블록 매핑 기반으로 일괄처리하는 쓰기 버퍼에서 버퍼 관리 정책이 성능에 미치는 영향 분석

이현섭*
백석대학교 컴퓨터공학부 교수

Analysis of the Impact of Buffer Management Policies on Performance in Batch Processing Write Buffers Based on Block Mapping

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

요 약 낸드 플래시 메모리 기반 저장장치는 비대칭의 데이터 처리 단위와 쓰기 전 소거(Erase Before Write)의 물리적 특징이 있다. 따라서 이러한 문제를 감추고 일반적인 저장장치와 동일하게 사용하기 위해 논리적인 주소와 물리적인 주소를 분할하여 관리하는 플래시 전환 계층(Flash Transfer Layer, FTL)을 사용한다. 최근에는 성능이 최적화된 페이지 매핑 기법을 널리 사용하고 있다. 그러나 이 방법은, 저장장치의 용량이 증가할수록 논리적 주소와 물리적 주소를 매핑하는 매핑 정보가 과도하게 증가하는 문제가 있다. 이를 해결하기 위해 매핑 정보의 오버헤드를 줄이기 위한 연구들이 진행되고 있다. 특히, 블록 매핑을 기반으로 쓰기 버퍼에 데이터를 모아서 일괄처리하는 연구는 매핑 정보의 오버헤드 뿐만 아니라 블록 매핑 기법의 성능 문제도 개선한 연구이다. 그러나 일괄처리 되는 블록을 선정하기 위한 버퍼 관리 정책을 최적화 하기 위한 연구가 필요하다. 따라서 본 연구에서는 블록 단위로 일괄처리하는 쓰기 버퍼 환경에서 버퍼 교체 알고리즘이 시스템 성능에 미치는 영향을 분석하였다. 기존의 LRU, FIFO와 같은 전통적인 교체 알고리즘 외에도 낸드 플래시 메모리의 특성을 고려한 새로운 교체 전략들을 제안하고 실험을 통해 성능을 평가하였다. 실험을 통해 버퍼 교체 정책이 성능에 미치는 영향을 분석하고 최적화 된 버퍼의 크기와 버퍼 교체 정책의 연구 방향을 제시한다.

주제어 : 낸드 플래시 메모리, 플래시 전환 계층, 블록 매핑 기법, 버퍼 관리 정책, 대용량 저장장치

Abstract NAND flash memory-based storage devices have asymmetric data processing units and the physical characteristic of erase before write. Therefore, in order to hide these issues and use them in the same way as general storage devices, a flash transfer layer (FTL) is used to manage logical addresses and physical addresses separately. Recently, performance-optimized page mapping techniques have been widely used. However, this method has the problem of excessive increase in mapping information for mapping logical addresses to physical addresses as the storage device capacity increases. To solve this problem, research is being conducted to reduce the overhead of mapping information. In particular, research on batch processing by collecting data in a write buffer based on block mapping has improved not only the overhead of mapping information but also the performance issues of block mapping techniques. However, further research is needed to optimize the buffer management policy for selecting blocks to be batch processed. Therefore, this study analyzed the impact of buffer replacement algorithms on system performance in a write buffer environment that processes data in blocks. In addition to traditional replacement algorithms such as LRU and FIFO, this study proposes new replacement strategies that consider the characteristics of NAND flash memory and evaluates their performance through experiments. Through experimentation, we analyze the impact of buffer replacement policies on performance and propose research directions for optimizing buffer size and buffer replacement policies.

Key Words : NAND Flash Memory, Flash Translation Layer, Block Mapping Method, Page Mapping Method, Large Capacity Storage

This paper was supported by 2025 Baekseok University Research Fund
*교신저자 : 이현섭(hyunseob@bu.ac.kr)
접수일 2025년 09월 09일 수정일 2025년 10월 03일 심사완료일 2025년 10월 10일

1. 서론

낸드 플래시 메모리는 비휘발성, 저전력, 높은 내구성 등의 장점으로 다양한 저장장치에 핵심 부품으로 사용되고 있다. 특히 대용량의 저장장치를 사용하는 시스템에서 SSD(Solid State Drive)와 eMMC(Embedded Multi Media Card), UFS(Universal Flash Storage) 등을 활용하면서, 주요 저장 매체인 낸드 플래시 메모리의 용량은 지속적으로 증가하고 있다[1, 2]. 그러나 플래시 메모리는 일반적인 저장장치와는 다른 독특한 물리적 특징과 이 특징을 관리하는 방법이 필요하다. 대표적인 특징 중 한 가지는 제자리 갱신이(In-place Update)이 안되는 특성이다. 이 때문에 데이터 갱신이 요청될 경우 해당 블록의 쓰기 전 지우기(Erase Before Write) 동작을 수행해야 한다. 또 다른 특징은 읽기 쓰기는 페이지 단위로 수행되는 반면 지우기 동작은 블록 단위로 수행되는 비대칭의 연산 단위 특징이다. 따라서 페이지 단위의 제자리 갱신이 발생하면 페이지가 포함된 블록 단위의 데이터를 지우고 다시 써야 하는 관리 오버헤드가 있다. 이러한 특징과 오버헤드로 일반적인 저장 시스템을 플래시 메모리에서 사용하는 것은 불가능하다. 이 문제를 해결하기 위해 논리적 주소와 물리적 주소를 매핑하는 플래시 변환 계층(Flash Translation Layer, FTL)이 사용되고 있다. 그러나 이 방법은 용량이 증가할수록 매핑 유지해야 하는 매핑 정보 또한 기하급수적으로 증가하게 되어 메모리 오버헤드가 심각한 문제로 대두되고 있다. FTL의 가장 일반적인 알고리즘은 페이지 매핑 방법이다. 페이지 매핑 방식은 가장 세밀한 매핑 단위를 제공하여 높은 성능을 보이지만 매핑 테이블의 크기가 매우 크다는 단점이 있다. 반면 블록 매핑 방식은 매핑 테이블의 크기를 대폭 줄일 수 있지만, 블록 내부에서 무효한 페이지가 발생할 경우 전체 블록을 다시 쓰기해야 하는 성능 저하 문제가 발생한다[3-5].

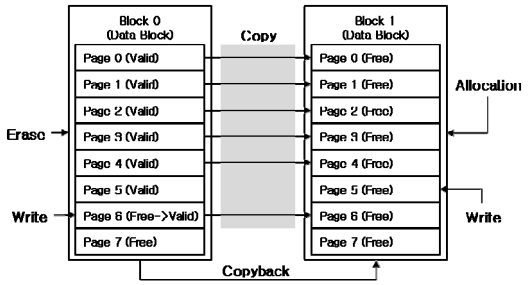
블록 매핑 방식의 가장 큰 문제점은 부분 업데이트 시 발생하는 성능 저하이다. 블록 내의 일부 페이지만 수정되더라도 전체 블록을 새로운 위치에 복사해야 하는 카피백(Copy-back) 연산이 필요하며, 이는 쓰기 지연시간을 크게 증가시킨다. 또한, 빈번한 가비지 컬렉션을 유발하여 시스템 전체의 성능을 저하시키는 원인이 된다. 이러한 문제를 해결하기 위한 기존 연구들은 주로 하이브리드 매핑 기법이나 로그 블록을 활용한 방식들이 제안됐으나, 여전히 매핑 오버헤드와 성능 간의 트레이드오프 문제가 완전히 해결되지 않았다.

FTL의 성능 개선을 위한 한 가지 방법은 쓰기 버퍼(Write Buffer)를 사용하는 것이다. 선행 연구에서는 블록 매핑의 성능 문제를 해결하기 위해 쓰기 버퍼를 활용한 일괄 처리(Batching) 방식의 선행 연구를 제안하였다. 이 방식은 여러 개의 쓰기 요청을 버퍼에 임시로 저장한 후, 한 블록 크기만큼의 데이터가 누적되면 한번에 플래시 메모리에 쓰기하는 방식이다. 이를 통해 카피백 연산의 빈도를 줄이고 쓰기 성능을 향상시킬 수 있다. 그러나 제한된 크기의 버퍼에서 어떤 블록의 데이터를 유지하고 어떤 블록을 플래시 메모리에 플러시(Flush)할지 결정하는 버퍼 교체 알고리즘이 전체 시스템 성능에 미치는 영향이 클 것으로 예상한다. 따라서 본 연구에서는 블록 매핑 기반의 버퍼 환경에서 다양한 버퍼 교체 알고리즘을 설계하고 이들의 성능을 비교 분석한다. 또한, 버퍼 자원의 크기를 조절하며 성능을 분석하는 실험을 진행하고 블록 매핑 전용 버퍼 자원의 최적화를 위한 연구를 진행한다. 이러한 실험과 분석을 통해 향후에는 블록 매핑에 최적화된 버퍼 교체 전략을 설계하기 위한 연구 방향을 제시한다.

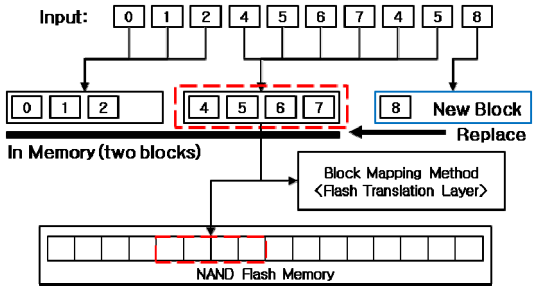
2. 배경 및 관련 연구

2.1 블록 매핑 기법에서 카피백 연산의 분석

플래시 메모리는 여러개의 메모리 칩을 구성되어 있고 각 메모리 칩은 여러개의 블록들로 이루어져 있다. 그리고 하나의 블록은 여러개의 페이지를 포함하고 있다. 이러한 구조로 되어 있는 플래시 메모리는 비대칭의 읽기/쓰기/지우기 단위와 성능을 가지고 있다. 일반적으로 읽기 및 쓰기 동작은 페이지 단위로 처리되는 반면 지우기 동작은 블록단위로 수행된다. 또한, 읽기, 쓰기, 지우기의 성능 차이는 1:10:100의 비율을 가지고 있다. 따라서 성능을 향상하기 위해서는 읽기 횟수가 증가하더라도 쓰기 횟수를 줄여야 하며, 쓰기 횟수가 증가하더라도 지우기 횟수를 줄여야 한다. 또 다른 특징 중 한 가지는 데이터의 쓰기 동작은 비어있는 페이지에만 가능하다는 것 이다. 즉, 이미 데이터가 저장된 페이지에 데이터 갱신이나 새로운 데이터를 써야하는 경우 해당 페이지를 지우고 데이터를 써야한다. 그러나 플래시 메모리에서 데이터를 지우는 단위는 블록 단위다. 따라서 플래시 메모리의 데이터 업데이를 위한 별도의 운영 방법이 필요하다.



[Fig. 1] Copyback operation



[Fig. 2] Batch process based mapping method

Fig. 1은 블록 매핑을 사용하는 플래시 메모리의 구조와 데이터가 쓰여있는 페이지에 변경된 데이터를 다시 쓰기 위한 카피백 연산[6-8]의 동작을 보여주고 있다. 그림의 예제에서는 하나의 블록이 0번부터 7번까지 8개의 페이지로 구성되어 있다. 그리고 0번 블록의 0번부터 5번까지는 이미 데이터가 쓰여있고 6번과 7번 페이지는 비어있다. 이때 비어있는 6번 프리 페이지 주소로 새로운 데이터의 쓰기 요청이 들어오면 추가적인 연산 없이 6번 페이지에 데이터를 저장한다. 그러나 이미 데이터를 저장하고 있는 0번에서 6번 페이지 중 특정 한 페이지에 데이터의 쓰기 요청이 발생하였을 때 카피백 연산을 수행해야 한다. 예를 들어 5번 페이지에 쓰기 요청이 발생하면 비어 있는 새로운 블록을 할당한다. 그림의 예제에서는 1번 블록을 할당 받았다. 그다음 0번 블록의 유효 데이터를 보유하고 있는 0번에서 6번 페이지를 1번 블록으로 복사 한다. 그리고 블록 매핑 정보를 0번 블록에서 1번 블록으로 변경한다. 마지막으로 이전 블록인 0번 블록을 삭제한다. 이러한 동작은 데이터 업데이트가 발생해도 플래시 메모리의 특성을 감추고 변경된 데이터를 저장할 수 있도록 해 준다. 그러나 내부적으로는 데이터를 복사하고 블록을 삭제하는 과정에서 단 한번의 업데이트가 많은 읽기, 쓰기, 지우기 동작을 야기하는 문제가 있다. 따라서 성능을 향상하기 위해서는 카피백 연산의 횟수를 줄이거나 지연시키는 연구가 필요하다.

2.2 일괄처리 기반 블록 매핑 기법

매핑 정보를 위한 자원 오버헤드에 최적화 된 블록 매핑 기법은 데이터 갱신이 발생할때 마다 카피백 연산이 발생한다. 따라서 집중적인 쓰기 연산을 수행할 때 심각한 성능 저하 문제를 가지고 있다. 이러한 문제를 해결하기 위해 쓰기 버퍼를 이용한 일괄처리 기반 블록 매핑 기법[9]이 제안되었다.

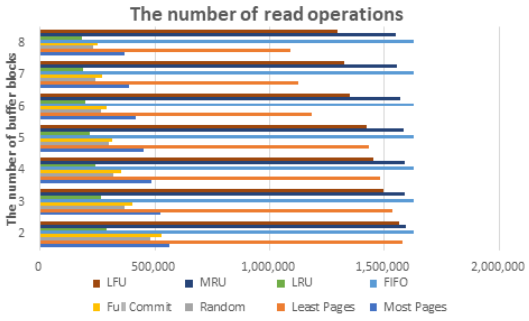
Fig. 2는 일괄 처리 기반 매핑 기법을 보여주고 있다. 그림에서는 메모리상에 2개의 쓰기 버퍼를 유지하고 있다. 그리고 쓰기 데이터에 대해 블록 단위로 유지와 관리를 하고 있다. 이때 새로운 블록에 속하는 페이지의 쓰기 데이터가 요청되면 쓰기 버퍼에 유지하고 있는 블록 중 특정 블록을 Victim으로 선택하여 플래시 메모리에 커밋한다. 그리고 이 동작은 블록 단위로 진행된다. 그러나 만약 Victim 블록과 플래시 메모리에 저장된 페이지의 구성이 상이한 경우 Victim 블록의 최신 페이지와 플래시 메모리 블록의 이전 페이지를 합병하여 블록 단위의 쓰기 데이터를 구성한 다음 플래시 메모리에 저장한다. 이 과정은 카피백 연산과 유사하며, 대량의 업데이트 된 페이지들을 모아서 한번에 일괄 배치 방법으로 저장한다. 따라서 버퍼에 유지되는 블록내의 페이지 개수 만큼 카피백 횟수를 줄일 수 있는 장점이 있다. 그림의 예제에서는 쓰기 버퍼에 0, 1, 2번 페이지가 속한 블록과 4, 5, 6, 7 페이지가 속한 블록이 유지되고 있다. 이때 새로운 블록에 속하는 8번 페이지의 쓰기 데이터가 전달되었다. 그러나 버퍼에는 새로운 블록 구성요소인 페이지를 유지할 공간이 없어서 4, 5, 6, 7 페이지가 구성된 블록을 Victim으로 선택하였다. 선택한 블록은 블록 매핑 기법을 통해 연관된 블록에 일괄 저장된다. 그런데 Victim 블록은 해당 블록의 모든 페이지의 최신 업데이트 페이지를 모두 유지하고 있다. 따라서 추가적인 읽기 쓰기 등 합병 동작 없이 매핑된 페이지를 한 번 소거하고 저장하여 이상적인 쓰기 성능을 보여주었다.

그림의 예제와 같이 블록 매핑 기법을 사용하는 저장 시스템에서 Victim으로 선정된 블록의 최신 업데이트 페이지가 버퍼에 모두 유지되고 있는 경우 블록 매핑을 사용하더라도 이상적인 성능을 보여줄 수 있다. 그러나 데이터를 관리하는 방법이나 버퍼의 설정값에 따라 버퍼 관리 정책의 성능은 예측할 수 없는 영향을 받을 수 있다. 따라서 블록 매핑에 영향을 주는 정책들을 분석하여

블록 매핑 기법 전용 버퍼 관리정책을 연구하는 것이 필요하다.

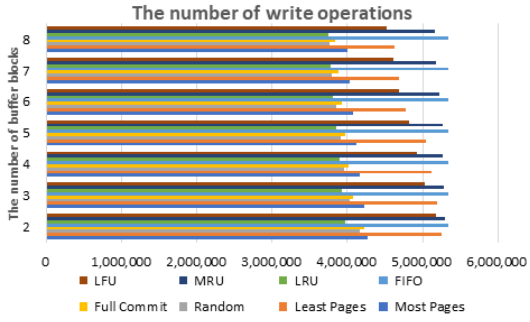
3. 실험 및 영향 분석

블록 매핑 기법 기반의 쓰기 버퍼에서 버퍼 교체정책의 성능을 측정하고 분석하기 위해 시뮬레이션 구현하고 성능을 분석하였다. 실험 환경은 블록당 256개의 페이지로 구성된 400GB의 저장장 공간으로 설정하였다. 또한, 각 페이지의 크기는 512B로 하였고, 쓰기버퍼의 크기는 최적화 실험을 위해 2개의 블록에서 8개의 블록을 유지할 수 있는 크기로 변경해 가며 실험을 진행하였다. 실험에 사용한 데이터는 엔터프라이즈 시스템에서 사용된 트레이스[10]를 사용하였다. 마지막으로 실험을 위해 LFU(least frequently used)[11], MRU(most recently used)[12], LRU(least recently used)[13], FIFO[14], Full Commit[15], Random, Least Pages, Most Pages의 버퍼 관리 정책 및 Victim 선정 기준을 이용하여 성능을 평가하였다. 이중 Random은 버퍼 내에서 임의의 블록을 Victim으로 선택하는 알고리즘을 적용하였다. Least Pages와 Most Pages는 버퍼내의 블록 중 가장 적은 페이지와 가장 많은 페이지를 유지하고 있는 블록을 선택하는 알고리즘을 적용하였다. 실험은 트레이스 데이터를 쓰기 버퍼에 블록 단위로 모아 놓고 버퍼에 새로운 블록의 영역에 데이터가 삽입될 때 Victim을 선정하여 블록 매핑으로 일괄처리하였다.



[Fig. 3] The number of read operations

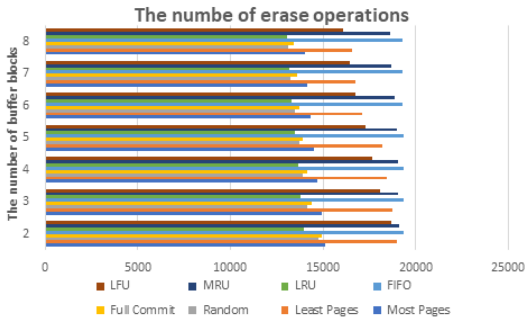
Fig. 3은 실험에서 발생한 읽기 횟수다. 그림의 결과와 같이 버퍼 블록의 개수가 증가할수록 성능이 향상하는 경향을 보인다. 가장 좋은 성능을 보인 버퍼교체 정책은 LRU고, 291,775회에서 180,396회의 읽기 요청을 수행했다. 가장 안좋은 성능을 보인 정책은 FIFO이고 1,627,281회에서 1,627,187회의 성능을 보였고 버퍼가 증가해도 큰 성능차이를 보이지 않았다. 버퍼의 크기와 관계 없이 높은 성능을 보인 정책은 LRU, Full Commit, Random, Most Pages 기법이었고 가장 안좋은 성능의 FIFO와 비교하여 각각 82.07, 67.5, 70.47, 65.36%의 우수한 성능 차이를 보였다. 최근 접근 횟수가 높은 LFU와 버퍼내에 가장 적은 페이지를 유지하고 있는 블록을를 커밋하는 Least Pages는 버퍼의 크기에 따라 2.92~33.04%와 4.02~20.49%의 상승효과가 있었으나 성능에 큰 영향을 미치지 못했다.



[Fig. 4] The number of write operations

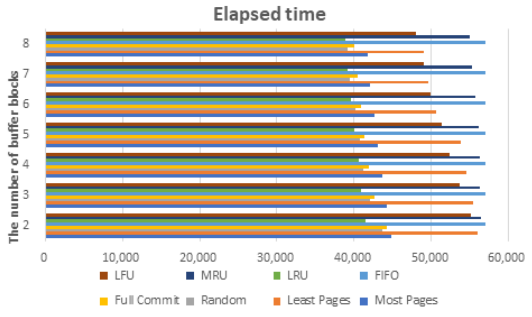
Fig. 4는 실험 중 발생한 쓰기 횟수다. 쓰기 횟수의 성능은 버퍼가 증가할수록 향상되었다. 버퍼 블록이 2개일 때 LFU, MRU, LRU, FIFO, Full Commit, Random, Least Pages, Most Pages의 순으로 각각 5,173,69, 5,293,243, 3,975,584, 5,349,461, 4,223,147, 4,169,028, 5,259,831, 4,273,815회 발생했다. 그리고 버퍼 블록이 8개 일 때 각각 12.68, 2.44, 5.93, 0.14, 9.05, 9.73, 11.96, 6.44%의 성능이 향상되었다. 가장 좋은 성능을 보인 교체 알고리즘은 LRU고 Random이 버퍼의 크기에 따라 4.87~0.62% 근소한 성능차이가 발생했다. 그다음 Full Commit과 Most Pages가 각각 6.23~2.7%, 7.5~6.91%의 성능차이가 발생했다. 이러한 성능의 차이는 버퍼의 크기가 증가할수록 감소하는 차이를 보였다. 그 외에 다른 교체 알고리즘은 약 30%이상 차이가 발생했다.

Fig. 5는 실험 중 발생한 지우기 횟수다. 지우기 횟수의 성능은 버퍼가 증가할수록 향상되었다. 버퍼 블록이 2개일 때 각각 18,659, 19,126, 13,971, 19336, 14,940, 14,726, 18,996, 15,122회 발생했다. 그리고 버퍼 블록이 8개 일 때 각각 14.74%, 2.71, 6.79, 0.18,



[Fig. 5] The number of erase operations

10.14, 10.91, 12.94, 7.27%의 성능이 향상되었다. 가장 좋은 성능을 보인 교체 알고리즘은 LRU고 Random이 버퍼의 크기에 따라 5.4~0.74% 근소한 성능차이가 발생했다. 그다음 Full Commit과 Most Pages가 각각 6.94~3.09%, 8.24~7.69%의 성능차이가 발생했다. 지우기 차이 또한 버퍼의 크기가 증가할수록 감소하여 쓰기 성능의 경향과 유사한 패턴을 보였다. 그 외에 다른 교체 알고리즘은 약 30%이상 차이가 발생했다.



[Fig. 6] Elapsed time

Fig. 6은 각 알고리즘의 경과 시간을 보여주고 있다. 버퍼의 개수가 2일 때 각각 순서대로 55,165, 56,437, 41,445, 57,055, 44,254, 43,643, 56,078, 44,814ms의 시간이 경과하였고, 버퍼의 개수가 8일 때 각각 48,081, 55,053, 38,881, 56,975, 40,003, 39,176, 49,052, 41,753였다. 경과 시간은 LRU가 가장 좋은 성능을 보였고, Random이 5.31~0.76%의 근소한 성능차이를 보였다. 그다음 Full Commit과 Most Pages가 각각 6.78~2.88%, 8.13~7.38%의 성능차이를 보였다. 이러한 경향은 버퍼의 크기가 증가할수록 차이가 줄었으며 특히 Random은 버퍼가 8개 일대 1%미만의 성능을 보였다. 실험의 결과를 통해, 유의미한 성능의 패턴을 보여준 버퍼교체 정책은 LRU, Random, Full Commit, Most Pages로 판단된다. 따라서 최근 사용한 데이터에 대한 접근 패턴을 버퍼에 유지하는 것, 임의의 데이터를 커밋하는 것, 모든 데이터를 커밋하는 것, 가장 많은 데이터의 블록을 모아서 한번에 커밋하는 방법등이 블록 매핑 기반 버퍼 관리정책에 효과가 있는 것으로 판단된다.

4. 결론

본 논문에서는 대용량 플래시 메모리 기반 저장장치에서 블록 매핑 전용 쓰기 버퍼의 성능을 최적화 하기 위한 특성을 분석하기 위해 다양한 버퍼 교체 알고리즘을 평가하였다. 또한 버퍼의 크기를 조절하며 버퍼의 자원에 따른 성능 개선 효과를 분석하였다. 실험에서는 최근 사용한 데이터에 대한 접근 패턴을 버퍼에 유지하는 특성, 임의의 데이터를 커밋하는 특성, 모든 데이터를 커밋하는 특성, 가장 많은 데이터의 블록을 모아서 한번에 커밋하는 특성이 블록 매핑의 쓰기 버퍼에 효과가 있음을 확인하였다. 또한, 버퍼의 크기가 2개에서 8개로 400% 증가하는 동안 쓰기 성능은 최대 약 12.84%의 성능만 증가 밖에 없었다. 따라서, 다양한 특성을 활용하고 자원 최적화 연구가 필요하다. 향후에는 성능향상에 효과가 있는 특성들을 반영할 수 있는 알고리즘과 버퍼 자원 증가하는 비율 만큼 성능을 향상할 수 있는 버퍼 교체 연구를 진행할 예정이다.

REFERENCES

[1] C.Kim and C.Lee, “Design of eMMC controller with multiple channels,” 2016 International SoC Design Conference (ISOCC). IEEE, pp. 317-318, 2016.

[2] N.Yang, J.Kim, G.Park, C.Kwon, S.Lee, S.Pae and S.Hwang, “A Study on System Level UFS M-PHY Reliability Measurement Method Using RDVS.” 2021 IEEE International Reliability Physics Symposium (IRPS), IEEE, pp.1-7, 2021.

[3] 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.

[4] 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.

[5] 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.

[6] D.Hong, M.Kim, J.Park, M.Jung and J.Kim, "Improving SSD performance using adaptive restricted-copyback operations," 2019 IEEE Non-Volatile Memory Systems and Applications Symposium (NVMSA), IEEE, pp.1-6, 2019.

[7] D.Huang, B.Ding, W.Tong and D.Feng, "SuperCopyback: Revisiting Copyback on Modern High-Performance NAND Flash-based SSDs," 2025 62nd ACM/IEEE Design Automation Conference (DAC), IEEE, pp.1-7, 2025.

[8] W.Chang, Y.Lim and J.Cho, J. "An efficient copy-back operation scheme using dedicated flash memory controller in solid-state disks," Proc. of the International Journal of Electrical Energy, Vol.2, No.1, 2014.

[9] 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.

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

[11] A.Najla, A.B.Dris and S.Alahmadi. "Randomized least frequently used cache replacement strategy for named data networking." International Conference on Computer Applications & Information Security (ICCAIS). pp.1-6, 2020.

[12] F.C.D..Assis, A.H.V.Dias and C.F.Helenice, "The most frequently used tests for assessing executive functions in aging," Dementia & neuropsychologia, Vol.9, No.2, pp.149-155, 2015.

[13] D.Lee, J.Choi, J.H.Kim, S.H.Noh, S.L.Min, Y.Cho, and C.S.Kim, "On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies," In Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp.134-143, 1999.

[14] A.C.Sembiring, J.Tampubolon, D.Sitanggang, and M..Turnip, "Improvement of inventory system using first in first out (FIFO) method," In Journal of Physics: Conference Series, Vol.1361, No.1, pp.012070, 2019.

[15] D.Debortoli, and R.Nunes, "Fiscal policy under loose commitment," Journal of Economic Theory, Vo1.45, No.3, pp.1005-1032, 2015.

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