ZNS 性能提升的研究综述

杨奕驰[1)]

[1)](华中科技大学计算机学院 武汉市 中国 430074)

摘 要 ZNS SSD 是近年来提出的一种新型固态硬盘（solid state drive，SSD），它以 Zone（分区）的方式管理和存取 SSD 内的数据。相比于传统 SSD，ZNS SSD 可以有效提升 SSD 的读写吞吐，降低写放大，减少 SSD 的预留空间。但是，ZNS SSD 要求 Zone 内必须采用顺序写模式，并且 Zone 上的存储分配、垃圾回收等任务都需要用户自行控制。因此 ZNS 也存在一些缺点和问题。本文首先介绍了 ZNS 的基本概念、优点和一些具有强相关性的技术，然后选择了六篇关于解决 ZNS 缺点，优化 ZNS 性能的前沿工作进行总结，充分展示了目前 ZNS 的研究现状。

关键词 分区命名空间，固态硬盘，日志结构合并树，并行性，数据放置，公平性

Literature review of ZNS performance improvement

Yichi Yang[1)]

[1)](Huazhong University of Science and Technology of Computer School, WuHan China 430074)

Abstract ZNS SSD is a new type of solid state drive (SSD) proposed in recent years, it uses Zone (partition) to manage and access the SSD data. Compared with traditional SSDS, ZNS SSDS can effectively improve SSD read/write throughput, reduce write magnification, and reduce SSD reserved space. However, ZNS SSDS require sequential write mode in zones, and storage allocation and garbage collection tasks in zones must be controlled by users. So ZNS also has some drawbacks and problems. This paper first introduces the basic concept of ZNS, advantages and some strongly related technologies, and then selects six articles on solving the shortcomings of ZNS and optimizing the performance of ZNS to summarize the frontier work, fully demonstrates the current research status of ZNS.

Key words Zone Namespace，Solid State Drive，Log-Sturctured Merge Tree，Parallelism，Data placement，Fairness

1. 引言

分区名称空间（ZNS）是一种新兴的存储接口，它可以匹配数据写入设备端后端闪存的方式，同时隐藏特定于闪存的管理的详细信息。具体来说，ZNS 提供了一个区域的概念，每个区域都被映射到一个或多个 flash 物理块，并将它们的操作约束公开给主机（例如，在块中的顺序写入和写入前擦除）。ZNS 的主要优点是使底层的固态驱动器（SSD）更加高效。由于 ZNS 允许主机在区域上管理数据，因此 SSD 可以摆脱对许多约束的管理，并变得更轻。例如，ZNS SSD 不需要为垃圾收集保留大量过度配置的闪存块，因为主机只负责回收区域并对物理数据

收稿日期：2022年10月22日

布局做出决定。此外，一个区域的所有写入都应该按顺序执行，这使得块到块（B2B）映射足够，而不是页到页（P2P）映射。这还可以通过删除用于覆盖 TB 规模的后端 flash 地址空间的大量内部 Dram，从而使 SSD 更便宜。虽然有这些优点，但是 ZNS SSD 要求 Zone 内必须采用顺序写模式，并且 Zone 上的存储分配、垃圾回收等任务都需要用户自行控制，这些都限制了 ZNS 性能，因此本文总结了一些目前最前沿的研究成果，展示了科研工作者是如何设计框架以避免 ZNS 的缺点带来的性能限制。

2. ZNS 及介绍

ZNS SSD (Zoned Namespaces SSD)是 2019 年由西部数据和三星公司推出的新一代 SSD，目前受到了工业界和学术界的广泛关注.由于基于闪存的 SSD 只有在块被完全擦除以后才能重写，传统的 SSD 通 过使用闪存转换层(flash translation layer，FTL) 来适应这一特性，但由于闪存块存在物理限制 (擦除操作 以块大小进行，而读写操作以页大小进行)，因此经 常需要进行垃圾回收(garbage collection，GC)同时也带来了预留空间(over provisioning，OP)和写放大(write amplification，WA)等问题，ZNS SSD 可以有效提升 SSD 的读写吞吐，降低写入时的写放大，减少 SSD 本身的预留空间，并且还能解决传统 SSD 在空间占用达到一定程度时由于内部垃圾回收导致的性能不稳定的问题，因此利用 ZNS SSD 来构 建数据库系统是一个趋势.

图 1 展示了 ZNS SSD 和传统 SSD 数据放置对比，在 ZNS SSD 上数据由主机程序显式地控制放置.虽然 ZNS SSD 具有如此多优点，但它同样带来了一些挑战.与传统基于闪存的 SSD 相比 ZNS SSD 移除了 FTL，将 Zone 的空间直接交由主机程序控制管理，由主机程序来处理垃圾回收、磨损均衡、数据放置等，这扩大了用户数据布局的设计空间.由用户程序来决定数据的放置、生命周期和垃圾回收时机，通过有效合理的组织数据，可以提高系统的性能.但 ZNS SSD 同样给主机程序的设计带来了新的要求，比如一个 Zone 内有一个写指针只能进行顺序写、不同 Zone 性能有差异、何时进行 Zone-Reset 操作等.

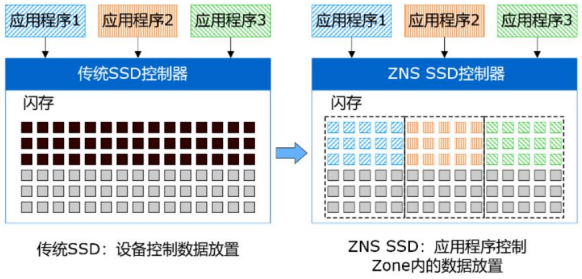

图 1 ZNS SSD 和传统 SSD 数据放置对比

2.1 ZNS的特性

ZNS 和高密度闪存技术对海量存储容量的高要求和卓越的读取性能使其在此类数据中心场景中具有广阔的应用前景。

高密度的闪存。ZNS 可以降低写放大因子，使底层存储更加可靠，因为主机需要显式地在顺序和回收区域(即 gc)中执行写。存储供应商使用最高密度的闪存技术(如三层单元(TLC)和四层单元(QLC))构建数据中心/企业 SSD 是有益的。由于最高密度的闪存可以将存储容量提高三到四倍，它可以使 SSD 在性能成本方面优于 HDD。然而，具有讽刺意味的是，它们的低持久性和较短的生命周期特性会增加总拥有成本(TCO)，并挑战在各种数据密集型计算中广泛采用此类 SSD。例如，QLC 的每擦除程序数(P/E)比 MLC 低 10 倍。ZNS 可以解决这个缺点，因为它允许主机直接回收应用程序知识的区域，并且只允许设备级的顺序写操作。这个特性显著提高了闪存块的利用率，而不会放大写操作，从而降低了 P/ e，使高密度闪存在许多数据密集型计算领域实用。注意，即使写操作应该按顺序执行，但读服务没有限制;可以读取区域中的任意地址。许多读取密集型应用程序可以享受 ZNS 支持的低成本、高性能功能。

强大的读取能力。利用 ZNS SSD 的一个特定用例场景是大规模推荐系统，用于 AI/ML 服务的分布式键值存储(KVS)，以及数据中心的社交图数据管理。例如，Facebook 和百度的推荐系统将用户 TB ~ pb 规模的嵌入表保存在企业 SSD(而不是 HDD)中，并随机读取嵌入表的特征向量。Facebook 还在 MySQL 表中维护社交图数据，所有对应的表行都由 KVS (RocksDB)存储和服务。请注意，推荐系统和 RocksDB 场景以大多数顺序的方式偶尔更新大规模用户数据(例如，配置文件和图表)，同时集中读取

数据，用于几个 AI/ML 服务或图表分析的训练和推断。

2.2 RockDB

Facebook 开发了 RocksDB，这是一个基于日志结构合并树(LSM-Tree)的开源键值存储。与其他键值存储不同，它支持在刷新和压缩例程中使用多线程。LSM-Tree 使用异地更新减少随机写入，并有一个压缩例程来收集异地更新生成的无效数据。它们使 LSM-Tree 具有快速的写入性能，并有助于有效地利用空间。这就是为什么 LSM-Tree 是键值存储的后端，以及 LSM-Tree 被各种键值存储后端采用的主要原因。

图 2 显示了由 Memtable、预写日志(WAL)文件和排序序列表(SSTable)文件组成的 RocksDB 架构。Memtable 是一个存在于内存中的写缓冲区，它聚集用户写 I/O，直到达到缓冲区阈值。缓冲创建粗粒度的写 I/O，并允许更多地利用磁盘。SSTable 由压缩结果生成。它是带有 LSM-Tree 的 RocksDB 的核心。RocksDB 压实采用水平压实。分层压缩限制重叠表的数量，以实现最佳的读取性能。但是，当它执行排序合并时，它会遇到高写放大的问题。因此，位于级别 1 的 SSTable 由于写入放大而出现问题。

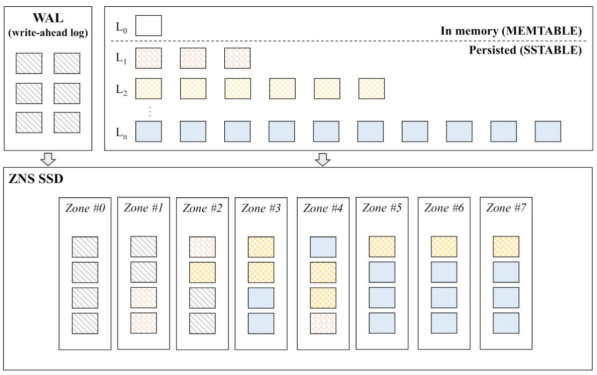

图 2 RocksDB 架构与 ZenFS。Ln 表示 RocksDB 中 LSM-tree 的第 n 级。Zone #0~#7 代表 ZNS SSD 的部分 Zone

2.3 ZenfS

ZenFS 是 Western Digital 为 RocksDB 开发的文件系统插件。它位于 RocksDB 和 Linux 内核之间。要使用符合可移植操作系统接口(Portable Operating System Interface, POSIX)的 I/O 功能，需要使用 libzbd 获取文件描述符。在获得文件描述符后，ZenFS 可以直接 I/O 的形式将 RocksDB 数据刷新到 ZNS SSD 中，并带有符合 posix 的函数。由于数据必须按顺序记录在每个分区中，因此需要更改 I/O 调度程序，以确保 I/O 顺序为。

图 3 显示 ZenFS 管理每个区域，划分为日志和数据区域。日志区域中的区域数在编译时是静态定义的。相比之下，数据区域中的区域数量定义为区域总数减去日志区域区域的数量。

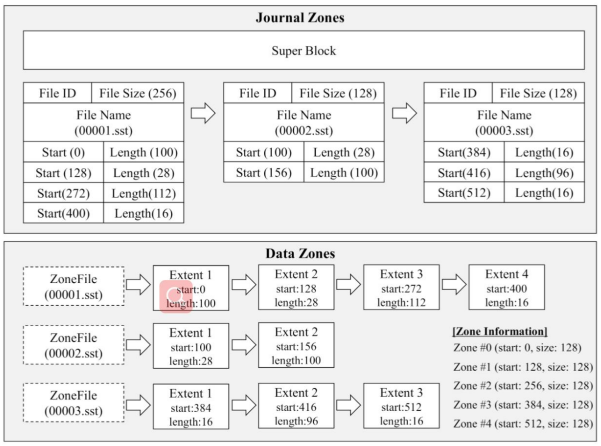

图 3 ZenFS 的整体架构

日志区域包含 ZenFS 超级块和 zonfile 元数据。超级块区域有 ZenFS 创建信息。日志区域中的其余区域包含分区文件元数据，例如映射区域信息的 WAL 和 SSTable。zonefile 元数据包含文件 ID、文件大小、文件名和 extent 元数据数组。

zonefile 是 ZenFS 用来处理 RocksDB 文件的抽象文件。它的元数据被上传到内存中，并在系统崩溃时写入日志区域进行恢复。它的数据以数据区域的范围为单位写入。区段中写入分区的分区文件数据的连续部分。extent 中，start 表示第一次写入数据时，WP (write pointer)在分区中的位置;length 表示在分区中连续写入的大小。

图 4 描述了 ZenFS 中区域的分配。图中 L3 (Level 3)数据请求一个 zone。在这种情况下，zone allocator 会检查 ZNS SSD 中的所有 zone，发现只有无效数据的 zone。如果分区分配器找到一个只包含无效数据的分区，那么所有找到的分区都将被擦除。例如，在图中，区域#2 只有无效数据。因此，选择 Zone #2 作为擦除目标并擦除。在擦除所有分区后，分区分配器将找到与所请求的数据具有最小级别差异的分区。在图中，区域分配器发现 zone #1 与请求的数据具有最小的级别差异。因此，分区分

配器返回 zone #1 以写入请求的数据。此外，分配结果被写入 zonfile 元数据。

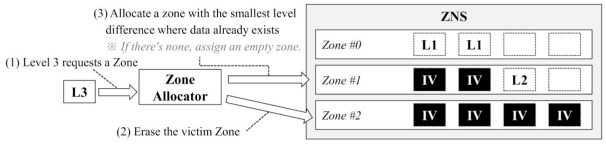

图 4 ZenFS 中的分区分配例程。L1、L2 和 L3 分别表示在 RocksDB 中包含来自第 1、2 和 3 级的 SSTable 文件的区段。IV 表示被删除的 SSTable 文件的无效范围。

3. ZNS 的性能优化研究

3.1 ZNS SSD 上的基于 LSM 的紧凑感知区域分配键值存储

与传统的基于块的 SSD 不同，分区命名空间(ZNS) SSD 通过分区块接口公开存储，完全消除了设备内垃圾收集(GC)的需要，并将这一责任移交给应用程序。因此，应用程序感知的数据放置决策为主机上的应用程序提供了执行高效 GC 的机会。同时，RocksDB for ZNS SSD 通过 ZenFS(一个用户级文件系统)使用基于生命周期的 zone 分配算法(LIZA)将具有相似失效时间(生命周期)的数据放置在同一个 zone 中，并在回收 zone 时最大限度地减少有效数据复制的 GC 开销。然而，Lee 等人发现 LIZA 通过根据 LSM -树的层次结构级别预测每个 SSTable 的生命周期来分配区域，由于 SSTable 生命周期的不准确预测，在最小化写放大(WA)问题方面非常低效。

并且 Lee 等人观察到 LSM -树中 sstable 的删除时间完全由压缩过程决定，因此在 ZenFS 中提出了一种新的压缩感知 zone 分配算法(CAZA)，允许新创建的 sstable 在未来合并后一起删除，该算法能够感知 LSM -树的压缩过程。CAZA 将具有重叠关键范围的 sstable 放置在同一区域，以便选择用于 LSM -树压缩的 sstable 位于同一区域内。具体来说，CAZA 的设计考虑了 LSM -树的以下特点。首先，sstable 只在压缩过程中被删除和失效，并相应地创建新的 sstable。其次，在压缩过程中，选择作为受害者的 SSTable 和与该 SSTable 的密钥范围重叠的 SSTable 一起被删除并失效。

实验结果表明，与写密集型合成工作负载的 LIZA 相比，CAZA 减少了多达两倍的区域清理期间产生的有效数据副本，并至多减少了 7.4%的 WA（写放大）。

3.2 ZNS SSD 的高效键值数据放置

基于日志结构合并树(LSM-Tree)的键值存储由于其顺序写入特性而具有较高的 I/O(输入/输出)性能而备受关注。但是由于压缩导致的写操作过多，会导致 SSD (Solid-state Drive)的寿命缩短。Oh 等人发现，现有的一些旨在通过使用分区命名空间 ZNS 来减少垃圾收集开销的研究虽然实现了主机自行决定数据存放于 SSD 盘的位置，但是由于没有考虑键值数据的生命周期和热度，在性能提升方面存在一定的局限性。

因此 Oh 等人在本文中提出了一种方法，通过优化基于 LSM - 树的键值存储的基于区域的数据放置，在压缩过程中快速保护空区域。此外，Oh 等人还提出了一种考虑数据生命周期的垃圾收集技术，以最大限度地减少写放大。

Oh 等人通过修改 RocksDB 的 ZenFS 实现了前文提到的方法，并在 QEMU 5.1.0 中使用仿真 ZNS 进行性能评估。实验结果表明，空间效率最高可提高 75%。如果再使用改进的垃圾收集算法，将性能还能再提高 10%。

3.3 通过并行 I/O 机制加速小区域 ZNS SSD 的 RocksDB

分区命名空间(ZNS)是一种新颖的存储接口，它向主机系统提供与物理介质(例如 NAND 闪存)一致的逻辑区域。由于物理介质的错位更新方案，基于 log-structured merge -tree (LSM-Tree)的键值存储(如 RocksDB)非常适合 ZNS SSD 的只能追加约束。

Im 等人发现虽然 ZenFS (RocksDB 的一个文件系统插件)在大区域 ZNS SSD 上显示了不错的吞吐量，但在小区域 ZNS SSD 上的性能不足，因为小区域没有空间来利用内部并行性。

因此 Im 等人提出了两种并行 I/O 机制，通过将 I/O 数据并行分布到多个区域来利用外部并行性。根据 lsm 树层次的特点，将所提出的 I/O 机制应用于 ZenFS。此外，他们还提出了一种动态分区管理策略，通过将生命周期相似的数据块放置在同一个分区中，使放置在一个分区中的文件几乎同时被删除，从而最大限度地利用空间。这样就可以在不进行垃圾收集的情况下重用这些区域。实验结果

表明，与传统的 ZenFS 相比，该方法取得了显著的改进。它的将平均 I/O 性能提高了 7.5 倍，同时保持几乎相同程度的空间效率。

3.4 Fair-ZNS：通过自平衡 I/O 调度来增强 ZNS SSD 的公平性

NVMe ZNS (Zoned Namespace)是一种新型的存储接口，它将逻辑地址空间划分为固定大小的分区，每个分区严格遵循写约束，并使用写指针。

Liu 等人发现由于 ZNS 的顺序写约束，I/O 请求不会像传统的块接口 SSD 那样任意调度。当多个应用同时访问一个 ZNS SSD 硬件时，该约束会导致 I/O 阻塞恶化，造成严重的不公平。为了解决这一问题，Liu 等人提出了一种专门用于 ZNS SSD 的自平衡 I/O 调度，称为 Fair-ZNS，以平衡多个应用程序之间的减速并确保公平性。Fair-ZNS 通过最大减速值识别不公平请求，并将这些请求强制调度到队列头部，克服顺序写约束。为了消除暴力调度的负面影响，FairZNS 部署了一个自平衡协调器来微调请求的顺序。实验评估表明，与目前的 ZNS SSD 相比，Fair-ZNS 缓解了 I/O 阻塞，平均等待时间缩短了 8.3 倍，公平性提高了 2.3 倍，最大速度平均降低了 5.1 倍。

3.5 ZNS+：支持存储内区域的高级分区命名空间接口

NVMe 分区命名空间(ZNS)作为一种新的存储接口出现，其中的逻辑地址空间被划分为大小固定的区域，并且每个区域必须按顺序写入，以实现对闪存友好的访问。由于 ZNS 的顺序只写区域方案，需要日志结构文件系统(LFS)访问 ZNS 固态硬盘(SSD)。

Han 等人发现虽然 SSD 可以在当前的 ZNS 接口下进行简化,但其对应的 LFS 必须承担段压缩开销。因此 Han 等人为了解决这个问题，提出了一个新的 LFS -aware ZNS 接口(称为 ZNS+)，其中主机可以将数据复制操作卸载到 SSD 以加速段压缩。ZNS+还允许使用稀疏的顺序写请求覆盖每个区域，这使得 LFS 可以使用基于线程日志的块回收，而不是段压缩。他们还为支持 ZNS+的 LFS 提出了两种文件系统技术。支持回拷的块分配考虑 SSD 内不同拷贝路径上的不同拷贝成本。混合段回收在段压缩和线程日志之间根据开销选择合适的块回收策略。ZNS+已经在一个 SSD 模拟器和一个真正的 SSD 上成功实现。实验结果表明，ZNS+存储系统的文件系统性能是普通 ZNS-based 存储系统的 1.33 ~ 2.91 倍。

3.6 一种新的 LSM 风格的 ZNS SSD 垃圾收集方案

ZNS SSD 虽然通过将不同的工作负载分离到单独的区域提高性能和降低 WAF(写放大因子)，但是主机需要直接管理 ZNS SSD，例如 zone reset，并且它们有顺序写入约束。为了解决这个麻烦，一种简单的方法是应用传统的方案，如 LFS(日志结构文件系统)使用的段清理或 FTL (Flash 转层)使用的垃圾收集。唯一的区别是它基于区域单元而不是段单元应用垃圾收集。然而，Chio 等人分析之后发现，这种简单的方法会导致很长的垃圾收集延迟，因为一个区域的大小(例如 0.5 GB 或 1gb)比一个段的大小(2 MB 或 4 MB)大得多。

因此 Chio 等人探讨如何为 ZNS(分区命名空间)ssd(固态硬盘)设计一个垃圾收集方案。他们的工作首先展示了基于区域单元的朴素垃圾收集由于区域的巨大大小而导致很长的延迟。为了克服这个问题，设计了一种新的方案，特们称之为 LSM ZGC(日志结构化合并风格的区域垃圾收集)，它利用了以下三个特性:基于段的细粒度垃圾收集，以分组方式读取有效和无效数据，并将不同的数据合并到单独的区域。Chio 等人还利用区域的内部并行性，并通过隔离热数据和冷数据来减少候选区域的利用率。实验结果表明，该方案的性能平均提高了 1.9 倍。

4. 总结与展望

ZNS SSD 和传统的 SSD 的不同点在于可以将，可以通过将不同的工作负载分配到不同的区域以此提高 SSD 的性能。ZNS SSD 能够充分利用底层介质的存储容量，同时发挥介质本身的特性优势，它可以根据数据的类型和访问频率采用不同的数据分区方式，以减少整体写放大，从而延长硬盘寿命，同时，还可以改进的 I/O 访问延迟，支持主机与存储设备协作放置数据。

虽然 ZNS SSD 仍然存在 Zone 必须采用顺序写模式，Zone 上的存储分配、垃圾回收等任务都需要用户自行控制，等一些小缺点，但相信在本文总结的六篇工作的带动下，会有越来越多类似的工作出现，并且会推动 ZNS SSD 不断优化。

参 考 文 献

[1] Hee-Rock Lee, Chang-Gyu Lee, Seungjin Lee, and Youngjae Kim. 2022. Compaction-Aware Zone Allocation for LSM based Key-Value Store on ZNS SSDs. In Proceedings of 14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage ’ 22). ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3538643.3539743

[2] Oh, G.; Yang, J.; Ahn, S. Efficient Key-Value Data Placement for ZNS SSD. Appl. Sci. 2021, 11, 11842. https://doi.org/10.3390/app112411842

[3] Minwoo Im, Kyungsu Kang, and Heonyoung Yeom. 2022. Accelerating RocksDB for Small-Zone ZNS SSDs by Parallel I/O Mechanism. In 23rd International Middleware Conference (Middleware ’22 Industrial Track), No vember 7–11, 2022, Quebec, QC, Canada. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3564695.3564774

[4] R. Liu, Z. Tan, Y. Shen, L. Long and D. Liu, "Fair-ZNS: Enhancing Fairness in ZNS SSDs through Self-balancing I/O Scheduling," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, doi: 10.1109/TCAD.2022.3232997.

[5] Kyuhwa Han, Hyunho GwakDongkun Shin,Joo-Young Hwang. ZNS+: Advanced Zoned Namespace Interface for Supporting In-Storage Zone Compaction[C]Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation.2021：147-162

[6] Choi G, Lee K, Oh M, et al. A New LSM-style Garbage Collection Scheme for ZNS SSDs[C]//12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 20). 2020.