计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190700037

面向非易失性内存文件系统的NVM模拟与验证方法

王鑫鑫[1]　诸葛晴凤[2]　吴　林[1]
1 重庆大学计算机学院　重庆 400044
2 华东师范大学计算机科学与软件工程学院　上海 200062
(wang_xinx@cqu.edu.cn)

摘　要　现有非易失性内存文件系统都以DRAM模拟非易失性内存(Non-Volatile Memory,NVM)进行测试,而没有充分考虑两者间的写时延和写磨损特性差异,使得测试结果无法准确反映文件系统在NVM物理设备上的写性能以及对NVM造成的磨损情况。现有NVM模拟器准确度不高,且仿真接口不完备,无法满足内存文件系统对NVM的仿真需求。对此,提出一种面向非易失性内存文件系统的NVM模拟与验证方法。首先,结合非易失性内存文件系统本身的数据读写特性,提出内存文件系统中NVM写时延的模拟方案;其次,跟踪内存文件系统对NVM的读写操作,以验证文件系统对NVM物理设备的写磨损分布情况。选取多个典型内存文件系统实现上述方法。实验结果表明,提出的写时延模拟方法能够将写时延的模拟误差平均降低65%,写磨损验证方法能够较准确地反映内存文件系统对不同粒度NVM页面的磨损分布情况。

关键词:非易失性存储器;内存文件系统;写时延;磨损均衡
中图法分类号　TP316.89

Method for Simulating and Verifying NVM-based In-memory File Systems

WANG Xin-xin[1],ZHUGE Qing-feng[2] and WU Lin[1]
1 College of Computer Science,Chongqing University,Chongqing 400044,China
2 Shool of Computer Science and Software Engineering,East China Normal University,Shanghai 200062,China

Abstract　Most existing NVM-based file systems conduct experiments by simulating NVM with DRAM. However,they ignore the differences between NVM and DRAM,and make it impossible to accurately reflect the performance and wear distribution of file systems on NVM devices. The accuracy and interfaces of existing NVM emulators are not sufficient to support the simulation requirements of NVM-based file systems. This paper proposes a method for simulating NVM write latency and verifying wear distribution of NVM-based file systems. The accuracy of latency simulation is improved by injecting software-created delay according to the I/O characteristics of file system. To depict the wear distribution of NVM physical devices caused by the NVM-based file system,every update operation to NVM page is tracked. Experimental results show that the proposed method can reduce the error rate of write latency simulation by 65% on average,while accurately reflect the wear distribution of NVM.

Keywords　Non-volatile memory,In-memory file system,Write latency,Wear leveling

1　引言

新型非易失性内存,如相变存储器PCM[1]和3D Xpoint[2]等,由于具有诸多优秀的特性,逐渐发展成为存储级内存[3-5],并受到学术界和工业界的广泛研究。NVM可直连内存总线,并具有按字节寻址和非易失性等优良特性。以NVM为存储介质的新型内存架构,为文件系统的研究与设计带来了新的机遇和挑战[6-9]。目前学术界和工业界设计并实现了多个基于此架构的非易失性内存文件系统[10-15]。这类文件系统充分利用NVM可字节寻址和非易失性等特点,优化了传统面向磁盘文件系统的I/O软件栈,从而提高了文件系统的读写性能。

现有面向非易失性内存的文件系统在使用DRAM代替NVM进行测试的过程中,往往没有充分考虑两者之间的硬件特性差异,导致文件系统性能不足以反映真实情况。其中,DRAM与NVM之间的写时延差异能够对内存文件系统的性能产生较大的影响[16-18]。因此,在硬件技术尚未成熟的情况下,采用软件模拟的方式弥补两者之间的写时延差异,能够更为真实地反映出NVM内存架构之上的文件系统的写性能。

到稿日期:2019-07-03　返修日期:2020-01-06　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家高技术研究发展计划(863计划)(2015AA015304);国家自然科学基金(61472052)
This work was supported by the National High Technology Research and Development Program of China (2015AA015304) and National Natural Science Foundation of China (61472052).
通信作者:诸葛晴凤 (qfzhuge@gmail.com)

其次,区别于 DRAM,NVM 在写入新的数据之前须擦除已有的内容,且允许擦除操作的次数有限[19]。而现有内存文件系统在实现过程中很容易忽视 NVM 硬件的磨损情况。通过本文实验发现,非易失内存架构上的不同内存文件系统对 NVM 的写磨损情况差异极其显著,且均存在不同程度的磨损不均衡现象。由于现有硬件并未提供相应的软件接口,上层应用无法得知底层物理设备的实时磨损情况。因此,有必要在文件系统层面实现对 NVM 写磨损的验证方案,以便真实地反映 NVM 存储设备的磨损状况,从而推动内存文件系统的进一步优化。

现有部分 NVM 仿真模型不仅可扩展性较差,可能引入较大的性能损失;而且由于无法确定数据从高速缓存写回 NVM 的准确时刻,使得其模拟的准确度不足以满足内存文件系统对 NVM 写时延模拟的需求。与此同时,现有 NVM 模拟器所提供的仿真接口不完备,不足以支持内存文件系统对 NVM 写时延与写磨损的模拟。

综上所述,在使用或模拟 NVM 的过程中,DRAM 与 NVM 之间的写时延、写磨损差异是不可忽略的,并且可能对内存文件系统的具体实现和性能产生较大的影响。而现有模拟器不足以满足文件系统对 NVM 写时延和写磨损的仿真需求。因此,本文提出一种面向非易失内存文件系统的 NVM 模拟与验证方法,以有效构建内存文件系统在 NVM 写时延和写磨损方面的仿真需求。其中,写时延模拟与写磨损验证两部分内容互不影响,其共同目标均是使文件系统的性能更加接近于真实情况。本文的主要贡献如下。

(1)考虑内存文件系统本身对物理设备的数据读写特性,在保证数据一致性的流程中引入软件时延,以提高 NVM 写时延模拟的准确度。

(2)跟踪不同粒度 NVM 页面的写次数,模拟内存文件系统对 NVM 硬件的写磨损实时状态;区分内存文件系统中不同类型数据的更新操作导致的写磨损分布。

(3)在多个典型内存文件系统中实现上述写时延模拟和写磨损验证方法,使用标准测试工具 FIO[20] 和 Postmark[21] 进行测试。实验结果表明,本文提出的模拟方案能够将 NVM 写时延的模拟误差平均降低约 65%,提出的写磨损验证方法能够较准确地反映文件系统对 NVM 的写磨损分布情况。

2 相关工作

在硬件技术尚未成熟的情况下,许多研究者以硬件辅助或软件的形式建立 NVM 仿真模型[12,16-17,22],从而观察应用程序或系统软件在不同 NVM 硬件配置下的性能表现。

PMEP[12] 是 Intel 提出的持久性内存模拟平台,其将机器内存划分为普通内存和模拟的非易失性内存,以实现对不同 NVM 时延和带宽的模拟。NVM 与 DRAM 访问时延不同,会导致末级缓存失效(LLC miss)时处理器阻塞的时长不同。PEMP 在特定 CPU 微指令以及硬件的支持下,以固定周期计算由末级缓存失效导致的 CPU 阻塞时长,并按照 NVM 与 DRAM 之间的时延差异比例增加对应的软件时延。由于仿真过程受到特定硬件的限制,因此该方案的扩展性差。全系统模拟器(Full-system Simulators)是一类模块化的硬件仿真模型,如 Gem5[22] 通过模块化的方式对多种处理器模型、存储结构和网络等多个组件进行模拟。该类仿真工具的主要缺点在于难以很好地支持复杂应用程序或者系统软件,并且会带来较大的性能损失。

Quartz[16] 是一款轻量级的 NVM 性能模拟器,其将 NUMA 架构中远端节点的内存模拟为 NVM,并根据数据读写过程中高速缓存是否命中来记录对 NVM 的访问次数,从而得到 DRAM 模拟环境下应增加的软件时延。该方法通过周期性地读取硬件寄存器得到相应的状态信息,并以固定时间间隔增加时延。在多线程应用程序访问时延的模拟过程中,Quartz 根据多个线程之间的依赖关系,动态调整访问寄存器的时间间隔,从而达到较高的准确度。

类似地,HME[17] 也能够实现对不同 NVM 读写时延和带宽的模拟,并提供 nvm_malloc 等非易失性内存空间分配接口。其同样基于 NUMA 架构实现对 NVM 读写时延的模拟。与 Quartz 不同,考虑到多次访问寄存器可能会对本地节点上的应用程序产生较大的影响,HME 将访问硬件寄存器的任务单独放置在远端节点,并通过中断指令周期性地将计算结果返回至本地节点。其次,HME 模拟 NVM 读写时延时采用了不同的仿真模型。考虑到文件写流程中高速缓存的状态不同,其将 NVM 写操作划分为 Write back 和 Write through 两种类型,分别统计程序运行过程中两种类型写操作发生的次数,并增加相应的软件时延。

上述模拟器主要通过判断高速缓存是否命中,并区分不同的写操作类型,来实现对 DRAM 与 NVM 之间时延差异的模拟。但由于高速缓存内部的存储与控制逻辑全部由硬件实现,对用户完全透明,系统运行过程中从软件层面无法得知数据从高速缓存写回 NVM 的真正时刻,因此现有模拟方案的准确度不能满足系统软件对 NVM 写时延模拟的需求。此外,现有 NVM 模拟器所提供的仿真接口功能不完备,无法利用现有仿真工具实现内存文件系统中 NVM 写时延的模拟和硬件写磨损情况的验证。

3 系统设计与实现

本文提出面向非易失性内存文件系统的 NVM 写时延模拟和写磨损验证方法。首先,结合内存文件系统在文件读写过程中对非易失性存储器的访问特性,如读写粒度和访问时间等,在文件系统保证数据一致性的流程中引入软件时延,以实现对不同 NVM 读写时延的模拟,并验证 NVM 时延效应对文件系统写带宽的影响。其次,跟踪内存文件系统读写过程中对 NVM 的更新操作,动态反映内存文件系统对不同粒度 NVM 页面的写磨损情况,并对不同类型更新操作导致的磨损进行区分和对比。

如图 1 所示,本文将上述模拟和验证方法实现于内核态文件系统。Latency Emulator 和 Page Counter 分别代表写时延模拟和写磨损验证两个独立模块。应用程序运行在文件系统之上,并通过标准的文件读写接口访问 NVM 设备。写时延模拟和写磨损验证的详细设计方案分别见 3.1 节和 3.2 节。

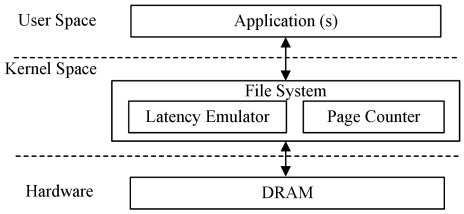


图 1 系统整体框架

Fig. 1 System framework of proposed method

本文提出的模拟和验证方法与现有 NVM 模拟器存在以下区别。

(1)现有开源模拟器均是在存储设备层模拟读写时延,并提供应用程序编程接口。上层应用需要修改和重新编译自身代码才能进行 NVM 的模拟。而本文提出的模拟和验证方法是基于非易失性内存文件系统的,这使得上层应用程序可直接在文件系统之上运行,无须修改和重新编译自身代码,从而有效节省了开发成本。

(2)在提高写时延模拟准确度的基础上,能够完成对非易失性存储器写磨损情况的验证,并且所提方法具有较强的通用性,能够适用于现有大部分非易失性内存文件系统。

3.1 写时延仿真模型

NVM 写时延仿真的关键在于如何精确统计对 NVM 的访问。由于高速缓存的存储与管理机制对软件层面完全透明,且其采用的置换策略和底层优化技术相对复杂,使得现有模拟方案不足以满足这一基本要求。因此,本文提出一种基于非易失性内存文件系统的 NVM 写时延模拟方案。

现有非易失性内存文件系统在保证数据一致性的过程中,往往需要通过 clflush 等指令适时地将数据写入 NVM,并且数据读写过程中通常涉及对文件元数据和日志信息的更新操作。本文利用这一特性实现内存文件系统中对 NVM 写时延的模拟。在内存文件系统保证数据一致性的流程中,根据对 NVM 的读写粒度和访问次数引入相应的软件时延,以更加准确地实现对不同 NVM 写时延的模拟。记 Δ 为文件系统写操作流程中须引入的软件时延,其仿真模型为:

$$\Delta=\delta*\lceil len/m\rceil*(NVM_{\text{lat}}-DRAM_{\text{lat}}) \tag{1}$$

其中,NVM_{lat} 与 $DRAM_{\text{lat}}$ 分别表示 NVM 与 DRAM 硬件的写时延;写操作过程中通过 clflush 指令写入 NVM 的数据量为 len,以 m 为粒度计算 NVM 与 DRAM 之间的写时延差异;δ 是与应用程序运行过程中末级缓存失效(LLC miss)比例有关的变量,用于确保稳定的时延模拟准确度。δ 的取值可通过对多次运行过程中的末级缓存失效比例进行建模分析得到。同时,由于内存文件系统写流程中除调用 clflush 之外,还可能涉及对文件元数据和日志信息的多次更新,因此在上述模型的基础上,须根据对 NVM 的访问次数增加额外的软件时延,以达到仿真的目的。

实际应用环境中,由于 NVM 硬件可能存在不同的底层细节以及读写特性,因此可根据实际情况对仿真模型中的参数进行调整。该时延仿真方案的核心思想是在文件系统更新 NVM 数据的流程中增加软件时延,与文件系统的具体实现方式相耦合。但由于不同文件系统均遵循 VFS 的读写接口,因此该模拟方案能够通用于现有的大多数非易失性内存文件系统。

3.2 写磨损验证模型

现有大多数非易失性内存文件系统在设计与实现过程中的重点在于提高文件系统的读写性能和实现一致性等,往往容易忽略对底层物理设备的磨损情况,导致在大量文件读写操作之后出现对 NVM 设备磨损不均衡的现象,进而缩短了 NVM 的使用寿命。与此同时,硬件层面并没有提供相应的接口,使得上层应用无法得知底层设备的实时磨损情况,尤其是不同粒度页面的实时磨损情况。基于以上分析,本文提出内存文件系统中对 NVM 写磨损的验证方法。

通过跟踪文件系统运行过程中对物理设备的所有更新操作,模拟并展示不同文件系统对非易失性存储器的写磨损分布情况,进一步分析现有内存文件系统存在的不足。其中,文件系统对 NVM 设备的更新操作主要包含其对文件数据、元数据以及日志信息的更新操作。具体做法是在图 1 所示的 Page Counter 模块中,为文件系统所在的 NVM 物理区间建立多个访问计数器,并在文件系统向 NVM 写入数据时增加相应位置上的计数器。由于现有基于文件系统的 NVM 磨损均衡方案均以固定大小的块粒度为基本单元,因此该方案能够支持对不同粒度 NVM 页面的写磨损分布情况的验证。

上述写磨损验证模型能够适用于不同非易失性内存文件系统。具体地,由于文件系统组织文件数据的索引结构不同,因此写流程中对文件数据页面的磨损不同,须根据文件系统的具体实现方式对数据页面进行定位。如 PMFS[12] 需要对 B-Tree 按层次遍历;SIMFS[15] 则需要根据文件页表结构得到页面的具体物理位置。同样,元数据和日志页面作为保证数据完整性和一致性的重要数据,需要根据文件系统的不同管理方式更新其访问计数器。此外,上述写磨损验证模型能够有效区分不同类型数据的更新操作导致的 NVM 磨损,如内存文件系统对文件数据、元数据和日志等数据的更新操作。

3.3 具体实现

本文提出的写时延模拟和写磨损验证方法均实现于文件系统层面,而非硬件层面。因此,理论上该方案能够适用于所有直连内存总线并可按字节访问的新型非易失性存储设备。实现过程中,需要针对模拟的硬件配置信息对模型中的参数进行相应的调整,以提高方案的准确度。

本文选取了多种典型的非易失性内存文件系统实现上述模拟和验证方法。以内存文件系统 SIMFS[15] 为例,实现写时延模拟方法时,须在更新文件数据、元数据和日志的多个位置,根据对 NVM 的读写粒度及访问次数增加相应的软件时延。由于 NVM 与 DRAM 的访问时延均为纳秒级别,因此实现过程中利用 x86 平台的读取时间戳指令,以达到纳秒级精度模拟访问时延的目的。具体做法是,利用 x86 平台的 RDTSC 指令,读取当前处理器的时间戳。该指令得到的时间戳在每个 CPU 时钟周期都会加 1,是软件能够达到的最高精度。RDTSC 的时间戳间隔与当前 CPU 的频率有关,因此在增加时延的过程中须综合考虑当前 CPU 频率和 NVM 与 DRAM 的写入速度差异。

同样,在内存文件系统 SIMFS 中实现写磨损验证方法,需根据该文件系统对文件数据、元数据和日志的具体组织形式,在其更新数据所在物理页面之后增加相应的访问计数器。

SIMFS 使用文件页表管理和组织文件数据,并将文件元数据所在区域固定在 super block 区域之后,因此对文件数据和元数据所在页面的定位可通过地址偏移计算得出。此外,NVM 写磨损验证过程会引入额外的系统开销,且粒度越小,开销越大。因此,使用时可根据实际需求选择不同粒度的写磨损验证方案。具体地,以文件系统中的 inode 区域和数据区域为例[18,23]。文件系统往往以 inode 为单位对单个文件的元数据进行更新,因此在对 inode 区域进行磨损验证时,较小的粒度更有利于观察当前页内的磨损情况。而对于数据区域,其更新操作一般以 4kB 大小的页为单位,因此磨损粒度应随之增加。

4 系统测试及分析

本文选取多个典型内存文件系统实现上述写时延模拟和写磨损验证方法,对比不同写时延模拟方法的准确度,并指出 NVM 时延效应对不同文件系统写带宽的影响;模拟并验证多个内存文件系统对物理设备的写磨损情况。实验使用的 NUMA 架构服务器的详细配置信息如表 1 所列。

表 1 服务器配置信息

Table 1 Server configuration

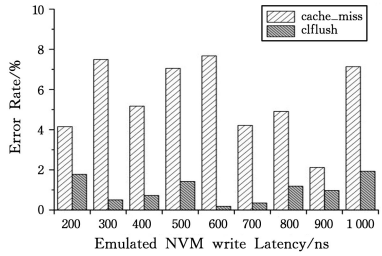
参数	取值
CPU 型号	Intel Xeon E5-2620 2.4 GHz
内存大小和主频	128 GB 2133 MHz
节点数	2
系统与内核版本	Ubuntu 15.04 Linux 4.4.30

4.1 写时延模拟方案的对比

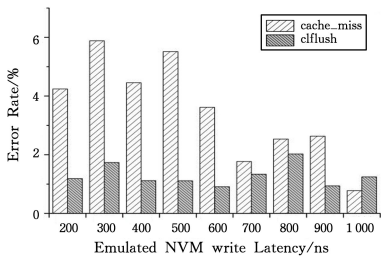
本文实现了两种 NVM 写时延模拟方案,分别是本文提出的基于 clflush 的写时延模拟方案和现有模拟器中 NUMA 架构下基于高速缓存的模拟方案。首先,以内存文件系统 SIMFS 为例,比较不同写时延模拟需求下两种方案的准确度。

将文件系统运行在 DRAM 上的写时延作为基准值 L_1,模拟之后的写时延为 L_2,将两者之差与理论计算得出的软件时延 D 作对比,从而计算得出写时延模拟方案的模拟误差。模拟误差的计算公式为:$Error\ rate = abs((L_2 - L_1) - D)/D$,其中 $abs()$ 表示取绝对值。模拟误差越小,表明模拟的准确度越高。实验过程中,以 100 ns 为间隔,选取 200~1000 ns 中不同大小的时延值作为模拟的 NVM 写时延,并使用标准测试工具 FIO[20] 得到文件系统的不同 I/O 大小写时延。

以 4 kB 为粒度对文件进行写操作,测试不同 NVM 写时延下两种方案的模拟误差,结果如图 2(a)所示。可以看出,本文提出的写时延模拟方案得的模拟误差明显低于现有方案,模拟误差平均降低了 78.7%。I/O 粒度为 1 MB 时,两种方案的模拟误差对比如图 2(b)所示。两者的模拟误差最大值分别为 2.02%和 5.88%,且在大多数情况下本文提出的基于 clflush 的写时延模拟方案能够实现较低的模拟误差,并且结果相对稳定。基于高速缓存的写时延模拟方法误差较高的主要原因在于无法准确判断文件系统数据读写过程中高速缓存是否命中,并且模拟过程中高速缓存的状态可能受到处理器上其他应用程序的影响。



(a)I/O 粒度为 4 kB



(b)I/O 粒度为 1 MB

图 2 不同 NVM 写时延模拟方案的误差对比

Fig. 2 Error rate of different NVM write latency simulation method

表 2 列出了模拟 NVM 写时延为 500 ns 时,不同写粒度情况下两种写时延模拟方案的对比情况。将文件系统运行在内存上的写时延作为基准值,分别计算两种方案的模拟误差。

表 2 模拟 NVM 写时延为 500 ns 时两种仿真方案的误差对比

Table 2 Error rate of different simulation schemes (NVM write latency=500 ns)

写粒度	加入时延之前/us	应增加的软件时延/us	cache miss 方案模拟之后的时延/us	误差/%	clflush 方案模拟之后的时延/us	误差/%
4 kB	1.96	3.38	5.57	7.05	5.38	1.42
8 kB	2.86	6.38	8.69	8.50	9.12	1.82
16 kB	5.58	12.38	19.07	8.99	17.65	2.46
32 kB	10.48	24.38	36.40	6.35	33.46	5.71
64 kB	18.30	48.38	63.03	7.53	66.63	0.10
128 kB	40.13	96.38	124.39	12.57	131.83	4.85
256 kB	78.18	192.38	245.87	12.83	268.36	1.14
512 kB	153.53	384.38	516.17	5.65	521.68	4.22
1 MB	292.84	768.38	1103.58	5.51	1069.76	1.11
2 MB	672.22	1536.38	2103.00	6.87	2217.12	0.56
均值	—	—	—	8.19	—	2.34

从表可以看出,平均情况下本文提出的模拟方案能够将写时延模拟误差降低为现有方案的 31.05%。此外,由于现

有 NVM 硬件技术尚未成熟，因此本文暂未将模拟性能与文件系统运行在真实硬件之上的性能进行对比。从实验数据可看出，本文提出的方法与现有模拟方案在写时延模拟之后的性能变化趋势大致相同。而对于特定硬件的模拟，则可对模拟过程中的参数进行相应的调整，以提高模拟方案的准确度。综上分析，在模拟 NVM 不同写时延的过程中，考虑文件系统本身对 NVM 的读写粒度和访问次数等特性，能够更加准确地实现对 NVM 访问时延的模拟，并且避免对硬件寄存器的频繁访问，有效降低模拟过程中产生的额外系统开销。

另外，本文选取内存文件系统 PMFS[12]，NOVA[13] 和 SIMFS[15] 来实现基于 clflush 的 NVM 写时延模拟方案，以验证不同 NVM 写时延对文件系统写带宽的影响。NVM 写时延分别为 0 ns，200 ns 和 500 ns 时，不同文件系统的写带宽随 I/O 粒度大小的变化情况如图 3 所示。

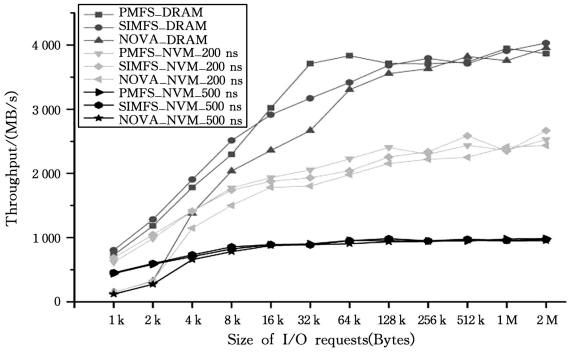


图 3 NVM 写时延对文件系统写带宽的影响

Fig. 3 Impact of NVM write latency on file system writeper for mance

从图 3 可以看出，不同 NVM 访问时延对文件系统的写带宽有着极其显著的影响。

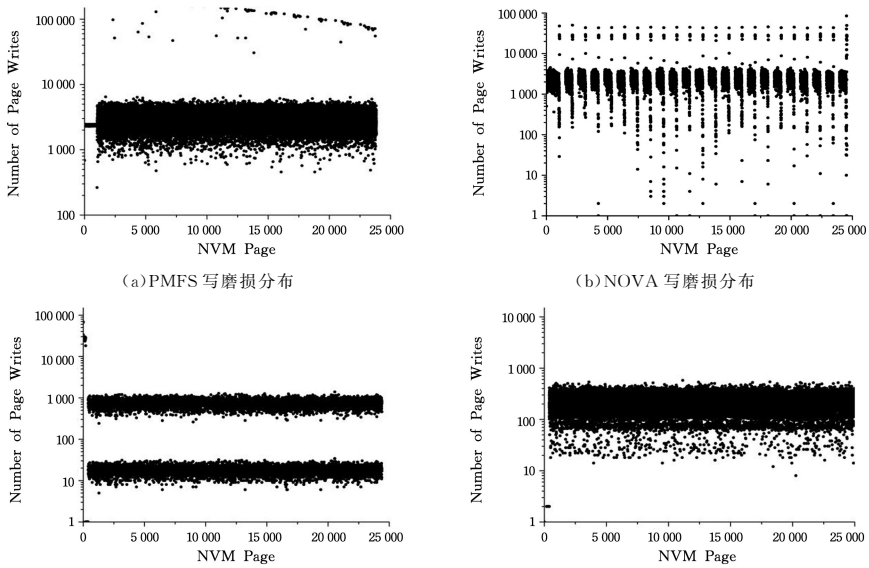
NVM 写时延为 200 ns 和 500 ns 时，文件系统写带宽平均约为无额外时延情况的 1/2 和 1/3。同时，随着 NVM 写时延的增加，不同文件系统之间的写带宽差异逐渐减小。NVM 写时延为 500 ns 时，3 种内存文件系统的写带宽基本一致。其主要原因是硬件访问时延的增加，使得文件系统自身软件栈优化技术所带来的性能提升逐渐减弱。NVM 写时延模拟器的使用，能够更为真实地反映非易失性内存架构之上的内存文件系统的数据读写性能，进而推动非易失性内存文件系统对自身软件栈做出进一步的优化。

4.2 写磨损方案的验证

选取内存文件系统 PMFS，NOVA 和 SIMFS 来实现本文提出的写磨损验证方法，并分析不同文件系统在大量文件读写过程中对 NVM 的写磨损分布情况。使用测试工具 Postmark[21] 向目标文件系统连续提交 100 万次文件访问请求，统计读写结束之后每个 NVM 页面的写次数，以及文件数据、元数据和日志的写磨损情况。每个文件的大小为 4 kB，I/O 操作的大小为 100 B，读写的比例为 1∶9。

当 NVM 页面大小为 4 kB 时，不同内存文件系统对 NVM 物理页面的写磨损分布情况如图 4 所示。图中横坐标以内存页面为单位，纵坐标表示每个物理页面的写次数。

从图 4 中可看出，3 种文件系统均存在不同程度的磨损不均衡现象。其中，内存文件系统 PMFS 对物理页面的写磨损分布情况如图 4(a)所示。由于每次读写操作均会导致文件元数据的更新，因此元数据所在区域的写次数明显高于其他页面，约为100 000 次，而剩余页面的写次数约为 1 000～10 000 次。



(a)PMFS 写磨损分布 (b)NOVA 写磨损分布

(c)SIMFS 写磨损分布 (d)实现磨损均衡之后的 SIMFS 写磨损分布

图 4 不同文件系统对 NVM 的写磨损分布情况(页面大小为 4 kB)

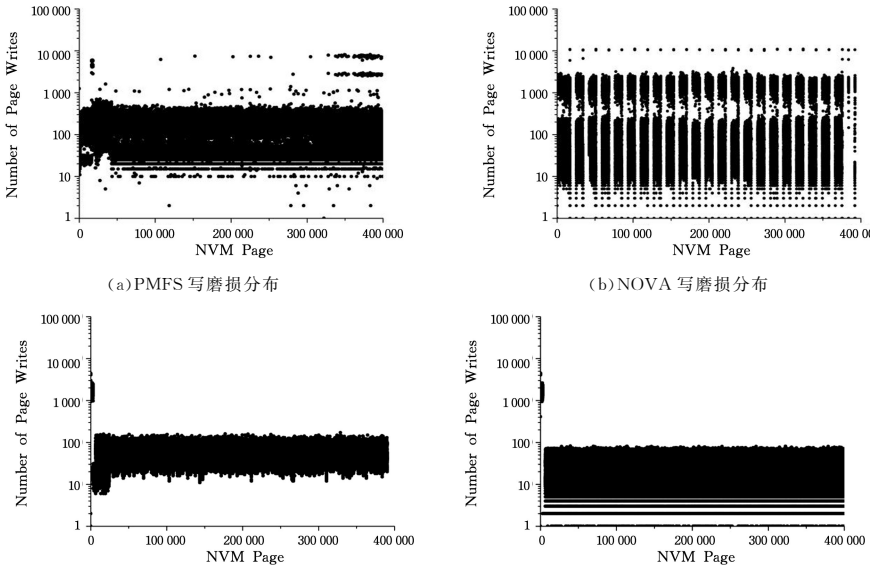
Fig. 4 NVM wear distribution by different file systems (page size=4 kB)

NOVA 文件系统同样存在上述问题，使得少数页面的平均写次数明显大于其他文件数据页面的写次数。同时，NOVA 文件系统将空闲数据块划分为多个子列表，并根据 I/O 请求从不同的子列表中分配空闲页，因此产生了如图 4(b)所示的条状磨损分布情况，该部分页面的写次数分布约在 1～3 500 之间。

内存文件系统 SIMFS 使用文件页表的形式对文件数据进行组织和管理,因此在大量读写过程中导致多级页表中较高等级的页面写次数较少,而较低等级的页面承受的写次数较多,使得对物理页面的写次数相差较为明显,如图 4(c)所示。其中一部分页面的写次数约为 1000 次,而剩余页面的写次数约为 10～100 次。同时,由于 SIMFS 将文件元数据区域固定在 super block 区域之后,因此文件元数据所在页面的平均写次数远高于文件数据和日志页面的平均写次数。

上述实验分析充分暴露了 3 种文件系统对 NVM 的写磨损不均衡问题。以 SIMFS 为例,选取实现磨损均衡之后的文件系统写磨损分布情况作为对比,如图 4(d)所示。可明显看出,实现磨损均衡之后的文件系统能够达到较为均衡的写磨损分布。该部分着重说明本文提出的写磨损验证方案能够较好地体现文件系统在不同优化方案下的写磨损状态,从而促进对文件系统的进一步优化。而具体的磨损均衡方法或文件系统优化方法不在本文的讨论范围之内。

在页面大小为 256B 时,不同文件系统对 NVM 的写磨损分布情况如图 5 所示。可以看出,随着统计粒度的减小,文件系统所在物理区间的平均写次数均有所下降,但仍能够反映出文件系统对物理设备的基本磨损状况。



(a)PMFS 写磨损分布 (b)NOVA 写磨损分布

(c)SIMFS 写磨损分布 (d)实现磨损均衡之后的 SIMFS 写磨损分布

图 5 不同文件系统对 NVM 的写磨损分布情况(页面大小为 256B)

Fig. 5 NVM wear distribution by different file systems (pagesize=256B)

综上所述,不同类型的内存文件系统在读写过程中对 NVM 的磨损分布情况存在较大的差异,因此有必要对 NVM 写磨损情况进行验证。而目前软件层面想要得知硬件内部的真实磨损状况较为困难,因此如何有效判定写磨损验证方法的准确性仍有待进一步研究。写磨损验证方法的使用,使得非易失性内存文件系统在大量读写过程中能够暴露出在文件数据、元数据和日志页面管理过程中存在的部分问题。在实际的空闲页面分配过程中,应尽可能采用动态分配[18]的策略,优先选择写次数较少的页面,从而改善物理设备的写磨损分布情况。

结束语 本文针对非易失性内存文件系统在 NVM 写时延和写磨损方面的仿真需求,设计并实现了面向非易失性内存文件系统的 NVM 模拟和验证方法。首先,在文件系统保证数据一致性的流程中引入软件时延;其次,提出能够准确刻画 NVM 写磨损分布情况的验证方法。实验结果表明,提出的写时延模拟方法能够有效降低模拟误差,并且写磨损验证方案能够较准确地反映文件系统在不同优化状态下的写磨损分布情况。然而,由于硬件技术尚未成熟,本文目前并未将所提方案的性能与文件系统运行在真实硬件之上的性能进行对比。未来的研究工作中,我们将进一步完善模拟与验证模型,使其能够达到更高的模拟准确度。

参 考 文 献

[1] LEE B C,ZHOU P,YANG J,et al. Phase-change technology and the future of main memory[J]. IEEE Micro,2010,30(1):143-143.

[2] Intel Optane Memory[EB/OL]. https://www.intel.com/content/www/us/en/architecture-and-technology/optane-memory.html.

[3] FREITAS R F,WILCKE W W. Storage-class memory:The next storage system technology[J]. IBM Journal of Research and Development,2008,52(4/5):439-447.

[4] JUNG M,SHALF J,KANDEMIR M. Design of a large-scale storage-class RRAM system[C]// Proceedings of the 27th international ACM Conference on International Conference on Supercomputing. ACM,2013:103-114.

[5] XU C,CHEN P Y,NIU D,et al. Architecting 3D vertical resistive memory for next-generation storage systems[C]// 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE,2014:55-62.

[6] LUO L,LIU Y,QIAN D P. Survey on in-memory computing

technology[J]. Ruan Jian Xue Bao/Journal of Software, 2016, 27(8): 2147-2167.

[7] ARULRAJ J, PAVLO A, DULLOOR S R. Let's talk about storage & recovery methods for non-volatile memory database systems[C]// Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015: 707-722.

[8] 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. IEEE Computer Society, 2010: 385-395.

[9] WU C, ZHANG G, LI K. Rethinking computer architectures and software systems for phase-change memory[J]. ACM Journal on Emerging Technologies in Computing Systems, 2016, 12(4): 33.

[10] 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. ACM, 2009: 133-146.

[11] 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. ACM, 2011: 39.

[12] DULLOOR S R, KUMAR S, KESHAVAMURTHY A, et al. System software for persistent memory[C]// Proceedings of the Ninth European Conference on Computer Systems. ACM, 2014: 15.

[13] XU J, SWANSON S. NOVA: a log-structured file system for hybrid volatile/non-volatile main memories[C]// Proceedings of the 14th Usenix Conference on File and Storage Technologies. USENIX Association, 2016: 323-338.

[14] OU J, SHU J, LU Y. A high performance file system for non-volatile main memory[C]// Proceedings of the Eleventh European Conference on Computer Systems. ACM, 2016.

[15] SHA E H M, CHEN X, ZHUGE Q, et al. A new design of in-memory file system based on file virtual address framework[J]. IEEE Transactions on Computers, 2016, 65(10): 2959-2972.

[16] VOLOS H, MAGALHAES G, CHERKASOVA L, et al. Quartz: A lightweight performance emulator for persistent memory software[C]// Proceedings of the 16th Annual Middleware Conference. ACM, 2015: 37-49.

[17] DUAN Z, LIU H, LIAO X, et al. HME: A lightweight emulator for hybrid memory[C]// 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2018: 1375-1380.

[18] WU L, ZHUGE Q, SHA E H M, et al. DWARM: A wear-aware memory management scheme for in-memory file systems[J]. Future Generation Computer Systems, 2018, 88: 1-15.

[19] SHEN Z R, XUE W, SHU J W. Research on New Non-Volatile Storage[J]. Journal of Computer Research & Development, 2014, 51(2): 445-453.

[20] Fio: Flexible I/O Tester[EB/OL]. http://freecode.com/projects/fio.

[21] PostMark[EB/OL]. http://openbenchmarking.org/test/pts/postmark.

[22] BINKERT N, BECKMANN B, BLACK G, et al. The gem5 simulator[J]. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1-7.

[23] CHEN X, SHA E H M, ZENG Y, et al. Efficient wear leveling for inodes of file systems on persistent memories[C]// 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2018: 1524-1527.

WANG Xin-xin, born in 1996, postgraduate. His main research interests include non-volatile memory and in-memory file systems.

ZHUGE Qing-feng, born in 1970, Ph.D, professor, Ph.D supervisor, is a member of China Computer Federation. Her main research interests include parallel architectures, embedded systems, supply-chain management, real-time systems, optimization algorithms, compi-lers, and scheduling.