사물인터넷융복합논문지 Vol. 12, No. 2, pp. 1-6, 2026 https://doi.org/10.20465/KIOTS.2026.12.2.001

낸드 플래시 메모리 기반 저장장치에서 대용량 대역폭 데이터를 고속 처리하기 위한 내부 RAID 기반 저장 기술 설계

이현섭*
백석대학교 컴퓨터공학부 교수

A Design of an Internal RAID Based Storage Technology for High Speed Processing of Large-Capacity, High-Bandwidth Data in NAND Flash Memory Based Storage Devices

Hyun-Seob Lee*
Professor, Division of Computer Engineering, Baekseok University

요 약 전통적인 플래시 메모리 기반 저장장치에서는 플래시 메모리 칩의 처리 속도가 호스트 인터페이스의 전송 속도보다 빨라, 저장장치의 성능이 인터페이스의 I/O 대역폭에 의해 제한되었다. 그러나 저장장치 인터페이스 기술이 SATA, SAS를 거쳐 NVMe로 진화하면서 상황이 근본적으로 변화하였다. NVMe 인터페이스는 PCIe 버스를 활용하여 기존 대비 수십 배 이상의 전송 대역폭을 제공하며, 이는 플래시 메모리의 처리 속도를 크게 상회한다. 이에 따라 성능 병목 지점이 인터페이스 계층에서 플래시 메모리 처리 계층으로 이동하였으며, 저장장치 설계에 새로운 접근 방식이 요구되고 있다. 본 논문에서는 이러한 문제를 해결하기 위해 SSD 내부에서 독립적으로 동작하는 다수의 플래시 메모리 칩을 논리적 그룹으로 구성하고, 각 그룹에 내부 RAID 기법을 적용하여 고대역폭 데이터를 병렬 처리하는 아키텍처를 제안한다. 제안하는 방법은 증가된 대역폭을 효과적으로 활용하여 시스템 처리량을 극대화할 뿐만 아니라, RAID의 패리티 기반 오류 정정을 통해 대용량 데이터 처리 시 발생하는 비트 오류, 칩 장애, 데이터 손실에 대한 강건한 복구 기능을 제공한다. 결과적으로 본 연구는 차세대 고속 인터페이스 환경에서 고성능과 고신뢰성을 동시에 만족시키는 효율적인 저장 시스템 솔루션을 제시한다.

주제어 : 플래시 메모리, 내부 RAID, 대역폭, 데이터 복구, 저장장치 성능

Abstract In traditional flash memory-based storage devices, the processing speed of the flash memory chips exceeded the transfer speed of the host interface, limiting the storage device's performance to the I/O bandwidth of the interface. However, the situation fundamentally changed as storage device interface technology evolved from SATA and SAS to NVMe. The NVMe interface leverages the PCIe bus to provide tens of times more transfer bandwidth than its predecessors, significantly exceeding the processing speed of flash memory. Consequently, the performance bottleneck has shifted from the interface layer to the flash memory processing layer, demanding a new approach to storage device design. This paper proposes an architecture to address this issue. It organizes multiple flash memory chips operating independently within an SSD into logical groups and applies an internal RAID technique to each group, enabling parallel processing of high-bandwidth data. The proposed method not only maximizes system throughput by effectively utilizing the increased bandwidth but also provides robust recovery capabilities against bit errors, chip failures, and data loss during large-scale data processing through RAID's parity-based error correction. Consequently, this research presents an efficient storage system solution that simultaneously satisfies high performance and high reliability in next-generation high-speed interface environments.

Key Words : Flash Memory, Internal RAID, Bandwidth, Data Recovery, Storage Device Performance

This paper was supported by 2026 Baekseok University Research Fund
*교신저자 : 이현섭(hyunseob@bu.ac.kr)
접수일 2025년 10월 12일 수정일 2026년 04월 01일 심사완료일 2026년 04월 13일

1. 서론

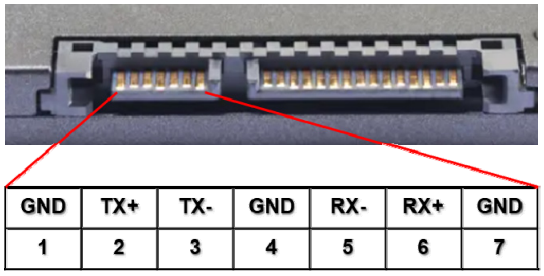
전통적인 플래시 메모리 기반 저장장치 환경에서는 플래시 메모리의 처리 속도가 데이터 전송 속도보다 빠르기 때문에 전송 속도가 성능의 병목 현상을 일으켰다. 그러나 대용량 저장 시스템에서 SATA(Serial ATA)[1-3], SAS(Serial Attached SCSI)[4-6]를 거쳐 NVMe(Non-Volatile Memory Express)[7-9]로 이어지는 저장장치 통신 기술의 발전으로 전송 대역폭이 플래시 메모리의 처리 속도를 초과하게 되었다. NVMe 인터페이스는 PCIe(Peripheral Component Interconnect Express) [10, 11] 버스를 활용하여 기존 인터페이스 대비 수십 배 이상의 전송 대역폭을 제공하며, 이는 플래시 메모리의 내부 처리 속도를 크게 상회한다. 이에 따라 저장장치 성능의 병목 지점이 인터페이스 전송 계층에서 플래시 메모리 미디어 처리 계층으로 이동하게 되었으며, 이러한 패러다임의 전환은 저장장치 설계에 있어 새로운 접근 방식을 요구하고 있다. 본 논문에서는 이러한 문제를 해결하기 위해 플래시 메모리 기반 SSD(Solid State Drive) 내부에서 독립적으로 동작하는 다수의 플래시 메모리 칩을 논리적 그룹으로 구성하고, 각 그룹에 내부 RAID(Redundant Array of Independent Disks) 기법을 적용하여 고대역폭 데이터를 병렬로 처리하는 새로운 아키텍처를 제안한다. 제안하는 방법은 증가된 인터페이스 대역폭을 효과적으로 활용하여 시스템의 전체 처리량(throughput)을 극대화할 수 있을 뿐만 아니라, RAID의 패리티(parity) 기반 오류 정정 메커니즘을 통해 대용량 데이터 처리 과정에서 발생할 수 있는 비트 오류, 칩 장애, 그리고 예기치 않은 데이터 손실에 대한 강건한 복구 기능을 제공한다. 결과적으로 본 연구는 차세대 고속 인터페이스 환경에서 요구되는 고성능과 고신뢰성을 동시에 만족시키는 안정적이고 효율적인 저장 시스템 솔루션을 제시한다.

2. 배경

2.1 SATA 인터페이스

SATA 인터페이스는 데이터 전송핀 7핀과 전원 커넥터 15핀으로 구성된다. 이 중 데이터 전송 핀은 2개의 송신 핀과 2개의 수신 핀을 가지고 있다.

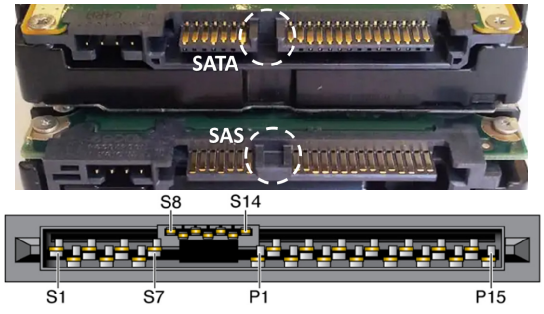
Fig. 1은 SATA의 데이터 전송 핀의 구조를 보여주고 있다. 7핀의 데이터 전송 핀 중 2번, 3번 핀인 TX+와 TX-는 데이터를 송신하기 위한 핀으로 노이즈를 최소화하기 위해 Twisted Pair 구조로 데이터를 송신한다. 그리고 5번 6번 핀인 RX-와 RX+는 데이터를 수신하기 위한 핀이다. SATA는 이렇게 단일 연결 포트를 통해 SATA I, SATA II SATA III에서 각각 1.5 Gb/s, 3.0 Gb/s, 6.0 Gb/s의 속도로 데이터를 송수신한다.



[Fig. 1] SATA Data Transfer Interface

2.2 SAS 인터페이스

SAS는 총 29핀으로 구성된 고성능 인터페이스다. 각 핀의 구조는 2개의 7개의 데이터 전송핀과 15개의 전원 커넥터로 구성된다. 따라서 SAS와 비교하여 더 빠르고, 안정적이며 2개의 데이터 핀을 통해 듀얼 포트를 진행한다.



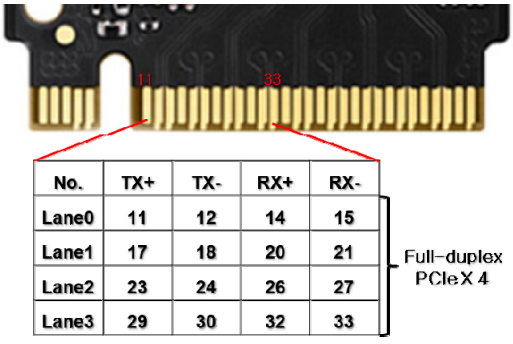
[Fig. 2] SAS Interface

Fig. 2는 SAS 인터페이스를 보여주고 있다. 그림과 같이 SAS 인터페이서는 SATA 인터페이스와 비교하여 연속하여 연결된 컨넥터의 형태를 가지고 있다. 그리고 막혀있는 컨넥터의 뒷면은 그림과 같이 7개의 추가적인 핀이 구성된다. 즉, 그림에서 S1에서 S7까지의 핀 구성은 SATA와 동일하고, S8에서 S14까지 데이터 전송핀이 한 셋트 추가되어 있다. 따라서 SATA와 비교하여 듀얼 포트 연결이 가능하고 이는 전송 대역폭과 전송 속도가 2

배 이상 증가되는 효과가 있다. SAS-3의 경우 약 12 Gb/s의 속도로 데이터 전송이 가능하고, SAS-4의 경우 24 Gb/s의 전송이 가능하다. 따라서 SATA대비 최소 2배 이상의 데이터 전송 대역폭과 속도를 가지고 있다.

2.3 NVMe 인터페이스

NVMe는 CPU와 직접 통신이 가능한 PCIe 버스 위에서 동작하여 응답 시간이 짧은 고속 프로토콜이다. 핀의 구성은 총 67개의 핀이 양면으로 배열되어 있고, 홀수번 핀과 짝수번 핀의 위치가 양면으로 분할되어 있다. 이 중 데이터를 전송하는 단일 연결핀은 TX 2핀과 RX 2핀으로 구성되어 있고 총 4개의 레인(lane)이 PCIe 연결이 가능하다.

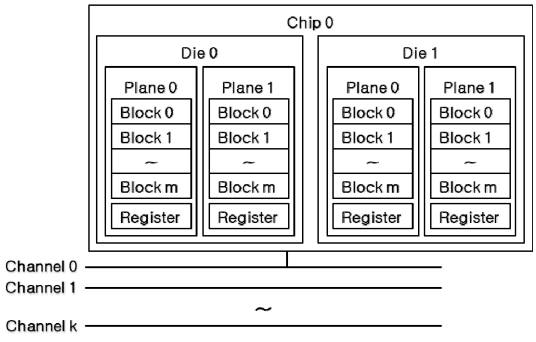


[Fig. 3] NVMe Data Transfer Interface

Fig. 3은 NVMe의 데이터 전송 핀의 구조를 보여주고 있다. NVMe는 그림과 같이 4개의 PCIe 레인을 통해 병렬 전송이 가능하다. 0번 레인의 TX+, TX-, RX+, RX-핀은 11, 12, 14, 15번 핀이다. 그리고 1번 레인은 17, 18, 20, 21번 2번 레인은 23, 24, 26, 27번 3번 레인은 29, 30, 32, 33번 레인을 통해 데이터를 전송한다. 각 레인은 전이중 통신(full-duplex) 방식으로 데이터를 송수신하며, 독립된 통신 포트로 데이터 전송을 병렬 처리한다. 따라서 NVMe는 고대역폭의 데이터를 빠르게 전송할 수 있다. NVMe는 PCIe 4.0을 기준으로 약 64 Gb/s의 속도로 동작하며, SATA, SAS 등 기타 인터페이스와 비교하여 대용량 데이터를 고속으로 전송 가능하다.

2.4 낸드 플래시 메모리 칩의 구성과 병렬 처리

낸드 플래시 메모리의 독립적 운용이 가능한 구성은 채널(channel), 칩(chip), 다이(die), 플래인(plane)이다. 먼저 채널은 플래시 메모리 칩을 연결하는 독립적인 데이터 버스다. 따라서 독립적으로 동작하기 때문에 서로 다른 채널이 완전히 병렬로 데이터를 전송할 수 있다. 그 다음 각 채널마다 버스에 n개의 플래시 메모리 칩이 연결되어 있고 칩 인에이블(chip enable, CE) 신호를 통해 채널의 버스를 공유하고 있는 칩들 중 하나를 활성화 시킬 수 있다. 다이는 물리적인 칩 패키지 내에서 인터리빙 방식으로 병렬 동작이 가능한 독립적인 구성으로, 1개, 2개, 4개, 8개 또는 그 이상의 다이가 포함 가능하다. 마지막으로, 플레인은 다이 내에서 독립적으로 동작할 수 있는 메모리 영역이다. 일반적으로 다이당 2개 또는 4개의 플레인으로 구성된다[12, 13].



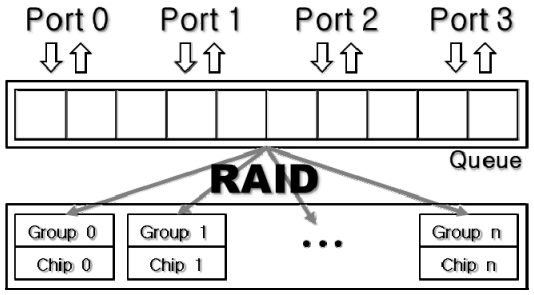
[Fig. 4] Structure of NAND Flash Memroy Chip

Fig. 4는 낸드 플래시 메모리 칩의 구조를 보여주고 있다. 그림의 예제에서는 k개의 채널로 버스가 연결되어 있다. 각 채널은 독립된 버스라서 독립적 운용이 가능하다. 그리고 각 채널에는 n개의 칩이 연결되어 있다. 그림에서는 0번 칩 한개에 대한 내구 구조의 예제를 보여주고 있다. 그림과 같이 하나의 칩은 여러개의 다이로 구성되며 예제에서는 0번과 1번 두 개의 다이로 구성된 예제를 보여주고 있다. 그림과 같이 각각의 다이도 물리적으로 독립되어 있기 때문에 독립적인 병렬 운용이 가능하다. 그리고 각 다이는 여러개의 플래인으로 구성되어 있다. 그림에서는 2개의 플래인으로 구성된 다이 구조의 예제를 보여주고 있다. 각각의 플래인은 독립적으로 구성되어 있기 때문에 병렬적 운용이 가능하다. 마지막으로 각각의 플래인은 직렬로 연결된 m개의 블록들로 구성되어 있다. 각각의 블록 내부에 페이지 단위 데이터는 플래인과 연결된 레지스터 버퍼를 통해 버스로 연결된다. 따라서 독립적인 병렬 처리가 가능한 마지막 물리적 구성 단위는 플래인이다.

3. 인터페이스 대역폭 기반 내부 RAID 설계

3.1 핵심 아이디어

본 논문에서는 인터페이스의 대역폭에 최적화 된 데이터 처리를 위한 내부 RAID를 설계한다. 이를 위해 RAID 단위는 한번에 전송될 수 있는 인터페이스 최대 대역폭 단위이고, 이 데이터는 큐를 통해 버퍼에 유지하다가 RAID 단위가 모이는대로 칩 단위의 논리적 그룹으로 플래시 메모리에 저장된다.

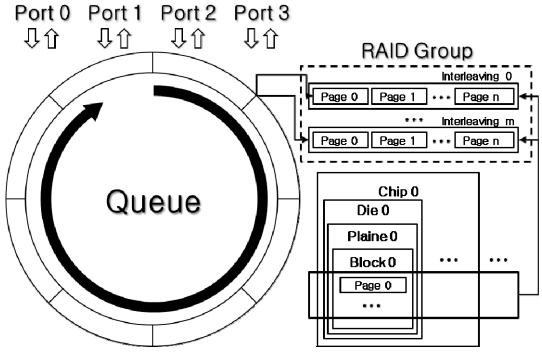


[Fig. 5] Key Idea

Fig. 5는 핵심 아이디어를 보여주고 있다. 그림의 예제에서는 저장장치가 4개의 포트로 데이터를 전송한다. 그리고 데기큐에서 각 포트에서 전송된 데이터를 모아서 처리한다. 이때 데이터는 칩에서 처리할 수 있는 단위로 RAID 5 그룹을 구성하며 RAID에서 한번에 처리되는 데이터 전송 단위는 각 포트에서 한번에 전달될 수 있는 최대 크기와 패리티 데이터로 구성한다. 예를 들어 각 포트로부터 한번에 전송되는 데이터가 2개 블록 크기의 데이터이고, 플래시 메모리에서 인터리빙을 통해 한번에 저장할 수 있는 단위가 1개 블록이라고 가정할 경우, RAID에서 한번에 처리되는 데이터는 인터페이스에서 한번에 전송되는 최대 대역폭인 4개의 블록과 1개의 패리티 블록이다. 따라서, RAID의 데이터를 저장하는 그룹은 5개의 칩 단위로 5개의 블록이 묶여서 큐에 쌓인 데이터를 저장 및 관리한다.

3.2 큐잉 정책 및 RAID 그룹의 설계

저장 시스템에서 외부 데이터 전송 인터페이스의 대역폭과 내부 데이터 처리량이 균형을 이룰 경우, 추가적인 제어 설계 없이도 데이터 전송과 처리가 효율적으로 수행된다. 그러나 두 구성 요소 간의 성능이 비대칭적인 경우, 인터페이스와 저장매체 사이에서 발생하는 속도 차이로 인해 병목 현상이 발생할 수 있다. 이러한 문제를 완화하기 위해 일반적으로 버퍼링 기법이 활용되지만, 대역폭 격차가 큰 시스템에서는 단순 버퍼만으로는 충분한 성능 보장이 어렵다. 특히 NVMe와 같이 초고속 인터페이스를 사용하는 환경에서, 단일 입출력 단위당 처리량이 제한적인 낸드 플래시 기반 저장장치는 내부 처리 속도가 상대적으로 느려 성능 불균형이 두드러진다. 이에 본 연구에서는 이러한 구조적 비대칭성을 완화하기 위해, 큐를 기반으로 한 효율적인 큐잉 정책을 설계하여 인터페이스와 내부 처리 모듈 간의 데이터 처리량 차이를 최소화한다.



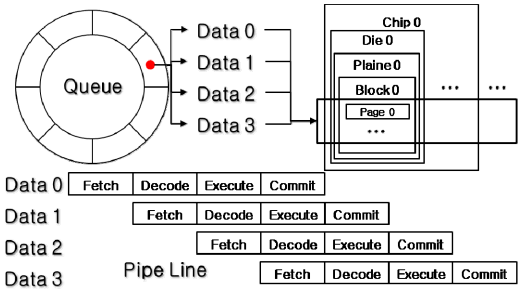
[Fig. 6] Design of Queuing Policy and RAID Group

Fig. 6은 큐잉 정책의 설계를 보여주고 있다. 그림에서는 4개의 포트로 전송되는 데이터를 원형 큐로 전송한다. 큐의 단위는 포트를 통해 한번에 전송될 수 있는 단위다. 그 다음 큐의 단위를 처리할 수 있는 크기의 RAID 그룹을 생성하여 플래시 메모리에서 처리한다. RAID 그룹의 구성은 큐의 최소 단위 데이터를 처리할 수 있는 복수의 인터리빙 그룹으로 구성하며, 인터리빙 그룹은 채널, 칩, 다이, 플레인의 병렬 동작을 고려하여 각 플래인 내에 같은 순번째 페이지들을 묶어서 조직한다. 또한 각 인터리빙 그룹 마다 하나의 페이지에 패리티를 저장하여 데이터의 무결성을 확보한다.

3.3 인터리빙 기반 RAID 그룹의 설계

내부 RAID를 구성하여 데이터 처리량을 확장하기 위해서는 병렬처리 가능한 최소 저장 단위를 하나의 RAID 그룹으로 구성하여 인터리빙 단위를 최대화 해야 한다. NVMe의 경우 PCIe Gen4 기준으로 각 레인당 약 1.9692 GB/s가 가능하다. 낸드 플래시 메모리의 경우 칩당 다이를 32개로 가정하고 다이당 플래인을 8개로,

페이지의 크기를 16 KB로 설계했을 때 칩당 인터리빙 최대 속도는 약 4 MB/s의 처리 속도가 나올 수 있다. 따라서 플래시 메모리 칩을 PCB의 양면을 이용하여 128개를 설치하였을 때 최대 처리량은 512 MB/s까지 가능하며 512개의 칩을 사용할 경우 데이터 처리량은 2 GB/s가 된다.

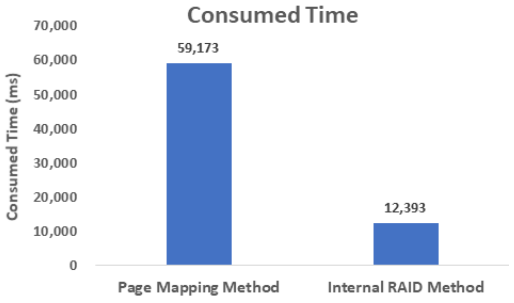


[Fig. 7] Design of Interleaving Based RAID Group

Fig. 7은 인터페이스 대역폭 대비 내부 처리량이 작은 저장 시스템에서 인터리빙 기반 RAID 그룹의 설계를 보여주고 있다. 그림에서는 큐에 전달 된 데이터를 처리할 수 있는 인터리빙 구조를 4개의 데이터 셋으로 구분하였다. 그 다음 파이프라인을 통해 데이터 셋을 처리한다. 명령인출(fetch) 과정은 큐에서 실행 가능한 요청을 선택 및 스케줄하는 단계다. 명령해석(decode)은 수신된 명령을 해석하고 논리 주소와 물리주소를 매핑하여 접근할 칩, 다이, 플레인, 페이지의 위치를 결정하는 단계다. 명령실행(execute) 단계는 플래시 메모리에 읽기/쓰기를 수행하는 단계다. 마지막으로 커밋(commit) 단계는 ECC등 에러를 정정하고 처리 결과에 대한 메타 데이터를 업데이트 및 전달하는 단계다. 이 과정에서 거대 RAID 그룹 데이터를 인터리빙과 파이프 라인을 통해 내부 데이터 처리량을 최적화 한다.

4. 시뮬레이션 결과 분석

내부 RAID 기법의 성능을 분석하기 위해 시뮬레이션을 통해 소모된 시간을 측정하였다. 실험에 사용된 데이터는 엔터프라이즈 환경에서 수집된 주소 데이터[15]를 사용하였다. 플래시 메모리의 구조는 512개의 칩, 32개의 다이, 8개의 플래인으로 설정하였다. 플레인 당 512개의 블록으로 구성되고 각 블록은 256개의 페이지로 가정하였다. 페이지의 크기는 16KB로 가정하였다. 동일 처리하기 위한 인터리빙 단위는 페이지 단위의 병렬성을 최대화 하여 131,072개의 페이지로 구성하였고 레이드 단위도 동일하게 2GB로 설정하였다. 호스트 인터페이스는 NVMe m.2와 PCIe Gen4로 가정하였다. 레인 당 데이터 전송속도는 2 GB/s로 가정하였다.



[Fig. 8] Simulation Result

Fig. 8은 트레이스 데이터를 처리하는 동안 제안하는 내부 RAID 기법의 소비된 시간을 측정한 결과다. 제안하는 방법의 우수성을 측정하기 위해 FTL(Flash Transfer Layer)[15, 16] 기법 중 가장 최적화된 성능을 보이고 있는 페이지 매핑 기법과 성능을 비교하였다. 그림의 결과와 같이 페이지 매핑은 약 58,173ms의 시간이 소모되었고 내부 RAID 기법은 12,393ms의 시간이 사용되었다. 이 결과는 제안하는 기법이 약 79.05% 성능 향상할 수 있음을 보여주었다.

5. 결론

본 논문에서는 고대역폭의 인터페이스 대비 부족한 데이터 처리량의 SSD(Solid State Drive)에서 내부에서 독립적으로 동작하는 인터리빙 그룹을 구성하고 이 그룹을 기반으로 RAID 기법을 적용하여 고대역폭 데이터를 병렬로 처리하는 새로운 아키텍처를 제안하였다. 결과적으로 본 연구는 차세대 고속 인터페이스 환경에서 요구되는 고성능과 고신뢰성을 동시에 만족시키는 안정적이고 효율적인 저장 시스템 솔루션을 제안하였다. 그러나 제안하는 기법은 쓰기 성능에 편중 된 설계였다. 향후에는 내부적으로 발생하는 가비지 컬렉션과 읽기/지우기 동작 또한 최적화 하기 위한 연구를 진행 할 예정이다.

REFERENCES

[1] R.Niwase, H.Harasawa, Y.Yamaguchi, W.Kaijie and H.Amano, "A cost/power efficient storage system with directly connected FPGA and SATA disks," 2023 IEEE 16th International Symposium on Embedded Multicore/ Many-core Systems-on-Chip (MCSoC), pp.51-58, 2023.

[2] X.Li, Z.Yu, L.Han, B.Wang, F.Chen, and S.Zhou, "Research on Secure Storage with SATA Command Filtering Based on Logical Isolation Architecture," 2024 4th International Conference on Electronic Information Engineering and Computer Communication, pp.1042-1047, 2024.

[3] I.Saukani, E.Nuraini, S.Nurhadi, A.S.H.Sumarno, R.T.T.Saptawati, P.Prasetyo and F.I.S.Ms, "Format Methods on Storage Media (Hard Disk) for Optimization Data Storage Capacity," Asian Journal Science and Engineering, Vol.2, No.2, pp.126-132, 2023.

[4] J.H.Park, S.Choi, G.Oh, and S.W.Lee, "Sas: Ssd as sql database system, "Proceedings of the VLDB Endowment, Vol.14, No.9, pp.1481-1488, 2021.

[5] S.Maneas, K.Mahdaviani, T.Emami and B.Schroeder, "Reliability of SSDs in enterprise storage systems: a large-scale field study," ACM Transactions on Storage, Vol.17, No.1, pp.1-27, 2021.

[6] B.H.Xiao, K.Xiao, J.X.Li, C.F.Xiao, S.Cao and Z.Q.Liu, "Flexible electrochemical energy storage devices and related applications: recent progress and challenges," Chemical Science, Vol.15, No.29, pp.11229-11266, 2024.

[7] R.Wertenbroek, Y.Thoma and A.Dassatti, "A portable linux-based firmware for NVMe computational storage devices," ACM Transactions on Storage, Vol.21, No.2, pp.1-36, 2025.

[8] S.H.Kim, J.Shim, E.Lee, S.Jeong, I.Kang and J.S.Kim, "Empowering storage systems research with nvmevirt: A comprehensive nvme device emulator," ACM Transactions on Storage, Vol.19, No.4, pp.1-26, 2023.

[9] G.Haas and V.Leis, "What modern nvme storage can do, and how to exploit it: High-performance i/o for high-performance storage engines," Proceedings of the VLDB Endowment, Vol.16, No.9, pp.2090-2102, 2023.

[10] M.Jung, "Hello bytes, bye blocks: Pcie storage meets compute express link for memory expansion (cxl-ssd)," Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems, pp.45-51, 2022.

[11] K.G.Nirupama and N.B.GVL, "Interoperable SoC Framework for PCIe Design Verification Challenges," 2024 IEEE 5th Women in Technology Conference, pp.1-6, 2024.

[12] H.S.Lee, "A Safety IO Throttling Method Inducting Differential End of Life to Improving the Reliability of Big Data Maintenance in the SSD based RAID," Journal of Digital Convergence, Vol.20, No.5, pp.593-598, 2022.

[13] H.S.Lee, "Performance analysis and prediction through various over-provision on NAND flash memory based storage," Journal of Digital Convergence, Vol.20, No.3, pp.343-348, 2022.

[14] SNIA, http://iotta.snia.org/traces/block-io/388.

[15] H.S.Lee, "A Study on the Performance Measurement and Analysis on the Virtual Memory based FTL Policy through the Changing Map Data Resource," Journal of Internet of Things and Convergence, Vol.9, No.1, pp.71-76, 2023.

[16] H.S.Lee, "A Design of Enhanced Block Mapping Method Based on Batch Processing," Journal of Internet of Things and Convergence, Vol.11, No.4, pp.71-76, 2025.

이 현 섭(Hyun-Seob Lee) [종신회원]

- 2013년 2월 : 한양대학교 컴퓨터공학과 (공학 박사)
- 2012년 3월 ~ 2021년 2월 : 삼성전자 책임연구원
- 2021년 3월 ~ 현재 : 백석대학교 컴퓨터공학부 조교수

〈관심분야〉

인공지능, 저장시스템, 임베디드 시스템