基于非易失性存储器的存储系统技术研究进展

舒继武,陆游游,张佳程,郑纬民

清华大学计算机科学与技术系,北京 100084

摘要 非易失性存储器(NVM)主要包括两类,即适用于外存的、块寻址的闪存和适用于内存的、字节寻址的持久性内存。相比于传统磁盘,闪存具有性能高、能耗低和体积小等优势;相比于DRAM(动态随机存储器),持久性内存如PCM(相变存储器)、RRAM(阻变存储器)等,具有非易失、存储密度高以及同等面积/内存插槽下能给多核环境的CPU提供更多的数据等优点,这些都为存储系统的高效构建带来了巨大的机遇。然而,传统存储系统的构建方式不适用于非易失性存储器,阻碍了其优势的发挥。为此,分析了基于非易失性存储器构建存储系统的挑战,从闪存、持久性内存两个层次分别综述了它们在存储体系结构、系统软件以及分布式协议方面的变革,总结了基于非易失性存储器构建存储系统的主要研究方向。

关键词 非易失性存储器;闪存;持久性内存;存储系统

近年来,非易失性存储器(non-volatile memory, NVM)技术得到了快速发展。非易失性存储器主要分为块寻址和字节寻址两类。以闪存(flash memory)为代表的块寻址非易失性存储器已经广泛应用于嵌入式系统、桌面系统及数据中心等系统中。字节寻址的非易失性存储器主要包括相变存储器(phase change memory, PCM)、阻变存储器(resistive random-access memory, RRAM)、自旋矩存储器(spin-transfer torque RAM, STT-RAM)等。而在字节寻址的非易失性存储技术方面,NVDIMM是已经商用的一种模拟持久性内存存储器件。NVDIMM采用闪存与DRAM(动态随机存储器)的混合形式,并采用电容或后备电源保证DRAM数据掉电不丢,以模拟持久性内存。2015年,英特尔与镁光公司联合发布的3D XPoint存储技术,是最接近商用的、真正意义上的、字节寻址的非易失性存储器,并计划于2017年进入市场。随着非易失性存储器硬件技术的成熟,基于非易失性器件构建存储系统成为当前的研究热点之一。

在应用方面,高性能计算和大数据分析对数据的存储与处理的要求越来越高,对非易失性存储器的使用有着迫切的需求。数据密集型应用的比例逐渐超越计算密集型应用,成为了计算机系统中的主流应用。然而,外存存储性能的提升远远落后于计算性能的提升,传统存储系统的性能远不能满足高性能计算与大数据应用的需求。为此,非易失性存储器因具有性能高、能耗低和体积小等优点,对于数据密集型应用的性能提升具有重要意义。

然而,传统存储系统的构建方式不仅不利于发挥非易失性存储器在性能方面优势,还暴露了易失性存储器在耐久性(使用寿命)、读写不对称等方面的劣势。本文首先分析非易失性存储器在现有存储系统中应用的矛盾问题,讨论基于非易失性存储器构建存储系统的挑战,然后从闪存与持久性内存两个层次综述现有相关研究进展,总结基于非易失性存储器的存储系统的研究趋势。

1 基于非易失性存储器的存储系统构建的挑战

在近半个世纪里,磁盘占据着存储介质的主导地位。磁盘的机械式部件限制了其延迟和带宽性能的提升。现有存储系统中积累了大量针对磁盘机械式特性进行优化的设计。非易失性存储器与磁盘存储存在较大差异。非易失性存储器是电子式存储器件,数据读写无需机械式部件驱动,数据读写访问延迟可从毫秒级降至微秒甚至纳秒级。除此之外,非易失性存储在读写方式、读写对称性、耐久性等方面与传统磁盘存储相差较大。这些差异对现有存储系统的设计带来的巨大的挑战,包括存储体系结构、系统软件和分布式协议等方面。

1.1 体系结构

非易失性存储器相比于磁盘有延迟低和带宽高的特点。现有存储系统

收稿日期:2016-05-30;修回日期:2016-06-27

基金项目:国家自然科学基金项目(61327902, 61232003);北京市科委项目(D151100000815003)

作者简介:舒继武,教授,研究方向为网络(云/大数据)存储系统、存储可靠性与安全等,电子信箱:shujw@tsinghua.edu.cn

引用格式:舒继武, 陆游游, 张佳程, 等. 基于非易失性存储器的存储系统技术研究进展[J]. 科技导报, 2016, 34(14): 86-94; doi 10.3981/j.issn.1000-7857.2016.14.010

中的硬件接口、通知机制、软硬件接口等均限制了非易失性存储器低延迟与高带宽优势的发挥。

在硬件接口上,磁盘的SATA或SCSI等硬件接口不适用于非易失性存储器。在带宽上,闪存设备中由于内部可实现不同级别并发,其带宽可高达数GB/s,而SATA 3.0的理论带宽为600 MB/s,SATA或SCSI等硬件接口成为了新的带宽瓶颈。在延迟上,如图1所示[1],闪存比磁盘的访问延迟低6个数量级,PCM等持久性内存的访问延迟比磁盘低12个数量级,而SATA协议及其SATA在设备内部和主机驱动上的复杂协议解析,都引入了额外的软件延迟,从而成为新的延迟瓶颈。因而,NVM需要采用带宽更高、延迟更低的PCIe接口或DDR接口[2,3]。

在通知机制上,由于磁盘性能远低于CPU性能,传统存储系统采用中断方式以避免CPU的等待。在非易失性存储系统中,NVM延迟降低,相比之下,中断方式所引发的上下文切换等开销较大。因而,基于非易失性存储器的存储系统一般采用轮询等通知机制[4]。

在软硬件接口上,主机与磁盘采用简单的读写接口。闪存的存储单元不可覆盖写,闪存设备内部通过异地更新方式更新数据,无效数据通过垃圾回收释放空间。若主机不能及时通知闪存设备其中数据的无效性,闪存设备的垃圾回收将引入较多不必要的数据移动,影响闪存的性能和寿命。因此,软硬件接口上需要增加trim命令以提供该语义传递。类似地,软硬件接口还需要更丰富的命令,以提高NVM设备管理的效率[5~8]。

1.2 系统软件

非易失性存储器对系统软件的挑战主要体现在两个方面,即软件开销和软件层次设计。

在软件开销上,NVM硬件的开销大幅降低,软件系统中的开销比例逐渐增大。图2对比了不同存储介质上软件开销在存储总开销中的占比。由图2中可见,在传统磁盘存储系统中,软件占比仅为0.30%;在DDR接口的NVM上,软件占比超过90%。因此,基于非易失性存储器的存储系统中的软件需要高效的设计。

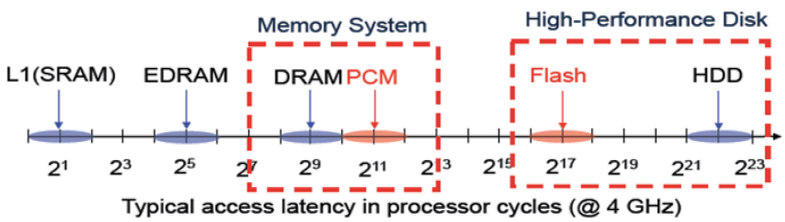


图1 不同存储介质的延迟比较[1]

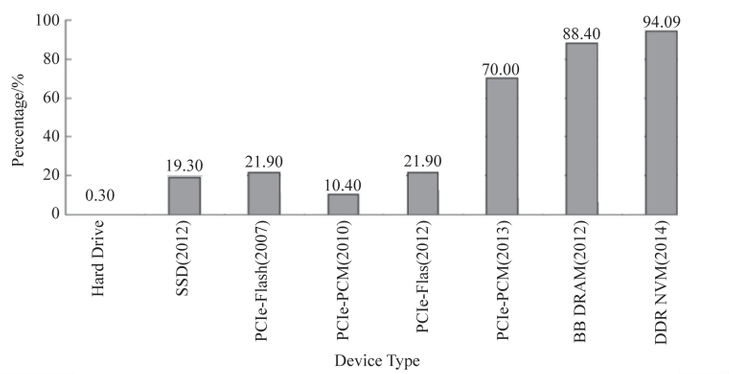


图2 不同存储设备上软件开销在存储总开销中所占的比例[9]

在软件层次设计上,系统软件不仅需要自身设计的高效性,也需要与非易失性存储器硬件之间实现更优的功能划分。传统系统软件在NVM上的主要矛盾,体现在以下4个方面[10]:

1)维度缺失。NVM存储器多数存在耐久性问题,即数据写入造成存储单元的磨损,在有限次写入之后不能可靠存储数据,达到寿命极限。传统软件系统仅考虑磁盘存储介质,因而缺乏对存储介质耐久性维度的考虑。

2)分工欠妥。闪存的存储单元不可覆盖写,采用异地更新方式更新数据。异地更新方式天然地提供了数据多版本,利于实现数据一致性。相比传统系统软件的一致性实现,效率大幅提升。因而,系统软件与存储硬件之间需要考虑新的功能分工。

3)功能冗余。NVM设备(如闪存固态盘)在设备内部采用嵌入式CPU与RAM处理地址转换、垃圾回收等工作。系统软件(如文件系统)需要对存储空间进行管理。这些功能产生冗余,对NVM的性能、可靠性和寿命均有影响。因而,系统软件与存储硬件可以通过软硬件协同的方式简化系统中的繁冗设计。

4)优化错配。传统系统软件过多地针对磁盘的特性进行优化,如顺序化读写等。然而,NVM的随机读写性能与顺序读写差距缩小,但读写呈现不对称特性。因而,传统系统软件需要考虑NVM的新特性进行相应的设计优化。

1.3 分布式协议

非易失性存储器的特性对现有分布式协议提出了新要求。例如,在非易失性存储器的持久性利用方面,闪存以及字节寻址NVM用于分布式存储系统客户端缓存时,提供了数据持久性,可延缓数据写回时间。分布式存储系统中多客户端间的数据访问需要保证一致性,要求数据及时写回持久性存储。因而,分布式客户端缓存一致性协议需要均衡客户端数据持久性与一致性。同样,在非易失性存储器的耐久性利用,分布式存储系统中的负载均衡协议需要考虑数据迁移对非易失性存储器

寿命的影响。

非易失性存储器的新特性也为实现新的分布式协议提供了机遇。例如,NVM在访问并发性与随机访问上优势明显。在分布式共享存储中,NVM的访问并发性可缓解多个客户端并发访问的性能问题。闪存的异地更新特性也利于数据并发写入,延迟冲突检测,提高并发性。因而,利用该特性可设计新的分布式共享数据访问协议。又如,NVM提供了低延迟数据存储,基于新型高速网络的RDMA也提供了低延迟的数据网络交互。因而,这也提供了RDMA与NVM相结合的分布式数据访问协议的新机遇。

2 闪存存储系统的研究进展

面对闪存存储系统在应用中的上述挑战,研究人员致力于对现有的存储结构、系统软件以及分布式协议进行变革。例如,研究去掉闪存设备上的FTL(flash translation layer),设计基于裸闪存的存储系统;优化闪存集群上的数据迁移策略,延长闪存寿命;研究闪存与特定应用负载特征相结合的软件定义存储等。

2.1 存储结构的变革

2.1.1 硬件接口

闪存在实际应用中的形态,由早期的闪存固态盘、闪存卡,逐渐发展到闪存阵列和闪存集群系统。

在硬件接口上,由于传统SATA接口的速度限制不能发挥闪存的高并发性能,新的接口,例如PCIe和DIMM将被应用到高性能闪存设备上。在体系结构上,将闪存移动到离CPU更近的位置上,将获得主存级的传输带宽。在使用新的设备接口的同时,闪存设备的内部结构也在向着简化、高效的方向发展。传统磁盘阵列的访问,需要经过主机HBA、网络交换、阵列控制器等访问模块。PCIe固态盘RAID卡是早期的闪存固态盘加速卡形式,其访问也需要经过RAID控制器、元数据处理模块后才能访问数据[2]。以FusionIO为代表的裸闪存卡,可以使主机直接访问闪存芯片,缩减IO传输路径,提供了极低的访问延迟。

闪存卡难以实现存储容量的扩展及存储共享,多家存储厂商推出了闪存阵列,其中包括采用固态盘与传统阵列控制器的演进式设计,以及采用闪存芯片与全新阵列控制器的革新式设计。基于传统存储阵列的演进式设计,它们支持HDD规格的SSD,并且将SSD仅当作HDD。这意味着在write cliff出现时,性能下降比较严重。一些传统存储控制器厂商重新构建了存储控制器,使其能够支持闪存特有的行为和性能,同时还保持与传统阵列相同的高级特性。与使用HDD的同级别阵列相比,这些阵列的平均性能有接近5倍的性能提升[11]。基于全闪存阵列的革新式设计,是专门为闪存技术从头开始构建设计,采用积极的垃圾回收和数据缓存算法,通常性能(视工作负载而定)约提升10倍(使用超过90%的容量)[3]。全闪存阵列厂商包括Pure Storage、EMC XtremIO、Solid Fire等。

在闪存的集群系统方面,FAWN[12]和Gordon[13]是2个代表性的工作。

FAWN[12]是卡内基梅隆大学(CMU)基于闪存介质构建的可扩展、低能耗、高性能的集群系统。与闪存卡和闪存阵列仅关注于IO子系统的性能及可靠性的设计不同,FAWN从集群整体设计的角度考虑闪存与处理器的匹配,以降低整体能耗。通过采用低频低能耗CPU与闪存存储相匹配,提高系统各组成部分在数据密集型计算中的利用率。FAWN实现的Key Value存储系统,可提供每焦耳364次查询,相比普通桌面系统每焦耳1.96次查询,性能提升数百倍。

Gordon[13]是加州大学圣地亚哥分校(UCSD)设计的,与FAWN关注的高性能、低能耗不同,Gordon主要工作是设计闪存转换层来匹配处理器和内存芯片的性能,发挥闪存芯片间的并发特性,设计包括地址动态映射、合成大物理页,同时采用并发与流水机制。Gordon在单板上集成256 G的闪存和2.5 G的DRAM,目前已经应用到San Diego超算中心,用于天体物理、基因组测序等数据密集型计算领域。

2.1.2 通知机制

随着闪存设备的延迟越来越低,文献[4]发现传统基于中断的软硬件通知机制开销增大,低延迟访问造成的频繁中断,上下文切换代价已经超出了循环等待的代价。UCSD的Moneta[14]系统采用了spin等待方式避免频繁中断带来的高昂代价。文献[15]通过减少I/O路径上的上下文切换,降低软件延迟。Linux Block IO[16]在多路CPU上,对存储设备建立多个请求队列,减少竞争锁的开销来降低软件延迟。

在I/O路径上,因软件的模块化需求,软件系统对软件的多个层次进行了抽象,软件层次化屏蔽了异构特性,但也导致了不必要的处理与转换。在闪存存储系统上,去掉设备的FTL,从文件系统或者应用层程序直接管理裸闪存,成为一个研究热点。例如,SDF[17]和LOCS[18]就是使用了通道开放型(open-channel)的SSD,闪存的内部并行直接由应用程序进行控制;RFFS[19]和ALFS[20]则简化了FTL并在其上实现了闪存文件系统。

2.1.3 软硬件接口

在软件接口上,传统的基于块的IO接口,由于缺乏丰富的语义支持,上层软件不能发挥闪存的特性,例如异地更新、内部并发等;而闪存设备也不能对IO请求做优化,例如有效的冷热分组、高性能的请求调度等。一些研究工作通过扩展现有的软件接口对系统进行优化,例如,使用Atomic Write[7]来利用闪存的特性完成原子更新操作,使用TRIM来提供数据显示删除的语义,使用额外的数据标识来显示数据的冷热程度[21],等等。

2.2 系统软件的变革

传统文件系统过多地针对磁盘设计。一方面,文件系统的自身设计与磁盘特性密切相关,例如,传统文件系统大多假设文件系统的随机读写性能较差,继而采取缓存、预取、数据分组、碎片整理等机制,甚至用对磁盘磁道对齐等方式提高系统性能,这些优化对闪存作用有限,甚至额外的软件管理引入造

成数据写放大、处理延迟增加等问题。另一方面,文件系统与闪存设备的自管理机制,在很多方面产生了功能冗余,例如重复的空间管理、垃圾回收和地址映射等,这些冗余会造成元数据管理的双倍开销、文件系统优化失效以及额外的写放大等问题。

针对这些问题,面向闪存构建文件系统也是近年来的研究热点。图3示意了闪存文件系统的结构演变。早期,闪存多用于嵌入式系统,图3(a)为嵌入式闪存文件系统的结构示意图。在嵌入式系统中,闪存以裸闪存形式存在,嵌入式闪存文件系统在文件系统中实现闪存转换层FTL的相关功能,包括地址映射、垃圾回收和磨损均衡等。这类文件系统包括JFFS2[22]、UBIFS[23]和YAFFS[24]等。

随着闪存密度提升和价格下落,闪存逐步被桌面机、服务器和数据中心采用。在该环境下,闪存最初以固态盘SSD的形式出现。如图3(b)所示,固态盘提供与传统磁盘相同的读写接口,并采用闪存转换层FTL转换为闪存命令。由于固态盘与传统磁盘的读写接口兼容,传统文件系统可无缝地部署于固态盘上。现有不少研究在该使用模式下针对SSD特性优化文件系统。F2FS[25]是三星公司针对SSD进行定向优化的文件系统。F2FS是针对闪存的特性,改良传统的日志型文件系统的一个例子。文件系统日志的每个段与底层flash的block保持一致大小,并对数据进行冷热分组,以此来提高SSD的垃圾回收(garbagecollection, GC)效率。同时利用地址映射表,以避免元数据更新时一直更新到根的滚雪球式的更新,减少元数据写入量,提高SSD寿命。

固态盘内部闪存转换层FTL要求较强的嵌入式处理器和设备内缓存,尤其是盘内闪存容量较大时。FusionIO公司提出主机端FTL的设计VFSL(虚拟文件系统层),以充分利用主机的处理器与内存资源。如图3(c)所示,普林斯顿大学与FusionIO公司基于主机端FTL联合研究了DFS文件系统[26],将文件系统的块分配操作下移至FTL,避免了文件系统与FTL的重复数据分配工作。DFS还通过利用Fusion-IO的虚拟文件系统层VFSL(virutal file system layer)提供的原子写、log等机制,实现了元数据更新的原子性,无需额外的log操作。

尽管主机端FTL的设计有效利用了主机端资源,但文件系统与FTL之间仍采用简单的读写接口,限制了语义信息的使用。针对该问题,清华大学研制了闪存耐久性感知的对象式闪存存储系统OFSS[27],如图3(d)所示。OFSS是闪存存储系统中需要引入耐久性(考虑闪存寿命)的一个例子。传统文件系统的设计主要是基于磁盘的设计,关注于顺序读写的优化,较少考虑数据写入量的问题。在闪存系统中,闪存寿命随着写入数据量的增加而降低,而传统文件系统机制中日志机制、同步元数据更新、页边界对齐更新等多种机制设计引入了极大的更新数据量,因而文件系统的机制需要重新考虑以满足闪存系统的寿命需求。针对这些问题,OFSS[27]提出了对象式闪存转换层的设计,结合文件系统的空间管理功能与闪存转换层的地址映射,以发挥文件系统的文件语义,同时利用闪存特性。基于对象式闪存转换层,又提出了延迟持久化元数据、粗粒度空闲空间元数据管理、紧凑数据组织的技术,以减少传统文件系统引入的写入数据量。

ReconFS[28]是清华大学研制的对目录树管理进行优化的闪存文件系统。ReconFS是闪存颠覆了传统文件系统目录树构建的一个例子。闪存文件系统命名空间(目录树)构建了文件系统的结构,但目录树结构的维护引入了较大的元数据开销,具体表现为写入频繁和分散小写的特征。元数据的频繁、分散小写不仅损害了存储系统的性能,也加速了闪存磨损,降低了闪存系统寿命。针对此问题,ReconFS[28]提出了新型文件系统命名空间的设计,并基于此实现了可重构文件系统ReconFS。可重构文件系统ReconFS通过分离易失性目录树与持久性目录树的管理,并通过嵌入式索引机制与日志持久化机制,提高了闪存文件系统目录树的一致性与持久性。在系统意外掉电时,由于闪存存储的高IOPS与高带宽,可重构文

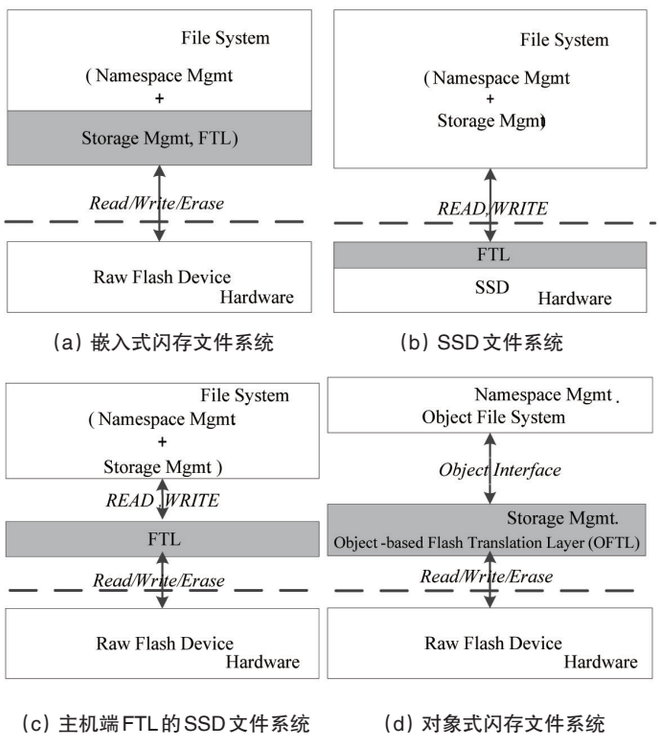


图3 闪存文件系统的结构演变

件系统ReconFS在数秒内完成文件系统目录树的重构。

闪存设备相比于传统磁盘还有许多其他特性,如内部并发等。如何在闪存文件系统层次充分利用这些特性仍需要进一步研究。

2.3 分布式协议的变革

与磁盘相比,闪存具有随机访问速度快、访问低延迟、读写不对称以及有限的寿命等特点,给分布式协议的设计带来了革新的机遇。

一部分研究工作,致力于对现有技术的改进,例如改进分布式存储系统上客户端的缓存机制。文献[29]通过大量的实验,提出使用write-through的缓存策略,能够有效减少维护一致性的开销。DuraCache[30]利用write-through和分配更多的ECC空间,以延长闪存设备的寿命。文献[31][32]提出在闪存缓存上使用write-back的缓存策略,以获得更高的读写性能;还提出了多种替换、恢复算法,以避免掉电时由于write-back导致的数据不一致的问题。

在分布式存储系统中,数据迁移保证了负载均衡以提高整体性能。然而,在分布式闪存存储环境中,数据迁移量增大会导致闪存寿命缩短。针对该问题,EDM[33]提出了耐久性感知的数据迁移策略,在保证高性能的同时减少系统数据写入量,延长闪存设备的寿命。

另一部分研究工作,致力于设计新的协议以发挥闪存设备的特性。CORFU[34]利用多闪存卡构建了低延迟、高带宽的共享日志型、分布式存储系统。通过将闪存设备直连到网络,CORFU去除了服务器,降低了集群复杂性、访问延迟和能耗。CORFU的单节点功率仅为15 W,而一台普通的服务器的功率为250 W左右。CORFU具有极低的读写延迟(低于1 ms),同时还保证了集群系统的强一致性,具有快速的故障恢复能力(约650 ms)。CORFU目前的不足是,系统所用的硬件需要定制,不能利用现有的闪存产品。Tango[35]则实现了基于CORFU的共享数据结构,能保证集群系统中元数据操作的一致性、持久性、原子性和隔离性。

3 持久性内存存储系统的研究进展

字节寻址的非易失性存储器与DRAM性能接近,提供了内存级的数据持久性,可在内存级构建持久性存储系统。持久性内存存储系统改变了传统存储系统两级存储的模式,即从易失性内存和非易失性外存两级结构变化至非易失性内存单级结构。这种结构上的变化也带来了处理器缓存管理等存储结构、内存管理等系统软件以及分布式协议上的变革。

3.1 存储结构的变革

持久性内存存储系统与传统存储系统的一个主要区别在于易失性-持久性边界的变化[36]。如图4所示,在传统存储系统中,易失性-持久性边界位于主存-外存之间;在持久性内存存储系统中,易失性-持久性边界位于处理器缓存-主存之间。存储系统的一致性要求数据原子性写入持久性存储介质中,即处理器缓存中的数据须原子性写回持久性内存。由于处理器缓存替换由处理器硬件管理,因而需要处理器硬件高效支持缓存数据写回。

持久性内存的一致性开销主要分为顺序性和持久性两个方面[36]。顺序性是指处理器数据需要按照数据依赖关系顺序写回持久性内存;持久性是指处理器数据从L1、L2等多级易失性缓存中替换到持久性内存。针对顺序性上的开销,微软研究院提出在处理器缓存中增加epoch原语,允许程序插入epoch指令,由处理器硬件保证写入顺序性,即epoch指令后的写入必须等待epoch指令前的数据写回后方可执行[37,38]。清华大学提出一种放松一致性的技术LOC(loose-ordering consistency),通过引入预测持久化技术提前写回数据,弱化了顺序性要求的性能影响[39]。密西根大学提出了strand persistency的一致性语义,即多个无依赖关系的I/O操作可并行写回,降低了严格顺序性的开销[40]。英特尔在新型处理器上增加了CLFLUSHOPT、CLWB和PCOMMIT指令,以支持持久性内存的高效顺序性写回[41]。

针对持久性上的开销,微软研究院提出全系统持久性WSP技术,通过后备电源等硬件技术提供系统掉电后的数据备份,避免了数据因一致性导致的写回开销[42]。宾州州立大学提出Kiln技术,将非易失性存储器引入处理器缓存的末级缓存LLC中,通过非易失性LLC提供数据新版本的持久化,降低了持久化开销[43]。除了硬件支持外,清华大学提出软件级降低持久化开销的方法——模糊持久化,在处理器缓存管理中引入类steal和no-force的缓存管理方法,减少了数据不必要的持久化与拷贝,降低了一致性开销[44]。

非易失性存储器在存储结构上的影响还体现在存储层次的变化,如DRAM与NVM的混合内存组织,以及DRAM作为NVM缓存等组织方式。3D XPoint技术的性能指标介于内存与闪存之间[45],进一步将持久性存储分层,推动对多级持久性存储进行结构方面的研究。

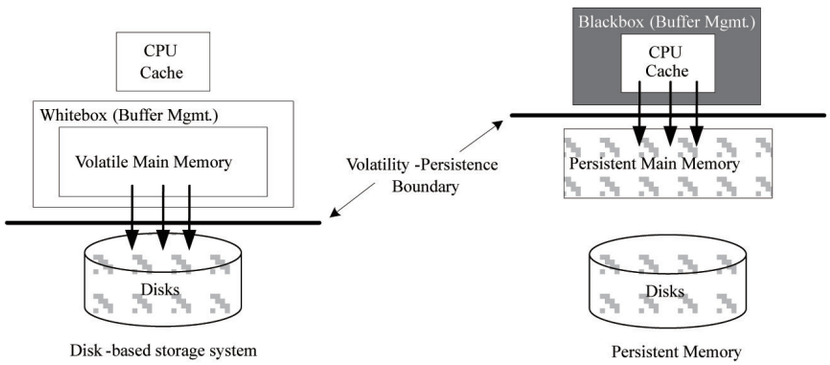


图4 易失性-持久性边界的变化[36]

3.2 系统软件的变革

持久性内存的存储结构从两级存储转变为单级存储。这种结构上的转变不仅要求文件系统进行变革以适应内存级器件特征,同时也催生了内存级维护持久性数据结构这一新型编程模型。

3.2.1 持久性数据结构

传统存储系统中,数据结构包括内存格式和磁盘格式两种格式。在数据的持久化过程中,数据结构从内存格式转换成磁盘格式后再写回外存。在持久性内存中,数据结构可以持久化于内存中,而无需进行格式转换。利用这一特性,威斯康辛大学、加州大学圣地亚哥分校分别研究提出持久性堆栈结构Mnemosyne[46]和NV-Heaps[47],提供用户新的非易失性编程接口,是NVM在主存级中新的应用模式。

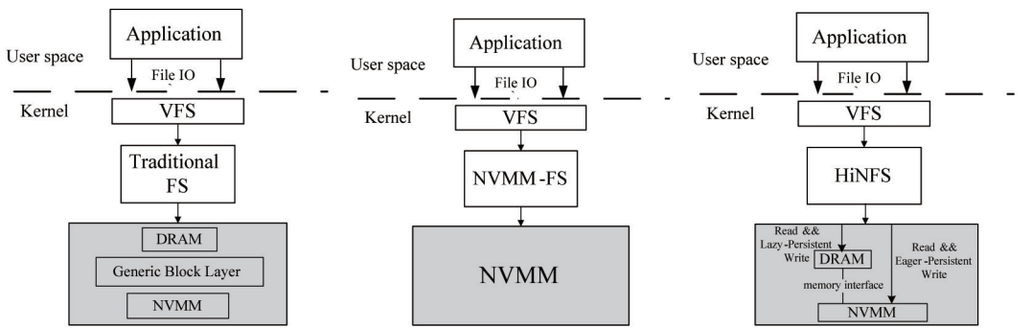
持久性堆栈结构Mnemosyne与NV-Heaps提供了用户态的编程接口,使程序能够在用户态访问非易失主存,避免了传统存储系统中的冗长路径,包括文件系统、设备驱动等。在持久性堆栈中实现持久性数据结构,保证系统掉电后数据的一致性和可访问性,从而使得程序可直接在用户态进行持久性数据结构的高效访问。

3.2.2 持久性内存文件系统

持久性内存文件系统(persistent memory file system, PMemFS)最初通过RAMDISK模拟块设备的形式兼容传统文件系统。如图5(a)所示,持久性内存可以通过RAMDISK形式模拟成块设备,以兼容传统文件系统,包括Ext4、XFS、BtrFS等。传统文件系统无需修改,可直接构建于以持久性内存模拟的RAMDISK块设备之上。RAMDISK形式使得传统文件系统快速受益于内存级的数据持久化,相比于外存性能有数量级的提升。

持久性内存的访问与传统外存设备的访问存在较大差异,RAMDISK块设备形式向文件系统隐藏了较多特性,阻碍了文件系统发掘字节寻址非易失性存储器的优势。在持久性内存中,NVM通过内存总线接入处理器,存储访问延迟极低。同时,与外存块粒度访问不同,持久性内存可由处理器以字节粒度访问。这些差异对持久性内存文件系统提出了新的挑战。如图5(b)所示,现有研究考虑基于持久性内存重新构建新的文件系统,以充分挖掘持久性内存的优势,如BPFS[37]、SCMFS[48]和PMFS[49]等。

微软研究院于2009年研究提出的字节寻址的持久性内存文件系统



(a) RAMDISK块设备的PMemFS　(b) 字节粒度的PMemFS　(c) I/O路径优化的PMemFS

图5 持久性内存文件系统(PMemFS)的结构演变[50]

BPFS[37],旨在利用NVM的字节寻址特性。BPFS以字节寻址方式管理持久性内存,并引入树状结构在持久性内存上构建文件系统数据结构,避免了数据在文件系统缓存与文件系统映像间的数据拷贝;以短路影子页(short-circuit shadow paging)方式提供数据的原子更新,在数据重启后仍能正确访问其中数据[37]。

德克萨斯州农工大学(TAMU)于2011年研究提出内外存融合管理的SCMFS文件系统[48],旨在利用操作系统中现有的内存管理模块提供持久性内存的数据分配与管理。与BPFS一样,SCMFS在持久性内存中无需数据拷贝。SCMFS文件系统针对文件系统的连续数据块访问,尽可能在内存中为一个文件连续分配空间。通过这类的数据组织的优化,提供了优越的性能[48]。

Intel公司于2014年研究提出NVM直写的PMFS文件系统[49],绕开了文件系统缓存直接访问非易失性内存。PMFS是面向持久性内存提供POSIX接口的文件系统,因而兼容传统应用程序。PMFS通过内存映射方式绕开了文件系统缓存,从而避免在持久性内存中的数据拷贝。相比于以内存模拟块设备方式的文件系统RAMDISK,PMFS性能提升可达1个数量级[49]。

当前,以字节粒度、融合内外存管理和NVM直写等持久性内存构建方法,更高效地发挥了字节寻址NVM的性能优势,已基本形成共识。然而,如何在高速硬件上进一步降低软件开销,仍然是持久性内存中的一大挑战。在这个方面,清华大学于2016年提出的I/O路径优化的高性能持久性内存文件系统HiNFS[50]。如图5(c)所示,针对非易失性存储器的读写不对称特性,基于

DRAM与NVM的比较优势,HiNFS对文件系统不同类似的操作进行分类处理,以最优I/O路径满足不同操作的要求,进一步提高了持久性内存文件系统的性能。HiNFS是持久性内存文件系统精细化设计向前迈进的一小步。如何在接近于处理器速度的高速硬件上实现持久性内存文件系统的精细化设计,是未来值得研究的问题。

3.3 分布式协议的变革

持久性内存的硬件技术尚未成熟,仅有小容量NVDIMM形式的持久性内存商用,因而在分布式持久性内存存储系统上的研究还比较少。然而,持久性内存提供了内存级数据持久性,为现有分布式内存系统提供了机遇。例如,现有的分布式数据库系统及分布式内存文件系统可利用持久性内存的数据持久性,而无需将数据备份至外存。

持久性内存延迟较低对现有分布式文件系统的访问路径提出了挑战。如何设计低延迟的文件系统元数据与数据访问协议,是分布式持久性内存文件系统中需要研究的问题。同时,如何利用如RDMA等高速低延迟网络互联构建分布式持久存储系统,也是值得研究的方向。

4 结论

非易失性存储器是电子式存储器件,数据读写通过电信号驱动。电信号读写方式改变了以磁盘为代表的机械式读写方式,推动了存储子系统从机械式到电子式的跨越发展。这使得计算机系统中计算、存储和通信均实现了电子式访问,具有里程碑的意义。同时,一些字节寻址的非易失性存储器具有与内存接近的读写性能,但具有非易失、存储密度高等优点。而非易失性存储器尽管带来了低延迟、高带宽、低功耗等优点,但同时也有耐久性、读写不对称等新特性。本文综述了近年来在闪存存储系统构建和持久性内存存储系统构建的研究中,基于非易失性存储器的特性,在存储结构、系统软件和分布式协议这三个方面的革新技术研究而取得的一些突破性进展。然而,基于非易失性存储器构建存储系统,仍有许多亟需研究的挑战,主要体现在以下几个方面:

1)存储结构的创新与优化。非易失性存储器的特性呈现多元化,且从处理器缓存级至外存级均有匹配的新型器件出现。如何在现有存储层次结构中选用合适的存储器件、设计相应的管理方式,以及如何优化或变革现有存储层次,包括多级持久化存储的设计等,都有待深入研究。此外,非易失性存储器的性能与处理器性能进一步拉近,如何协同非易失性存储与多核/众核处理器的机制设计(如持久性事务内存等)也是一种创新性研究。

2)精细化软件系统设计。非易失性存储硬件性能相比于传统磁盘存储的性能提升极大,存储系统中相应的软件开销显得尤为突出。针对软件系统的优化,采用软硬件结合设计以及细粒度精细化软件设计,将是未来存储系统的研究方向之一。

3)新型分布式系统构建。新型高速存储硬件和高速网络硬件动摇了传统分布式系统中存储与通信的条件假设,且这些硬件均提供了新的访问特性与访问模式。例如,结合RDMA与NVM访问方式可以构建高效的分布式存储系统。这些新型分布式存储系统的构建需要重新思考分布式存储协议的设计,并实施相应的研究。

总之,非易失性存储器对存储结构、系统软件和分布式协议的革新是接近于颠覆性的。这些颠覆性的研究对于存储系统,乃至计算机系统,将产生较大的变革。

参考文献(References)

[1] Qureshi M K, Gurumurthi S, Rajendran B. Phase change memory: from devices to systems[M]. San Rafael: Morgan & Claypool Publisher, 2011.

[2] 陆游游, 舒继武. 闪存存储系统综述[J]. 计算机研究与发展, 2013, 50(1): 49-59.
Lu Youyou, Shu Jiwu. A survey on flash based storage systems[J]. Journal of Computer Research and Development, 2013, 50(1): 49-59.

[3] Fusion IO. The fusion-io difference[EB/OL]. [2015-05-06]. http://www.fusionio.com/load/- media-/lqaz4e/docsLibrary/FIO_SSD_Differentiator_Overview.pdf.

[4] Yang J, Minturn D B, Hady F. When poll is better than interrupt[C]//Conference on File and Storage Technologies (FAST). San Jose, CA, USA: USENIX, 2012: 25-32.

[5] Nellans D, Zappe M, Axboe J, et al. Ptrim ()+ exists (): Exposing new FTL primitives to applications[C]//The 2nd Annual Non-Volatile Memory Workshop (NVMW). La Jolla, CA, USA: UCSD, 2011: 17-17.

[6] Prabhakaran V, Rodeheffer T L, Zhou L. Transactional flash[C]//Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (OSDI). Berkeley, CA, USA: USENIX, 2008: 147-160.

[7] Ouyang X, Nellans D, Wipfel R, et al. Beyond block I/O: Rethinking traditional storage primitives[C]//Proceedings of the 17th IEEE International Symposium on High Performance Computer Architecture (HPCA). San Antonio, Texas, USA: IEEE, 2011: 301-311.

[8] Lu Y, Shu J, Guo J, et al. LightTx: A lightweight transactional design in flash-based SSDs to support flexible transactions[C]//Proceedings of the IEEE 31st International Conference on Computer Design (ICCD). Asheville, North Carolina, USA: IEEE, 2013: 115-122.

[9] Swanson S, Caulfield A M. Refactor, reduce, recycle: Restructuring the I/O stack for the future of storage[J]. Computer, 2013, 46(8): 52-59.

[10] 陆游游. 闪存文件系统关键技术研究[D]. 北京: 清华大学, 2015.

Lu Youyou. Research on key technologies for flash-based file systems[D]. Beijing: Tsinghua University, 2015.

[11] Hewlett Packard Enterprise. StoreServ7450[EB/OL]. [2015-05-01]. http://www.hp.com/hpinfo/newsroom/press.kits/2014/ HPDiscover2014/3PAR.

[12] Andersen D G, Franklin J, Kaminsky M, et al. FAWN: A fast array of wimpy nodes[C]// Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP). Big Sky, Montana, USA: ACM, 2009: 1-14.

[13] Caufield A M, Grupp L M, Gordon S S. Using flash memory to build fast, power-efficient clusters for data-intensive applications[C]//Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). New York, NY, USA: ACM, 2009: 217-228.

[14] Caulfield A M, De A, Coburn J, et al. Moneta: A high-performance storage array architecture for next-generation, non-volatile memories[C]//Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Atlanta, GA, USA: IEEE Computer Society, 2010: 385-395.

[15] Yu Y J, Shin D I, Shin W, et al. Optimizing the block I/O subsystem for fast storage devices[J]. ACM Transactions on Computer Systems (TOCS), 2014, 32(2): 6-10.

[16] Bjørling M, Axboe J, Nellans D, et al. Linux block IO: Introducing multi-queue SSD access on multi-core systems[C]//Proceedings of the 6th International Systems and Storage Conference (SYSTOR). Haifa, Israel: ACM, 2013: 22-31.

[17] Ouyang J, Lin S, Jiang S, et al. Software-defined flash for web-scale internet storage systems[C]//Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS). Salt Lake City, Utah, USA: ACM, 2014: 471-484.

[18] Wang P, Sun G, Jiang S, et al. An efficient design and implementation of LSM-tree based key-value store on open-channel SSD[C]//Proceedings of the 9th European Conference on Computer Systems (EuroSys). New York, NY, USA: ACM, 2014: 16-29.

[19] Zhang J, Shu J, Lu Y. RFFS: A log-structured file system on raw-flash devices[C/OL]. WiP Session of the 14th USENIX Conference on File and Storage Technologies (FAST), Santa Clara, CA, USA, February 22-25, 2016. [2016-03-01]. http://storage.cs.tsinghua.edu.cn/~lu/papers/fast16wip.pdf.

[20] Lee S, Liu M, Jun S, et al. Application-managed flash[C]// Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST). Santa Clara, CA, USA: USENIX, 2016: 339-353.

[21] Kang J U, Hyun J, Maeng H, et al. The multi-streamed solid-state drive[C/OL]. 6th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 14), Philadelphia, PA, USA, June 17-20, 2014. [2015-05-01]. https://www.usenix.org/system/files/conference/hotstorage14/hotstorage14-paper-kang.pdf.

[22] Woodhouse D. JFFS2: The journaling flash file system, version 2[EB/OL]. [2015-05-06]. http://sourceware.org/jffs2.

[23] MTD Subsystem for Linux. UBIFS-UBI file-system[EB/OL]. [2015-05-01]. http://www.linux-mtd.infradead.org/doc/ubifs.html.

[24] Aleph One Limited. YAFFS[EB/OL]. [2015-05-01]. http://www.yaffs.net.

[25] Lee C, Sim D, Hwang J, et al. F2FS: A new file system for flash storage[C]//Proceedings of the 13th USENIX Conference on File and Storage Technologies(FAST). Santa Clara, CA, USA: USENIX, 2015: 273-286.

[26] Josephson W K, Bongo L A, Flynn D, et al. DFS: A file system for virtualized flash storage[C]//Proceedings of the 8th USENIX Conference on File and Storage Technologies(FAST). Berkeley, CA, USA: USENIX, 2010: 85-99.

[27] Lu Y, Shu J, Zheng W. Extending the lifetime of flash-based storage through reducing write amplification from file systems[C]//Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST). Berkeley, CA: USENIX, 2013: 257-270.

[28] Lu Y, Shu J, Wang W. ReconFS: A reconstructable file system on flash storage[C]//Proceedings of the 12th USENIX Conference on File and Storage Technologies(FAST). Berkeley, CA, USA: USENIX, 2014: 75-88.

[29] Holland D A, Angelino E, Wald G, et al. Flash caching on the storage client[C]//Proceedings of the 2013 USENIX Conference on Annual Technical Conference (ATC). San Jose, CA, USA: USENIX, 2013: 127-138.

[30] Liu R, Yang C, Li C, et al. Duracache: A durable ssd cache using MLC NAND flash[C]//Proceedings of the 50th Annual Design Automation Conference (DAC). Austin, Texas, USA: ACM, 2013: 166-171.

[31] Koller R, Marmol L, Rangaswami R, et al. Write policies for host-side flash caches[C]//Proceedings of the 11th USENIX Conference on File and Storage Technologies(FAST). San Jose, CA, USA: USENIX, 2013: 45-58.

[32] Qin D, Brown A D, Goel A. Reliable writeback for client-side flash caches[C]//Proceedings of the 2014 USENIX Annual Technical Conference (ATC). Philadelphia, PA, USA: USENIX, 2014: 451-462.

[33] Ou J, Shu J, Lu Y, et al. EDM: An endurance-aware data migration scheme for load balancing in SSD storage clusters[C]//IEEE 28th International Parallel and Distributed Processing Symposium(IPDPS). Phoenix, Arizona, USA: IEEE, 2014: 787-796.

[34] Balakrishnan M, Malkhi D, Prabhakaran V, et al. Corfu: A shared log design for flash clusters[C]//Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation(NSDI). San Jose, CA, USA: USENIX, 2012: 1-14.

[35] Balakrishnan M, Malkhi D, Wobber T, et al. Tango: Distributed data structures over a shared log[C]//Proceedings of the 24th ACM Symposium on Operating Systems Principles(SOSP). Farmington, Pennsylvania, USA: ACM, 2013: 325-340.

[36] Lu Y, Shu J, Sun L. Blurred persistence: efficient transactions in persistent memory[J]. ACM Transactions on Storage, 2016, 12(1): 1-13.

[37] Condit J, Nightingale E B, Frost C, et al. Better I/O through byte-addressable, persistent memory[C]//Proceedings of the ACM SIGOPS 22nd Symposium

on Operating Systems Principles(SOSP). Big Sky, Montana, USA: ACM, 2009: 133-146.
[38] Moraru I, Andersen D G, Kaminsky M, et al. Persistent, protected and cached: Building blocks for main memory data stores[EB/OL]. [2015-05-06]. http://www.researchgate.net/publication/268269110.
[39] Lu Y, Shu J, Sun L, et al. Loose-ordering consistency for persistent memory[C]//Proceedings of the IEEE 32nd International Conference on Computer Design(ICCD). Phoenix, Arizona, USA: IEEE, 2014: 216-223.
[40] Pelley S, Chen P M, Wenisch T F. Memory persistency[C]//Proceedings of the 41st ACM/IEEE International Symposium on Computer Architecture (ISCA). Minneapolis, Minnesota, USA: ACM/IEEE, 2014: 265-276.
[41] Intel corporation: Intel architecture instruction set extensions programming reference[EB/OL]. [2015-05-06]. https://software.intel.com/sites/default/ files/managed/0d/53/319433-022.pdf.
[42] Narayanan D, Hodson O. Whole-system persistence[C]//Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS). New York, NY, USA: ACM, 2012: 401-410.
[43] Zhao J, Li S, Yoon D H, et al. Kiln: Closing the performance gap between systems with and without persistence support[C]//Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture(MICRO). Davis, CA, USA: ACM, 2013: 421-432.
[44] Lu Y, Shu J, Sun L. Blurred persistence in transactional persistent memory[C]//Proceedings of the 31st International Conference on Massive Storage Systems and Technology (MSST). Santa Clara, CA, USA: IEEE, 2015: 1-13.
[45] Intel and Micro produce breakthrough memory technology[EB/OL]. [2015-05-06]. http://newsroom.intel.com/community/ intel_newsroom/blog/2015/07/28/intel-and-micron-produce-breakthrough-memory-technology.
[46] Volos H, Tack A J, Swift M M. Mnemosyne: Lightweight persistent memory[C]//Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). New York, NY, USA: ACM, 2011: 91-104.
[47] Coburn J, Caulfield A M, Akel A, et al. NV-Heaps: Making persistent objects fast and safe with next-generation, non-volatile memories[C]//Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). New York, NY, USA: ACM, 2011: 105-118.
[48] Wu X, Reddy A L. SCMFS: A file system for storage class memory[C]//Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC). Seattle, WA, USA: ACM, 2011: 39-52.
[49] Dulloor S R, Kumar S, Keshavamurthy A, et al. System software for persistent memory[C]//Proceedings of the 9th European Conference on Computer Systems (EuroSys). Amsterdam, The Netherlands: ACM, 2014: 15-28.
[50] Ou J, Shu J, Lu Y. A high performance file system for non-volatile main memory[C]//Proceedings of the 2016 European Conference on Computer Systems (EuroSys). London, UK: ACM, 2016: 12-25.

Research progress on non-volatile memory based storage system

SHU Jiwu, LU Youyou, ZHANG Jiacheng, ZHENG Weimin

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Abstract Non-volatile memory includes both sector-addressable flash memory and byte-addressable persistent memory. Flash memory has advantages like high performance, low energy consumption and light weight compared to hard disk drives, and brings opportunities to high-efficiency storage systems. Byte-addressable persistent memory (e.g., PCM, RRAM etc.), on the other hand, has high density over volatile DRAM and enlarges the memory space for multi-core CPUs. However, traditional storage systems fail to fully exploit the benefits of the emerging non-volatile memory. This paper first analyzes the challenges of storage system based on nonvolatile memory, and then discusses the revolutions in terms of storage architecture, system software and distributed protocols for flash memory and persistent memory, respectively. Finally, it points out the open problems for future research on storage systems based on the non-volatile memory.

Keywords non-volatile memory; flash memory; persistent memory; storage systems

（责任编辑　韩星明）