Open Zone 갯수 제한과 Zone-finish 명령어 사용에 따른 LSM-ZNS 스토리지의 성능 분석

이상윤[1], 박재완[1], 변성진[1], 김영재[1]
[1]서강대학교 컴퓨터공학과
{tkddbs9801, brian7567, sjbyeon, youkim}@sogang.ac.kr

Performance Analysis of LSM-ZNS Storage with Open Zone Limits and Zone-finish Commands

Sangyun Lee[1], Jaewan Park[1], Sungjin Byeon[1], Youngjae Kim[1]
[1]Sogang University

요 약

ZNS SSD에서 RocksDB와 ZenFS를 활용하여 open zone limit과 zone-finish 명령어가 전체 시스템 성능과 ZNS SSD의 수명에 미치는 영향을 분석한다. ZNS SSD는 제한된 DRAM으로 인해 open 상태의 zone 수가 제한되며, 이로 인해 GC 오버헤드와 성능 저하가 발생한다. ZenFS는 생명주기가 유사한 데이터를 같은 zone에 배치해 GC 효율성을 높이지만, 제한된 open zone limit으로 인해 일부 파일은 서로 다른 생명주기의 파일과 함께 배치되어 추가적인 GC가 요구된다. 이를 해결하기 위해 zone-finish 명령어를 사용해 open zone을 조절하면 GC 오버헤드는 줄어들지만, 지연 시간이 증가하거나 NAND 수명이 단축될 위험이 있다. 본 논문은 다양한 open zone limit 조건과 zone-finish의 적용 여부를 비교하여 RocksDB + ZenFS에서 적절한 open zone 수와 zone 관리 방안을 제시한다.

1. 서 론

ZNS(Zoned Namespace) SSD [1]는 논리 블록 주소를 고정된 크기의 zone으로 나누어 관리함으로써 데이터 관리의 효율성을 높이고 고성능 스토리지 시스템을 구현하는 데 유망한 솔루션으로 각광 받고있다. ZNS SSD에서는 쓰기 동작이 각 zone의 WP(Write Pointer)에서만 순차적으로 가능하다. 각 zone은 한개의 WP를 가지고 있으며, open 상태인 zone에서만 쓰기가 허용된다. 데이터가 쓰여지고 WP가 zone의 끝에 도달하면, 해당 zone은 full 상태로 전환된다. Full 상태의 zone을 다시 사용하기 위해서는 zone-reset 명령어를 실행하여 zone을 empty 상태로 되돌려야 한다.

그러나 ZNS SSD에서는 동시에 open 상태로 유지할 수 있는 zone의 수, 즉 open zone limit [1, 2, 3]이 존재한다. 제조사들은 이 수를 일반적으로 14개에서 256개 사이로 제한하고 있다 [3, 2]. 이는 NAND device 내부의 DRAM 용량이 일반적으로 1GB 이하로 제한되어 있기 때문이다 [4, 2]. 이 DRAM은 각 zone의 WP를 관리하는 mapping table과 host가 실행한 쓰기를 buffering하는 데 사용된다.

LSM-tree [5]는 쓰기 집중형 Key-Value Database에서 널리 사용되며, ZNS SSD와 마찬가지로 순차쓰기를 기반으로 한다. RocksDB와 같은 LSM 기반의 Key-Value Database는 ZNS SSD에서 데이터를 효율적으로 관리하기 위해 ZenFS와 같은 파일 시스템을 활용한다. ZenFS는 open zone에 SST 파일을 배치하는 zone 할당과 full 상태의 zone에서 유효한 SST 파일을 다른 zone으로 복사하고 해당 zone을 reset는 zone cleaning 기능을 수행한다. 그러나 ZenFS는 open-zone limit으로 인하여 생명 주기가 다른 파일들을 동일한 zone에 배치할 수밖에 없는 상황에 직면하며, 이는 valid data copy로 인한 GC(Garbage Collection) 오버헤드를 증가시킨다.

이 문제를 해결하기 위한 한 가지 접근법은 open zone의 수를 늘리는 것이다. 그러나 이는 ZNS SSD의 제한된 DRAM 용량으로 인해 현실적으로 한계가 있다. 다른 접근법으로는 zone-finish 명령어를 사용하여 zone을 조기에 닫고 새로운 zone을 open하여 데이터의 생명 주기를 더 세밀하게 구분하는 방법이 있다.

본 논문에서는 RocksDB+ ZenFS 환경에서 open zone limit이 시스템 성능과 ZNS SSD에 미치는 영향을 실험을 통해 면밀하게 조사한다. 5.1에서 zone-finish 명령어가 있는 경우와 없는 경우에 대해 open zone limit의 수를 변경 시켜가며 쓰기를 실행한다. 이를 통해, zone-finish command의 영향을 분석하고, open zone limit에 따른 성능 지표들을 측정한다. 측정 결과 zone-finish command가 없는 경우 open zone limit이 20까지 증가함에 따라 유효 데이터 복사량이 약25% 감소 하였다. Zone-finish command가 있는 경우 open zone limit이 30까지 증가함에 따라 유효데이터 복사량이 매우 큰폭으로 감소하였으나, 전구간에서 zone-finish command가 없는 경우에 비해 유효 데이터 복사량이 많고 처리량이 낮아지는 것을 보였다.

2. 배경지식

2.1 LSM based Key-Value Store

LSM-tree [5]는 순차쓰기 기반 자료구조로, 쓰기 집중 키-값 데이터베이스에서 널리 채택된다. LSM-tree는 Memtable과 Disk의 Sorted String Table(SST) 파일로 구성된다. Memtable이 가득 차면 level 0의 SST 파일로 변환되어 디스크에 영구적으로 기록된다. level 0의 SST 파일들이 충분히 많아지면, level 0의 SST 파일은 level 0의 SST 파일과 키 범위가 중복되는 level 1의 SST 파일의 키-값들과 merge-sort한다. 그 결과, 중복되고 삭제된 키-값이

"본 연구는 2024년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학 사업 지원을 받아 수행되었음"(2024-0-00043)

제외된 level 1의 SST 파일이 생성하고, merge-sort에 참여한 SST 파일이 삭제된다, 이것을 compaction이라는 동작이라고 한다. 즉 LSM-tree는 SST 파일에 대해 overwrite를 통해 데이터를 수정하는 것이 아니라, merge-sort를 통해 새로운 SST 파일을 생성함으로써 데이터를 수정하는 append-only I/O pattern을 보인다. 따라서, append-only pattern을 보이는 LSM-tree는, zone의 WP에 오직 순차쓰기만이 가능한 ZNS와 잘 어울린다.

ZenFS : ZenFS는 LSM 기반 Key Value DB인 RocksDB를 ZNS에서 구동하기 위해 특수하게 디자인된 유저 레벨 파일 시스템이다 [3]. ZenFS는 RocksDB를 위해 두가지 기능을 수행한다. 첫번째, SST 파일을 'open' 상태의 zone에 배치하는 Zone Allocation과, 'full' 상태의 zone에 대해 빈 공간을 회수하고 empty 상태로 바꾸는 Zone Cleaning을 수행한다. Zone Allocation은 CAZA [6]와 같은 알고리즘을 통해 삭제 시기가 같은 SST 파일을 같은 zone에 배치한다. Zone Cleaning은 'full' zone들 중, 가장 valid SST 파일이 적은 zone을 선정하여, 그 valid SST파일을 다른 zone으로 안전하게 copy한다. 그 다음, zone-reset을 실행함으로서 그 zone을 다시 'empty' 상태로 바꾼다.

3. ZNS 디바이스 자원의 한계점

3.1 Open Zone 갯수 제한

ZNS SSD의 제약된 DRAM resource로 인해 동시에 쓰기가 가능한 zone의 갯수를 제한하고, 이를 open zone limit으로 정의하고 있다. 그런데, ZNS SSD를 사용하는 소프트웨어는 open zone limit 으로 인해 생명주기가 다른 파일을 같은 zone에 배치하는 문제점이 있다. ZNS SSD에서 생명주기가 다른 파일을 같은 zone에 배치하는 것은 GC 시 valid data copy의 증가를 유발한다 [6]. 구체적으로, ZenFS는 새로운 파일을 배치할 때, 그 파일과 생명주기가 같은 파일들이 배치되어 있는 zone을 선택한다. 그런데 생명주기가 같은 파일들이 배치되어 있는 zone이 모두 full zone이라면, 그 zone에 배치할 수 없다. 또한, 쓰기가 가능한 open zone에 배치되어 있는 파일들이, 배치할 새로운 파일과 모두 생명주기가 다른 경우, ZenFS는 생명주기가 다르지만 쓰기가 가능한 open zone에 새로운 파일을 배치할 수 밖에 없고, 이는 valid data copy를 유발한다.

3.2 Zone-finish 명령어 사용

ZNS SSD를 사용하는 소프트웨어는 배치할 파일과 배치되어 있는 파일들의 생명주기가 다른 상황을 방지하기 위해, zone-finish command를 사용할 수 있다. zone-finish command는 open zone 의 DRAM resource를 반환하고 그 zone의 WP는 zone의 끝으로 변경되며 full zone이 된다. 이를 통해 open zone resource를 확보한 후, ZNS SSD를 사용하는 소프트웨어는 새 empty zone을 열어 생명주기가 다른 파일들을 분리할 수 있다. 예를 들어, 새로운 파일을 배치할 때 생명주기가 같은 zone이 없다면, 그 파일과 생명주기가 다른 zone에 zone-finish command를 사용한다. open zone resource를 확보하였으므로 새로 empty zone을 열어 생명주기가 다른 파일을 서로 분리할 수 있다.

그러나, open zone resource 확보를 위해 zone-finish command를 사용하는 것은 많은 문제를 야기한다. WP가 zone의 시작점에 가까울수록, 이를 zone의 끝까지 이동시키는 zone-finish command의 지연시간은 더 크다. WP가 zone의 시작점에 가까울 때 zone-finish command를 사용한다면 이는 높은 지연시간을 나타낼 것이다. 또한, full zone은 zone-reset command를 통해 empty zone으로 만들어지는 단계를 겪게 된다. 이 단계는 해당 zone 내부의 유효한 데이터를 복사하는 GC과정을 포함한다. GC는 유효한 데이터를 다시 복사해야하므로 NAND flash의 수명을 악화시킨다. zone-finish command는 지연시간을 늘리거나 NAND flash의 수명을 악화시킬 수 있으므로 조심스럽게 사용할 필요가 있다.

4. 실험 방안

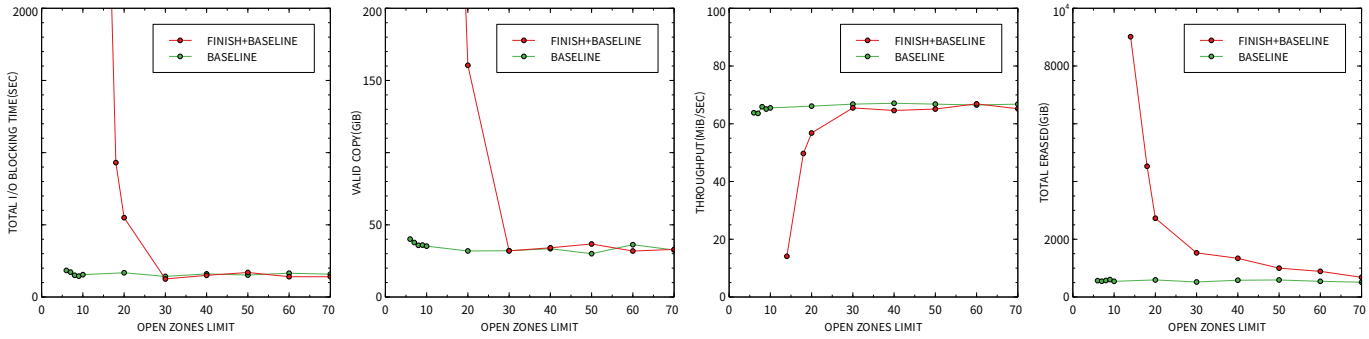
이 논문에서, 현재의 RocksDB+ZenFS 환경에서 적절한 open zone의 개수는 몇이며, 과연 어느 정도까지 SST 파일의 생명주기를 구분할 수 있는지를 측정하고자 한다. 또한, zone-finish command를 사용하며, 이것이 전체적인 성능과 ZNS SSD에 어떤 영향을 미치는지를 관찰하고자 한다. 이를 위해 크게 두가지 기준을 바꿔가며 실험한다.

- 첫째는 open zone limit이다. open zone limit을 늘리는 것은 데이터마다 생명주기를 구분하기 위한 가장 직관적인 방법이지만, 이 방법은 NAND flash device의 제한된 DRAM 크기로 인해 제약이 존재한다. 그러나, 적은 DRAM 사용을 통한 비용 절감은 ZNS의 큰 장점 중 하나이고, DRAM의 크기를 늘려 open zone limit을 늘리는 방법은 ZNS의 비용 절감이라는 장점을 상쇄한다.
- 둘째는 zone-finish command이다. 데이터를 세밀하게 zone마다 생명주기 별로 구분한다면, 해당 zone의 데이터들은 비슷한 시기에 무효화 될 가능성이 높다. 이렇게 무효화된 데이터들이 많은 zone은 유효한 데이터의 양이 적으므로 GC의 오버헤드는 줄어든다. 또한, zone-finish-command는 zone이 전부 채워질 경우 지연시간은 비워져 있는 경우보다 300배 정도 빠르며 [7], NAND flash의 수명또한 아낄 수 있다. 즉, zone-finish command가 zone이 적절히 채워져 있고, zone-reset command가 무효화된 데이터가 많은 zone에 적절히 사용된다면, 이는 open zone limit를 해결할 수 있는 방법이 될 수 있다. 그러나, 3.2에서 언급하였듯, zone에 유효한 데이터가 많은 경우에 zone reset command를 사용하거나, zone이 많이 채워지지 않은 경우에 zone-finish command를 사용한다면, 이는 오히려 NAND flash의 수명을 악화시키거나 지연시간을 늘리는 결과를 가져올 수 있다.

5. 실험 및 평가

5.1 실험환경

실험을 위해, FEMU를 기반으로한 ZNS SSD 에뮬레이터인 ConfZNS [8]를 사용한다. 전체 ZNS SSD의 용량은 80GB이며,



(a) Total I/O Blocking Time (b) Valid Copy (c) Throughput (d) Total Erased

그림 1: fillrandom 워크로드에 대한 open zone limit 및 zone-finish 정책의 효용성 평가.

zone size는 1GB다. read의 지연시간은 65us, write의 지연시간은 450us, erase의 지연시간은 2ms로 각각 설정하였다. CPU는 64 Intel(R) Xeon(R) Gold 5218R CPUs를 사용하였고, 32GiB의 메모리를 사용한다. 또한, 우리는 RocksDB v7.4과 ZenFS v2.1을 변경하여, zone-finish를 사용하는 환경과 그렇지 않은 환경을 비교한다. 워크로드로는 RocksDB의 쓰기 집중 벤치마크 중 하나인 fillrandom을 사용하였고, 72GB의 fill random을 실행한다.

5.2 결과 및 분석

그림1-(a)는 쓰기를 위해 기다린 시간을, 그림1-(b)는 GC 시 유효한 데이터의 복사량을, 그림1-(c)는 성능을, 그림1-(d)는 zone이 지워진 양을 나타낸다.

특히, zone-finish command를 사용하여 open zone limit을 해결하려 할 때, open zone limit가 30 아래인 경우에서 굉장히 높은 valid copy량과 매우 낮은 througput을 보인다. 좋지 않음을 볼 수 있다. 이 경우 open zone limit가 낮기 때문에, 원하는 zone에 배치하기 위해 zone-finish/reset command를 잦게 호출한다. zone-finish/reset command를 자주 호출하므로 GC도 자주 일어나며, zone이 거의 채워지지 않은 경우에도 zone-finish command를 사용하게 되어 지연시간 또한 올라간다. 따라서, 쓰기를 위해 기다린 시간이 올라가며, 유효한 데이터의 복사량 또한 올라간다. 이로인해 성능은 내려가며, zone이 지워진 횟수는 높아지게 된다.

그림1-(d)를 보면, zone-finish command를 사용하는 경우에 open zone limit가 30 이상에서도 zone이 지워진 양이 꾸준히 줄어들며, 점차 zone-finish command를 사용하지 않는 경우와 가까워지고 있음을 볼 수 있다. 이는 zone-finish command는 open zone limit가 30 이상인 경우에도 꾸준히 사용되지만, 그 빈도가 점차 줄어들고 있음을 의미한다. 즉, open zone limit가 30 이상일 때, RocksDB+ ZenFS의 zone 배치 알고리즘은 open zone 내에서도 충분히 생명주기별로 잘 구분하고 있으며, zone-finish command의 경우 지표에 악영향을 주지 않고 적절히 수행되고 있음을 보여준다. 그럼에도 불구하고, 그림1-(a), 그림1-(b), 그림1-(c)를 보면, open zone limit가 30 이상인 모든 경우에서 쓰기를 위해 기다린 시간, 유효한 데이터 복사량, 성능이 일정하다. 이를 통해, 현재의 RocksDB+ ZenFS의 zone 배치 알고리즘은, SST 파일의 생명주기를 약 30종류로 구분하여 zone을 배치하려하고 있다고 볼 수 있다.

그러나 zone-finish command를 사용하지 않는경우, open zone limit가 약 20정도에서도 이미 최적의 성능을 보이고 있음을 볼 수 있다. 이는 현재 실험 환경에서, RocksDB+ ZenFS의 zone 배치 알고리즘은 SST 파일의 생명주기를 약 30종류로 구분하려 하지만, 실제로는 약 20종류의 zone으로도 충분히 데이터 별로 생명주기를 구분할 수 있음을 나타낸다.

6. 결론 및 향후 연구

본 연구는 RocksDB+ ZenFS에서 open zone limit과 zone-finish 명령어가 전체 시스템 성능과 ZNS SSD에 미치는 영향을 분석한다. 또한 현재의 zone 할당 알고리즘이 SST 파일을 생명주기별로 분리하는 정도를 실험한다. 그러나 zone-finish command를 사용하는 경우, 사용하지 않는 경우에 비해 전체적으로 더 높은 유효 데이터 복사량과 더 낮은 처리량을 보였다. 향후, 관찰을 통해 zone-finish command를 실행하는 기준을 변경하여 zone-finish command를 사용하여 성능을 개선하는 방향으로 연구를 확장하고자 한다.

참고 문헌

[1] "NVM Express® Zoned Namespace Command Set Specification Revision 1.1a." https://nvmexpress.org/specifications/, 2022.

[2] K. Han, H. Gwak, D. Shin, and J. Hwang, "Zns+: Advanced zoned namespace interface for supporting in-storage zone compaction," in *15th USENIX Symposium on Operating Systems Design and Implementation*, OSDI '21, pp. 147–162, 2021.

[3] W. Digital, "Zenfs." https://github.com/westerndigitalcorporation/zenfs.

[4] Samsung, "Samsung v-nand ssd 990 pro." https://download.semiconductor.samsung.com/resources/data-sheet/samsung$_nvme_ssd_990_pro_datasheet_rev.2.0.pdf$.

[5] P. O'Neil, E. Cheng, D. Gawlick, and E. O'Neil, "The log-structured merge-tree (lsm-tree)," *Acta Informatica*, vol. 33, pp. 351–385, 1996.

[6] H.-R. Lee, C.-G. Lee, S. Lee, and Y. Kim, "Compaction-aware zone allocation for lsm based key-value store on zns ssds," in *14th ACM Workshop on Hot Topics in Storage and File Systems*, 2022.

[7] K. Doekemeijer, N. Tehrany, B. Chandrasekaran, M. Bjørling, and A. Trivedi, "Performance characterization of nvme flash devices with zoned namespaces (zns)," in *2023 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 118–131, 2023.

[8] I. Song, M. Oh, B. S. J. Kim, S. Yoo, J. Lee, and J. Choi, "Confzns: A novel emulator for exploring design space of zns ssds," in *16th ACM International Conference on Systems and Storage*, SYSTOR '23, 2023.