ZNS 与日志结构数据管理研究综述

李邦宇[1)]

[1)](华中科技大学计算机学院 武汉市 中国 430074))

摘 要 本文是作为 ZNS SSD 相关技术与日志结构数据管理方式相结合的研究综述。本文首先将从分区命名空间这种新兴技术与想法的来源开始，介绍 ZNS SSD 的相关基础概念，以及与传统 SSD 分区管理的区别，在此基础上将介绍 ZNS SSD 技术中的软件模型以及 zone 状态转换概念，其后将介绍与日志结构数据管理相结合的 ZNS SSD 技术研究现状。

关键词 分区命名空间 SSD 软件模型 日志结构数据

Literature review of NVM-based file system research

Bangyu Li[1)] Zhiwei Song[2)] Liangxing Liu[3)]

[1)2)3)](Huazhong University of Science and Technology of Computer School, WuHan China 430074)

Abstract This article is a research overview on the combination of ZNS SSD related technologies and log-structured data management methods. This article will first start with the source of emerging technologies and ideas such as partition namespaces, and introduce the related basic concepts of ZNS SSD and the differences from traditional SSD partition management. On this basis, it will introduce the software model and zone status in ZNS SSD technology Convert the concept, and then introduce the research status of ZNS SSD technology combined with log-structured data management.

Key words zone namespace; SSD; File System; software model; log structure data

1. 引言

将闪存用于固态驱动器的根本挑战是，所有的计算机都是围绕硬盘驱动器如何工作的概念构建的，而闪存的行为不像硬盘驱动器。闪存的组织结构与硬盘驱动器截然不同，

收稿日期：2022年1月10日

因此针对闪存的性能优化一直都是相关研究的聚焦点。

作为持久化存储设备的另一个代表，磁盘没有固有结构来规定扇区大小等功能。选择 512 字节扇区的长期标准仅仅是为了方便，随着现实生产环境中出现了达到多 TB 范围的驱动器容量，企业驱动器现在也会选择支持 4K 字节扇区。

为了使 SSD 能够与与磁盘兼容，SSD 也提供 512B 或者 4KB 大小的扇区抽象给软件使用。这隐藏了许多 SSD 的复杂特性，例如页面和擦除块大小、磨损均衡和垃圾收集。这种抽象也是为什么 SSD 控制器和固件比硬盘驱动器控制器更大、更复杂（并且更容易出错）的部分原因。对于大多数用途，块设备抽象仍然是正确的折衷方案，因为它允许未经修改的软件享受闪存的大部分性能优势，并且写入放大等缺点是可控的。

多年来，存储工业界一直在寻求块存储抽象方式的替代方案。如 Open-Channel SSD，将闪存的接口细节直接暴露给主机，部分 SSD 固件的功能也放在主机端运行。从所有功能运行在 SSD 设备端，到所有功能（FTL）放在主机端，各种方案被提出。工业界一致认为，早期的标准，如 LightNVM1.x 协议，暴露了太多的细节，需要软件处理不同厂商的不同闪存，还需要处理 SLC,MLC,TLC 等概念特性，不太靠谱。新标准则应该找到一个更好的平衡点和抽象层次，既可以很容易的批量适配闪存，又可以绕过传统 SSD 的低效能。

NVMe 协议标准就给出了一个不同视角的解决方案，定义了不一样的数据访问和组织方式。此方案中多半特性是可选的，原有软件还是可以按原来的逻辑运行。指令与流（Directives and Streams），NVM 集合(NVM Sets)，可预测延迟模式(Predictable Latency Mode)，各种对齐方式和操作粒度等，都在最近的几版 NVMe 标准中被加入，这样也给软件和 SSD 提升了工作契合度。

受硬盘市场的影响，SMR(Shingled Magnetic Recording，叠瓦式磁记录)技术方案最近发展势头良好，SMR 将数据磁道作部分重叠（像屋顶上的瓦片一样），从而提升存储密度。缺点也很明显，因为有重叠部分，磁道数据不能像以前那样任意修改，所以 SMR 硬盘会把多个磁道分成一组，在组内只能顺序写。这种方式对随机写的影响巨大，这也导致人们对 SMR 硬盘评价褒贬不一。不过在服务器存储市场，SMR 硬盘还是有存活空间的：它需要操作系统，文件系统，甚至应用软件主动管理分区存储。

SMR 硬盘上使用的分区存储模型，对闪存也是适合的，NVMe ZNS 便是由此发展而来。SSD 中 Page 和 Block 的概念，与 SMR 硬盘中的 SMR 分区概念是相似的

2. ZNS 概念与软件模型

2.1 ZNS SSD相关概念

ZNS SSD 的相关概念可以先从 ZNS 存储说起。ZNS SSD 是 Zoned Storage 设备的一种实现 Zoned Storage 设备是将 address space 分成多个 zones，不同于标准块设备的读写，Zoned Storage 设备没有读限制但有写限制：必须顺序写，每个 zone 有一个 write pointer 去记录下一次要写的位置，数据不能覆写，要写之前，必须先利用特别的命令进行擦除，这个限制几乎就是为 NAND 量身定制一样。

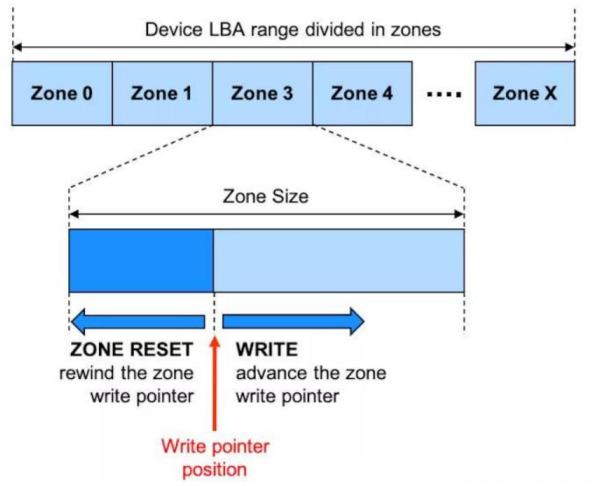

图 2-1 Zoned 存储示意图

传统的硬盘分区，是在文件系统层面。由于 FTL 闪存映射表这个中间层的存在，逻辑分区和实际闪存地址并不是固定的对应关系，起不到在数据分类存放的作用。不同的数据最终会混杂地存储在一起。

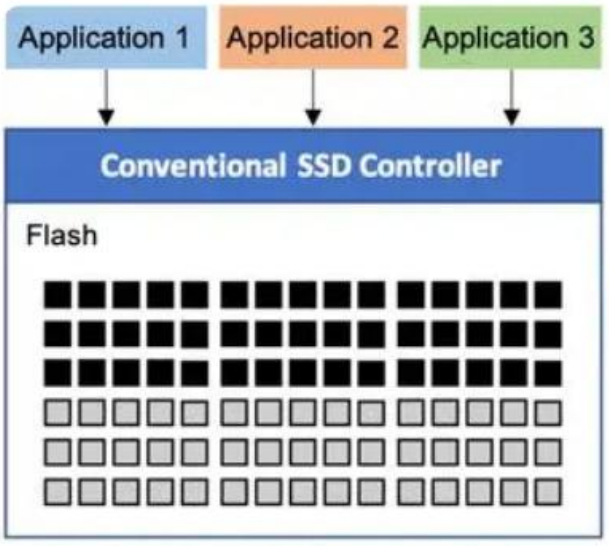

图 2-2 传统 SSD 设备地址映射示意图

而支持 ZNS 分区(Zone)命名空间的固态硬盘可以根据数据类型的不同选择单独的存放位置，从而降低因垃圾回收带来的写入放大（减少对闪存磨损）。此外，ZNS 的一个 Zone 区域可以定义为一个或多个 Block 闪存块的大小（一个 Block 通常为 16MB），在 Zone 区域内只支持顺序写入（但支持随机读取），这样就可以达到降低写放大的目的。

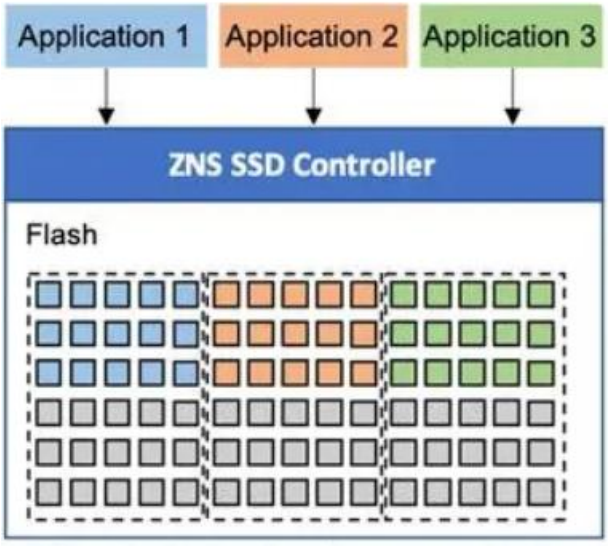

图 2-3 ZNS SSD 设备地址映射示意图

ZNS 在一定程度上是受到了 SMR 机械硬

盘的启发，闪存和 SMR 叠瓦磁记录机械盘有一定的相同之处：前者的最小擦除和读取/写入单位大小不同，后者由于原理的限制不支持对单一磁道直接覆盖写入（会破坏到相邻磁道的内容）。在引入分区命名空间之后，可以优化固态硬盘的性能以及写放大表现。

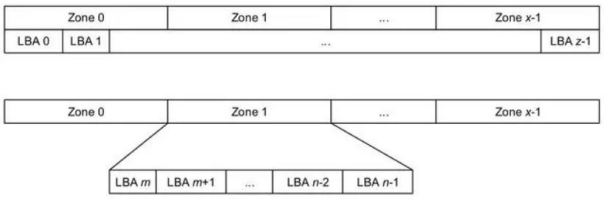

图 2-4 ZNS SSD 里的 zone 示意图

ZNS 分区命名空间还能减少对 DRAM 缓存容量的需求。传统上的 4K 扇区映射需要按照 1GB：1MB 的比例去配备 DRAM 缓存，才能确保良好的存取性能。而在采用 ZNS 分区之后，每个映射分区的容量在十几兆乃至几百兆，而不是过去的 4KB。对于 1TB 以下的家用固态硬盘来说，1GB 的缓存可能无所谓，但对于几十 TB 容量的大容量企业级固态硬盘，ZNS 对于 DRAM 缓存容量需求的降低是非常有利的。

2.2 ZNS SSD软件模型

传统块存储设备的驱动程序需要做一些修改才能支持分区存储设备。首先是要适应在一个 Zone 内只能顺序写的限制，主机软件还需要管理数据的存放位置，对每一个 Zone 要进行状态跟踪，这似乎听起来比预想的要复杂一些。SMR 硬盘是这样，ZNS 也是类似，技术上 SCSI 和 ATA 还分别扩展了命令集（如 ZBC 和 ZAC）来支持这些特性。

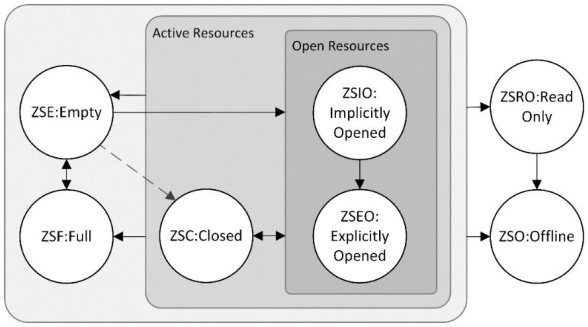

图 2-5 ZNS SSD 中 Zone 的状态转换图

支持 ZNS 功能的 SSD，每个 Zone 有 7 种可能的状态。一个区即使没有存满也有可能进入 Full 态，就像刻录光盘这种场景，没有多的数据再需存入。

当 SSD 中的闪存芯片出错时，有可能使当前 Zone 进入只读（Read Only）态和离线（Offline）态。ZNS 虽然能有效降低写放大，但也需要磨损均衡算法。一般来说，只读（Read Only）态和离线（Offline）态只有整个盘的生命末期才会出现，因此很多软件对这两种状态并不需要做太多的动作，一旦出现，只需将整个盘作废即可。

剩下还有三种状态：implicitly opened(隐式打开)，explicitly opened(显式打开)，Closed(关闭)。处于上述三种状态的 Zone 称为 Active Zone。硬盘在某个工作时间内，Active Zone 的个数可能会有限制。除了状态信息，还需要一些额外的跟踪信息，因此 Active Zone 的个数限制会加剧。每一个 Active Zone，都需要维护一个写指针(Write Pointer)，这个写指针说明了 Zone 被写到哪个地方，下一个写入点在哪。对于 Full

和 Empty 状态的 Zone，写指针是不需要的。

写入之前，Zone 需要被打开。直接发写命令会隐式打开当前 Zone，通过 Zone 管理命令可以显式地打开一个 Zone。这两种打开是有区别的，对于隐式打开的区，SSD 控制器可以随时关闭此区，而对于显式打开的 Zone，主机需要显式地关闭。当打开的 Zone 个数到达最大限制时，如果之前打开方式都是显式的，则打开新 Zone 会失败。有部分 Zone 是隐式打开的话，则打开新 Zone 时会导致部分隐式打开的需要被关闭。Zone 的打开与关闭状态，有利于硬盘维护内部有限的资源（如 buffer）。在一定程度上，这只是 SMR 硬盘遗留的产物，但闪存芯片也是有相似的限制。目前闪存芯片的页大小普遍在 16KB，但 ZNS SSD 支持的 LBA 大小一般是 4KB（甚至 512B），这会导致闪存芯片的某些 Cell 处于部分编程(Partially Programmed)状态，即使按照页大小对齐写入，这种情况还是会发生，因为 Page 数据到闪存的物理存储单也有一个映射过程。如图 2-6 所示，Page 4 和 Page1 的数据被存入同一个 Worldline。

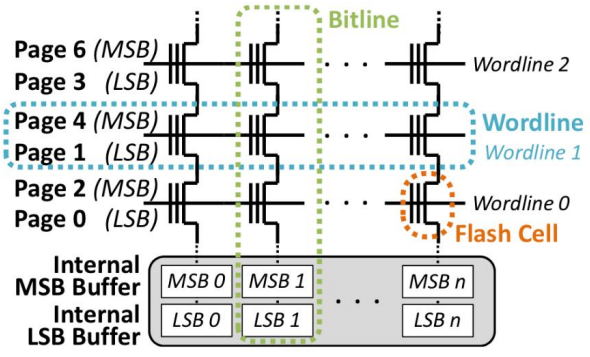

图 2-6 部分编程问题

对于部分编程的存储单元，可能会发生读扰动错误(read disturb error)，从中（甚至从临近的存储单元）读取数据时，这部分存储单元中的电压会发生改变。Open Channel SSD 直接则不允许读部分编程的 Page，而分区存储模型则不想对此做限制。ZNS SSD 一般会将最近的写入数据缓存起来，读命令直接从缓存中返回数据。这种缓存会消耗存储资源，所以会限制 Active Zone 的个数。当某个 Zone 要关闭时，如果存在部分编程的存储单元，硬盘一般有两个选择：一是使用填充数据完成编程操作，填上这个洞，还要期待主机之后不会用到这个 Zoner 的所有空间（这有点不靠谱）；另外一种选择是，即使关闭，也一直缓存着最近写入的数据。支持的 Active Zone 越多就会消耗越多的内存资源，但相比传统 SSD，这点内存资源也不算多，所以在实际使用时，这种方案似乎更合理。一个同时支持分区存储和块存储的 SSD，很可能有那个资源来同时保持所有 Zone 都是 Active Zone。

3. ZNS SSD 与日志结构数据管理方式

基于日志的数据管理系统将存储用作仅附加介质，将随机写入转换为顺序写入，当日志持久保存在硬盘上时，这会带来显着的好处。尽管固态驱动器（SSD）提供改进的随机写入功能，但由于局部性和空间效率，顺序写入仍然具有优势。然而，基于闪存的 S

SD 的固有属性在与随机写入块接口一起使用时会带来重大缺点，导致写入放大、不均匀磨损、日志堆叠和垃圾收集开销。

为了解决这些缺点，Stavrinos 等人指出了 ZNS SSD 的可能性及其替代基于块的 SSD 的潜力。因为 ZNS SSD 可以通过具有顺序写入语义且必须显式重置的区域向主机提供增加的容量、减少的写入放大并开放数据放置和垃圾收集。

在这个基础上，Bjørling 等人则展示了 ZNS 在基于日志的文件系统和 RocksDB 上的优势。他们认为，分区命名空间 (ZNS) 接口代表了主机软件和基于闪存的 SSD 之间的新功能划分，并且介绍了 ZNS 接口并解释了它如何影响 SSD 硬件/固件和主机软件。通过公开闪存擦除块边界和写入顺序规则，ZNS 接口要求主机软件解决容量过度配置，过高的垃圾收集开销等问题，，同时继续管理 SSD 内的介质可靠性。ZNS SSD 通过将擦除块内管理数据组织的责任从 FTL 转移到主机软件，消除了上述问题。同时，它们展示了启用对 ZNS SSD 的支持所需的工作，并展示了与在具有相同物理硬件的块接口 SSD 上运行相比，RocksDB 的修改版本如何利用 ZNS SSD 实现更高的吞吐量和更低的尾部延迟。实验表明，RocksDB 的 99.9% 随机读取延迟在 ZNS SSD 上至少低 2-4 倍，写入吞吐量高 2 倍。

Devashish R. Purandare 则在 Stavrinos 的基础上，提出了跨基于日志的系统实现这一目标所需的具体设计决策；同时，还探索了这些系统内部操作所需的进一步变化，如压实和检查点，以充分利用 ZNS 的潜力。在本文中，他还讨论了常用的基于日志的数据管理系统（包括文件系统、LSM 树、数据库和事件/共享日志）如何发展以从仅附加分区存储中获益。在此基础上，作者总结了数据管理系统的一些有益的设计做法，例如 Group Append。Group Append 将允许应用程序将子块追加到尾部，控制器会将它们分组到一个块中。由于带有这些追加的区域专用于特定日志，因此基本 LBA 或区域编号将足以进行查找。

Kyuhwa Han 等人则提出，由于 ZNS 的顺序只写区域方案，需要日志结构文件系统来访问 ZNS 固态驱动器。虽然在当前的 ZNS 接口下可以简化 SSD，但其对应的 LFS 必须承担段压缩开销。为了解决这个问题，他提出了一个新的 LFSaware ZNS 接口，称为 ZNS+ 及其实现，其中主机可以将数据复制操作卸载到 SSD 以加速段压缩。 ZNS+ 还允许每个区域被稀疏顺序写入请求覆盖，这使 LFS 能够使用基于线程日志记录的块回收而不是段压缩。Kyuhwa Han 等人还为 ZNS+ 感知 LFS 提出了两种文件系统技术。copyback-aware 块分配考虑了 SSD 内不同复制路径的不同复制成本。混合段回收根据成本在段压缩和线程日志记录之间选择合适

的块回收策略。

4. 总结

当创新的硬件和硬件接口出现时，软件系统应该改变以利用这些创新。使用 ZNS SSD 的计算存储、分类存储和分区存储就是这样的创新。分区命名空间（ZNS）SSD 通过具有顺序写入语义且必须显式重置的区域而特别适合对于日志结构的数据管理系统。因此，在这个方面的研究仍将如火如荼。

参 考 文 献

[1] Theano Stavrinos, Daniel S. Berger, Ethan Katz-Bassett, and Wyatt Lloyd. 2021. Don’t be a blockhead: zoned namespaces make work on conventional SSDs obsolete. [C]Proceedings of the Workshop on Hot Topics inOperating Systems. Association for Computing Machinery, 144–151

[2] Matias Bjørling, Abutalib Aghayev, Hans Holmberg, Aravind Ramesh, Damien Le Moal, Gregory R. Ganger, and George Amvrosiadis. 2021. ZNS: Avoiding the Block Interface Tax for Flash-based SSDs.[C] In 2021 USENIX Annual Technical Conference (USENIX ATC 21). USENIX Association, 689–703.

[3] 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

[4] Matias Bjørling, Abutalib Aghayev, Hans Holmberg, Aravind Ramesh, Damien Le Moal, Gregory R. Ganger, and George Amvrosiadis. ZNS: Avoiding the Block Interface Tax for Flash-based SSDs. [C]In 2021 USENIX Annual Technical Conference (USENIX ATC 21). USENIX Association, 689–703.

[5] Alberto Lerner and Philippe Bonnet. 2021. Not Your Grandpa’s SSD: The Era of Co-Designed Storage Devices.[C] In Proceedings of the 2021 International Conference on Management of Data (Virtual Event, China) (SIGMOD/PODS ’21). 2852–2858.

[6] Theano Stavrinos, Daniel S. Berger, Ethan Katz-Bassett, and Wyatt Lloyd. 2021. Don’t be a blockhead: zoned namespaces make work on conventional SSDs obsolete. [C] In Proceedings of the Workshop on Hot Topics in Operating Systems. Association for Computing Machinery, 144–151.