ZNS를 이용한 파일시스템 성능 개선 연구

Team: OS 을~매나 맛있게요?

부산대학교 전기컴퓨터공학부 정보컴퓨터공학전공
School of Electrical and Computer Engineering, Computer Engineering Major
Pusan National University

2022년 5월 16일

지도교수: 안 성 용 (인)

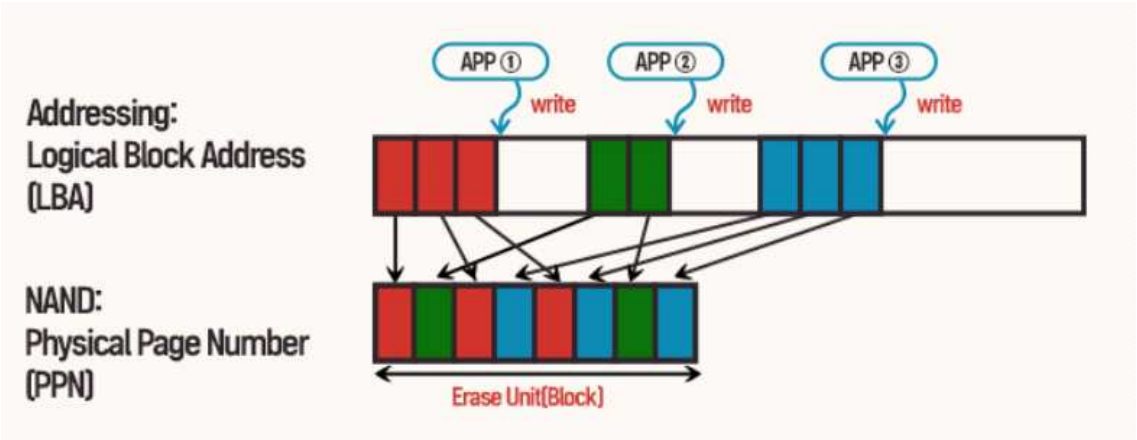
목 차

1. 과제 배경 및 목표

1 과제 배경

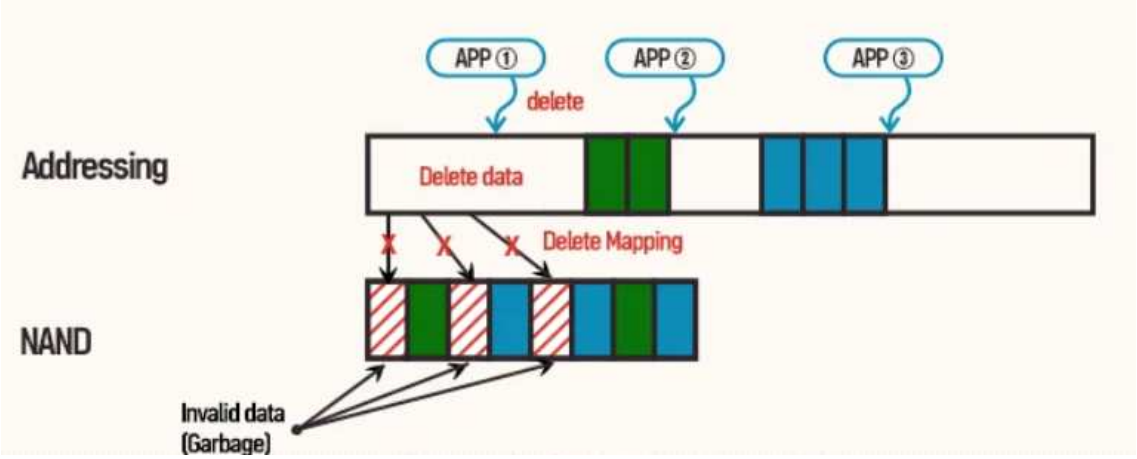
현재 flash-based SSD는 수 십 년된 블록 인터페이스를 유지하며 over-provisioning, page mapping tables을 위한 DRAM(Dynamic random-access memory), garbage collection 오버헤드를 완화시키기 위한 host software의 복잡성 측면에서 상당한 비용이 초래되고 있다.

최근 4차 산업의 핵심기술 중 하나인 데이터 센터(Data center)의 성능과 비용 효율을 위해 세계 유수 기업들은 치열한 기술경쟁을 벌이고 있다. 과거 데이터센터는 하나의 좋은 성능의 서버(Server)에서 하나의 운영체제와 응용프로그램을 구동했다. 반면 차세대 데이터 센터는 하드웨어 풀(Pool)위에 다수의 가상 운영체제와 응용프로그램이 구동되는 형태이다. 즉, 다양한 종류의 프로그램이 하나의 저장장치를 동시에 읽고 쓰는 것을 의미한다. 따라서 데이터 센터에서는 속도뿐 아니라 다중 사용 환경(Multi-tenant)에서도 서로 간섭 없이 일정한 성능을 제공하는 서비스품질(Quality of Services, QoS)을 갖춘 저장 장치가 요구된다.



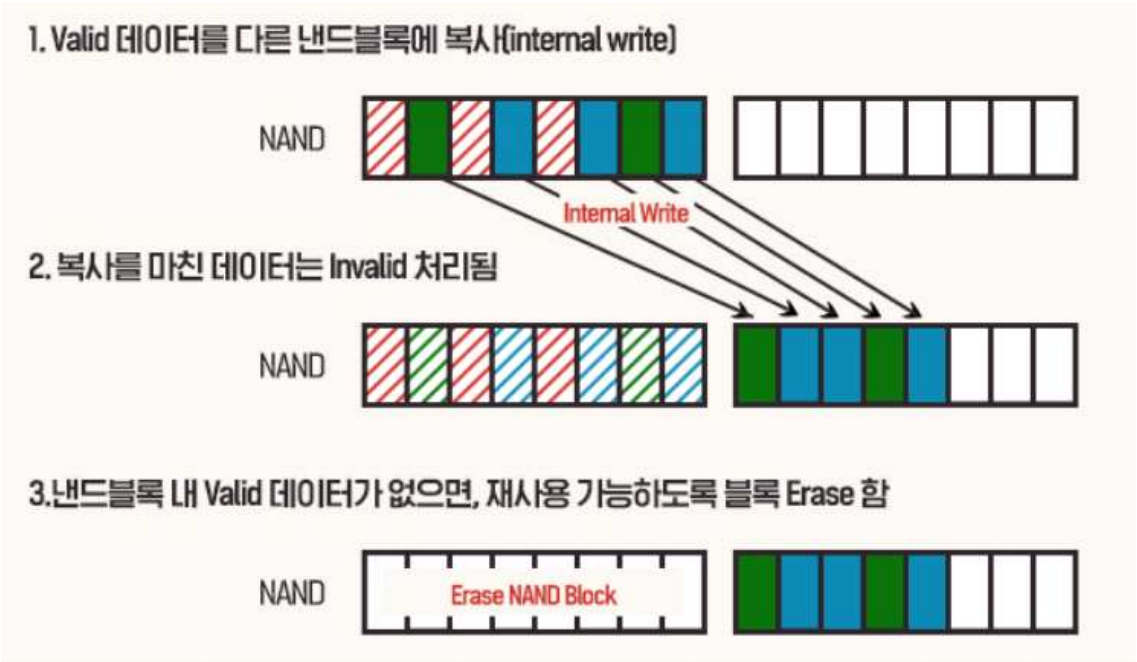
[그림 1. SSD에서의 Write data process]

응용프로그램(APP)이 SSD의 각자의 영역(LBA)에 데이터를 저장(write)하면, SSD는 하나의 낸드블록에 순차적으로 데이터를 저장하고 데이터가 저장된 낸드주소(PPN)과 논리주소(LBA)를 mapping한다.



[그림 2. SSD에서의 Delete data process]

APP에서 SSD의 데이터를 삭제하면, SSD는 논리주소(LBA)와 낸드주소(PPN)의 mapping만 삭제한다. NAND에는 기존 데이터가 가비지(Invalid data)로 남아있게 된다.



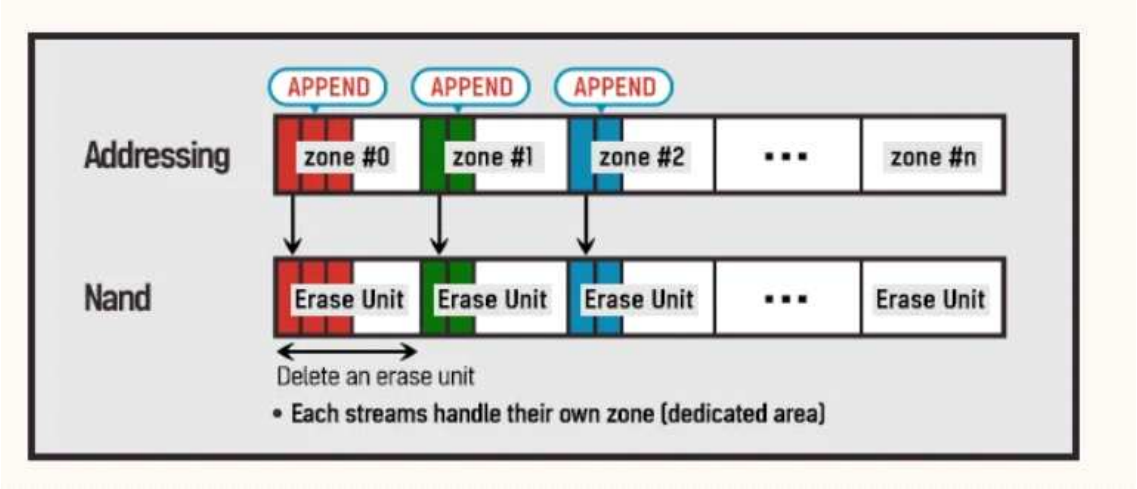
[그림 3. SSD에서의 Garbage Collection process]

Write 와 Delete Data를 반복하면서 NAND Block이 가득 채워지면 Valid data만 모아 다른 NAND Block에 복사를 한다. 복사를 마친 기존의 Valid data는 Invalid data로 처리되고 NAND block 내에 Valid 데이터가 없으면, 재사용 가능하도록 블록 Erase를 한다.

SSD는 Garbage Collection이 발생한다는 단점이 있다. Garbage Collection이 발생하면 진행 중이던 Read/Write process가 잠시 멈추기 때문에 서비스품질(QoS)이 떨어진다는 것이다.

이러한 이유로 글로벌 데이터센터 업체와 반도체 제조사들은 오랫동안 응용프로그램 사이의 성능 간섭과 Garbage Collection 발생 문제의 해결책으로 ZNS 솔루션을 제시했다.

2021년 6월 2일, 삼성전자는 ZNS(Zoned Namespace) 기술을 적용한 차세대 기업 서버용 Solid State Drive(ZNS SSD)를 출시하였다. ZNS는 SSD 전체 저장 공간을 작고 일정한 용량의 구역(Zone)으로 나누고, 용도 및 사용 주기가 같은 데이터를 동일 구역에 저장해 SSD를 효율적으로 활용할 수 있게 만드는 차세대 기술이다.



[그림 4. ZNS에서의 Write Data Process]

기존 SSD에서는 여러 응용프로그램이 논리적으로 각자 원하는 영역에 데이터를 저장하더라도, 실제로는 하나의 NAND 블록에 순차적으로 데이터가 저장된다.
반면, ZNS SSD에서는 그림 4와 같이 여러 응용프로그램이 각자 정해진 Zone에 순차적으로 데이터를 저장한다. 여기서 Zone은 논리적인 공간처럼 Nand 블록에도 나누어져 있다. 하나의 Zone 안에는 비슷한 데이터끼리 모여 있으며, 순차적으로 저장했다가 존 단위로 지우기 때문에 Garbage가 발생하지 않는다. 따라서 SSD에서의 Garbage Collection이 없어도 됨으로써 발생했던 여러 오버헤드를 제거할 수 있다.
그 결과, ZNS SSD의 이점으로는
- SSD를 무려 4배 오래 사용할 수 있다.
- Garbage Collection에 의한 Read/Write 동작의 정지가 발생하지 않으므로 저장속도가 30% 향상되고, QoS의 안정성이 25% 향상된다.
- ZNS의 효율적인 저장구조로 인해 SSD 내 OP(Over Provisioning)영역을 총 SSD 용량의 6%만큼 줄일 수도 있다.
- SSD를 동일 크기의 Zone 단위로 나눠 사용하고 Zone 내에서 append로만 데이터를 저장하므로 기존 SSD에서 DRAM의 많은 부분들을 차지하던 Mapping table이 없어져서 DRAM을 효율적으로 사용할 수 있다.

2 기존 문제점

2.1 연구 tool

FEMU는 USENIX FAST'18에서 처음 소개된 QEMU 기반의 플래시 에뮬레이터로 SSD 연구를 위한 목적으로 개발되었다. 주요 특징으로는 기존 QEMU와 Open-Channel SSD를 위한 파생인 qemu-nvme와 달리 디스크에 데이터를 쓰는 것이 아닌 메모리에 쓴다.

현재 기업에서 쓰는 ZNS 기술은 배포되지 않았기 때문에 ZNS 인터페이스를 FEMU를 통해 연구를 진행할 예정이다.

2.2 실험 환경

CPU	Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz (4-Core)
메모리	4GByte
OS	Ubuntu 18.04, Lonux Kernel 5.10.0

[표 1. 시스템 구성]

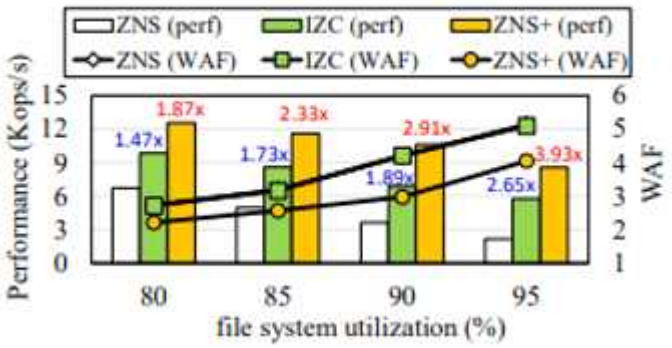
2.3 파일 시스템

현재 개발된 대부분의 파일 시스템은 일반적으로 Zoned Storage 모델에 적용하기 어렵다. 그러나 f2fs, btrfs 및 zfs와 같은 일부 파일 시스템은 순차적인 쓰기 패턴을 나타내며 영역을 지원할 수 있도록 이미 적용 중이다. 모든 쓰기가 순차적인 것은 아니지만(예시로 슈퍼블록 및 일부 메타데이터 쓰기) 이러한 파일 시스템은 엄격한 로그 구조적 쓰기, 부동 슈퍼블록 및 유사 기능으로 확장하여 격차를 해소할 수 있다.

3 과제 목표

3.1 WAF의 크기를 1에 수렴하게 만든다.

WAF는 파일 시스템(데이터 블록 쓰기, 노드 블록 쓰기, 메타데이터 업데이트, 세그먼트 압축, 내부 플러그 포함)이 호출한 총 쓰기 트래픽을 사용자 워크로드에 의해 생성된 쓰기 트래픽으로 나눈 값이다. IZC와 ZNS의 WAF값은 비슷하다. 파일 시스템 활용도가 높아지면 세그먼트 압축이 더 많은 수의 유효한 블록을 복사해야하고 세그먼트 압축이 더 자주 호출되기 때문에 WAF가 증가한다. 스레드 로깅은 노드 및 메타데이터 업데이트 수를 줄이기 때문에 ZNS+는 IZC보다 낮은 WAF 값을 표시한다. ZNS를 통한 IZC 또는 쿤+의 성능 향상은 파일 시스템 활용도가 높을수록 세그먼트 재활용 비용이 더 많이 들기 때문에 파일 시스템 활용도가 높아질 수 록 증가한다.



[그림 5. Performance at various file system utilization]

3.2 hot, cold data를 구별하여 data를 효율적으로 접근하고 처리할 수 있게 개선한다. LFS는 hot/cold에 관계없이 데이터를 쓰고, 세그먼트 정리 시 느리게 데이터 분리를 시도한다. 핫/콜드 데이터가 처음 기록될 때 분류할 수 있다면 개선의 여지가 많다.

3.3 메타 데이터를 수집한다.

3.4 기존의 파일시스템을 ZNS에서 사용이 가능하도록 수정을 한다. 혹은 ZNS환경에서 사용 가능한 filesystem을 개발한다.

[구성원별 역할]

학 번	이 름	구성원별 역할
201924556	임경민	- 파일 시스템 성능 개선 혹은 추가를 위한 구현 - 파일 시스템 성능 분석 - WAF의 값을 1에 근사하게 맞추기 위한 알고리즘 구현
201624431	김상현	- 파일 시스템 성능 개선 혹은 추가를 위한 구현 - 파일 시스템 성능 분석 - WAF의 값을 1에 근사하게 맞추기 위한 알고리즘 구현
201824637	이선후	- 파일 시스템 성능 개선 혹은 추가를 위한 구현 - 파일 시스템 성능 분석 - WAF의 값을 1에 근사하게 맞추기 위한 알고리즘 구현

[References]

[1] Matias Bjørling, Western Digital; Abutalib Aghayev, The Pennsylvania State University; Hans Holmberg, Aravind Ramesh, and Damien Le Moal, Western Digital; Gregory R. Ganger and George Amvrosiadis, Carnegie Mellon University, “ZNS: Avoiding the Block Interface Tax for Flash-based SSDs”

[2] Kyuhwa Han, Sungkyunkwan University and Samsung Electronics; Hyunho Gwak and Dongkun Shin, Sungkyunkwan University; Joo-Young Hwang, Samsung Electronics, “ZNS+: Advanced Zoned Namespace Interface for Supporting In-Storage Zone Compaction”

[3] Changwoo Mina, Kangnyeon Kimb, Hyunjin Choc, Sang-Won Leed, Young Ik Eome abdeSungkyunkwan University, Korea, acSamsung Electronics, Korea, “SFS: Random Write Considered Harmful in Solid State Drives”

[4] Gijun Oh , Junseok Yang and Sungyong Ahn, “Efficient Key-Value Data Placement for ZNS SSD”