本科毕业论文（设计）

基于差量压缩的 QLC-SSD 寿命优化关键技术研究

RESEARCH ON KEY TECHNOLOGIES FOR LIFESPAN OPTIMIZATION OF QLC-SSD BASED ON DELTA COMPRESSION

杨大荣

哈尔滨工业大学

2024 年 **5** 月

密级：公开

本科毕业论文（设计）

基于差量压缩的 QLC-SSD 寿命优化关键技术研究

本　　科　　生：杨大荣

学　　　　号：200110727

指 导 教 师：夏文教授

　　邹翔宇助理教授

专　　　　业：计算机科学与技术

学　　　　院：深圳校区计算机学院

答 辩 日 期：2024 年 5 月

学　　　　校：哈尔滨工业大学

摘　要

以 NAND 闪存技术为核心的固态硬盘近年来得到了飞速发展。新一代闪存技术四级单元固态硬盘（Quad-level Cell Solid-State Drives, QLC-SSD）带来了可观的存储密度，但其寿命急剧下降，成为了限制其全面普及的最大挑战。针对这一问题，本文观察到现实世界数据往往具有高度的冗余性和局部相似性，同时在高密度的 NAND 闪存上具有多级电压编程的特性，以及不必要的异地更新是 NAND 闪存上寿命消耗的关键因素，而现有的工作并没有很好地兼顾空间和寿命来充分解决 QLC-SSD 上面临的寿命问题。

本文提出 IUDW 系统，该系统：（1）利用现实数据冗余特性和数据消冗技术来优化 QLC-SSD 的寿命问题。本文采用基于差量压缩技术的页面更新数据的去重优化方案实现了 NAND 闪存上的细粒度数据更新，并采用基于无损压缩技术的页面内部数据消冗及其管理方案来平衡空间和性能的开销。（2）创新地将高密度闪存上的多级电压编程特性和数据消冗技术相结合。本文基于多级电压编程特性来进行差量的存储，在高密度闪存上实现原地更新。（3）创新地提出了适用于高密度闪存上且无需额外空间开销的混合更新框架。本文结合上述技术，实现了最大限度的就地更新策略，减少所有不必要的寿命消耗。

实验结果表明：在微基准测试中，IUDW 系统相对于最先进的方案在单个页面上能够带来 5-100 倍的更新次数提升；在公开的真实世界负载下，相对于原始方案，IUDW 系统能够减少 70% 空间消耗和 90% 的寿命消耗；在开销评估中，IUDW 系统的读时延开销约在 9%，编程时延开销约在 5.7%。

关键词：四级单元固态存储器；NAND 闪存；差量压缩；寿命优化

Abstract

In recent years, solid-state drives based on NAND flash technology have undergone rapid development. The emergence of the next-generation flash technology, Quad-level Cell Solid-State Drives (QLC-SSD), has significantly increased storage density. However, its lifespan has sharply declined, becoming the major challenge limiting its widespread adoption.

This thesis observes the high redundancy and local similarity in real-world data, the multi-level voltage programming characteristics of NAND high-density flash storage, and identifies unnecessary out-of-place updates as the key factors contributing to lifespan degradation on NAND flash storage. Existing works have not adequately balanced space and lifespan considerations to effectively address the lifespan issues faced by high-density flash storage technology, particularly QLC-SSD.

This thesis introduces the IUDW system, which: (1) optimizes the lifespan issues of QLC-SSD by utilizing the redundancy characteristics of real-world data and data deduplication techniques. The project implements a fine-grained data update scheme on NAND flash through a delta compression-based page update data deduplication optimization, achieving balanced space and performance overhead through a lossless compression-based internal data deduplication and management scheme. (2) Innovatively combines the multi-level voltage programming characteristics and data deduplication techniques on high-density flash storage. The project utilizes multi-level voltage programming characteristics for delta storage, achieving in-place updates on high-density flash storage. (3) Innovatively proposes a hybrid update framework suitable for high-density flash storage without additional space overhead. By integrating the above technologies, the project implements a maximum in-place update strategy, reducing all unnecessary lifespan consumption.

Experimental results demonstrate significant improvements with the IUDW system. In micro-benchmark tests, relative to state-of-the-art solutions, the IUDW system can bring about a 5-100 times increase in update frequency on a single page. Under publicly available real-world workloads, the IUDW system reduces space consumption by 70% and lifespan consumption by 90% compared to the original scheme. Overhead evaluation indicates minimal read latency overhead (approximately 9%) and programming latency overhead (approximately 5.7%).

Keywords: quad-level cell solid state drives, NAND flash, delta compression, lifetime optimization

目　录

第 1 章 绪论

1.1 课题背景及研究的目的和意义

1.1.1 固态硬盘的发展与挑战

数据的爆炸增长与高效数据存储之间存在矛盾。近年来，随着人工智能、AIoT、移动物联网等技术的推动，全球的数据规模呈现爆炸式的增长。据国际数据公司 IDC 统计显示，预计到 2025 年，全球数据量将达到 160ZB 的量级[1]。这些海量数据的高效存储成为一个具有挑战性的难题。

高密度闪存存储技术逐渐成为数据存储的主流。历经多代的发展，存储技术从光盘技术、磁带技术、磁性存储技术发展到现在的 NAND 闪存技术。目前使用最为广泛的是以机械硬盘（**H**ard **D**isk **D**rive，HDD）为代表的磁性存储技术和以固态硬盘（**S**olid **S**tate **D**rive，SSD）为代表 NAND 闪存技术。近年来，NAND 闪存技术得到了突飞猛进的发展。如图1-1所示，NAND 闪存的出货占比从 2010 年的仅不到 10%，到 2025 年预估达到 40%，并仍有进一步增长的趋势。以 NAND 闪存技术为核心的固态硬盘比机械硬盘具备更快的访问速度，在诸多场景有逐渐替代机械硬盘的发展趋势。近年来，凭借固态硬盘优秀的访问速度，存储厂商提出了内存语义的固态硬盘（**C**ompute **E**xpress **L**ink **S**olid **S**tate **D**rive，CXL-SSD）来提供大内存方案，进一步拓展了固态硬盘使用场景[2,3]。

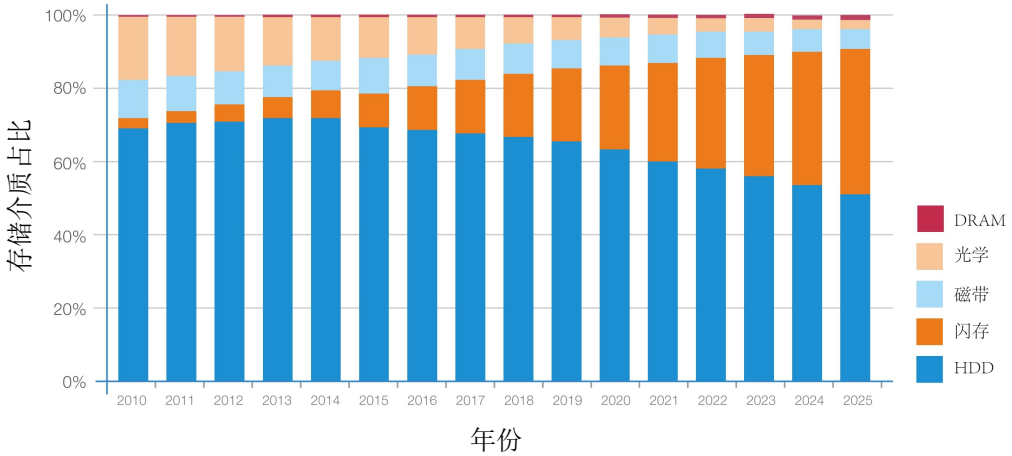


图 1-1　各种存储介质的出货占比[1]

QLC-SSD 兼具低成本和高吞吐的双重优势。固态硬盘存储介质的 NAND 闪存颗粒依次经历了从 SLC、MLC、TLC，以及到最新一代 QLC 颗粒的发展。如图1-2所

示，SLC（**S**ingle **L**evel **C**ell）颗粒表示每个存储单元能存储 1 比特数据，依此类推，QLC 颗粒表示每个存储单元可以存储 4 比特数据。从 SLC 到 QLC 颗粒，每个存储单位所能存储的位数不断增加，带来了更高的存储密度，极大降低了固态硬盘的成本。以 QLC 闪存颗粒为存储介质的固态硬盘（Quad-level Cell Solid-State Drives，QLC-SSD），有望能实现和机械硬盘一样的大容量和低成本，同时还兼具固态硬盘所具备的高速访问速度[4]。QLC-SSD 视为可以全面替代机械硬盘用作下一代主流的存储设备。

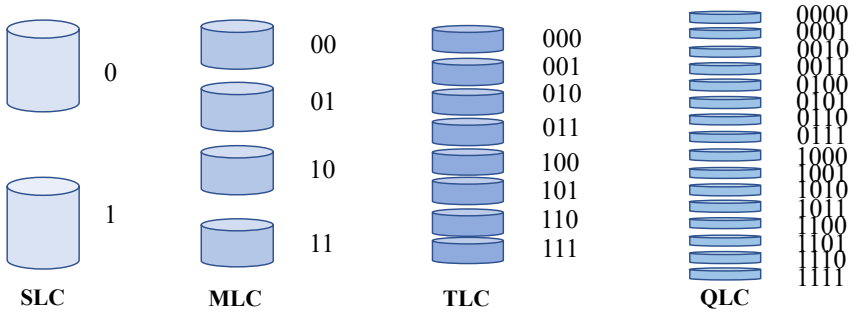


图 1-2　NAND 颗粒介绍及发展

第四代固态硬盘 QLC-SSD 的全面普及正面临寿命上的严峻挑战。从 SLC 颗粒到 QLC 颗粒，带来了可观的存储容量的同时，其使用寿命也在急剧下降。NAND 闪存的使用寿命和可擦除次数（**P**rogram/**E**rase Cycles，P/E Cycles）有关。如图1-3所示，从 SLC 颗粒到 QLC 颗粒，在不同制造工艺下，寿命从 5000-10000 的可擦除次数下降到了 70-800①，骤降了几个数量级。QLC-SSD 的寿命骤降成为了限制其全面普及的一个关键因素。

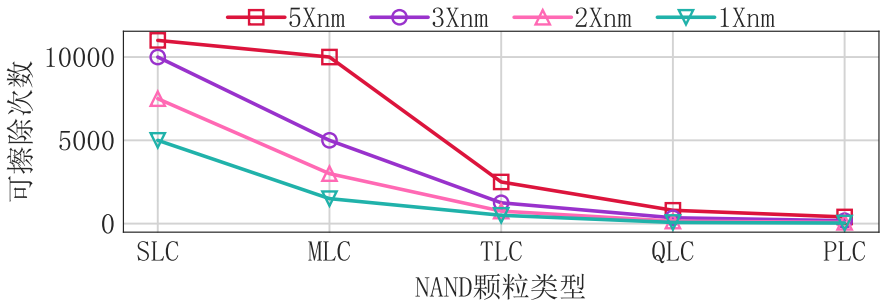


图 1-3　NAND 颗粒可擦写次数

1.1.2 固态硬盘存储结构和 I/O 特性

固态硬盘的内部存储结构：固态硬盘的闪存介质有寿命限制并且 I/O 单元与

① WD and Tosh talk up penta-level cell flash, https://blocksandfiles.com/2019/08/07/penta-level-cell-flash/

传统扇区概念不同，因此其内部通常除了存储介质 NAND 闪存外，还需要额外的控制器充当管理和转换层。如图1-4所示，固态硬盘的内部可以划分为：（1）固态硬盘的控制器。其核心是闪存转换层（**F**lash **T**ranslation **L**ayer，FTL），同时还包括与其相连接的硬件设备，如微处理器和 DRAM，在部分计算型存储设备上，可能还有专用硬件计算引擎。（2）闪存芯片阵列。其负责固态硬盘的全部数据存储，受闪存转换层的管理。为了便于管理和提高并行，闪存芯片阵列通常进一步设计为多维度的管理结构：通道、芯片、平面、块、页。

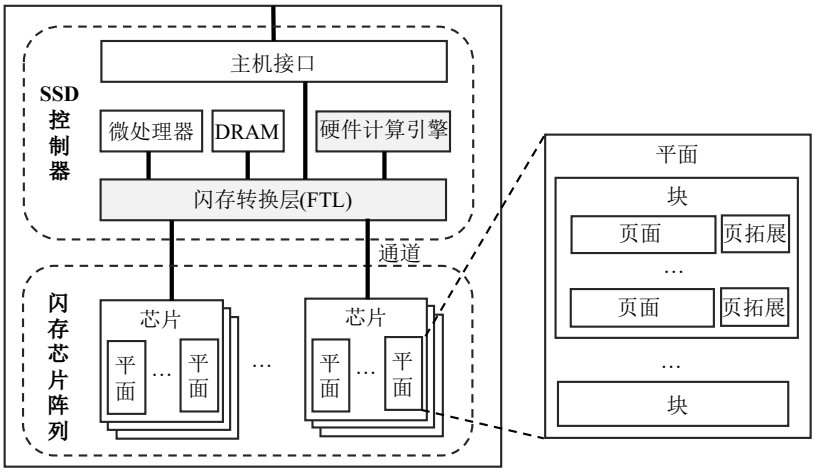


图 1-4　固态硬盘内部存储结构

固态硬盘上的 I/O 特性：（1）固态硬盘上的操作与操作单元。在固态硬盘中，通常包括读、编程和擦除操作。读操作的粒度通常为一个页，编程操作的粒度也为一个页，而擦除操作的粒度为一个块。读写操作和擦除操作的粒度并不一致，读写操作粒度更小，擦除操作粒度更大。（2）存储单元的编程原理及编程的单向性。NAND 闪存中的存储单元通过将存储单元的电压提高并保持到一定的阈值来表示不同的数据。电压的提高过程是单向递增且不可逆的。编程到最大电压 V_{max} 的存储单元只有通过擦除操作让电压回到最低水平才能再次编程[5,6]。（3）固态硬盘上的异地更新。固态硬盘上的存储单元通过擦除才能再次进行编程。如果需要对编程过的页面进行修改操作，需要先擦除这个页面然后才能重新执行写入操作。如表1-1所示，擦除操作相对于写操作的时延增加了几个数量级，会严重影响写入的性能。因此固态硬盘采用的方案是，重新分配一个未被编程过的新页面来执行写入操作，而暂时不对原来的页面进行擦除操作，这个方案称为异地更新。原来的页面会标记为失效，等到固态硬盘的闪存阵列上的未被编程页面接近没有时，固态硬盘再集中地进行一次擦除操作，这个过程被称为垃圾回收。

表 1-1　NAND 闪存颗粒的操作时延[7]

特性	SLC	MLC	TLC	QLC
读时延	25μs	50μs	75μs	100μs
写时延	200-300μs	600-900μs	900-1350μs	1500μs
擦除时延	1.5-2ms	3ms	5ms	6ms

1.1.3 数据消冗和差量压缩

现实中的数据存在着大量冗余。这体现在：（1）不同份数据之间会存在冗余。不同的数据块间往往会出现相似的内容，甚至是完全相同的内容。完全重复的内容可以通过块级数据去重的方式进行数据消冗，相似度高的内容可以通过差量压缩的方式对数据进行消冗。如表1-2所示，在几种不同的常见场景下，数据的块级消冗比率能达到 30%-80% 的块级消冗比率。（2）同一份数据内部也会存在冗余。同一份数据内存在的冗余数据，可以通过数据压缩的方式对同一份数据进行数据内部的数据消冗。如表1-3所示，以个人电脑的工作环境为例，高达 75% 的数据可以达到 60% 以上的压缩率①。

表 1-2　块级数据冗余情况[8]

研究机构	数据集	数据规模	冗余比率
微软	近九百个用户桌面的文件系统	162TB	≈42%
EMC	近一万个用于备份管理的系统	700TB	69-93%
美因茨大学	四个用于高性能计算的集群	1212TB	20-30%
IBM	四十三台服务器	44TB	18-53%

表 1-3　个人工作环境的数据压缩率

数据类型	数据占比	数据压缩率
微软 Outlook 应用	19%	60%
微软 Office 文件	23%	85%
Firefox 文件	9%	80%
微软 Communicator 应用	3%	70%
系统安全应用	21%	95%
其他	25%	-

差量压缩技术用于相似度很高的数据块间的数据消冗。在上述数据消冗技术中，差量压缩是对两个相似度很高的数据块，仅保留两者很小的差值 $Delta$，来消除两者很大部分的重复冗余内容的技术。差量压缩技术往往包括数据块的相似性

① Data compression in the intel solid-state drive 520 series, https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/ssd-520-tech-brief.pdf

检测和差量编码技术[8]。差量压缩技术在一些数据相似度高的场景下，如版本管理、系统上相似文件等，发挥着显著的作用。

1.1.4 研究的目的和意义

QLC-SSD 带来空间上的突破，但其寿命急剧下降。寿命瓶颈极大限制了 QLC-SSD 的全面普及应用。

本课题的目的在于：充分挖掘真实世界数据的特征，充分利用高密度闪存的多级电压编程特性，将差量压缩等数据消冗技术和多级电压编程特性相结合。提出一种兼顾存储空间和使用寿命的方式来充分解决 QLC-SSD 上面临的寿命问题。

本课题的意义在于：（1）推动新一代 QLC NAND 闪存存储技术的进一步普及。本课题能够充分解决高密度闪存上的寿命瓶颈，改善 QLC-SSD 的使用体验，推动下一代 NAND 闪存技术的普及。（2）探索计算型存储的新应用场景。本课题充分结合差量压缩等数据消冗方式，存算融合，为仍处于探索阶段的计算型存储探索新的新应用场景。（3）为数据中心等大规模数据存储场景提供新的方案。突破寿命瓶颈的 QLC-SSD 可全面替代其他存储设备（如机械硬盘）用于各种大规模数据存储的场景中，为大规模数据存储提供一种新的方案。

1.2 国内外在该方向的研究现状及分析

1.2.1 差量压缩和数据消冗

差量压缩是用于相似块之间进行数据消冗的技术。差量压缩技术主要包括数据块的相似性检测技术和相似块间的差量编码技术。相似性检测负责快速从海量数据块中定位和判断出高度相似的数据块。差量编码则负责对上述的高度相似块的差异值进行编码表示。

在相似性检测技术上，REBL 工作[9] 最早提出通过超级指纹的方式来降低查找和匹配相似块的计算开销。随后 TAPER 工作[10] 提出可以使用布隆过滤器的方法来优化相似文件的相似性检测。面向备份场景，DARE 工作[11] 利用去重系统现有的信息快速进行相似块的判断。近年来，Odess 工作[12] 提出通过内容采样的思想生成采样哈希集加速相似特征生成，大大降低了计算开销。

在差量编码技术上，最早的差量编码技术 Xdelta 工作[13] 提出可以通过不同的指令来表示相似数据的相同部分和不同部分，如通过 cp 指令表示相同部分、$diff$ 指令表示不同部分。后续 Ddelta 工作[14] 受去重中常用的分块算法的启发，用贪心检测相同内容的策略快速生成相似块的差量编码。此外，一种基于异或压缩的朴

素差量编码[15,16] 通过计算相似块的异或值得到稀疏的原始差量，并对原始差量进行无损压缩作为相似块的差量编码。

1.2.2 固态硬盘的寿命优化

第四代固态硬盘 QLC-SSD 的全面普及面临严峻的寿命挑战。在章节1.1.1和图1-3中提到，QLC-SSD 相对于 SLC-SSD 带来了 4 倍存储密度的提升，但其使用寿命骤降了几个数量级。严峻的寿命挑战极大限制了第四代闪存技术 QLC-SSD 的全面普及和 NAND 闪存技术的进一步发展。延长 SSD 的使用寿命，始终是计算机存储领域研究的热点话题。

近年来，诸多学术工作对固态硬盘的使用寿命进行了优化。有关工作的总结如表1-4所示，固态硬盘上的寿命优化所采用的核心技术手段总结包括：数据消冗技术、缓存技术、数据编码和布局方法。详细介绍如下：（1）数据消冗技术主要是借助块级去重、无损压缩和差量压缩等技术减少固态硬盘内部的写入流量。（2）缓存技术主要是利用 SLC 颗粒充当其他高密度闪存颗粒（如 QLC 颗粒）的缓存，来减少对高密度闪存的磨损。其中配置 SLC 颗粒，因其寿命长但容量小，故成本很高。（3）数据编码和布局技术主要是挖掘高密度闪存上的硬件特性，如多级电压编程特性、存储单元的高电压易损伤性等，重新设计固态硬盘上的数据存储的方式，以最大限度利用硬件的存储能力。

表 1-4　固态硬盘寿命优化研究现状小结

工作名称	主要技术手段
CA-FTL[17]	块级去重
LX-SSD[18]	块级去重
DAC[19]	差量压缩
Delta-FTL[15]	差量压缩
SLC-Append[16]	缓存技术、无损压缩、差量压缩
HyFlex[20]	缓存技术
Damage-Reduce[21]	数据布局、无损压缩
WOMv[22]	数据编码

在固态硬盘的数据消冗上，IBM 研究者指出将数据压缩内嵌到固态硬盘端来做是必要的[23]，CA-FTL 工作[17] 提出了在固态硬盘内部做数据去重，所有写入固态硬盘的页面均会先进行索引查找. 如果存在重复页面，则直接进行页面的复用，不去申请新的页面。其缺点在于去重带来的复杂映射表结构、高额的查询开销和内存开销等问题使其无法很好用于内存资源受限的固态硬盘中。LX-SSD 工作[18] 则提出将去重范围缩小到失效页面中，如果当前写入固态硬盘的页面和失效页面

中的某个页面重复，则重新启用该失效页面而不对其进行擦除，从而减少了擦除次数。DAC 工作[20] 维护了若干的基准块及其相似性特征指纹，所有写入固态硬盘的页面会和这些基准块进行相似性检测，对于和基准块高度相似的页面，只需写入和基准块的差量即可。Delta-FTL 工作[15] 则从页面更新的角度出发，挖掘更新数据的相似性，仅保留页面更新前后的差量 *Delta*，来减少非必要的冗余数据重复写入。数据消冗技术除了用于固态硬盘的寿命优化以外，还有一些利用数据压缩性来增强固态硬盘中的数据可靠性的工作[24]，或者灵活选择压缩算法来提高吞吐性能等工作[25]，本文不做深入研究。

在固态硬盘的缓存技术上，SLC-Append 工作[16] 聚焦高密度闪存上的 SLC 缓存，挖掘 SLC 颗粒上的部分编程特性，将页面更新前后的差量存储在对页面进行无损压缩后的剩余空间中来实现 SLC 缓存上的页内差量更新，进而间接地减少高密度闪存（如 QLC 颗粒）上的写入与擦除。HyFlex 工作[20] 提出根据前台 I/O 的速度来决定固态硬盘写入的数据是写往 SLC 缓存还是高密度闪存上，并根据后台 GC 的性能来动态调整 SLC 缓存的容量，通过缓存的动态调整来间接减少高密度闪存的写入和擦除。

在固态硬盘的数据编码和布局上，Damage-Reduce 工作[21] 从存储单元损伤的角度出发，研究一个页面无损压缩后，如何尽可能地重排数据的布局使得页面内所有存储单元的电压尽可能处于更低的状态，以减少对存储单元的损伤从而延长寿命。WOMv 工作[22] 充分挖掘了高密度闪存（如 QLC 颗粒）上的多级电压编程的特性，用空间开销换原地更新的思想，提出将数据进行更高维度的多级电压特性编码，在一次多级电压编程的完整过程中数据可以被多次表示（即意味着多次更新），最大限度地利用了存储单元全部电压等级所能数据的能力，避免更新带来的不必要擦除，延长 SSD 使用寿命。

1.3 课题的研究动机

1.3.1 高密度闪存的特性

闪存技术的两大核心是空间与寿命。和其他存储技术（如磁性存储技术）相比，除了存储空间的大小以外，NAND 闪存还额外具有使用寿命的概念。存储空间和使用寿命共同构成固态硬盘存储技术的核心指标。在章节1.1.1的图1-3中提到，高密度闪存加剧了空间与寿命的矛盾，存储密度提高，但使用寿命急剧下降，两者无法兼得，但缺一不可。1.3 倍的空间的开销就会让 QLC-SSD 退化为上一代 TLC-SSD。固态硬盘寿命问题的解决需要平衡固态硬盘上的空间开销。

高密度闪存上具有多级电压编程特性。在章节1.1.2提到，NAND 闪存通过电压状态来表示数据，并且在编程过程中电压只能单调递增。在高密度闪存上，电压还会呈现多级状态，存储单元的数据表示需要经过多级的电压编程。如图1-5所示，以 QLC 颗粒为例，其存储单元的阈值电压被划分为了 16 个等级，将存储单元的电压进行多级的升压，升到相应的等级即可表示不同的 4 比特数据。和 SLC 颗粒仅有两个等级不同，QLC 颗粒多达 16 个等级，15 次电压递增过程。每达到一个电压等级均可进行一次数据的表示。高密度闪存上的寿命问题可以充分利用其具有的多级电压编程特性。

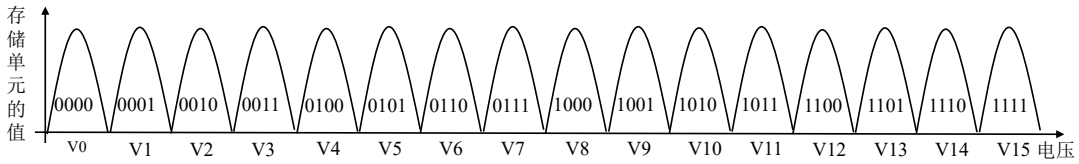


图 1-5　QLC 颗粒的阈值电压分布

现有的寿命优化方案（章节1.2.2所介绍）往往会陷入以下的问题：（1）没有充分挖掘高密度闪存上的多级电压编程特性。这些方案仅仅单一地从数据特征的维度或者是外置缓存（如 SLC 缓存）角度来缓解寿命问题，并没有挖掘高密度闪存上的多级电压编程特性。（2）对空间和读写性能做出了极大的牺牲。这些方案利用多级电压编程特性，但不可避免地带来了成倍的空间开销和性能开销。如图1-6所示，在高密度闪存上现在面临空间大但寿命很低的情况，而现有的方案为了实现更高的使用寿命，其空间利用率急剧下降，本文旨在提出一种能够充分兼顾空间和寿命的寿命优化方案。

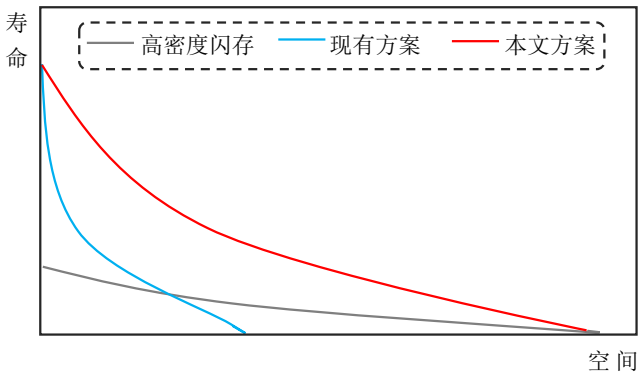


图 1-6　不同方案下的空间和寿命平衡

本文的第一个研究动机是：如何在充分平衡空间开销和寿命优化的前提下，尽可能地利用好高密度闪存上的多级电压编程特性来解决其寿命问题。

1.3.2 现实世界数据的特性

现实数据具有很好的数据更新特性。现实世界数据具有很好的数据更新特性，即往一个逻辑地址处写入的数据，后续会被多次更新。频繁更新的数据可以称为热数据，从不更新的数据可以称为冷数据。在现实世界场景中，数据的更新操作是很常见的，仅有很少的数据是冷数据。图1-7展示了几种场景下（负载信息见章节4.4.1的表4-4）的数据更新次数的占比情况，可以看到在 Mail（邮件服务器场景）负载下，几乎没有冷数据；在 Phones（个人手机场景）负载下，冷数据的占比仅有 40%；大多数数据的更新次数通常在 2-100 次。现实世界的数据容易发生更新操作。

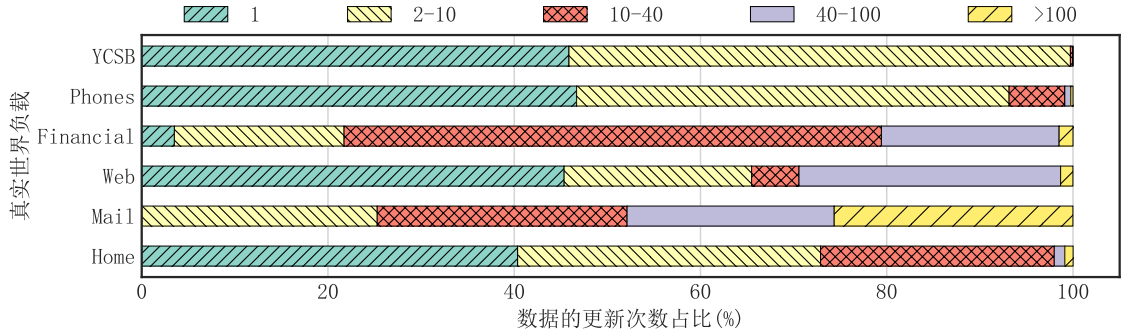


图 1-7　真实世界负载下的数据更新占比

现实数据具有很好的可压缩性。首先，现实世界数据往往具有很好的可压缩性。数据内部存在大量的数据冗余，如章节1.1.3的表1-3所示，在个人电脑的工作环境中，高达 75% 的数据可以达到 60% 以上的压缩率。其次，具有更新特性的数据往往具有很好可压缩性。从文件类型角度来看，如表1-5所示，表中覆盖了常用的文件类型。容易发生更新的文件类型，往往具有很好的可压缩性。如文本和文档一类文件能达到 40-90% 的压缩率。而压缩性相对较差的数据，其往往也不具有很好的数据更新特性。

现实数据具有很好的局部相似特性。发生数据更新的内容往往很好的局部相似性。微小的更新是常见的，更新前后的数据会有大部分的数据重复。在先前的 SLC-Append 工作中充分论证了这一特性[16]。以数据库系统和文件系统的元数据为例，如表1-6所示，不同的场景下的元数据改变内容的比例往往只有 8% 左右。在日常的数据库系统、版本管理系统、文档和代码更新等场景下，微小的更新也很常见的。

本文的第二个动机是：现实数据往往具有很好的更新特性和冗余特性，如何充分挖掘这些数据特性，将数据消冗余技术和高密度闪存上的多级电压编程特性进行互补地结合，来达到兼顾空间和寿命的寿命优化。

表 1-5　不同文件格式的平均压缩率和数据更新特性[26]

文件格式	文件类型	平均压缩率	数据更新特性
TXT	文本	42.28%	好
XML	文本	89.51%	好
XLS	文档	77.05%	好
PDF	文档	8.08%	差
BIN	程序	86.93%	好
AVI	视频	3.49%	差
MP3	音频	2.98%	差
WMA	音频	54.61%	差
BMP	图片	82.40%	差
JPG	图片	0.66%	差

表 1-6　不同文件格式的平均压缩率和数据更新特性[16]

场景	操作	元数据改变比例
数据库	记录插入	8.7%
数据库	记录更新	7.5%
文件系统	文件追加	8.0%
文件系统	文件更新	8.4%

1.3.3 异地更新是寿命消耗的关键原因

异地更新是固态硬盘寿命消耗的关键原因之一。本文聚焦从总体的擦除次数上来对寿命消耗做理论的分析。磨损均衡等引入擦除次数来延迟坏块的方式本文不做考虑。图1-8展示了固态硬盘寿命消耗的主要原因。固态硬盘寿命消耗通过擦除次数体现。而造成页面擦除的原因是页面被标记为了失效，标记为失效的页面在垃圾回收的时候会被全部集中擦除。在固态硬盘中，页面失效来源包括：（1）主动失效。页面可以通过 *discard* 操作（在不同协议中命令不同）来主动进行失效，如删除文件等。这种页面失效通常是无法避免的，也无法很好地缩减其对应的擦除次数。（2）异地更新（被动失效）。固态硬盘上对页面的更新采用的是异地更新机制。每次异地更新都会将旧的页面标记失效，重新申请新的页面进行写入。每次页面更新都会带来一次页面的失效，进而带来一次页面的擦除。

不必要的异地更新带来了固态硬盘上寿命不必要的过度消耗。一些异地更新是必要的，一个 NAND 闪存物理页的存储单元和最大电压 V_{max} 都是有限的，终究都会全部升满，不通过页面擦除无法被再次编程使用。而在达到此状态之前，页面仍有被继续使用的潜力，直接异地更新会极大地浪费这些可编程能力。本文的前两点研究动机充分地体现了页面仍然继续使用的潜力：（1）多级电压的数据表示能力没有被全部利用。QLC 颗粒的电压递增空间大小为 15[27]，具有 16 次的数据

表征能力，但只和电压空间为 1 的 SLC 颗粒一样，只用于表示了一次数据。此外，如图1-9所示，经过编程后的页面，仍还有很多存储单元没有达到最大电压，仍然有继续使用的潜力。（2）页面数据具有冗余性。页面内部数据具有冗余性，通过内部消冗可以让部分存储单元闲置而未利用。从更新角度来看，更新页面具有局部相似性，微小部分的修改可以充分借助闲置存储单元或上述未被利用的电压的数据表示能力等页面间隙来完成，而无需额外的页面申请。

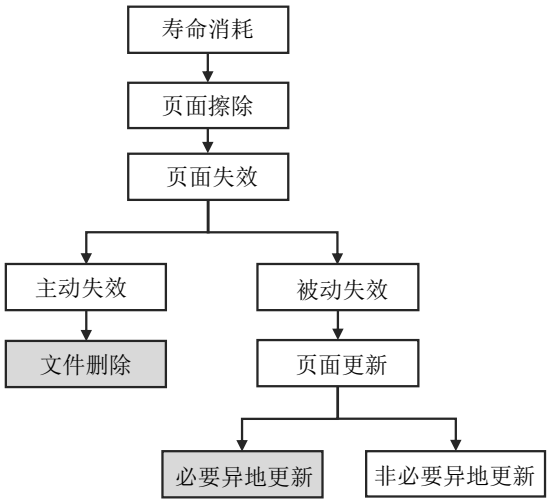


图 1-8　固态硬盘寿命消耗的原因

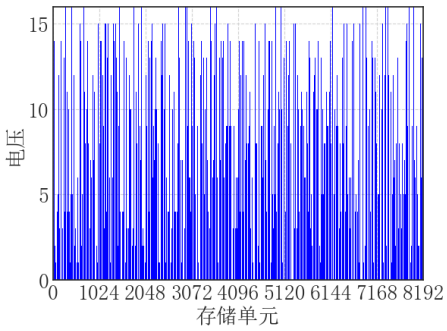


图 1-9　编程页面的电压分布

挖掘页面的极限编程潜力需要聚焦页面更新方式的优化。大部分的页面仍具有继续利用的潜力。然而现实数据的无序性和随机性使得页面电压分布呈现随机无规律的状态，如图1-9所示。由于电压编程的递增性，页面中达到更大级别的电压的存储单元无法进行二次编程来表示更小级别的电压的数据。而固态硬盘的写入粒度是页，页面更新前后的这种冲突是极易发生的。想要对页面再次进行编程操作就不得不将这些页面进行擦除操作，从而浪费掉这些仍有利用潜力的页面。想要充分挖掘页面继续利用的潜力，必须要对数据进行重新组织管理，对页面的更新方式进行重新设计和优化。

本文的第三个动机是：如何充分利用仍具有编程潜力的页面，进行最大限度就地更新，避免非必要的异地更新，进而最大程度地减少高密度闪存上敏感的擦除次数来达到高密度闪存上寿命优化的目的。

1.3.4 聚焦更新优化的方案的缺点

闪存上的寿命消耗问题可以通过针对页面更新操作的优化来解决。但面向高密度闪存的更新优化需要充分考虑以下三个关键因素：

• **关键点 1：细粒度更新。**SSD 的编程单元是页，微小的页面更新会带来大量重复数据的写入。面向高密度闪存上的更新优化需要实现细粒度的更新，以消除页面更新的冗余，仅保留更新前后的细小差异。细粒度更新需要充分挖掘数据更新的局部相似性。

• **关键点 2：无空间开销。**更新优化还需要充分考虑固态硬盘上空间和寿命的平衡，不应以过多的额外空间和性能开销作为代价。额外的空间开销会让高密度闪存退化为低密度闪存，并且造成多倍的读放大或写放大。空间开销的减少可以充分挖掘数据的冗余性。

• **关键点 3：原地更新。**高密度闪存上的更新优化需要能够充分利用高密度闪存的多级电压编程特性来实现原地更新，确保最大化利用每个存储单元的可编程潜力。原地更新需要根据多级电压编程特性来对数据进行高维编码来解决页面写入数据的随机性和无序性问题。

现有的页面更新优化工作都不能很好地适用于高密度闪存上。在章节1.2.2介绍了固态硬盘的寿命优化的诸多学术工作。而其中聚焦从固态硬盘的页面更新优化角度来解决寿命问题的工作包括：Delta-FTL，SLC-Append，WOMv 三项工作。表1-7展示了几项工作页面更新优化工作在高密度闪存上的不足之处。下面是几个工作的对比分析：

表 1-7　聚焦于页面更新优化工作的对比

工作名称	细粒度更新	无空间开销	原地更新
Delta-FTL	√	×	×
SLC-Append	√	√	×
WOMv	√	×	√
IUDW（本文）	√	√	√

• Delta-FTL 工作通过对页面更新进行差量压缩消除了页面重复数据，以实现了细粒度更新。但其缺点在于：（1）没有考虑空间和性能开销，带来了两倍的读放大和复杂映射结构。（2）没有考虑原地更新，高密度闪存上的存储单元编程能力没有充分利用。

• SLC-Append 工作在先前工作细粒度更新的基础上，通过无损压缩消除冗余实现页内更新。但其缺点在于：没有充分考虑原地更新，高密度闪存上的存储单元编程能力没有充分利用。SLC-Append 面向混合固态硬盘的 SLC 缓存，利用 SLC 缓存上特有的部分编程特性实现 SLC 缓存的页内追加，以减少 SLC 缓存的寿命消耗，来间接减少 QLC-SSD 上的写入流量。而在混合固态硬盘中 SLC 缓存容量很有限且对寿命消耗并不敏感，因而其对 QLC-SSD 上寿命优化效果极为有限。

• WOMv 工作通过对页面进行编码让无序的数据呈现一定的规律性，以利用多级电压编程特性来实现数据的原地更新。其缺点在于：没有考虑空间和性能开销，让无序的数据呈现规律性申请了额外的新页面用于原地更新的实现，带来了四倍（或两倍）的空间开销，同时带来四倍（或两倍）的读放大与写放大。在更新频率不高的情况下，高密度闪存还会退化为读写性能极低的低密度闪存（如从 QLC 颗粒退化为 SLC 颗粒或 MLC 颗粒）。

• 本文的 IUDW 系统将数据消冗技术和多级电压编程特性相结合，提出了适用于高密度闪存上的混合页面更新框架。IUDW 系统能够同时具备高密度闪存上页面更新的细粒度更新、无空间开销和支持原地更新的全部关键点。

1.4 本文主要研究内容

QLC-SSD 带来可观存储密度的同时，其使用寿命急剧下降，成为了限制其全面普及的严峻挑战。本文经过分析和总结得出：（1）现实数据的特性和高密度闪存的硬件特性能够帮助解决 QLC-SSD 的寿命问题。高密度闪存上具有多级编程特性，现实数据具有更新特性和冗余性。（2）影响固态硬盘使用寿命的关键因素在于大量且非必要的异地更新。大量且非必要的异地更新会带来了大量且非必要的页面擦除，从而对 QLC-SSD 寿命进行过度消耗。（3）现有的页面更新优化方案存在诸多问题和缺点。例如空间和性能成倍开销，未能充分挖掘更新潜力，不适用于高密度闪存等。

基于上述观察和研究动机，本文提出了 IUDW（**I**n-place **U**pdate with **D**elta in **W**OMv Code）存储系统，该系统围绕高密度闪存上面临的严峻的寿命挑战这一核心展开。针对固态硬盘寿命优化的若干关键问题（在章节1.3.4做了相关分析），提出了一种面向高密度闪存的实现细粒度更新、平衡空间和性能开销、充分发挥高密度闪存特性的存储系统。本文的方案借助差量压缩和无损压缩等数据消冗技术分别实现了页面的细粒度更新和页内更新，并创新地将数据消冗技术和多级电压编程特性进行结合来实现高密度闪存上的差量原地更新，同时采用混合更新框架来进行高密度闪存上的最大程度就地更新。

实验结果表明：在大量的非必要异地更新（即在压缩性高和局部相似性好）的场景下，（1）IUDW 方案相对于最先进的 SLC-Append 方案和 WOMv 方案在单个页面上能够带来 5-100 倍的更新次数提升；（2）在公开的真实世界负载下，相对于原始方案，IUDW 方案能够减少 70% 空间消耗和 90% 的寿命消耗；（3）开销评估表明，IUDW 方案的开销最坏约在读时延开销 9%，编程时延开销 5.7%。

图1-10总结了本文的主要研究内容。该方案很好地考虑了固态硬盘的两大核

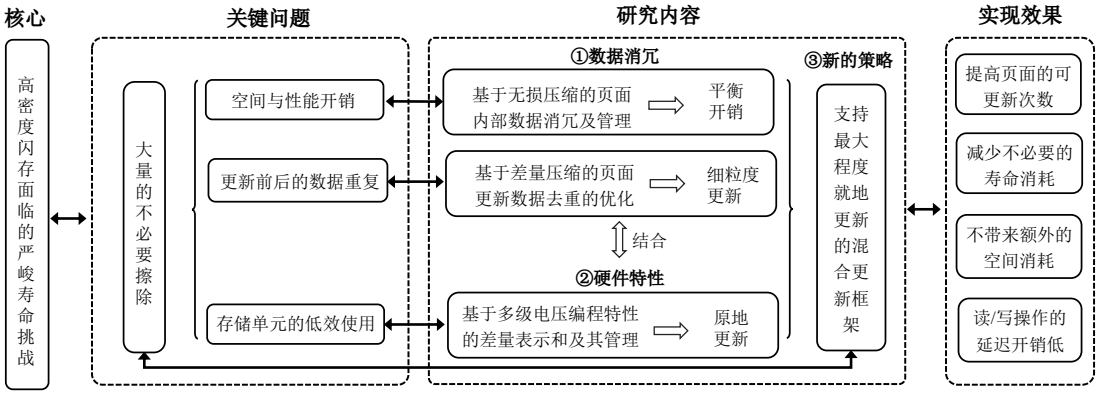


图 1-10　本文主要研究内容总结

心，寿命和空间，不带来额外的空间和性能开销，并最大程度将非必要的异地更新以就地更新的方式进行实现，几乎消除了所有非必要的异地更新，能够极大地减少页面的擦除次数，延长 QLC-SSD 的使用寿命。

总结本文的主要研究内容包括：

• 充分利用现实数据冗余特性和数据消冗技术来优化 QLC-SSD 的寿命问题。本文采用基于差量压缩技术的页面更新数据去重优化方案实现了 NAND 闪存上的细粒度数据更新，并采用基于无损压缩技术的页面内部数据消冗及其管理方案实现了 NAND 闪存上的页内更新。

• 创新地将高密度闪存上的多级电压编程特性和数据消冗技术相结合。IUDW 系统提出部分编码页的概念，并基于多级电压编程特性来进行差量的存储，在高密度闪存上实现原地更新。

• 创新地提出适用于高密度闪存上且无需额外空间开销混合更新框架。本文结合上述技术，实现最大限度的就地更新，并针对就地更新可能会遇到的问题，如和现有的异地更新兼容、写缓存兼容和垃圾回收等，提出了适用于高密度闪存上且无需额外空间开销的混合页面更新框架来减少所有不必要的寿命消耗。

• 对本文的方案设计微基准测试、宏基准测试和开销评估多个维度对存储系统 IUDW 进行全面评估。结果表明 IUDW 系统能够有效延长 QLC-SSD 寿命，并且带来很低的读写性能开销。

本文的第 2 章介绍本文的存储系统 IUDW 的系统总体结构设计，包括系统的核心思想和各个模块组件。第 3 章详细介绍了每个子模块的详细设计和实现。第 4 章介绍本文的多维度测试方法和存储系统 IUDW 的测试结果。

第 2 章 IUDW 系统总体结构设计

2.1 引言

本章旨在从宏观的角度来介绍本文的 IUDW（**I**n-place **U**pdate with **D**elta in **W**OMv Code）系统。本章主要介绍 IUDW 系统针对 QLC-SSD 寿命优化问题上的核心设计理念，以及整个系统的总体结构框架。通过本章能够对本文的 IUDW 系统的优化思想和系统总体运行有较为清晰的了解。系统方案中各个模块的有关技术点和细节实现将在第3章中分别进行介绍。

2.2 核心设计思想

高密度闪存上的寿命优化需要充分挖掘现实数据的冗余性和局部性。现实世界的数据往往会具有大量的数据冗余，让固态硬盘额外分配了诸多不必要的冗余空间，同时带来了固态硬盘上许多不必要的擦除。通过一些数据消冗的方式可以利用或者消除这些数据冗余，减少不必要的写入流量。在章节1.2.2中，有些工作提出通过块级去重的方式来进行数据消冗，但其索引的内存和性能开销使其并不适用于内存资源受限的固态硬盘内部。本文的方案主要聚焦于页面更新相关的两种数据冗余：页面内部的数据冗余和页面更新前后的数据重复。对于页面内部的数据冗余，本文通过对固态硬盘中的页面做无损压缩管理，为每个页面预留一定的空间，以支持页内更新的实现。对于页面更新前后的数据重复，本文通过差量压缩的方式仅保留更新的差量，去除页面更新前后的大部分重复数据，以实现细粒度的差量更新。

高密度闪存上的寿命优化需要合理并充分利用每一个存储单元的可编程能力。在章节1.3.1中介绍，QLC 闪存上具有 16 个不同的编程电压状态，在一次编程过程中电压递增空间大小为 15。因为现实数据具有随机性和无序性，对同一个存储单元前后更新，无法满足电压递增条件，使得每个存储单元只允许编程一次。在已有的方案中，在充分利用多级电压编程特性做了一定探索。数据的有序化表示需要通过升维，其结果是为了利用这一编程特性对空间和性能做了多倍开销的妥协。多级电压编程特性的利用需要空间成本。虽然这一特性带来了 NAND 闪存上原地更新的可能，但它并不适用于固态硬盘上的所有数据。本文经过观察发现，多级电压编程特性有其独特的运用场景。本文创新地将多级电压编程特性应用于固定大小空间下的小量数据的频繁更新写场景中，即微小差量的写入，可以巧妙地避

开多级电压编程特性的利用而带来的空间上的开销。即让每个闪存页面的其中一部分都可以得到多级电压编程的利用，充分利用 QLC-SSD 存储单元多级电压编程特性来让每个页面都具备一定的原地更新的能力。

高密度闪存上的寿命优化需要消除高密度闪存上所有非必要的异地更新。在章节1.3.3提到，必要的异地更新和主动的页面失效带来的固态硬盘寿命消耗无法避免，但是非必要的异地更新是带来大量且非必要的固态硬盘寿命消耗的关键因素。非必要的异地更新来源于现实数据的冗余性和局部性，以及 NAND 闪存存储单元的编程潜力没有充分被利用。本文的一个核心思想是从上述提到的现实数据的特性和 QLC-SSD 的硬件特性出发，消除高密度闪存上所有非必要的异地更新，将这些非必要的异地更新全部转化为最大限度就地更新，依此来减少存储单元上所有的不必要擦除次数，最大限度延长 QLC-SSD 的使用寿命。

高密度闪存上的寿命优化需要兼顾固态硬盘的空间和寿命，且不带来过多的性能开销。在章节1.2.2中，现有的诸多方案对固态硬盘的使用寿命进行优化的同时，带来了很多额外的空间开销和性能开销，如 WOMv 工作带来了两倍或四倍的空间开销和读写性能开销、Delta-FTL 带来两倍读性能开销等。本文方案的一个核心思想是高密度闪存上的寿命的优化，不应该以牺牲较多的空间或者牺牲较多的性能来实现。空间开销会让高密度闪存进行退化，1.3 倍的空间的开销就会让 QLC-SSD 退化为 TLC-SSD。因此，本文的一个核心思想是旨在围绕固态硬盘的两大核心展开，同时考虑空间和寿命的角度来解决 QLC-SSD 上面临的寿命问题，既不损失 QLC-SSD 的空间优势，又能解决 QLC-SSD 上的寿命挑战。

2.3 系统总体框架

在传统的固态硬盘设计中，QLC-SSD 内部主要包括固态硬盘硬件、固件管理程序和 NAND 闪存阵列，如图2-1所示（图中未展示垃圾回收、磨损平衡等功能）。上层应用的写请求到来时，固件管理程序分配新的物理页面，并在映射转换表中记录下映射关系。如果本次的写请求是页面更新，则会直接进行异地更新，标记原来的页面失效，然后分配新的页面并在映射转换表中重新记录映射关系。

本文 IUDW 系统对传统的固态硬盘设计进行优化，系统总体结构如图2-2所示。基于本文方案的 QLC-SSD 在传统的固态硬盘上引入了页面压缩模块、差量压缩模块、编码管理模块和更新优化模块共四大模块。其中页面更新模块和差量压缩模块共同负责对写入的数据进行数据消冗；编码管理模块负责充分利用多级电压编程特性，管理 NAND 闪存页面上确定大小的编码空间；更新优化模块负责根

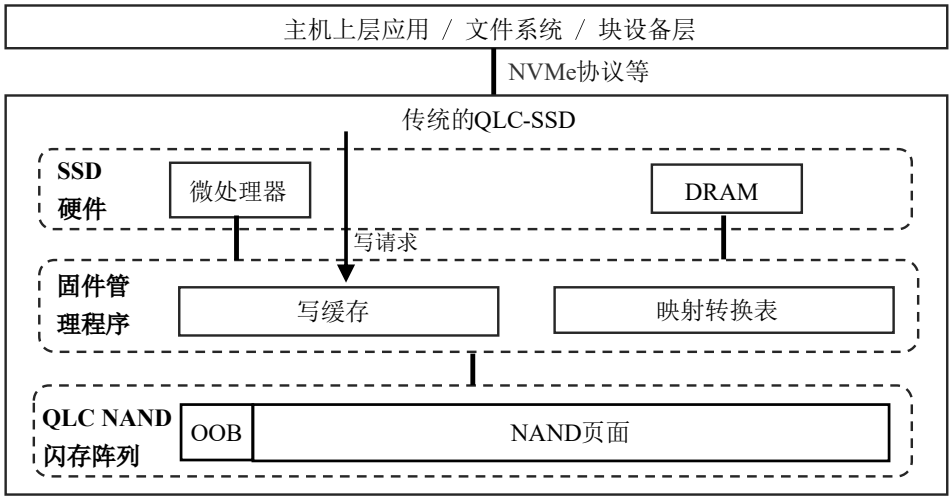


图 2-1　传统固态硬盘结构示意

据 NAND 闪存的页面状态来决定是继续执行就地更新，还是执行异地更新。下面对 IUDW 系统中的每个模块做概括性的介绍：

页面压缩模块。页面压缩模块主要从数据内部的冗余性出发，负责对固态硬盘每个页面都通过无损压缩的方式对页面数据进行内部的数据消冗，实现页内更新。当从主机的写请求到达后，如果写入的页面是新页面，页面压缩模块会尝试对写缓存中的写请求的数据进行无损数据压缩，并对新页面进行区域划分和维护。如果写入的页面不是新页面，即本次的写请求是更新操作，意味着页面已经被划分过，页面压缩模块会负责将页面压缩过的数据进行解压操作，还原初始的数据。此外在读请求到来的时候，页面压缩模块也会负责页面的解压缩，确保功能的正确性。页面压缩模块的无损压缩逻辑可以通过专用的硬件压缩引擎来实现。专用的硬件压缩引擎目前已经广泛地运用在计算型存储中。

差量压缩模块。差量压缩模块主要从数据的局部相似性出发，负责对固态硬盘每个页面上的更新都通过差量压缩的方式来减少大部分重复数据的重复写入，实现细粒度差量更新。当从主机来的写请求是更新写操作时，差量压缩模块会将页面压缩模块恢复的旧页面数据和写缓存中的当前新写入的数据进行差量编码。差量压缩模块也可以通过专用的硬件压缩引擎来实现。

编码管理模块。编码管理模块主要从 QLC 闪存的多级电压编程特性出发，负责对细粒度差量在给定空间下进行升维以实现有序的电压递增原地更新。新的页面在页面压缩模块处理过程中就会确定页面的区域划分，也就是确定了一个闪存页面上编码管理模块所能支配的空间大小。这个大小和页面数据的冗余程度有关，每个页面的编码空间大小各不相同。编码管理模块会在每个页面上有限的编码空间内最大程度地利用该空间内所有存储单元的再编程能力，将迭代的差量不断地

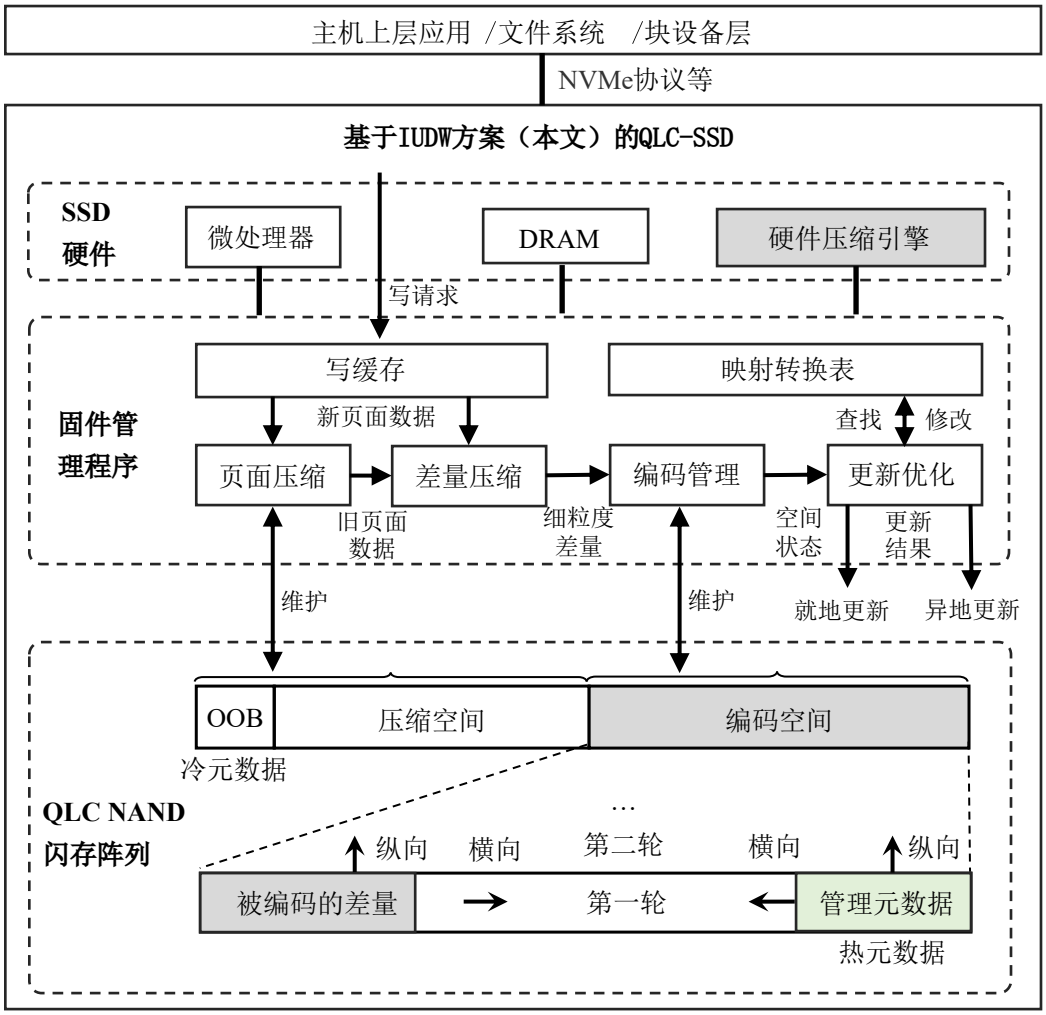


图 2-2 IUDW 系统总体结构

在编码空间中进行横向和纵向拓展更新，直到一个闪存页所分配的编码空间中的全部存储单元的电压都接近最大编程电压 V_{max}。不同页面中编码空间的状态各不相同，每个页面迭代的差量在编码空间中所处的位置也各不相同，编码空间中除了存在迭代的差量以外，还存在用于管理编码空间的元数据。元数据和差量共同在编码空间中进行滑动和更新。

更新优化模块。更新优化模块主要是负责将高密度闪存上不必要的异地更新都以就地更新的方式进行。仅当本次页面更新是必要的异地更新，更新优化模块才会进行异地更新。更新优化模块修改了传统固态硬盘朴素直接的异地更新逻辑，采用最大限度就地更新策略。更新优化模块会负责和映射转换表进行交互。当主机的写请求到来时，根据写请求中的逻辑地址且查找映射转换表获取旧的物理地址。如果该写请求需要申请新的页面，即触发了异地更新，更新优化模块才会修改映射转换表，否则不会修改映射表。对于写请求为更新的操作，更新优化模块会发

起一起页面的读取操作，获取闪存页面当前的全部数据供页面压缩和差量压缩模块使用。更新优化模块会对本次对旧的页面进行就地更新的数据进行电压合法性检查。如果所有存储单元均满足电压递增编程，则会进行就地更新。否则，会进行异地更新。更新优化模块在实现最大限度就地更新过程中还会面临来自 Cache 兼容和与 GC 冲突等挑战，更新优化模块也对应做了相应设计。

2.4 本章小结

本章主要介绍了本文在高密度闪存固态硬盘上面向页面更新优化的 IUDW 系统的核心设计思想和系统的整体结构。

首先，在核心设计思想上，本文的方案：（1）充分挖掘现实数据的冗余性和局部性；（2）对 QLC-SSD 存储单元的多级电压编程特性进行了合理利用；（3）消除高密度闪存上所有非必要异地更新；（4）旨在通过兼顾空间和寿命的方式来解决 QLC-SSD 的问题，不带来额外性能开销。

然后，在系统整体架构上，本文的 IUDW 系统核心分为四个模块：页面压缩模块、差量压缩模块、编码管理模块和更新优化模块。四个模块充分体现本文方案的上述核心设计思想。

第 3 章 IUDW 系统模块实现

3.1 引言

上一章避开模块的实现细节对系统的整体结构进行了总体性地介绍。本章则接着系统的总体性概貌，进一步对 IUDW 系统中每个模块的实现细节进行详细地介绍，讲解每个模块细节上的具体实现，以及这样实现的原因等。本章的小节划分围绕四个核心模块展开，页面压缩模块、差量压缩模块、编码管理模块和更新优化模块，每个小节介绍一个模块的细节实现。通过本章能够对本文方案的全部内容有完整性的了解。

3.2 页面压缩模块

3.2.1 两种页内布局

本文的方案提出了部分编码页的概念。固态硬盘中高密度闪存上的传统的不编码页没有空间开销，但也没有任何页面可更新能力。WOMv 工作提出的高密度闪存上的全部编码页具备可更新能力，但有对应的成倍空间开销。该方案提出的新的页面布局思想，部分编码页，能够让高密度闪存的页面不带来空间开销，同时具备可更新能力。在本文方案中，会维护以下两种页内布局，如图3-1所示：

一种是传统的不编码页。不编码页没有采用基于电压多级编程特性的编码，因此这些页面只能编程一次，也不具备任何的可更新能力。在固态硬盘中，每个页面都会有额外的很小部分页面拓展区域。因此不编码页的页面总体上包括两个部分：页面拓展区、非编码区。

一种是本文创新提出的部分编码页。部分编码页会将传统的页面划分为非编码区和非编码区。非编码区不会进行编码，即落入这个区域的修改都是被禁止的，可以用于存放一个页面无需发生修改的冷数据。编码区会进行基于电压多级编程特性的编码，这个区域可以用来多次更新，可以用来存一个页面发生频繁修改的热数据。页面的非编码区是出于降低页面所需的额外空间开销的考虑，而编码区则是让页面具备多次就地更新的能力。编码页的页面包括三个部分：页面拓展区、非编码区、编码区。

3.2.2 固态硬盘上的页面压缩及管理

部分编码页中的冷热数据：在部分编码页中，非编码区用于存储不会被修改

的冷数据，编码区用于存储会频繁修改的热数据。在本文方案中，非编码区所存放的不会修改的冷数据是第一次写入该页面的数据，称为页面的基准数据。而编码区所存放的频繁更新的热数据是后续再次写入该页面时的新数据和基准数据的差量编码，称为页面的更新差量。

部分编码页中的划分逻辑：页面的大小是固定的，原始的页面的基准数据等于页面大小，并不存在额外的空间来充当编码区。通过章节1.1.3的表1-3和章节1.3.2的表1-5，得知固态硬盘的写入页面中具有大量冗余，且易发生更新的数据类型存在更多数据冗余。即发生更新的固态硬盘页面的基准数据中往往存在大规模数据冗余。本文采用无损数据压缩的方式对页面的基准数据进行无损数据压缩，将压缩后的页面的基准数据的长度作为非编码区的长度，页面中剩下长度的空间作为编码区的长度。对于固态硬盘中的每个页面而言，本文方案中每个擦除过的闪存页面初次被编程，都会被页面压缩模块进行区域的划分。仅当无损压缩失败的时候，即页面的基准数据不可压缩，页面才会退化为传统的不编码页。部分编码页和不编码页的示意如图3-1所示。

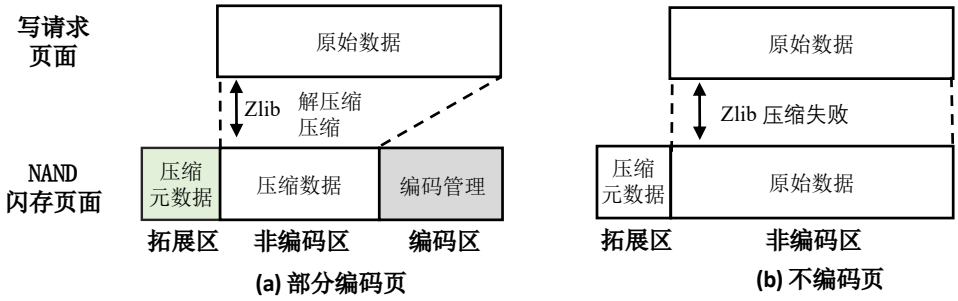


图 3-1　页面无损压缩管理和页面划分

页面的无损压缩及管理：图3-1展示了本文方案的页面无损压缩管理和页面的划分。该方案采用主流的压缩方案 Zlib[①]实现了页面基准数据的无损数据压缩。一般的透明压缩固态硬盘的压缩目的是让一个页面内能存入更多的压缩后的页面，其压缩管理和页面状态无关，统一修改全局的映射转换表结构即可。在此方案中，页面无损压缩并不用于存入更多的其他页面数据，而是为了预留空间给非编码区存储更新差量。每个页面的区域划分各不相同，因此需要为每一个页面维护一个无损压缩管理的元数据 compression meta，来记录页面各自的无损压缩信息。元数据维护了无损压缩的数据长度，也就是页面区域划分的信息，在后续页面读取时，该元数据能够：（1）提供编码管理模块确定编码区起始位置；（2）主机发起读请求时数据解压缩所必须的长度信息。元数据 compression meta 也是冷数据，会在页

① A massively spiffy yet delicately unobtrusive compression library, https://www.zlib.net/

面划分的时候确定，且后续不会发生改变。每个 NAND 闪存页面都会存在一个页面拓展区（**O**ut-**O**f-**B**and，OOB），用来存放页面特有的信息，且只能写入一次。本文将元数据存入在页面的拓展区。页面的拓展区，以 4KB 大小的页面为例，拓展区通常会预留 16B 左右（例如拓展区常常会存储该物理页面对应的逻辑地址用于恢复操作，其长度为 8B）。在本文方案的实现中，4KB 大小的页面，压缩管理元数据只需要额外的 2B 空间，空间开销完全可以接受。当压缩管理元数据为空的时候，表示当前页面的基准数据无损压缩失败，是传统的不编码页。

3.3 差量压缩模块

3.3.1 基于差量压缩的写请求逻辑

差量压缩算法的实现：在差量压缩算法的实现上，本文沿用先前工作多次使用的异或加无损压缩的朴素差量压缩算法。算法描述为将两份相似的数据进行异或运算，得到稀疏的原始差量，然后对稀疏的原始差量进行无损数据压缩作为最终的差量编码。在本文中，和无损压缩类似，差量压缩具体算法的设计不是本文所关注的核心。本文关注于提出一种将差量压缩和无损压缩等数据消冗技术和固态硬盘的内部机制相结合的框架，具体的差量压缩算法和无损压缩算法可以进行替换和调整。

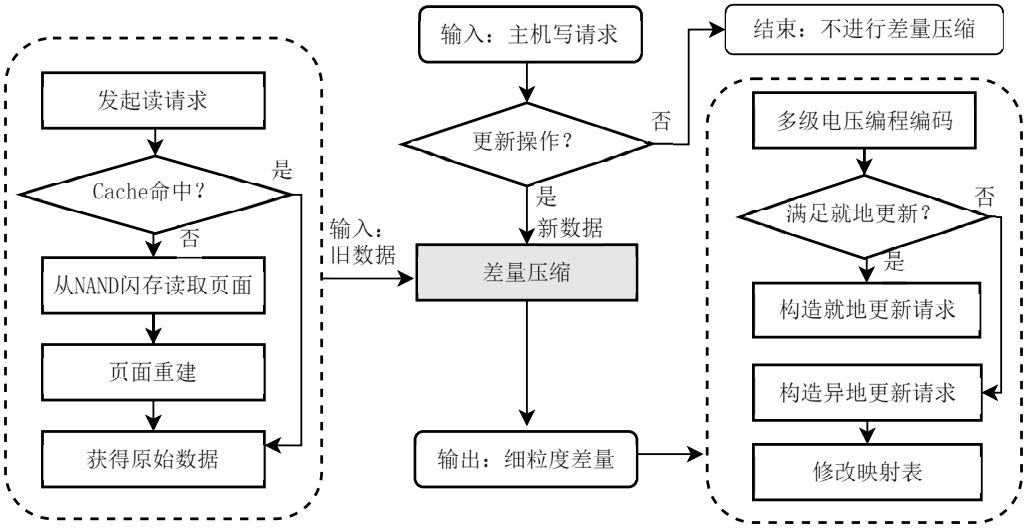


图 3-2　基于差量压缩的页面写请求逻辑

基于差量压缩的页面写请求逻辑：传统的页面写请求逻辑会保留新页面的全部数据。在本文的方案中，对于进行页面更新的写请求操作，利用数据的局部相似性，仅会通过差量压缩保留其差量部分来消除大量的重复冗余。如图3-2所示，如果主机到达的新的写请求是页面更新操作，会进入差量压缩处理模块，否则不会

进行差量压缩。差量压缩处理模块会从写缓存的写请求中获取本次需要写入新的页面数据，同时会获取通过更新优化模块的写前读请求和页面压缩模块的页面重建得到的页面旧数据。在写请求中，页面重建是对页面中非编码区中的基准数据进行页面数据的解压缩工作。对输入差量压缩模块的新的页面数据和旧的页面数据进行差量计算，得到本次更新操作的细粒度差量，消除更新操作中的大量重复数据。本次更新操作的细粒度差量会被送到编码管理模块中进行迭代存储。

3.3.2 基于差量压缩的读请求逻辑

基于差量解压缩的页面读请求逻辑：差量压缩模块在写请求的时候，对于页面更新，仅保留了页面更新前后的差量编码。在读请求时，对应的需要差量压缩模块进行对应的解码和复原操作。如图3-3所示，当主机发起读请求后，在写缓存中命中的读请求无需经过差量解压缩的步骤，直接返回读请求所需的原数据。当在写缓存中未命中时，会发起 NAND 闪存上的页面读取操作，页面的非编码区和编码区会分别送往页面压缩模块和编码管理模块进行解压缩和解码，分别得到页面的基准页面及其细粒度差量。基准页面和细粒度差量会进行差量编码的解压缩复原操作，重建原始的页面，并将得到的原始页面返回给来自主机的读请求操作。

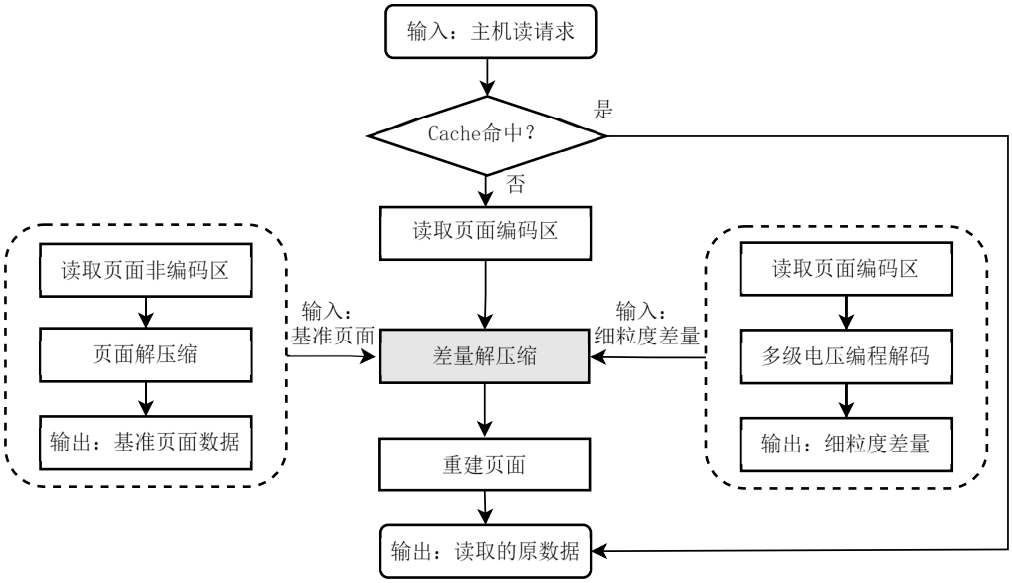


图 3-3　基于差量解压缩的页面读请求逻辑

3.4 编码管理模块

3.4.1 NAND 闪存上的可更新编码

多级电压编程特性及数据编码：在章节1.1.2中介绍了高密度 NAND 闪存上的

电压编程空间为 16。无序的数据会让 15 次的电压递增只能编程一次数据。WOMv 工作提出了一种高维的编码方式，通过一定的空间开销来利用多级电压编程特性，极限发挥一个 NAND 闪存的存储单元在一次电压递增编程的过程（电压达到 V_{max} 前）中表征数据的能力。图3-4展示了一个 4 比特存储单元仅用于表示 2 比特数据的多级电压编程特性编码。存储单元所表示的实际数据位数减少，但由于存储单元的可多级电压编程特性，该存储单元可以进行第二次，... 直到第五次的数据更新表示，直到第五次更新电压达到最高级别而达到可编程极限。在本文方案提出的部分编码页的编码区采用基于多级电压编程特性的数据编码方式，来充分发挥编码区中所有存储单元的极限表示数据能力，为频繁且微小的热数据更新提供最大可能的可更新次数。如表3-1所示，本文方案中采用了三种不同级别的数据编码分别实现了 IUDW(1,4)、IUDW(2,4) 和 IUDW(3,4) 方案，不同方案对应着一个存储单元（假设是 QLC 颗粒，大小为 4 比特）内能够表征差量信息的大小和该存储单元能够更新的极限次数。

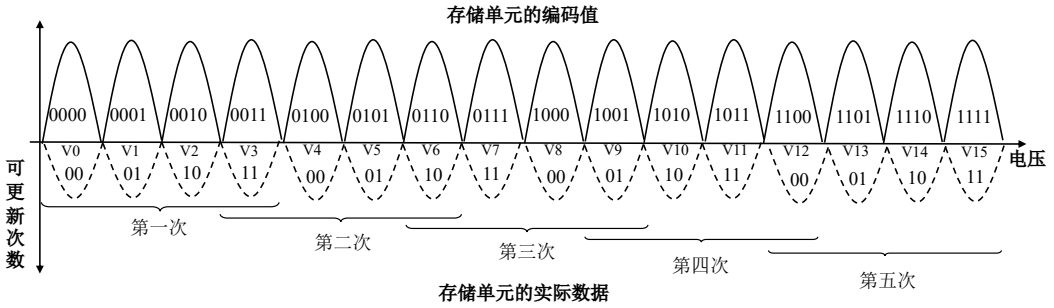


图 3-4　基于多级电压编程特性的编码

表 3-1　不同等级的本文方案的存储单元对比

特点	IUDW(1,4)	IUDW(2,4)	IUDW(3,4)
表示差量的位数	1	2	3
可以更新的次数	15	5	2
可以表示的位数	15	10	6

3.4.2 部分编码页的编码空间管理

编码空间的隐式划分和双向拓展：在本文方案中，迭代的最新的差量会动态地存在于编码空间中，对差量的定位和解码需要对应的编码元数据。每次的差量迭代，都会带来差量数据和编码元数据 code meta 的同时更新。在本文方案中，差量数据和编码元数据都是频繁更新的热数据。和无损压缩管理的元数据 compression meta 是冷数据不同，编码元数据是频繁更新的热数据，无法一同存储在页面拓展

区中。本文方案将差量数据和编码元数据同时存储于支持多次更新的编码空间中。在编码空间的布局上，本文方案采用隐式划分的方式，如图3-5所示，差量数据和编码元数据各处于编码空间的两侧，两种数据同时往内侧进行横向拓展。除了横向拓展，两种数据还会充分利用电压的多级编码特性进行纵向拓展。编码空间在布局上会最终形成差量数据和编码元数据的两端隐式空间划分，和各自在横向和纵向上的双向拓展。

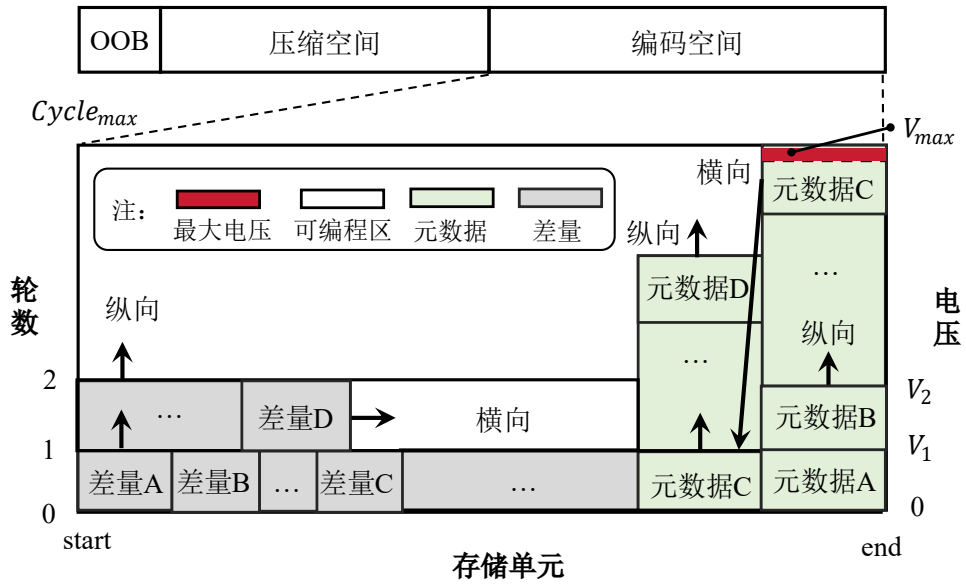


图 3-5　编码空间的多级电压编程和管理

编码元数据的可更新存储：本文方案的编码元数据包含差量在编码空间中的定位和编码类型等信息。在 4KB 大小的 NAND 闪存页面中，本文方案的编码元数据总大小仅为 3B。在不同等级的方案下，分别占用 NAND 闪存页面的 4B、6B 和 12B，是完全可接受的范围内。由于差量数据动态存在于编码空间中，起始位置和其大小均不固定，因此需要编码元数据的维护。但编码元数据也会在空间中进行滑动，起始位置并不固定。编码元数据的定位和解码是编码空间管理的一个关键问题。在本文方案中，对编码元数据采用了先纵向后横向的可更新拓展方式来解决这个问题。和差量数据有一点不同之处，在于编码元数据的大小 size 是固定的，本文方案充分利用了这一点。如图3-5所示，该方案将最大电压 V_{max} 作为编码元数据单元失效的标志。该方案会对编码元数据先尝试进行纵向更新，直到纵向更新达到最大电压 V_{max}。此时，编码元数据会更新失败，进行横向移动一个编码元数据的大小 size 的距离，重新进行纵向更新，以此往复。值得注意的是，横向移动并重新纵向更新未必一定要从空白的存储单元上开始，可以建立在被旧的差量数据编程过（见本文下一段）的存储单元上。在编码元数据的定位和解码时，会从编码

空间的编码元数据存储一侧开始，每次滑动大小为编码元数据的大小 size 的窗口，检查该窗口内是否出现最大电压 V_{max}，如果出现了最大电压，则会继续滑动，直到没有出现最大电压的窗口，该窗口内是当前有效的编码元数据。本质上，该策略是将横向移动前达到最大电压 V_{max} 的编码元数据单元内的全部存储单元从编码空间中进行丢弃，等效于编码空间在不断减小。

差量数据的可更新存储：差量数据的原始长度是不固定的，在 IUDW(3,4) 方案（表3-1所示）中，每 3 比特会编码为 4 比特，因而需要考虑差量数据的长度和对应的编码单位保持对齐的问题。在本文方案中，不对齐的差量数据首先会进行零填充，直到原始的差量数据的字节数和对应编码的编码单位的比特数保持对齐。为了避免编码空间过早出现存储单元达到最大电压 V_{max} 而缩小编码空间，差量数据在编码空间的拓展方式和编码元数据的方式正好相反。差量数据在编码空间中采用的拓展方式是先横向后纵向的可更新拓展方式。差量数据的定位和解码受编码元数据的维护，编码元数据中记录着当前编码空间中存储的差量数据的起始位置和结束位置。新的差量数据会接着当前旧的差量数据的结束位置继续横向拓展，直到抵达当前编码元数据的位置处停止。如图3-5所示，完成一次完整的横向拓展后，差量数据会进行一次纵向拓展，然后重新开始新的一轮横向拓展。不断迭代更新的输入到编码空间的差量会以这种方式在空间中进行不断拓展更新。

编码空间的可更新极限：如图3-5所示，图的横坐标代表了编码空间的全部存储单元，左侧的纵坐标代表了差量数据可以进行的完整横向拓展的次数，称为轮数。右侧的纵坐标代表了存储单元不同的电压级别。基于前面分析，在该方案中，元数据采用先纵向后横向的拓展方式，差量数据采用先横向后纵向的拓展方式。编码空间中的存储单元可更新的极限只看存储单元的纵向拓展，当纵向拓展程度达到最大编程电压 V_{max} 时，存储单元无法再进行页面更新，等价于可用于更新的编码空间的长度减小。当可用于更新的编码空间的长度减小到 0 时，编码空间达到其可更新的极限。

编码空间的编码和解码流程：基于上述设计，编码管理模块实现的编码和解码的流程为：（1）编码逻辑。编码管理模块接收到本次输入编码空间的新差量数据，从编码空间的编码元数据一侧开始滑动窗口寻找有效的编码元数据。然后编码管理模块根据编码元数据存储的旧差量数据的结束位置进行横向拓展更新，如果横向拓展更新达到上限，则回到编码空间中的差量数据一侧的起点重新开始新一轮的横向拓展更新。将新的位置信息存储在编码元数据中得到新的编码元数据，对新的编码元数据在原来的位置上进行纵向拓展更新，如果纵向拓展更新达到上限，则向前滑动一个编码元数据大小的窗口，重新进行纵向拓展。最后将新的差

量数据和新的编码元数据共同在编码空间中完成更新。（2）解码逻辑。编码空间的解码逻辑相对简单，编码管理模块从编码空间的编码元数据一侧开始滑动窗口寻找有效的编码元数据，然后根据编码元数据定位到当前编码空间中存储的差量数据，完成基于电压逐步编程特性的解码即可。

3.5 更新优化模块

3.5.1 混合页面更新框架

两种页面更新方式：如图3-6的步骤 7 所示。第一种是异地更新。异地更新为每一个写请求都重新申请了一个新的未编程页面，如果写请求的逻辑地址之前分配过页面，还会将这个旧的页面标记失效页面。第二种是就地更新。本文方案提出的部分编码页让高密度闪存页面具有一定的就地更新能力。在不违背页面内每个存储单元电压递增的原则下，就地更新意味着可以在原来的页面上多次编程，而无需申请一个新的页面。

混合页面更新框架：更新优化模块旨在以最大限度就地更新的方式将所有非必要的异地更新消除，以减少高密度闪存上不必要的寿命消耗。但一些必要的异地更新无法避免，如不编码页无法就地更新，部分编码页更新次数达到上限等。在 IUDW 系统的更新优化方案中，同时存在最大限度就地更新和异地更新两种页面更新方式的混合更新方案。在 NAND 闪存做就地更新的以往工作中，如 WOMv 工作和 SLC-Append 工作，均并未提及在固态硬盘中兼容中就地更新的方法。在固态硬盘内部，除了映射转换逻辑以外，通常还会设计如写缓存和垃圾回收等功能。为了解决让闪存上的就地更新兼容原有的异地更新逻辑，并和写缓存设计相适应，处理和垃圾回收发生冲突等问题，本文提出了一种在 NAND 闪存上实现最大限度就地更新的混合页面更新框架，这是在之前的工作中所忽视的。

3.5.2 旧页面地址的获取

异地更新的页面地址通过申请新的页面可以直接获取，就地更新的页面地址是旧的页面地址，需要查找映射表和写缓存结构。

映射转换表存放逻辑地址到页面地址的映射关系。在引入写缓存机制后，如果某个逻辑地址在写缓存中，映射转换表中存放的则是逻辑地址到 Cache 地址的映射关系；等到该逻辑地址被逐出缓存时，会将存在缓存中的页面地址写回映射转换表，也就是映射转换表重新存放逻辑地址到页面地址的映射关系。

如图3-6的步骤 1 所示，当主机的写请求到达写缓存，会为该写请求建立一个缓存结构，已有逻辑地址，但其页面地址为空。此时如步骤 2 所示，该写请求会

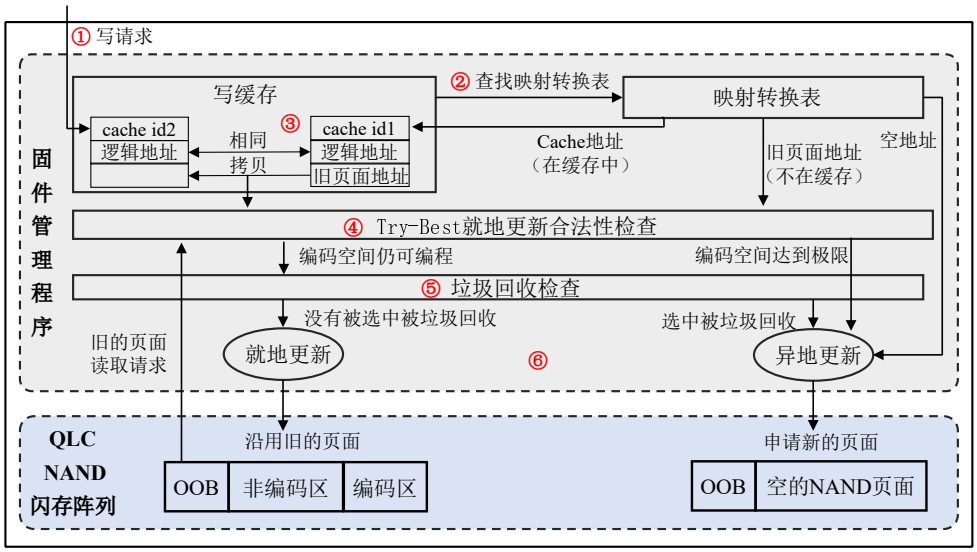


图 3-6　NAND 闪存上的混合页面更新框架

先查找映射转换表，如果映射转换表的地址：（1）为空。表明该写请求不是页面更新请求，该逻辑地址是第一次申请页面，直接进行异地更新逻辑分配新的页面即可。（2）旧的页面地址。该写请求对应的逻辑地址不在写缓存中，直接可以得到旧的页面地址。（3）Cache 地址。该写请求对应的逻辑地址在写缓存中命中，根据 Cache 地址查找对应的缓存结构。该缓存结构的页面地址即为本次写请求的旧的页面地址。同时将旧的页面地址拷贝到该写请求对应的写缓存结构中，以供下一个相同逻辑地址的写请求检索。

3.5.3 最大限度就地更新的两重检查

就地更新和垃圾回收的冲突：在本文方案的固态硬盘中，有两种场景会对编程过的有效页面再次进行操作。第一种是垃圾回收场景，会对选中的擦除单元中的有效页面进行擦除，并重新申请一个新的页面写入这个有效页面，这个过程又称为有效页的迁移。第二种场景是就地更新场景，编程过的有效页面可以被再次编程。如图3-7的异地更新所示，异地更新机制每次都将旧的页面进行失效，所写入的页面始终是新申请的页面，垃圾回收永远不会选中新分配的页面，即垃圾回收和异地更新的写入操作不会有冲突。但和异地更新不同，如图3-7的就地更新所示，就地更新每次写入的页面都是旧的页面，这些旧的页面所处的单元可能会被垃圾回收选中为擦除单元。垃圾回收对旧的页面的迁移和擦除，会和前台向旧的页面再次编程写入新数据同时进行而发生冲突，例如可能导致本次就地更新写入到迁移前的页面等诸多可能情况的问题。

最大限度就地更新的两重检查：如图3-6的步骤 4-7 所示，在章节3.5.2获取到

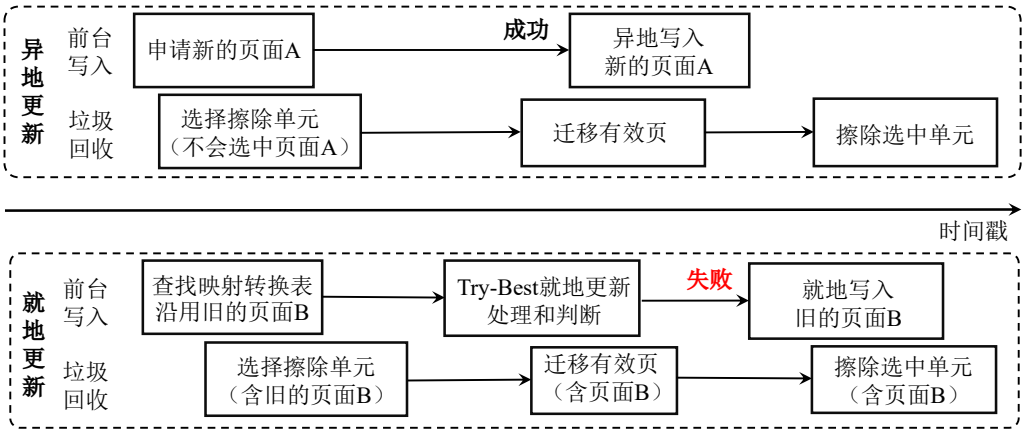


图 3-7　NAND 闪存上的就地更新与 GC 的冲突

本次就地更新的旧页面地址后，本文方案会对该页面地址发起一次页面读取操作，获取到旧页面的全部页面数据，即页面内每个存储单元的编程状态。旧页面的编程状态会用于确定本次就地更新发给旧页面的合法更新数据，或者用于和构造出的想要发往旧页面的数据进行合法性检查，确保本次就地更新不违背页面内每个存储单元电压递增的原则，即确保本次就地更新是合法的。否则，本次就地更新非法或者已经达到上限，会进行异地更新。本文前面提到，就地更新可能会和垃圾回收发生冲突。考虑到这种冲突发生的概率并不高和锁机制在固态硬盘内部带来的代价，在本文方案中，采用的是就地更新进行退让的冲突解决机制。具体说来，如步骤 6 所示，当确定了本次就地更新的合法性后，在对旧的页面正式发起就地更新前，本文方案额外添加一层检查，垃圾回收检查。垃圾回收检查会判断当前准备执行就地更新的旧页面对应的擦除单元是否被进行垃圾回收，如果没有被选中进行垃圾回收，则会正式发起就地更新的请求。否则，本次打算进行就地更新写入的页面被选中了垃圾回收，会将该页面进行退让性的丢弃，重新申请新的物理页进行异地更新。

3.6 本章小结

本章从各个模块角度介绍了本文如何充分利用数据冗余性和 NAND 闪存上的多级电压编程特性，来实现高密度闪存上兼顾空间和寿命的 IUDW 系统。该系统主要包括四个模块，实现细节总结如下：

- **页面更新模块**维护了两种页面布局，创新地提出了部分编码页的思想，将页面划分为编码区和非编码区，其中非编码区通过无损数据压缩消除数据冗余来获取。在页面初次进行编程的时候，会完成页面划分，且后续不会改变，用于页面划分管理的元数据会存放在页面拓展区中。

• **差量压缩模块**修改了以往的读写请求逻辑，对于写请求逻辑，会读取并重建旧的页面来保留页面更新前后的细粒度差量，大部分的重复数据会进行丢弃；对于读请求，会将读取到的原始 NAND 闪存页面的非编码区和编码区分别进行解压缩和解码，得到基准数据和细粒度差量，并通过差量解压缩恢复的页面数据。

• **编码管理模块**从多级电压编程特性角度出发，对差量数据和编码管理元数据都进行数据编码，两种数据共同存在于编码空间中。编码空间会进行隐式划分，差量数据和编码管理元数据从空间的两端，分别以先横向后纵向、先纵向后横向的拓展方式，来充分利用编码区中的所有存储单元的所有等级电压，实现最大程度地就地更新。

• **更新优化模块**在 NAND 闪存上实现最大限度就地更新，聚焦了就地更新如何兼容原有的异地更新、兼容写缓存和解决和垃圾回收操作面临的冲突等，提出了 NAND 闪存上的混合页面更新框架。

第 4 章 测试评估

4.1 引言

本章介绍对本文方案的全面测试。本章主要内容包括：章节4.2介绍用于验证本文方案的仿真模拟实验环境搭建。章节4.3介绍本文针对单个页面上的微基准测试的测试方法和测试结果，体现本文方案在页面可更新次数上相对于同类方案的优势。章节4.4介绍了本文方案的宏基准测试方法和在真实世界场景下的总体优化结果，验证了本文方案在空间消耗和寿命优化上的效果。章节4.5介绍系统的性能开销评估，表明本文方案的开销很低。

4.2 实验设置

4.2.1 实验环境构建

本文基于 SSD 仿真模拟环境进行设计实现和测试评估。原因如下：（1）SSD 封装的设计使其难以被修改。SSD 的 NAND Flash 存储芯片和其固件管理程序通常被存储厂商在出厂时就严格封装，其内部实现无法被用户感知，对用户完全透明，因而也无法对其管理程序进行修改。（2）最先进的 SSD 仿真模拟近乎还原 SSD 的时延特性，让 SSD 的设计和优化从工业界走进学术界。在最近 10 年的系统和存储领域最前沿会议的 391 篇有关 SSD 存储优化的论文中，超过一半的论文选择在 SSD 仿真模拟环境上设计实现[28]。

现有最先进 SSD 仿真模拟环境并非均满足本文要求。本文对 SSD 仿真模拟需求包括：（1）NAND Flash 颗粒能提供到最新的 QLC 颗粒的仿真模拟。（2）底层数据存储模拟需要提供到通过物理地址 PPA 进行数据的索引。经过调研，部分最先进的 SSD 模拟方案，FEMU-BBSSD[28]、NVMeVirt-SSD[29]、LightNVM[30] + FEMU-OCSSD[28] 架构，调研结果分别如表4-1所示。FEMU-BBSSD 虽然提供了 QLC 颗粒的支持，但其仅能提供逻辑地址索引数据。NVMeVirt-SSD 针对几种实际的 SSD 设备进行了仿真，但均不支持 QLC 颗粒和物理地址索引。近年来，也有部分 SSD 仿真模拟器针对存储单元的可靠性做了有关优化和研究[31]，但不是本文关注的重点。

本文采用 LightNVM+FEMU-OCSSD 协同架构进行设计和实现。LightNVM + FEMU-OCSSD 的架构采用了 Open-Channel SSD 的设计理念。该架构解耦了传统 SSD 的封装特性，将 SSD 的固件管理程序 FTL 和 SSD 的存储芯片 NAND Flash

表 4-1　SSD 仿真模拟环境调研

SSD 模拟方案	QLC 颗粒支持	物理地址索引支持
FEMU-BBSSD	√	×
NVMeVirt SSD	×	×
LightNVM + FEMU-OCSSD	√	√

分离开，将操作 NAND Flash 的接口向上暴露给操作系统，即将固件管理程序上移至操作系统实现，这允许开发者可以通过设计和优化位于操作系统的管理程序来完成对 NAND Flash 的管理和维护。其中，LightNVM 是上述提到的管理程序，是 Linux 操作系统上的一个内核模块；FEMU-OCSSD 则是上述提到的存储芯片 NAND Flash，其读取、写入和擦除受 LightNVM 的管理。在 LightNVM+FEMU-OCSSD 环境的构建过程中，本文修改了 FEMU OCSSD 的部分代码逻辑，用于修复并添加 NAND Flash 页面的拓展字段 OOB 的支持，这为本文方案的设计实现提供了必要的支持。

4.2.2 实验环境配置

本文在 Linux 内核版本 5.4 的 8 核、16G 的虚拟机上实现并完成测试评估，采用 Version 1.2 的 Open-Channel SSD。完整的系统环境配置参数如表4-2所示。

表 4-2　系统环境配置

系统配置项	系统参数
Linux 发行版	Ubuntu
Linux 内核版本	5.4.0-64
CPU 核心数	8
系统内存	16G
LightNVM 版本	5.4
FEMU-OCSSD 版本	Version 1.2

FEMU-OCSSD 通过内存模拟的 NAND Flash，所能模拟的 SSD 的容量大小有限。参考同类工作，在测试评估中，本文配置了 2 个通道、8 个存储芯片、16GB 大小的 SSD 供测试评估，每个 NAND Flash 的页面大小是 4KB，拓展字段的大小是 16B，完整的 SSD 参数配置如表4-3所示。

4.3 微基准测试

4.3.1 测试方法

在测试指标上，微基准测试聚焦于单个物理页的极限编程次数。讨论一个

表 4-3　SSD 参数配置

SSD 配置项	SSD 参数
SSD 容量	16G
Channel 数	2
每个 Channel 的 Chip 数	4
每个 Chip 的 Plane 数	1
每个 Plane 的 Block 数	1024
每个 Block 的 Page 数	512
每个 Page 的大小	4KB
拓展字段大小	16B

NAND Flash 页面，在不擦除的情况下所能进行的最大编程次数。SSD 的寿命情况和页面的擦除次数有着直接关系。而同时当一个页面达到完全编程前，即页面所有存储单元均达到 V_{MAX} 电压前，该页面的可更新次数越多，即意味着失效擦除旧页面和重新申请新页面的次数越少。因此聚焦于单个物理页面的极限编程次数的微基准测试可以反映不同方案对 SSD 寿命的改善情况。

在测试方法上，微基准测试采用高频率的就地密集写负载。针对不同的内容局部性程度、不同的数据可压缩性程度（在章节4.3.2标准化为 LC 因子）的场景，构建了超高频率的就地密集写的测试负载，该测试负载下会对同一个逻辑地址 LBA 进行频繁的更新写操作。在此工作负载下，观察不同对比对象为同一的逻辑地址分配新的物理地址 PPA 时，所能进行的平均写操作数。

在测试对象上，微基准测试选取了相关领域最先进的两个代表性工作。WOMv 工作[22] 旨在通过对全部数据进行编码换取全部数据的可更新次数；SLC-Append 工作[16] 旨在将高密度闪存颗粒（如 QLC）当作 SLC 使用以换取页面内的可追加写操作。这两项工作和本文出发点类似，均从尽可能实现 NAND Flash 的页面的就地更新的角度来优化 SSD 的使用寿命。

4.3.2 内容的局部-可压缩性因子

本文对负载的数据内容可压缩性和局部性敏感。在不同特征的数据内容下，本文方案的优化收益会呈现差异化的效果。影响本文的数据内容特征包括：（1）数据的可压缩性；（2）数据的局部访问特性。数据的可压缩性指的是数据在无损压缩前后的变化程度，可以通过压缩率 Com 来表示，如式4-1所示，其中 $Size_{new}$ 表示压缩后的数据大小，$Size_{ori}$ 表示压缩前的数据大小。数据的局部访问特性指的是同一份数据更新前后数据内容的变化程度，可以通过变化率 $Diff$ 来表示，如式4-2所示，其中 $Size_{all}$ 表示原数据的大小，$Size_{diff}$ 表示数据的改变量。在真实负载中，数据往往会呈现不同程度的可压缩性和局部特性。

$$Com = \frac{Size_{new}}{Size_{ori}} \tag{4-1}$$

$$Diff = \frac{Size_{diff}}{Size_{all}} \tag{4-2}$$

本文的测试评估中将数据内容的可压缩性和局部性标准化为一个影响因子 LC。本文的方案受数据的可压缩性 Com 和数据的局部性 $Diff$ 的共同作用，因而通过定义数据的一个新的特征来表示这种影响。本文定义了数据的局部-可压缩性因子 LC，如式4-3所示。当数据的可压缩性越好，即 Com 越小，LC 值也会越小，反之越大。当数据的改变量越少，即 $Diff$ 越小，LC 值也会越小，反之也越大。在本文中，LC 因子需要满足小于 1 的条件，否则视为数据变化量太大，或者数据可压缩性太差，而无法进行就地更新。

$$LC = \frac{Diff}{1 - Com} \tag{4-3}$$

4.3.3 测试结果分析

微基准测试的结果如图4-1所示。图中，IUDW 为本文所实现的方案，IUDW(x,4) 表示本文采用的 4 bits 表征 x bits 的方式对差量进行编码的就地更新方案。WOMv(x,4) 表示 WOMv 工作对全部数据进行 WOMv(x,4) 编码的就地更新方案。横坐标表示从 0.05 变化到 0.60 的 LC 因子，纵坐标表示每个页面能进行的极限写入次数。本文没有对比 WOMv(3,4) 方案，原因在于 WOMv(3,4) 方案的缺点使其应用于 SSD 上的无比低效且困难，相对于前两种 WOMv 方案，WOMv(3,4) 方案的逻辑地址和物理地址的页面不对齐，带来极其复杂的映射管理结构，但仅仅提供额外一次的就地更新，故本文不将 WOMv(3,4) 方案作为对比对象。

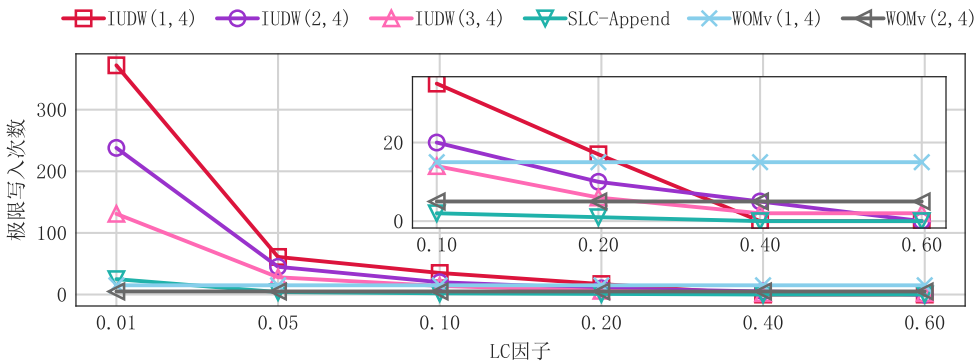


图 4-1　单个页面极限写入次数测试

从测试结果中可以看出：（1）在较小的 LC 因子下（如 LC 值为 0.01 时），本文的 IUDW 方案带来极为显著的写入次数提升，相对于其他方案可以带来 5 倍到 100 倍的写入次数提升。LC 因子小于 0.10 时，本文方案始终优于其他工作。（2）当 LC 因子增大，LC 因子达到 0.20 左右，IUDW 方案的极限写入次数开始下降，极限更新次数上会小于 WOMv 方案，但由于 WOMv 方案的实际写入数据只有一个页的实质上的极限写入量仍可能优于 WOMv 方案（见图4-2的分析）。（3）本文的 IUDW 方案和 SLC-Append 方案均对 LC 因子敏感，IUDW(1,4) 方案变化最大，IUDW(3,4) 方案变化相对较平缓。WOMv 方案则在不同的 LC 因子下基本保持稳定。此测试结果得出的结论为：**在较小的 LC 因子下，IUDW 方案相对于其他方案能够带来 5 倍到 100 倍的写入次数提升，但效果会随 LC 因子的增大而降低。**

在真实世界负载中，较小 LC 因子的数据是非常常见的。一个典型的场景是事务处理程序 OLTP，数据往往具备高度的可压缩性，并且会涉及频繁的记录更新操作，每次记录的更新往往只有几字节到几十字节。此外，如版本管理程序 Git 也会带来小数据的频繁更新，如文件系统或去重系统等具有元数据管理结构 (meta data) 的场景会对元数据进行频繁但细小的更新，等等均会带来很小的 LC 因子。

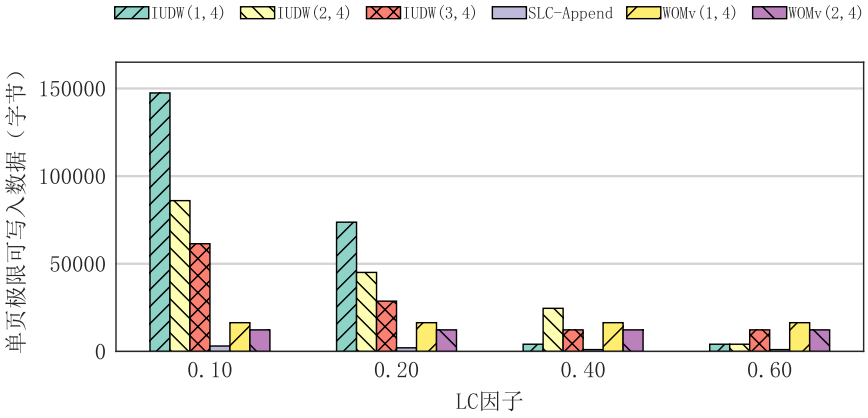


图 4-2　单个页面极限可写入数据

在较小的 LC 因子下，IUDW 方案仍可能保持更优，即使极限写入次数更低。在图4-1中，当 LC 因子达到 0.40 后，IUDW(2,4) 和 IUDW(3,4) 方案的极限写入次数已经小于 WOMv(1,4) 和 WOMv(2,4) 方案。但如图4-2所示，当 LC 因子达到 0.40 时，IUDW(2,4) 和 IUDW(3,4) 方案的单个页面极限可写入数据仍会超过 WOMv(1,4) 和 WOMv(2,4) 方案。这意味着在 SSD 的寿命延长上，IUDW 方案仍会更优。带来极限写入次数和极限写入数据量对比结果未必相同的原因是，WOMv 方案消耗了大量空间来争取可更新次数，但单个页面下实际表示的数据只有原来的一半或四分之一。因此两者共同作用的结果未必会更优。

4.4 宏基准测试

4.4.1 测试负载及处理

本文收集了六个具有代表性的真实世界负载用于宏基准测试。真实世界负载（I/O Trace）是在某个特定场景下通过操作系统的块设备层或文件系统层采集到的一段时间内的所有 I/O 请求操作。对真实世界负载进行重放，得到的指标可以很好地评估一个方案在真实世界中的优化效果。因此其被广泛应用于计算机系统结构的文件系统和存储领域的研究中[15,16,22,32]。本文收集了六个典型的工作负载，分别来自数据库环境、金融机构①、智能手机、个人工作环境、邮件服务器和网络服务器②等 I/O 密集的场景，详细信息如表4-4所示。

表 4-4　真实世界负载信息

负载名称	负载描述	负载收集
YCSB	用 RocksDB 运行了 YCSB 负载的记录	以色列理工学院[33]
Financial	大型金融机构运行的 OLTP 应用的记录	惠普公司/IBM 公司
Phone	智能手机上运行如聊天，视频等常用应用下的记录	VISA 实验室[34]
Home	研究小组进行开发、实验等活动的桌面下的记录	佛罗里达国际大学
Mail	部门邮件服务器上的记录	佛罗里达国际大学
Web	部门网络邮件代理和在线课程管理服务器上的记录	佛罗里达国际大学

原始的真实世界负载无法直接用于重放，需要进行预处理。主要原因有：（1）真实世界负载需要缩减。在章节4.2.2中提到，FEMU-OCSSD 仿真模拟环境所能模拟的 SSD 容量有限，本文的测试中配置了 16GB 容量的 SSD。真实世界负载往往采集于实际的 SSD，而目前实际的 SSD 已经可以达到 512GB 及以上的容量。因此需要对原始的真实世界负载进行缩减，以使其能够在 16GB 的模拟 SSD 上完成重放。但在缩减过程中，会面临一个挑战，即如何保证缩减后的负载的特征和缩减前的负载特征保持相同。（2）真实世界负载需要标准化。真实世界负载采集的格式各不相同，同时并非所有字段对宏基准测试都有用。需要对这些记录进行处理和标准化，保留有效字段。

本文采用的真实世界负载预处理的方法为：（1）只保留负载中的所有写请求。本文核心是面向 SSD 写请求逻辑的优化，读请求并不会影响到宏基准测试的结果。考虑到重放负载的时间成本，本文仅保留了负载中的全部写请求操作。（2）保证缩减前后的逻辑地址的更新频率次数不变。首先，本文对更新频率次数进行较细粒度的区间划分，如分为 1 次、2-5 次、5-10 次、...、150-300 次、大于 300 次若干

① Oltp application i/o, https://traces.cs.umass.edu/index.php/ Storage/Storage

② Fiu iodedup, http://iotta.snia.org/traces/block-io/391

区间，遍历负载，建立区间和逻辑地址的映射关系。第二步，统计缩减前的每个区间下的逻辑地址数，根据不同负载的缩减比例，计算缩减后每个区间理应分配的逻辑地址数。第三步，重新遍历负载，如果逻辑地址对应的区间还有剩余配额，则将该逻辑地址纳入候选子集中。第四步，根据候选子集生成新的子负载，根据子负载的写流量和空间大小设置微调因子，重新执行第二步直到满足条件，得到最终的子负载。（3）重新旧的映射逻辑地址到新的逻辑地址空间。将得到的子负载的逻辑地址映射为一个新的逻辑地址空间，这个地址空间的最大逻辑地址应该落于 16GB SSD 的可寻址范围中。（4）只保留记录中的有效字段。确定子负载及其逻辑空间后，对所有记录进行标准化，仅保留宏基准测试所需要用到的字段，最后将缩减后的标准化负载进行输出。

表 4-5　负载处理前后信息（单位：块）

负载名称	缩减前空间大小	缩减后空间大小	缩减前写流量	缩减后写流量
YCSB	460 839 400	928 304	1 030 048 168	2 269 736
Financial	1 089 517	449 827	34 364 277	11 526 821
Phone	77 551 632	800 664	367 456 496	3 620 136
Home	10 387 508	899 410	136 881 792	8 468 902
Mail	5 251 744	150 304	332 205 808	9 074 352
Web	1 984 632	317 016	89 421 616	10 882 472

经过上述预处理得到的负载满足测试需求并保持了原有负载的特征。表4-5展示了六个真实世界负载进行缩减处理前后的空间大小和写流量。图4-3展示了六个负载缩减前后的特征，横坐标表示不同更新次数区间的占比，纵坐标表示缩减前后的不同负载。从图中可以看出，在缩减前后，所有的负载的更新次数区间分布基本保持一致。

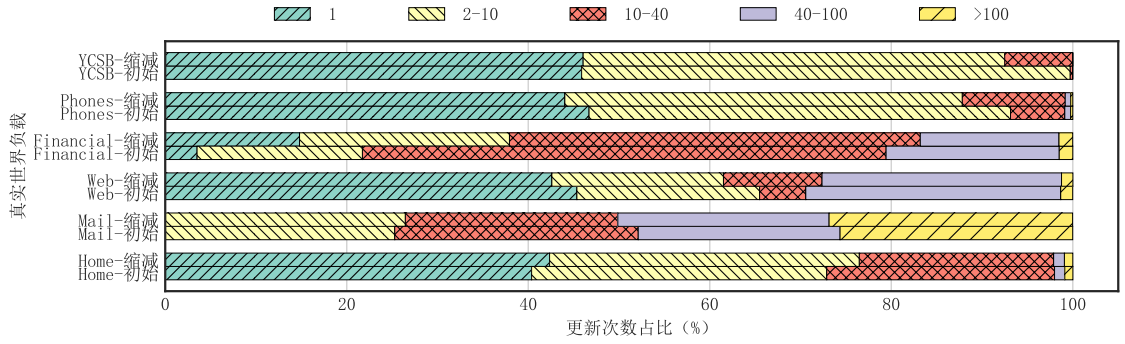


图 4-3　负载缩减前后的更新次数区间分布

4.4.2 测试方法

在测试指标上，宏基准测试同时关注 SSD 的两个最核心指标，空间和寿命。

SSD 的空间能够体现 SSD 所能存储的实际数据的最大数据量，空间的消耗情况可以通过**物理页的申请数**来体现。相同的负载下，物理页的申请数越多意味着空间消耗越快，SSD 的所能存储的实际数据量越小，空间越小，同时也会越早触发垃圾回收（GC）从而降低 SSD 的性能。SSD 的寿命情况能够体现 SSD 所能够使用的最长期限，SSD 寿命的消耗情况可以通过**物理页的擦除数**来体现。相同的负载下，物理页擦除次数越多意味着 SSD 的磨损速度越快，其寿命消耗也对应更快。

在测试方法上，宏基准测试采用真实世界负载重放的方式。在章节4.4.1中提到的六种真实世界负载还原了真实世界的六种场景，每个场景下进一步构建不同 LC 因子的子场景，并记录所有子场景下的物理页的申请数和物理页的擦除次数。研究本文方案在真实场景下的 SSD 空间和寿命上的消耗情况和改善情况。

在测试对象上，宏基准测试对比了原始的 SSD 方案和本文所实现的方案。和微基准测试一致，IUDW 为本文所实现的方案，IUDW(x,4) 表示本文采用 4 bits 表征 x bits 的方式对差量进行编码的就地更新方案。

4.4.3 测试结果分析

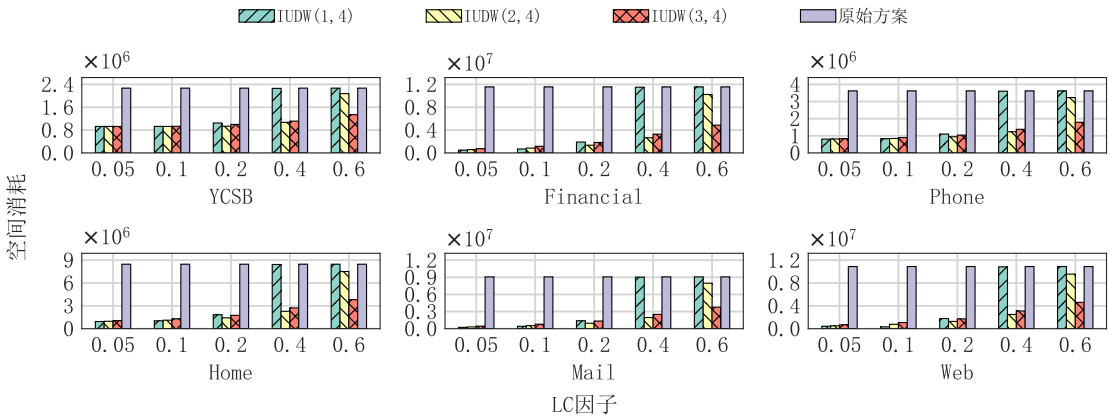


图 4-4　真实负载下的空间消耗对比

图4-4展示了真实负载下的空间消耗对比。横坐标表示不同的 LC 因子，纵坐标表示物理页的申请数。

从图中可以看出：（1）在各种真实世界负载下，本文的 IUDW 方案对空间的消耗均优于原始的方案，能够达到 70% 及以上的空间消耗的优化。空间消耗上的优化效果主要因为真实世界负载中具有频繁的重复块的更新写入操作，而在章节4.3.3的微基准测试结果中表明本文的 IUDW 方案能极大限度地进行就地更新，从而无需申请新的页。（2）空间消耗的优化效果会因负载而异。如在 Web 负载下的优化优于在 Phone 负载下，原因是 Web 负载的原地更新的特征会更为明显，即

更趋向于对已有的块进行更新操作。（3）本文的 IUDW 方案的优化效果会受 LC 因子的影响。在低 LC 因子下，空间消耗的改善具有极为明显的效果，但随着 LC 因子的增大，优化效果会逐渐降低。原因是高 LC 因子的数据往往是不可压缩的，并且改变量很大，这类数据并不适用于本文的 IUDW 方案。（4）不同的 IUDW 对象的优化效果在不同 LC 因子下表现不同。IUDW(1,4) 的变化趋势相对更加迅速，而 IUDW(3,4) 的变化趋势更加平缓。在低 LC 因子下 IUDW(1,4) 的优化效果显著优于 IUDW(3,4)；而在较高的 LC 因子下，则是 IUDW(1,4) 方案会急剧退化，但 IUDW(3,4) 仍然能保持较好的优化效果。通过图4-4得出的结论为：**在低 LC 因子下，本文 IUDW 方案能够减少 70% 以上的空间消耗，效果会受 LC 因子和负载的影响，不同 IUDW 方案在不同的 LC 因子场景下具有各自的优势。**

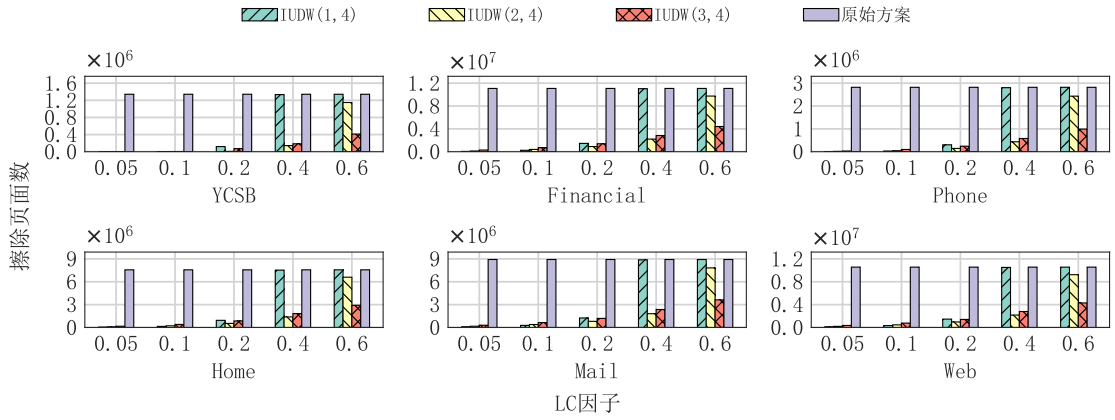


图 4-5　真实负载下的擦除页面数对比

图4-5展示了真实负载下的擦除页面数对比。横坐标表示不同的 LC 因子，纵坐标表示擦除的页面数。

从图中可以看出：（1）在较低 LC 因子下，IUDW 方案几乎不带来页面的擦除，相对于原始方案能够减少 90% 以上的擦除。原因是在章节4.3.3的微基准测试结果中表明，低 LC 因子下 IUDW 方案能够提供高达 20-70 次的可更新次数，大部分的物理页均能够在原来的页内完成就地更新而无需将原有的页面标记擦除并申请新的页。（2）擦除页面数的优化同样受 LC 因子的影响。随着 LC 因子的增大，IUDW 对于擦除页面数的优化效果会退化。（3）不同的 IUDW 方案在擦除页面数的优化上也会在不同 LC 因子下呈现不同优势。在低 LC 因子下，IUDW(1,4) 方案对擦除页面数的减少效果远优于其他方案。在高 LC 因子下，IUDW(3,4) 方案会具有更好的优化效果。通过图4-4得出的结论是：**在低 LC 因子下，本文的 IUDW 方案能够减少 90% 以上的页面擦除数，效果会随 LC 因子的增加而退化，不同 IUDW 方案在不同的 LC 因子下展示不同优势。**

4.5 开销评估

4.5.1 读写路径分析

为分析不同方案的开销情况，需要对不同方案的读写流程做关键路径的分析。公平和简单起见，开销评估不考虑 SSD 内部的并行设计，仅聚焦于单个页面的读取和编程操作。（1）页面的读取。如图4-6所示，在原始方案中，单个页面的读取包括单个页面的传输时延和单个页面的读取时延，在 SSD 领域统称为 *tR*。在 WOMv 方案中，由于使用了额外的页面来编码数据，对于原有的一个页面数据会带来多倍的页面传输和页面读取。在 IUDW 方案中，不会带来额外的页面传输和页面读取，但会额外引入页面解压。（2）页面的编程。如图4-7所示，在原始方案中单个页面的编程包括单个页面的传输时延和单个页面的编程时延，在 SSD 领域统称为 *tPROG*。在 WOMv 方案中，一个页面的读取带来了额外的多倍读取和多倍写入。IUDW 方案中，会存在两种可能的写逻辑。第一种是该页面是初次编程，会将原数据压缩，带来额外的页面压缩延迟。第二种是该页面是再次编程，即就地更新，会通过页面读取和解压将原数据进行重建，额外引入一次页面读取和页面解压。其中 SLC-Append 方案的关键路径基于 SLC 缓存，无法很好地量化 SLC 缓存和 QLC 主存的开销，本文对 SLC-Append 方案不做额外的对比。

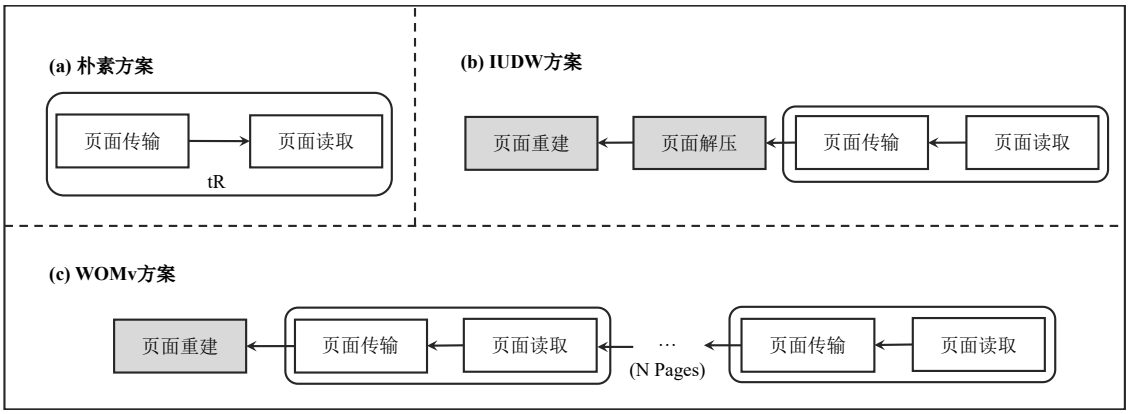


图 4-6　不同方案的页面读取路径

针对上述分析中的关键路径，本文对各个阶段的时延进行调研。经过调研：（1）压缩速率。计算型存储近年得到了飞速发展，目前已经有成熟的商用的压缩型 SSD，ScaleFlux CSD-3000①，其顺序写的吞吐量可以达到 4.8GB/s。CSD-3000 采用额外的硬件压缩引擎来处理专项处理 SSD 内部的压缩和解压缩。与图像专用处理器（GPU）类似，专项的压缩和解压缩计算引擎的速率比传统的处理器快多个数量

① Get better utilization, efficiency and TCO with ScaleFlux CSD 3000, https://scaleflux.com/products/csd-3000/

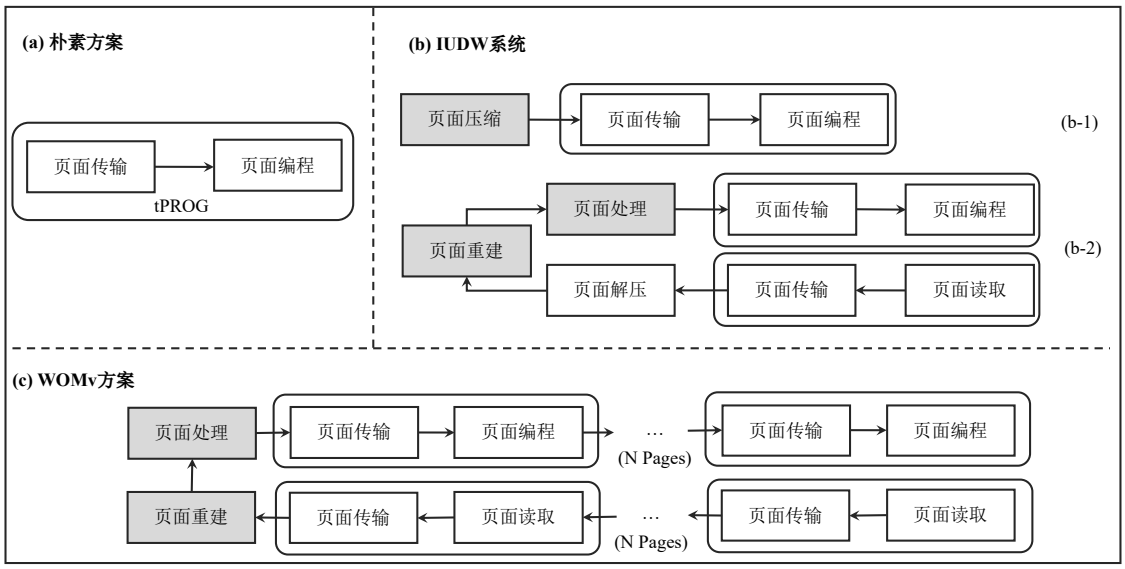


图 4-7　不同方案的页面写入路径

级。最新的压缩解压缩专项的硬件压缩引擎（如 Intel QAT 方案）目前已经能达到 160Gb/s①。考虑到不同的硬件成本，本文研究了从 500MB/s，到 10GB/s 的不同计算引擎速率下的开销情况。（2）QLC 读取和编程时延。目前最先进且公开读取和编程时延的 QLC SSD 是 Intel 于 2021 年发布的 3D-NAND QLC SSD[4]。本文参考其读取和编程的时延用于开销评估。（3）页面的重构和处理时延。对应图4-6和图4-7中的深色部分，其逻辑并不复杂，仅涉及编码的替换，其时延相对于复杂的计算密集型的压缩和解压缩和 QLC NAND Flash 的时延可忽略不计。最终读写路径的时延参数配置如表4-6所示。

表 4-6　读写路径的时延参数

时延项	时延参数
tPROG(μs)	1630
tR(μs)	85
压缩速率 (GB/s)	0.5/1/5/10
页面处理 (μs)	-

4.5.2 开销结果分析

根据表4-6的时延参数，对各种方案的开销进行评估，结果如图4-8所示。图中的横坐标表示压缩的速率，纵坐标表示单个页面读取或者编程的时延。从图4-8可以看出：（1）读时延开销评估。在最坏情况下，IUDW 方案的读时延开销仅在 9.0%，

① Intel QAT: Performance, Scale, and Efficiency, https://www.intel.com/content/www/us/en/architecture-and-technology/intel-quick-assist-technology-overview.html

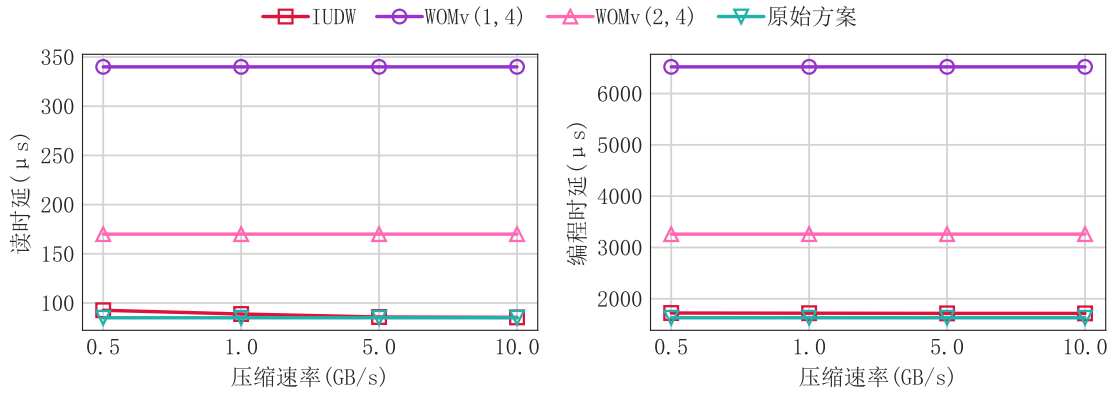


图 4-8　不同方案的开销对比

并随着压缩速率的增大而进一步减小。而 WOMv 方案会带来两倍或四倍的读时延开销。（2）编程时延开销评估。在最坏情况下，IUDW 方案的编程时延开销仅在 5.7%，并随着压缩速率的增大而进一步减小。IUDW 方案在单个页面的编程时延上除了压缩时延外，额外引入了读时延，但由于单个页面的编程时延远大于读时延和压缩时延，整体的编程时延开销并不会太大。而 WOMv 方案同样会带来两倍或四倍的写时延开销。

4.6 本章小结

本章介绍了本文方案的实验环境，设计了微基准测试对该方案在单个页面上的可更新次数的能力进行有关评估，设计了宏基准测试来验证本文方案在真实世界负载下的总体优化效果，包括空间消耗和寿命消耗上的优化等，最后对本文方案的读操作和写操作进行了性能评估。

在实验设置部分，本文采用 LightNVM + FEMU-OCSSD 的 SSD 仿真模拟环境。在微基准测试中，本文方案相对于 SLC-Append 方案和 WOMv 方案在单个页面上能够带来 5-100 倍的更新次数提升。在宏基准测试中，在真实世界负载下，相对于原始方案，本文方案能够减少 70% 空间消耗和减少 90% 的寿命消耗。其中本文的 IUDW(1,4) 方案更适合低 LC 因子的场景，而 IUDW(3,4) 方案更适合于高 LC 因子的场景。开销评估表明，本文方案的开销最坏情况为，读时延约在 9% 的开销，编程时延约在 5.7% 的开销。

结　论

本文提出了 IUDW 系统，一个基于差量压缩和多级电压编程特性来解决高密度闪存上的寿命挑战的新型存储系统。IUDW 系统兼顾空间和寿命，在不带来额外的空间和性能开销的前提下，充分优化了 QLC-SSD 上的寿命问题。测试结果表明，IUDW 系统能够在数据冗余性和局部相似性较好的情况下，特别显著地减少 QLC-SSD 上的寿命消耗，并且具备很低的开销。本文的贡献点有：

1. 充分利用现实数据冗余特性和数据消冗技术来优化 QLC-SSD 的寿命问题。本文采用基于差量压缩技术的页面更新数据去重优化方案实现了 NAND 闪存上的细粒度数据更新，并采用基于无损压缩技术的页面内部数据消冗及其管理方案实现了 NAND 闪存上的页内更新。

2. 创新地将高密度闪存上的多级电压编程特性和数据消冗技术相结合。IUDW 系统提出部分编码页的概念，并基于多级电压编程特性来进行差量的存储，在高密度闪存上实现原地更新。

3. 提出适用于高密度闪存上且无需额外空间开销混合更新框架。本文结合上述技术实现了最大限度的就地更新，并针对就地更新可能会遇到的问题（如和现有的异地更新兼容、写缓存兼容和垃圾回收等）提出了适用于高密度闪存上且无需额外空间开销的混合页面更新框架来减少所有不必要的寿命消耗。

IUDW 系统仍然存在着诸多可以改进的空间，下一步的研究可以集中在以下的几个方面：首先，IUDW 系统不同多级电压编码参数在不同 LC 因子下呈现不同的优化优势。后续可以针对这点观察，提出弹性的 IUDW 系统。即一个页面的编码区会存在多种编码方式，而不仅局限于现在的一种。其次，不相似的更新数据或者压缩性差的数据也会重复执行 IUDW 系统流程。IUDW 系统可以通过引入更新前后页面的相似性检测、压缩性快速检测等方式来减少这些不必要的流程。最后，就地更新框架可以额外引入对并行的考虑。现在的 SSD 内部都会通过并行操作来加速，后续可以研究 IUDW 系统的混合更新框架如何兼容这些并行操作。

参考文献

［1］ REINSEL D, GANTZ J, RYDNING J. IDC White Paper Data Age 2025[M]. 2017.

［2］ JUNG M. Hello Bytes, Bye Blocks: PCIe Storage Meets Compute Express Link for Memory Expansion (CXL-SSD)[C]//Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems. 2022: 45-51.

［3］ YANG S P, KIM M, NAM S, et al. Overcoming the Memory Wall with CXL-Enabled SSDs[C]//Proceedings of 2023 USENIX Annual Technical Conference. 2023: 601-617.

［4］ KHAKIFIROOZ A, BALASUBRAHMANYAM S, FASTOW R, et al. 30.2 a 1tb 4b/cell 144-tier Floating-Gate 3D-NAND Flash Memory with 40mb/s Program Throughput and 13.8 gb/mm 2 bit Density[C]//Proceedings of 2021 IEEE International Solid-State Circuits Conference: volume 64. IEEE, 2021: 424-426.

［5］ 李绪金. NAND Flash 固态存储可靠性关键技术研究[D]. 哈尔滨工业大学, 2018.

［6］ 魏德宝. 基于错误特征的 NAND Flash 存储策略研究[D]. 哈尔滨工业大学, 2016.

［7］ LIANG S, QIAO Z, TANG S, et al. An Empirical Study of Quad-level cell (qlc) Nand Flash SSDs for Big Data Applications[C]//Proceedings of 2019 IEEE International Conference on Big Data (Big Data). IEEE, 2019: 3676-3685.

［8］ 夏文. 数据备份系统中冗余数据的高性能消除技术研究[D]. 武汉: 华中科技大学, 2014.

［9］ KULKARNI P, DOUGLIS F, LAVOIE J D, et al. Redundancy Elimination within Large Collections of Files.[C]//Proceedings of USENIX Annual Technical Conference, General Track. 2004: 59-72.

［10］ JAIN N, DAHLIN M, TEWARI R. TAPER: Tiered Approach for Eliminating Redundancy in Replica Synchronization.[C]//Proceedings of USENIX Conference on File and Storage Technologies: volume 5. 2005: 21-21.

［11］ XIA W, JIANG H, FENG D, et al. Combining Deduplication and Delta Compression to Achieve Low-overhead Data Reduction on Backup Datasets[C]// Proceedings of 2014 Data Compression Conference. IEEE, 2014: 203-212.

［12］ ZOU X, DENG C, XIA W, et al. Odess: Speeding up Resemblance Detection for Redundancy Elimination by Fast Content-defined Sampling[C]//Proceedings of 2021 IEEE 37th International Conference on Data Engineering. IEEE, 2021: 480-491.

［13］ MACDONALD J. File System Support for Delta Compression[D]. Berkeley: Department of Electrical Engineering and Computer Science, University of California at Berkeley, 2000.

［14］ XIA W, JIANG H, FENG D, et al. Ddelta: A Deduplication-inspired Fast Delta Compression Approach[J]. Performance Evaluation, 2014, 79: 258-272.

［15］ WU G, HE X. Delta-FTL: Improving SSD Lifetime via Exploiting Content Locality[C]//Proceedings of Proceedings of the 7th ACM european conference on Computer Systems. 2012: 253-266.

［16］ ZHANG X, LI J, WANG H, et al. Reducing Solid-State Storage Device Write Stress through Opportunistic In-place Delta Compression[C]//Proceedings of 14th USENIX Conference on File and Storage Technologies. 2016: 111-124.

［17］ CHEN F, LUO T, ZHANG X. CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory Based Solid State Srives[C]// Proceedings of 9th USENIX Conference on File and Storage Technologies. 2011.

［18］ ZHOU K, HU S, HUANG P, et al. LX-SSD: Enhancing the Lifespan of NAND Flash-based Memory via Recycling Invalid Pages[C]//Proceedings of Proc. 33rd Int. Conf. Massive Storage Syst. Technol. 2017: 1-13.

［19］ PARK J, LEE S, KIM J. DAC: Dedup-assisted Compression Scheme for Improving Lifetime of NAND Storage Systems[C]//Proceedings of Design, Automation & Test in Europe Conference & Exhibition, 2017. IEEE, 2017: 1249-1252.

［20］ SHI L, LUO L, LV Y, et al. Understanding and Optimizing Hybrid SSD with High-Density and Low-cost Flash Memory[C]//Proceedings of 2021 IEEE 39th International Conference on Computer Design. IEEE, 2021: 236-243.

［21］ LI J, ZHAO K, ZHANG X, et al. How Much Can Data Compressibility Help to Improve NAND Flash Memory Lifetime?[C]//Proceedings of 13th USENIX Conference on File and Storage Technologies. 2015: 227-240.

［22］ JAFFER S, MAHDAVIANI K, SCHROEDER B. Improving the Reliability of Next Generation SSDs using WOM-v Codes[C]//Proceedings of 20th USENIX Conference on File and Storage Technologies. 2022: 117-132.

［23］ ZUCK A, TOLEDO S, SOTNIKOV D, et al. Compression and SSDs: Where and How?[C]//Proceedings of 2nd Workshop on Interactions of NVM/Flash with Operating Systems and Workloads. 2014.

［24］ GU Z, LI J, PENG Y, et al. Elastic RAID: Implementing RAID over SSDs with Built-in Transparent Compression[C]//Proceedings of the 16th ACM International Conference on Systems and Storage. 2023: 83-93.

［25］ MAO B, JIANG H, WU S, et al. Elastic Data Compression with Improved Performance and Space Efficiency for Flash-based Storage Systems[C]//Proceedings of 2017 IEEE International Parallel and Distributed Processing Symposium. IEEE, 2017: 1109-1118.

［26］ YANG H, QIN G, HU Y. Compression Performance Analysis of Different File Formats[J]. arXiv preprint arXiv:2308.12275, 2023.

［27］ LIU S, ZOU X. QLC NAND Study and Enhanced Gray Coding Methods for Sixteen-level-based Program Algorithms[J]. Microelectronics journal, 2017, 66: 58-66.

［28］ LI H, HAO M, TONG M H, et al. The CASE of FEMU: Cheap, Accurate, Scalable and Extensible Flash Emulator[C]//Proceedings of 16th USENIX Conference on File and Storage Technologies. 2018: 83-90.

［29］ KIM S H, SHIM J, LEE E, et al. NVMeVirt: A Versatile Software-defined Virtual NVMe Device[C]//Proceedings of 21st USENIX Conference on File and Storage Technologies. 2023: 379-394.

［30］ BJORLING M, GONZALEZ J, BONNET P. LightNVM: The Linux Open-Channel SSD Subsystem[C]//Proceedings of 15th USENIX Conference on File and Storage Technologies. 2017: 359-374.

［31］ LEE J, PARK H, CHOI G, et al. ACE: An Analog Cell Emulator for Dependability Study of NAND Flash Memory[C]//Proceedings of 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume. IEEE, 2023: 28-34.

［32］ QIU J, PAN Y, XIA W, et al. Light-Dedup: A Light-weight Inline Deduplication Framework for Non-Volatile Memory File Systems[C]//Proceedings of 2023 USENIX Annual Technical Conference. 2023: 101-116.

［33］ YADGAR G, GABEL M, JAFFER S, et al. SSD-based Workload Characteristics and Their Performance Implications[J]. ACM Transactions on Storage, 2021, 17 (1): 1-26.

［34］ YANG Q, JIN R, ZHAO M. SmartDedup: Optimizing Deduplication for Resource-constrained Devices[C]//Proceedings of 2019 USENIX Annual Technical Conference. 2019: 633-646.

攻读学士学位期间取得创新性成果

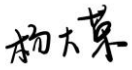
（一）参与的科研项目及获奖情况

［1］ **杨大荣**，林颀聪，孙赫辰. BoesFS: 基于 eBPF 实现的沙盒文件系统. 第三届全国大学生计算机系统能力大赛操作系统设计赛功能挑战赛道全国一等奖，2023

哈尔滨工业大学本科毕业论文（设计）
原创性声明和使用权限

本科毕业论文（设计）原创性声明

本人郑重声明：此处所提交的本科毕业论文（设计）《基于差量压缩的 QLC-SSD 寿命优化关键技术研究》，是本人在导师指导下，在哈尔滨工业大学攻读学士学位期间独立进行研究工作所取得的成果，且毕业论文（设计）中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本毕业论文（设计）的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名：  日期： 2024 年 5 月 27 日

本科毕业论文（设计）使用权限

本科毕业论文（设计）是本科生在哈尔滨工业大学攻读学士学位期间完成的成果，知识产权归属哈尔滨工业大学。本科毕业论文（设计）的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存本科生上交的毕业论文（设计），并向有关部门报送本科毕业论文（设计）；（2）根据需要，学校可以将本科毕业论文（设计）部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）本科生毕业后发表与此毕业论文（设计）研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉本科毕业论文（设计）的使用权限，并将遵守有关规定。

作者签名：  日期： 2024 年 5 月 27 日

导师签名：  日期： 2024 年 5 月 27 日

致　谢

由衷感谢导师夏文老师对我的精心指导。夏老师在大学四年中给予了我莫大的帮助。还在大三时，我有幸参加了操作系统大赛。夏老师每周都会进行进度把关并做问题指导，在决赛前更是和我们一起备赛至深夜凌晨，这种认真和负责让我和队友铭记在心。在推免过程，夏老师特别关心我的推免情况。在推免后的毕设科研过程中，夏老师几乎手把手领我走入学术的殿堂，从调研，到设计，到实验，到写作和答辩每个环节无微不至，还给了我参与学术交流的机会，并且每周都会和我同步课题的进度，这份感激之情溢于言表。在生活上，夏老师知道我家境困难，时常关心我的生活情况，让我遇到困难及时联系老师。很幸运能在人生生涯遇到这么认真和负责的老师。

由衷感谢生我养我的父亲和母亲，以及我至亲的妹妹。没有家里的支持，我没有条件走到今天一步。父亲和母亲文化水平不高，生活条件也不富裕。但他们相信知识可以改变命运，生活上省吃俭用，在我和妹妹的教育上却从不吝啬。父亲在学习上对我要求严格和教会我很多人生道理，母亲则在生活上对我无微不至。很幸运我能出生在这样一个家庭，一家人和和睦睦，并能给予自己追逐梦想的机会。

还要感谢课题组里的学长、参加比赛的队友、大学生活的舍友们和朋友们。感谢课题组里的黄浩学长、潘延麒学长和叶自立学长，几位学长在课题研究和论文写作上都给了我很多帮助。感谢队友林颀聪和孙赫辰能够选择信任我作为赛队队长，并和我一起并肩奋战，共同取得了不错的比赛结果，让我感受到了团队的合作感和归属感。感谢陪伴我大学四年的舍友，刘宇、徐明宇和孙赫辰，大家相互包容、相互支持，帮助我解决了很多生活上的困难。感谢我的大学朋友，何山、陈柏居、林颀聪和王佐文，大家其乐融融，都是我大学交过的好朋友，感谢你们给我的大学生活带来了这么多帮助。