，Akel A，et al. NV-Heaps: making persistent objects fast and safe with next-generation，non-volatile memories [J] . ACM SIGPLAN Notices，2011，46(3) : 105-118.

[4] ondit J，Nightingale E B，Frost C，et al. Better I/O through byteaddressable，persistent memory [C] / /Proc of the 22nd ACM SIGOPS Symposium on Operating Systems Principles. New York: ACM Press，2009: 133-146.

[5] ulloor S R，Kumar S，Keshavamurthy A，et al. System software for persistent memory [C] / /Proc of the 9th European Conference on Computer Systems. New York: ACM Press，2014: article No. 15.

[6] u Xiaojian，Qiu Sheng，R eddy A L N. SCMFS: a file system for storage class memory and its extensions [J]. ACM Trans on Storage，2013，9(3) : article No. 7.

[7] ha E H M，Chen Xianzhang，Zhuge Qingfeng，et al. A new design of in-memory file system based on file virtual address framework [J]. IEEE Trans on Computers，2016，65(10) : 2959-2972.

[8] u Jian，Swanson S. NOVA: a log-structured file system for hybrid volatile /non-volatile main memories [C] / /Proc of the 14th USENIX Conference on File and Storage Technologies. Berkeley，CA: USENIX Association，2016: 323-338.

[9] ha E H M，Jia Yang，Chen Xianzhang，et al. The design and implementation of an efficient user-space in-memory file system [C] / /Proc of the 5th Non-Volatile Memory Systems and Applications Symposium. Piscataway，NJ: IEEE Press，2016: 1-6.

[10] u Youyou，Shu Jiwu，Chen Youmin，et al.Octopus: an R DMAenabled distributed persistent memory file system [C] / /Proc of USENIX Annual Technical Conference. Berkeley，CA: USENIX Association，2017: 773-785.

[11] 伟，汪东升. 基于非易失存储器的事务存储系统综述 [J]. 计算机研究与发展，2016，53 (2) : 399-415. (Shi Wei，Wang Dongsheng. Survey on transactional storage systems based on non-volatile memory [J]. Journal of Computer R esearch and Development，2016，53(2) : 399-415.)

[12] 章玲，金培权，岳丽华，等. 基于 PCM 的大数据存储与管理研究综述 [J]. 计算机研究与发展，2015，52 (2) : 343-361. (Wu Zhangling，Jin Peiquan，Yue Lihua，et al. A survey on PCM-based big data storage and management [J]. Journal of Computer R esearch and Development，2015，52(2) : 343-361.)

[13] eza J，Luo Yixin，Khan S，et al. A case for efficient hardware /software cooperative management of storage and memory [C] / /Proc of the 5th Workshop on Energy-Efficient Design. Piscataway，NJ: IEEE Press，2013: 1-7.

[14] ailey K，Ceze L，Gribble S D，et al. Operating system implications of fast，cheap，non-volatile memory [C] / /Proc of the 13th USENIX Conference on Hot Topics in Operating Systems. Berkeley，CA: USENIX Association，2011.

[15] ang Jun，Wei Qingsong，Chen Cheng，et al. NV-Tree: reducing consistency cost for NVM-based single level systems [C] / /Proc of the 13th USENIX Conference on File and Storage Technologies. Berkeley，CA: USENIX Association，2015: 167-181.

[16] enkataraman S, Tolia N, R anganathan P, et al. Consistent and durable data structures for non-volatile byte-addressable memory [C] / / Proc of the 9th USENIX Conference on File and Storage Technologies. Berkeley, CA: USENIX Association, 2011: 61-75.

[17] u Youyou, Shu Jiwu, Sun Long. Blurred persistence in transactional persistent memory [C] / /Proc of the 31st Symposium on Mass Storage Systems and Technologies. Washington DC: IEEE Computer Society, 2015: 1-13.

[18] oraru I, Andersen D G, Kaminsky M, et al. Consistent, durable, and safe memory management for byte-addressable non-volatile main memory [C] / /Proc of the 1st ACM SIGOPS Conference on Timely R esults in Operating Systems. New York: ACM Press, 2013: 1-17.

[19] hang Zheng, Feng Dan, Chen Jianxi, et al. The design and implementation of a lightweight management framework for non-volatile memory [C] / /Proc of IEEE International Conference on Trust, Security and Privacy in Computing and Communications. Piscataway, NJ: IEEE Press, 2016: 1551-1558.

[20] ong H, Moon Y J, Lee S K, et al. Transforming legacy file systems into persistent memory exploiting file systems with MeLo . pdf.

[21] u Jiaxin, Shu Jiwu, Lu Youyou. A high performance file system for non-volatile main memory [C] / /Proc of the 11th European Conference on Computer Systems. New York: ACM Press, 2016: article No. 12.

[22] u Jiaxin, Shu Jiwu. Fast and failure-consistent updates of application data in non-volatile main memory file system [C] / /Proc of the 32nd Symposium on Mass Storage Systems and Technologies. Piscataway, NJ: IEEE Press, 2016: 1-15.

[23] ehgal P, Basu S, Srinivasan K, et al. An empirical study of file systems on NVM [C] / /Proc of the 31st Symposium on Mass Storage Systems and Technologies. Washington DC: IEEE Computer Society, 2015: 1-14.

[24] annan S, Gavrilovska A, Schwan K. R educing the cost of persistence for nonvolatile heaps in end user devices [C] / /Proc of the 20th IEEE International Symposium on High Performance Computer Architecture. Washington DC: IEEE Computer Society, 2014: 512-523.

[25] ai Miao, Coats C C, Huang Jian. HOOP: efficient hardware-assisted out-of-place update for non-volatile memory [C] / /Proc of the 47th ACM/IEEE Annual International Symposium on Computer Architecture.Piscataway, NJ: IEEE Press, 2020: 584-596.

[26] ee E, Yoo S, Jang J E, et al. Shortcut-JFS: a write efficient journaling file system for phase change memory [C] / /Proc of the 28th IEEE Symposium on Mass Storage Systems and Technologies. Piscataway, NJ: IEEE Press, 2012: 1-6.

[27] ee E, Yoo S H, Bahn H. Design and implementation of a journaling file system for phase-change memory [J]. IEEE Trans on Computers, 2015, 64(5) : 1349-1360.

[28] ee E, Jang J, Bahn H. DTFS: exploiting the similarity of data versions to design a write-efficient file system in phase-change memory [C] / /Proc of the 29th Annual ACM Symposium on Applied Computing. New York: ACM Press, 2014: 1535-1540.