FDP SSD의 Persistently Isolated와 Initially Isolated 모드에 대한 RocksDB 워크로드 기반 성능 평가*

노태완[O], 김영재[†]

서강대학교 컴퓨터공학과

{heize0502, youkim}@sogang.ac.kr

Performance Evaluation of Persistently and Initially Isolated Modes in FDP SSDs under RocksDB Workloads

Taewan Noh[O], Youngjae Kim[†]

Department of Computer Science and Engineering, Sogang University

요 약

본 논문은 NVMe Flexible Data Placement (FDP) SSD의 두 가지 데이터 배치 모드 – Persistently Isolated와 Initially Isolated – 를 구현하여 각 모드의 성능 차이를 정량적으로 평가한다. 두 모드는 모두 호스트가 데이터 수명을 기준으로 물리적 블록 배치를 제어할 수 있게 하여 Write Amplification Factor (WAF)와 latency를 개선하지만 내부 Garbage Collection (GC) 동작 시 데이터 이동 정책이 상이하여 성능에 미치는 영향이 다르다. 본 연구에서는 FEMU 기반 FDP 에뮬레이터를 확장하여 두 GC 모드를 구현하고 RocksDB를 대상으로 YCSB 벤치마크를 수행한다. 각 모드의 WAF와 평균 latency를 비교한 결과, 저부하 워크로드에서는 두 모드의 WAF와 평균 latency에 큰 차이가 없는 반면, 고부하 워크로드에서 Persistently Isolated 모드가 Initially Isolated 대비 약 10% 낮은 WAF와 약 6% 짧은 평균 latency를 보였다.

1. 서론

SSD는 HDD 대비 높은 성능을 제공하지만 내부 FTL 구조로 인해 Write Amplification (WA)와 Garbage Collection (GC) 오버헤드 문제가 존재한다. 특히 RocksDB [1]와 같은 LSM-Tree 기반 저장소는 데이터 수명이 다른 Sorted String Table (SST) 파일들이 동일 블록에 혼재되어 기록되며 이는 GC 비효율과 latency 증가로 이어진다.

이러한 문제를 해결하기 위해 NVMe 2.0 표준에서는 Flexible Data Placement (FDP) 기술을 도입하여 호스트가 Placement Identifier (PID)를 지정함으로써 데이터 배치를 직접 제어할 수 있도록 하였다 [3]. FDP는 데이터 수명별 분리를 통해 Write Amplification Factor (WAF)를 줄이고 NAND 내구성과 예측 가능한 성능을 제공할 수 있다 [3].

FDP 사양은 두 가지 주요 격리 정책을 정의한다: Persistently Isolated와 Initially Isolated. 전자는 초기 쓰기 시점에서는 데이터가 분리되지만 GC 과정에서 동일 Reclaim Group 내 다른 데이터와 섞일 수 있다. 반면 후자는 GC 후에도 데이터들의 강한 격리를 유지한다. 두 정책은 WAF, 그리고 평균 latency에 서로 다른 영향을 미칠 것으로 예상되지만 그 성능 차이를 체계적으로 분석한 연구는 찾아보기 어렵다.

본 연구에서는 FEMU [4] 기반 FDP 에뮬레이터 상에 두 격리 모드를 구현하고 RocksDB 환경에서 YCSB 벤치마크를 수행하여 성능을 정량적으로 평가한다. 이를 통해 두 격리 정책이 실제 워크로드에서 보이는 WAF 및 평균 latency 특성의 차이를 분석하였다. 분석 결과 Persistently Isolated 모드는 WAF와 평균 latency에서 Initially Isolated 모드보다 우수한 성능을 보였다.

2. 연구 배경

2.1. Flexible Data Placement SSD

FDP SSD는 NVMe 2.0 표준에서 새롭게 정의된 데이터 배치 메커니즘으로 호스트가 데이터의 물리적 위치를 직접 제어할 수 있도록 한다 [3]. 기존 SSD는 내부 FTL이 블록 할당과 주소 변환을 전담하기 때문에 애플리케이션이 데이터 수명 정보를 제공하더라도 실제 배치는 장치 내부 정책에 의해 결정되었다. 반면 FDP SSD는 PID 인터페이스를 통해 호스트가 데이터를 특정 Reclaim Unit (RU) 또는 Reclaim Group (RG)에 명시적으로 할당할 수 있다.

RU는 여러 NAND 블록으로 구성된 쓰기 단위이며 RG는 다수의 RU가 포함된 논리적 그룹으로 일반적으로 하나의

* 본 연구는 정부(과학기술정보통신부)의 재원으로 한국연구재단(RS-2025-00564249) 및 2025년 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업(2024-0-00043)의 지원을 받아 수행되었음.

† Corresponding author

NAND die 세트를 의미한다 [3]. FDP SSD는 호스트가 특정 PID로 쓰기를 요청하면 해당 PID가 참조하는 RU에 데이터를 배치하고 RU가 가득 차면 새로운 RU로 매핑을 갱신한다. 이를 통해 호스트는 데이터의 수명에 따라 데이터를 분리할 수 있어 GC 간섭을 줄이고 WAF를 낮출 수 있다 [3].

2.2. FDP 데이터 배치 모드

FDP 사양은 GC 시 데이터 배치 방식을 두 가지 정책으로 구분한다:

- **Persistently Isolated**는 쓰기뿐 아니라 GC 과정에서도 데이터 격리를 유지한다. 컨트롤러는 데이터를 기존 PID와 매핑된 RU에 재배치하며 다른 수명의 데이터와 혼합하지 않는다.
- **Initially Isolated**는 쓰기 시점에는 수명에 따라 데이터를 분리하여 RU에 배치하지만 GC가 발생할 때 컨트롤러가 데이터를 동일 Reclaim Group 내의 다른 RU로 이동시킬 수 있다. 즉, GC 후에는 데이터 격리가 유지되지 않을 수 있다 [3].

두 모드의 데이터 복제 위치는 **그림 1**과 같으며 각각의 모드는 설계상 명확한 trade-off를 가진다. Persistently Isolated는 강한 데이터 격리를 보장해 GC 간섭을 줄이지만 RU 재사용의 유연성이 떨어지고 공간 활용률이 낮을 수 있다. 반면 Initially Isolated는 GC 과정에서 데이터 이동이 자유로워 NAND 공간을 더 효율적으로 활용할 수 있지만 혼합된 데이터로 인해 WAF가 높아질 가능성이 있다. 따라서 워크로드 특성에 따라 어느 정책이 더 적합한지는 실험적 평가가 필요하다.

2.3. RocksDB with TorFS

RocksDB는 쓰기 효율을 극대화하기 위해 LSM-Tree 구조를 사용하는 고성능 key-value 저장소 시스템이다. 메모리에 저장된 데이터 (MemTable)는 주기적으로 디스크로 플러시(Flush)되어 SST 파일로 생성되고 이후 여러 레벨 간의 compaction 과정을 통해 병합된다. 이 과정에서 단기 데이터와 장기 데이터가 동일 물리 블록에 섞여 쓰이는 현상이 발생한다.

TorFS [2]는 FDP SSD용 사용자 수준 파일시스템으로 RocksDB의 각 SST 파일을 Level별로 서로 다른 PID에 매핑한다. 즉, RocksDB의 LSM-Tree에서 Level 0과 같이 단기적으로 갱신되는 SST 파일은 짧은 수명의 PID로, Level 1 이상과 같은 장기 데이터 파일은 긴 수명의 PID로 분리하여 기록한다. 이렇게 데이터 수명이 다른 파일을 별도의 RU에 배치함으로써 GC 간섭을 최소화하고 WAF를 줄이는 방식이다.

3. 연구 방법론

3.1. 실험 환경

본 연구는 FDP 기능을 지원하도록 확장 구현된 FEMU 상에서 수행되었다. FEMU는 QEMU 기반 NVMe SSD 에뮬레이터로 펌웨어 수준의 기능 검증과 정책 실험을 지원한다.

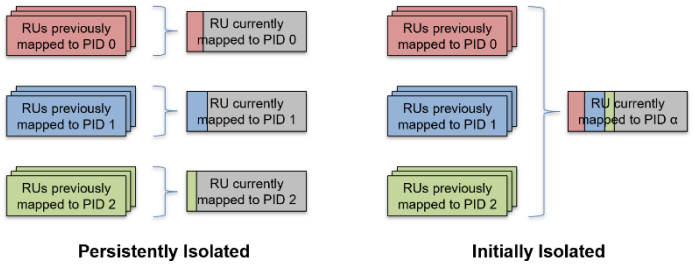

그림 1. 두 배치 모드의 GC 시 데이터 복제 위치

FEMU 내부의 NAND 구성은 1개의 RG로 이루어져 있으며 총 30개의 RU를 개당 1 GB로 설정하였다. 이러한 설정은 실제 NVMe FDP SSD의 구조를 단순화해 모사한 것으로 FDP의 논리적 데이터 격리 정책이 GC 동작과 WAF에 미치는 영향을 분석하기에 적합하다.

3.2. 데이터 배치 모드 구현

FDP의 두 가지 데이터 격리 정책인 Persistently Isolated와 Initially Isolated는 FEMU 내 FTL 계층에서 RU 할당 및 재배치 로직으로 구현되었다. 실험에서는 총 8개의 PID를 사용할 수 있도록 설정하였으며 각 PID는 호스트가 쓰기 요청 시 지정할 수 있다.

Persistently Isolated 모드에서는 RU의 메타데이터로 PID를 기록하여 각 RU를 명시적으로 추적한다. GC가 발생하면 삭제 대상 RU의 PID를 확인한 후 동일한 PID가 참조하는 RU로만 데이터를 복제하도록 제한함으로써 GC 수행 시에도 데이터 격리를 지속적으로 유지할 수 있도록 구현하였다.

반면, Initially Isolated 모드에서는 첫 쓰기 요청 시 호스트로부터 전달받은 PID를 기반으로 매핑된 RU에 데이터가 기록된다. 그러나 시스템의 빈 RU가 부족하여 GC가 발생하면 동일한 RG 내의 다른 RU로 유효 데이터를 복제하여 재배치하도록 구현하였다. 이를 통해 데이터는 RU 단위로 독립적으로 관리되지만 GC 시에는 동일 RG 내에서 자유롭게 이동할 수 있다.

3.3. 워크로드 및 측정 방법

실험 워크로드는 RocksDB 위에서 YCSB의 사용자 정의 워크로드를 구동하는 형태로 구성하였다. 모든 요청은 새로운 key-value 쌍에 대한 삽입으로만 구성되며 동일 키에 대한 갱신이나 읽기 요청은 발생하지 않는다. 키는 사전에 정의된 키 공간 전역에 걸쳐 분포하도록 생성되어 논리적으로는 랜덤한 쓰기 패턴을 형성한다. 이러한 워크로드 패턴은 디바이스의 WAF와 latency를 명확하게 관찰할 수 있도록 한다. 워크로드는 RocksDB 인스턴스 위에서 TorFS를 스토리지 계층으로 사용하여 실행되며 호스트 기준으로 각각 10.74 GB와 21.47 GB의 데이터를 기록한다. 측정 항목은 WAF 및 평균 latency로 구성되며 각 실험은 cold-start 상태에서 수행한 후 동일 워크로드를 두 FDP 모드에 대해 각각 5회 반복하여 평균값을 취하였다.

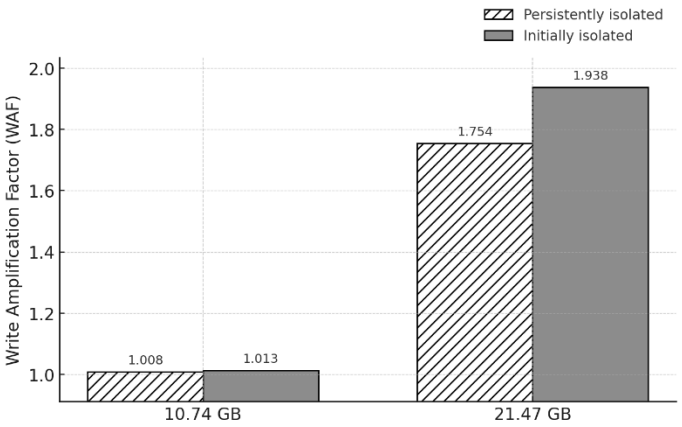

그림 2. 각 모드의 WAF 측정 결과

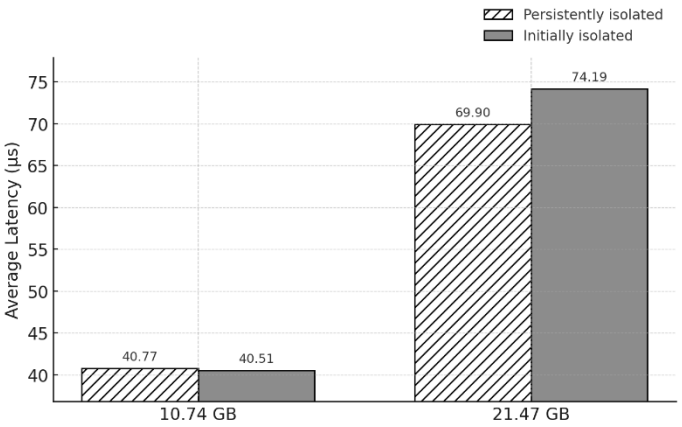

그림 3. 각 모드의 평균 latency 측정 결과

4. 실험 결과

4.1. WAF 측정 결과

그림 2는 두 모드의 WAF 측정 결과를 보여준다. 입력 데이터의 크기가 10.74 GB일 때는 GC 시 데이터 복제가 거의 발생하지 않아 WAF는 약 1.01로 거의 동일한 모습이다. 반면, 데이터 크기가 21.47 GB로 증가하면서 Persistently Isolated는 1.754, Initially Isolated 모드는 1.938로 WAF가 증가했으며 두 모드는 약 10% 정도의 차이를 보였다.

이는 Initially Isolated 모드의 GC 과정에서 수명이 다른 데이터들이 같은 RU에 누적됨으로써 추후 GC 발생 시 유효한 페이지들이 자주 복제되어 추가적인 쓰기 연산이 발생했기 때문으로 볼 수 있다. 반면 Persistently Isolated는 동일 수명의 데이터만 같은 RU에 배치되므로 유효 데이터 복제가 상대적으로 적어 WAF 증가 폭이 완만했다.

4.2. 평균 latency 측정 결과

그림 3은 두 모드에서의 평균 latency 변화를 나타낸다. 입력 데이터의 크기가 10.74 GB일 때 두 모드는 모두 약 41 µs 수준으로 거의 동일한 모습이다. 그러나 데이터 크기가 21.47 GB로 증가하면서 Persistently Isolated 모드는 69.9 µs, Initially Isolated 모드는 74.19 µs로 평균 latency가 증가했으며 두 모드는 약 6% 정도의 차이를 보였다.

이는 Initially Isolated 모드의 경우 수명이 다른 데이터가 동일 RU 내에 혼재되면서 GC가 더 자주 발생하고 그 과정에서 유효 페이지 복제가 늘어났기 때문으로 해석된다. 즉, GC가 본격적으로 활성화된 구간에서는 Persistently Isolated의 데이터 격리가 latency 상승을 완화하는 역할을 함을 확인할 수 있다.

5. 결론 및 향후 연구

실험 결과 쓰기 비중이 높고 데이터셋이 크며 수명 구분이 명확한 RocksDB 워크로드에서는 GC 최소화가 전체 쓰기 성능에 큰 영향을 미치므로 Persistently Isolated 정책이 더 적합하다고 할 수 있다. 그러나 다중 테넌트 환경이나 동적으로 데이터 수명이 변하는 워크로드에서는 RU를 유연하게 재활용할 수 있는 Initially Isolated가 공간 활용률 측면에서 더 효율적일 가능성이 여전히 남아있다.

향후 연구에서는 이러한 관점에서 Initially Isolated가 더 효율적으로 동작할 수 있는 워크로드 특성 및 응용 환경을 탐색하고 하이브리드 GC 정책 설계나 동적 모드 전환 기법을 연구하는 것이 필요하다.

[참고 문헌]

[1] Facebook, RocksDB: A Persistent Key-Value Store for Flash and RAM Storage, GitHub Repository.
https://github.com/facebook/rocksdb

[2] Samsung DS, TorFS: FDP Plugin for RocksDB, GitHub Repository.
https://github.com/SamsungDS/TorFS

[3] Samsung Electronics, NVMe FDP – A Promising New SSD Data Placement Approach,
Samsung Semiconductor Tech Blog, 2025.
https://semiconductor.samsung.com/news-events/tech-blog/nvme-fdp-a-promising-new-ssd-data-placement-approach/

[4] MoatLab, FEMU: Fast NVMe SSD Emulator for Full-Stack System Research, GitHub Repository.
https://github.com/MoatLab/FEMU