针对 ZNS 的系统软件优化技术论述

胡海川

华中科技大学武汉光电国家研究中心

摘 要 随着 SSD 技术的进步，SSD 的容量越来越大。与机械硬盘不同，SSD 不支持覆盖写的操作，需要擦除再写入。SSD 上擦除的过程非常耗时，并且闪存颗粒的擦除寿命是有限的。为了使 SSD 的寿命更长且性能更优，设备转接层（FTL）需要对 SSD 进行一些特殊处理，包括地址映射、垃圾回收和磨损均衡。垃圾回收会带来性能下降，地址映射在 SSD 容量庞大时会需要大量高速存储器空间，让 SSD 成本上升。这一些问题随着 ZNS SSD 的出现有了解决的可能。本文分析了 ZNS SSD 的性能和访问特性，罗列了一些基于这一些特性的系统软件改进案例，总结了目前 ZNS SSD 应用后系统软件的现状并展望未来 ZNS 更大范围使用后，还有哪一些可能的改进点。

关键词 操作系统；固态硬盘；分区命名空间

Abstract With the progress of SSD technology, the capacity of SSD is increasing. Unlike hard disks, SSDs do not support overwrite operation, and need to be erased the block and written again. The process of erasing block on SSD is time-consuming, and the erasing life of flash memory particles is limited. In order to prolong the service life and improve the performance of SSDs, the Flash Translation Layer (FTL) needs to perform some special processing on SSDs, including address mapping, garbage collection and wear balancing. Garbage collection will lead to performance degradation. When SSD capacity is large, address mapping will require a large amount of high-speed memory space, which will increase the cost of SSD. With the emergence of ZNS SSD, it is possible to solve these problems. This paper analyzes the performance and access characteristics of ZNS SSD, lists some system software improvement cases based on these characteristics, summarizes the current status of the system software after the application of ZNS SSD, and looks forward to some possible improvement points after the wider use of ZNS SSD in the future.

Key words operating system; Solid State Drive; Zoned Namespaces

1 介绍

随着闪存技术的发展，固态盘的容量越来越大，单 GB 的价格越来越低，这使得更多的设备开始使用大容量 SSD。相比较于机械硬盘，SSD 的写入方式比较特别，SSD 不能像传统的机械硬盘一样直接在目标区域覆盖写入，而必须先将数据擦除再写入，且擦除的最小单位大于写入的最小单位，这样不同于机械硬盘的写入方式给 SSD 的发展带来了一些问题。

在机械硬盘时代，操作系统会将存储设备看成一个地址连续的数组。如果要保持操作系统的抽象形式不变，那么 SSD 就需要一个设备转接层（FTL），因为擦写的速度非常慢，所以覆盖写入操作通常是将原先的页面标记为失效，然后获取一个新的页面写入，再将失效的页面定时回收。这样在操作系统看来逻辑连续的块设备实际上物理放置并不连续，因此需要一个地址转换装置。再加上闪存颗粒擦写次数有限，为了延长 SSD 的寿命，需要平衡各个块之间的擦写次数。这样就导致了 SSD 的管理逻辑非常复杂，传统的 SSD 会将垃圾回收，地址转换和擦写次数平衡等交由设备转接层（FTL）来管理。

随着 SSD 的容量越来越大，对于 SSD 的管理也越来越复杂，为了支持容量更大的 SSD，地址映射表的规模越来越大，存储地址映射表的成本越来

越高。为了让垃圾回收变得高效，需要配置额外的容量来保证出现长期持续写入的性能。有学者做了额外配置容量与性能的相关实验，实验表明，更多的额外容量的配置可以显著的提升连续大量写入的性能[1]。再加上用于存储地址映射表的存储器件价格高昂，额外的容量配置拉高了单 GB 闪存颗粒的成本，这导致了大容量高性能的 SSD 的配置成本非常高。这一点在大规模使用存储的数据中心尤为明显。在这个背景下，分区存储的模型诞生了，分区存储最早是用于大容量叠瓦式磁记录 HDD 的[2]，而在 SSD 内部管理复杂问题日益严重的今天，分区命名空间 SSD（ZNS SSD）给问题的解决带来了可能。

目前分区命名空间 SSD 处于起步的阶段，分区命名空间 SSD 可以带来很多优势，但是充分利用好分区命名空间 SSD 确实是一大挑战。首先分区命名空间 SSD 将设备的部分管理策略的制定交由用户，因此策略的优化对性能起到了重要作用；然后传统的系统软件与模块大多是基于传统的块设备而设计的，所以使用好分区命名空间 SSD 需要对与之相应的系统软件进行一定的改进。本文首先分析了分区 SSD 带来的优势和访问特点；然后介绍了目前有使用针对分区命名空间 SSD 的系统软件改进案例，包括使用分区命名空间 SSD 作为交换区设备、不同的空间管理策略；最后总结这一些改进的一些共同思想。

2 ZNS 的优势和访问特性

分区命名空间 SSD（ZNS SSD）将 SSD 分为多个区域，每一个区域内只能进行顺序写入，不能进行随机写入，但是可以进行随机读取。擦除操作只能对一整个区域进行，这样的设计让庞大的地址映射表不复存在，并且垃圾回收策略也变得更简单且自由。本章节中我们总结了目前文献中对于 ZNS SSD 的访问特性，并简要分析分区命名对 SSD 的影响以及优势。

2.1 垃圾回收开销分析

在传统的块设备 SSD 上，垃圾回收是将一个块上有效的页面提取并聚集，然后找一个新的块写入，最后将这个块擦除。论文中设计了一个实验[1]，实验设定了目标写入带宽，并且以目标写入吞吐量持续的向 SSD 中写入。为了对比的公平起见，块接口的 SSD 在逻辑上也按照 ZNS SSD 一样的分区，即在逻辑层面上写入的地址是一致的。这个实验结果如图 1(a)所示，可以看出，ZNS SSD 可以一直保持写入带宽达到目标吞吐量，但是块接口 SSD 在目标吞吐量较高的时候进行持续写入达不到目标吞吐量，并且随着 SSD 额外的空间越多，可以达到的目标吞吐量也越高。另一个实验在持续写入期间进行 4KiB 的随机读取，比较不同类型 SSD 之间的平均延迟[1]，实验结果如图 1(b)所示，可以看出在任何目标写入吞吐量的情况下，ZNS SSD 的随机读取延迟都要低于块接口 SSD。究其原因是因为长时间持续的写入后，尤其是覆盖写入，会产生很多被标记失效的页。块接口 SSD 定期执行垃圾回收，垃圾回收会影响写入的性能，给块接口 SSD 更多的保留空间可以缓解这个问题，但是 ZNS SSD 有更低的垃圾回收代价，因此吞吐量也更高。总而言之，使用 ZNS SSD 可以不需要保留分区，提升了设备的可用容量的同时获得了相比于块设备 SSD 更高的性能。

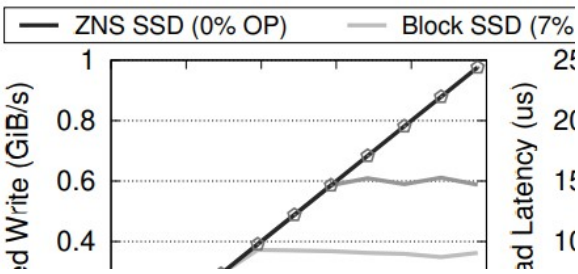

图 1 ZNS SSD 和块接口 SSD 的带宽比较结果

ZNS SSD 相比于块设备 SSD 牺牲了一定的灵活性，将更多的 SSD 的控制权交由了主机端，从而使得 SSD 的垃圾回收的开销减少，提高了吞吐量和读写延迟，那么影响 ZNS SSD 垃圾回收开销的因素有哪一些？论文[3]针对这个问题进行了分析，他设计了一个实验，实验测量了在不同分区占用比例下的垃圾回收效率，实验结果如图 2 所示，实验得出了随着分区内空间占用的越多，垃圾回收的效率有明显的下降的结论，这提示着设计针对 ZNS 的系统软件要适当的控制每一个分区内空间的使用比例。

2.2 对数据库的性能提升

有许多论文都对 ZNS SSD 用于数据库的性能进行了讨论[1,5,6]。其中这篇论文[1]讨论了在 RocksDB 上添加 ZNS SSD 的支持后的性能改变。

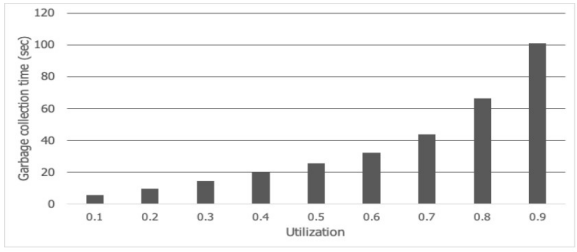

图 2 ZNS SSD 垃圾回收的时间随分区使用率的变化

实验首先测试了随机填充写和覆盖写情况下 ZNS SSD 和传统块设备 SSD 的性能区别，结果图如 3 所示，可以看到在填充写阶段，因为没有大量的失效块和垃圾回收操作，所以使用 ZNS SSD 和使用传统的 SSD 的性能相似，ZNS 有小幅度领先，但是覆盖写的测试中，因为块设备 SSD 需要执行大量的垃圾回收操作，所以拖累了整体性能。通常而言，数据库中的更新操作非常常见，因此更高的覆盖写性能可以显著提升数据库的整体性能。

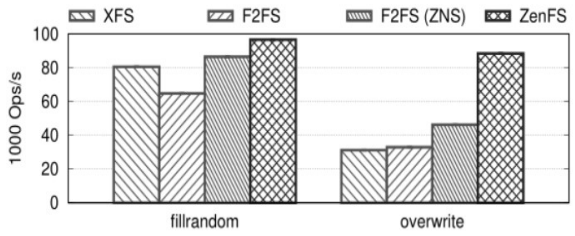

图 3 RocksDB 随机填充写性能和覆盖写性能对比

论文同时也对 RocksDB 中混合读写的性能进行了实验[1]，在数据库管理系统中，对数据的混合读写是非常常见的，传统的块设备 SSD 因为垃圾回收的缘故，在重负载的写入场景下会对读产生一定的影响，导致读的延迟升高。图 4 展示了实验的结果，图 4(a)是随机读取基准测试的尾延迟结果，可以看出单纯读取下 ZNS SSD 的尾延迟相比块设备 SSD 有一定的优势；图 4(b)是在限制 RocksDB 的写入速率为 20MiB 的情况下，同时进行写入和随机读取的延迟结果，可以看到此时两者的延迟差距被拉开；图 4(c)是完全放开 RocksDB 的写入速率，进行尾延迟测试的结果，可以看出此时 ZNS SSD 的领先进一步变大。综合以上的结果可以看出，虽然 ZNS SSD 限制了随机写的操作，但是在实际数据库应用中并没有出现一些场景下性能倒退的情况，而是在写入和读取性能上都相比于传统的块设备 SSD 有提升。提升的来源除了垃圾回收的开销减少之外还有 ZNS 允许根据应用特点将数据更合理的放置。

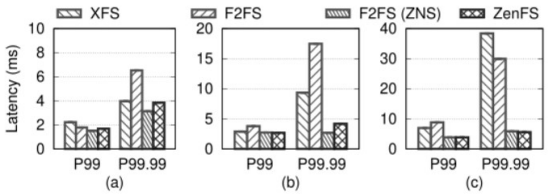

图 4 RocksDB 读取尾延迟对比

2.3 更好的性能隔离

随着云计算的快速发展，租用计算设备慢慢的成为一种重要的途径。对于 CPU 和内存来说，可预测性比较强，易于多租户场景下的划分使用。但是传统的块设备 SSD 因为性能的不可预测性以及读写会互相干扰，使得多租户场景下无法让所有租户的性能稳定，而 ZNS SSD 天然分区的特性给多租户的使用场景带来了转机[4]。利用 ZNS SSD 的优势，不仅可以做到充分发挥 SSD 的性能，还有利于多租户场景下的公平。

2.4 小结

通过以上实验的汇总分析，我们看到了 ZNS SSD 的应用优势，概括起来有以下几点：（1）更灵活的空间管理、放置策略和垃圾回收策略；（2）消除了保留空间（3）更高效的垃圾回收以及更好的持续读写性能（4）更好的性能隔离。ZNS SSD 给上层应用和系统更大的自由度，应用可以根据具体场景给出更优秀的策略，但是目前很多系统的优化手段和技术都是针对传统的块设备 SSD 的，因此要进一步充分发挥 ZNS SSD 的潜力需要进行一定程度的适配。

3 针对 ZNS 的系统软件优化技术

本章节中，将具体介绍几个典型的针对 ZNS SSD 的系统软件优化的例子，这一些例子都一定程度的使用了 ZNS SSD 的特性，为整体系统的性能带来了提升。

3.1 将ZNS SSD作为交换区

存储设备的一大用处是作为交换区设备，与机械硬盘时代不同，SSD 的读写速度非常快，交换区不再是内存不够使用的时候的被迫选择，主动使用一部分交换区可以减少内存的使用，从而提高内存的分配效率。但是目前经典的设计还是以交换区是内初不够时的备用选择为思想设计的，尽管从设备

上更换了读写速度更快的 SSD，但是却没有关注到交换区的交换性能。

对于改善交换区的性能，有学者做过一些研究[7]，这一些研究适用于专用的硬件。[4]尝试使用 ZNS SSD 替换传统的块设备 SSD 作为交换区的硬件，并且充分利用 ZNS SSD 的特性改进交换性能。

在传统的 SSD 上，交换区的性能很大程度上受限于垃圾回收，随着交换的数据量的增加，交换的带宽会降低，这极大的影响了交换的效率，并且这是在交换区仅仅使用 20%空间的情况下。并且交换区的垃圾回收也影响着性能隔离，因为交换区并不是内存不够的无奈之举，所以交换性能不稳定可能直接影响部分租户整个的系统性能，这会给云计算的资源分配等带来一定的影响。

基于以上的观察，以及 ZNS SSD 可能可以使用的优良特性，[4]针对 ZNS SSD 设计了一套垃圾回收方法——ZNGC，ZNGC 的示意图如图 5 所示。ZNGC 可以通过调用显示触发，也会在 ZNS SSD 中空分区较少的时候触发。整个过程具体包括一下四个步骤：Gather：将 ZONE 中缓存的页面的交换槽移除，且交换槽无效；并将还在使用的交换槽收集起来，为读取做准备；Read：将收集的交换槽读取放入缓存；Write：根据剩余的 ZONE 的情况分配并写入目标区域；Activate：之前收集的交换槽暂时还允许进程访问，待到映射表和页表更新完成之后，再将之前的交换槽真正的清理。实验表明，使用这样的方法作为交换区，可以做到随着设备空间占用上升，交换的性能损失较少，并且有很好的性能隔离。

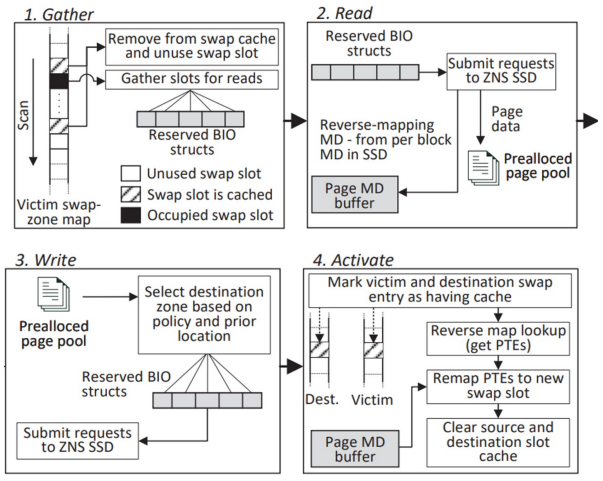

图 5 ZNGC 过程示意图

3.2 对日志操作的优化

日志记录在存储系统上是非常常见的，这篇论文[5]通过分析了日志文件系统、LSM 树（日志结构合并树）、数据库和事件日志和共享日志四个场景，从理论的角度分析挖掘 ZNS SSD 在这一些场景下的应用潜力，得出这一些场景比较适用于 ZNS SSD 的结论。

基于日志的文件系统比较适合于 ZNS SSD，因为他们的特性与平台的限制密切相关。日志文件系统的索引是直接索引的日志，日志再索引保存数据的块，通过使用内存保存索引的手段，日志文件系统在 SSD 上的逻辑上的覆盖写全部转换为事实上的追加日志操作。日志文件系统可以将不同的分区分别指定为检查点区域和主要存储数据的区域。通常日志文件系统写入日志的时候，日志不会出现覆盖写入，逻辑上的覆盖写入变成了日志，覆盖的数据会开辟新的空间存储，这与 ZNS SSD 的结构不谋而合，真正的物理覆盖写对于日志文件系统来说本就是鸡肋。但是 ZNS SSD 的高效垃圾回收和高持续写入是可以给日志文件系统带来性能提升的，因此日志文件系统在避开了 ZNS SSD 的限制的同时发挥了 ZNS SSD 的特性。

3.3 垃圾回收的加速

从第二章节中可以看出，垃圾回收会极大的影响 SSD 的读写性能，ZNS SSD 相比于传统的块设备 SSD 来说，通常不同程序的文件对应的块都用分区来进行一定的隔离，相比较块设备 SSD 中不同应用程序的文件混合在一起导致某个程序的文件覆盖写后 SSD 内很多块都会出现失效页，ZNS 可以很大程度减少这样的情况，进而减少垃圾回收的频率。但是垃圾回收依旧非常影响整体的系统性能，因此提升 ZNS SSD 的垃圾回收效率可以进一步提升系统性能。对于这一点很多学者都做出了研究成果[3,6]。

一种方法是在目前 ZNS 接口的基础上开发出更多的操作和原语[6]，论文作者将其命名为 ZNS+，他比传统的 ZNS 多出了三个新命令。ZNS+支持 compaction 加速和 compaction 避免，以减少 host 端执行垃圾回收时引起的开销。其中 compaction 加速将其中的数据拷贝操作下放给设备层，并且 ssd 可以利用存在的 copyback 机制来减少 copy 开销。具体过程如图 6 所示。

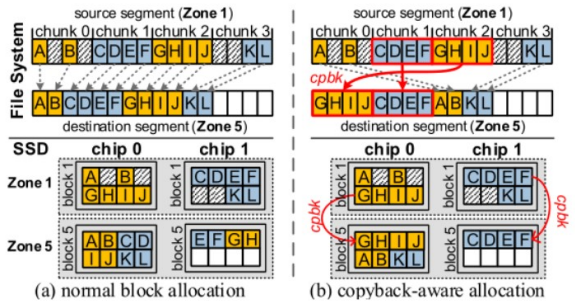

图 6 compaction 加速示意图

另一种解决方法是被称为 LSM_ZGC 的技术[3]，该技术的目标是降低每一个 Zone 的使用率，写入数据的时候优先选择使用率较低的 Zone，因为垃圾回收的代价随着 Zone 内空间的占用率增加呈现指数上升。在垃圾回收的时候，尽可能将数据写入新的 Zone 中而不是追加写入一个已经存在部分数据的 Zone。在垃圾回收的时候，将需要重置的分区内的数据读取出来，按照段为单位对于数据进行冷热识别并将冷热数据写入不同的区域中，冷热数据以使用率为度量标准，当一个段落的使用率超过某一个阈值的时候，认为这个段落的数据是热数据，反之则认为这个是冷数据。图 7 展示了 LSM_ZGC 技术垃圾回收过程的示意图。实验表明，这种方法可以提升大约 1.9 倍的性能。

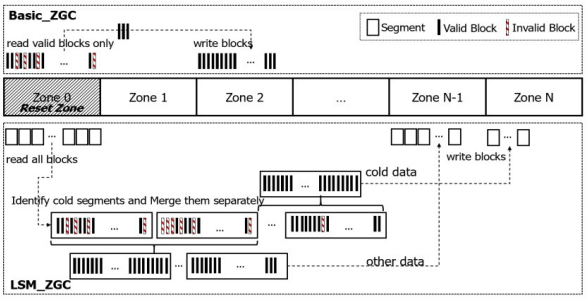

图 7 LSM_ZGC 技术相比于传统垃圾回收技术的过程对比示意图

4 评价与未来展望

ZNS SSD 将大多数控制任务交由主机完成，因此目前大多数的研究成果都是致力于给出一种优秀的 ZNS SSD 控制方案，包括提升垃圾回收的性能、给 ZNS SSD 设计一个优秀的数据放置策略以及与现有的系统进行适配。考虑充分利用 ZNS SSD 的特性为应用服务的研究还比较少。

未来，ZNS SSD 的分区特性还有很多值得挖掘的地方。将传统的块设备 SSD 替换成 ZNS SSD 之后，整个系统软件部分都要相应的做出一些优化，不局限于优化垃圾回收过程等 ZNS SSD 本身的性能。举例来说，将 ZNS 替换掉块设备的 SSD 之后，那么原先用块设备 SSD 与持久化内存混合的文件系统也可以有相应的适配技术。如果使用 ZNS SSD 组磁盘阵列，那么是否可以有一些新的方式或者新的技术。这一些都有待挖掘。

参 考 文 献

[1] Bjørling M, Aghayev A, Holmberg H, et al. {ZNS}: Avoiding the Block Interface Tax for Flash-based {SSDs}[C]//2021 USENIX Annual Technical Conference (USENIX ATC 21). 2021: 689-703.

[2] Le Moal D, Bandic Z, Guyot C. Shingled file system host-side management of shingled magnetic recording disks[C]//2012 IEEE International Conference on Consumer Electronics (ICCE). IEEE, 2012: 425-426.

[3] 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.

[4] Bergman S, Cassel N, Bjørling M, et al. {ZNSwap}:{un-Block} your Swap[C]//2022 USENIX Annual Technical Conference (USENIX ATC 22). 2022: 1-18.

[5] Purandare D R, Wilcox P, Litz H, et al. Append is Near: Log-based Data Management on ZNS SSDs[C]//12th Annual Conference on Innovative Data Systems Research (CIDR'22). 2022.

[6] Han K, Gwak H, Shin D, et al. ZNS+: Advanced zoned namespace interface for supporting in-storage zone compaction[C]//15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21). 2021: 147-162.

[7] Lee G, Jin W, Song W, et al. A case for hardware-based demand paging[C]//2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020: 1103-1116.