Journal of The Korea Society of Computer and Information
Vol. 24 No. 8, pp. 9-17, August 2019

www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.08.009

# Block Unit Mapping Technique of NAND Flash Memory Using Variable Offset

Seung-Woo Lee*, Kwan-Woo Ryu*

## Abstract

In this paper, we propose a block mapping technique applicable to NAND flash memory. In order to use the NAND flash memory with the operating system and the file system developed on the basis of the hard disk which is mainly used in the general PC field, it is necessary to use the system software known as the FTL (Flash Translation Layer). FTL overcomes the disadvantage of not being able to overwrite data by using the address mapping table and solves the additional features caused by the physical structure of NAND flash memory. In this paper, we propose a new mapping method based on the block mapping method for efficient use of the NAND flash memory. In the case of the proposed technique, the data modification operation is processed by using a blank page in the existing block without using an additional block for the data modification operation, thereby minimizing the block unit deletion operation in the merging operation. Also, the frequency of occurrence of the sequential write request and random write request Accordingly, by optimally adjusting the ratio of pages for recording data in a block and pages for recording data requested for modification, it is possible to optimize sequential writing and random writing by maximizing the utilization of pages in a block.

▸Keyword: Nand Flash Memory, FTL, Garbage Collection, Mapping algorithm, Block-level mapping

## I. Introduction

최근 모바일기기 분야의 성장세는 일반 PC분야의 성장세를 크게 앞섰으며 이러한 결과는 모바일기기 분야에 하드웨어 및 소프트웨어의 지속적인 연구와 기술개발로 인해 가능하였다 [1]. 특히 낸드 플래시 메모리는 이러한 연구와 기술개발로 인해 모바일기기 분야에서 요구되는 경량화, 저전력, 내구성 등의 조건을 대부분 만족하며 모바일기기 분야의 저장장치로 주로 사용되고 있다. 낸드 플래시 메모리는 일반 PC분야에서 주로 사용되는 하드디스크와는 다르게 기계장치의 구동 없이 전기적으로 동작함으로 상대적으로 더 빠른 입출력이 가능한 장점이 있다.

하지만 낸드 플래시 메모리는 물리적인 구조로 인해 데이터 쓰기 요청 시 반드시 빈 페이지를 대상으로만 쓰기 요청을 처리할 수 있으며 이 때문에 이미 데이터가 기록되어있는 페이지의 경우 데이터 수정 요청 시 데이터 덮어쓰기가 불가능하다.

이러한 단점을 가진 낸드 플래시 메모리를 일반 PC분야에서 사용되어온 하드디스크를 기반으로 개발된 운영체제 및 파일시스템과 함께 사용하기 위해서는 필수적으로 FTL(Flash Translation Layer)로 알려진 시스템 소프트웨어를 이용하여야 한다.

FTL은 기본적으로 파일시스템과 낸드 플래시 메모리 사이에서 동작하는 시스템 소프트웨어이며 FTL이 제공하는 가장 기본적인 기능은 파일시스템에서 발생하는 논리적인 섹터 단위의 입출력 요청에 대응하는 낸드 플래시 메모리의 물리페이지 주소를 서로 기록하는 주소 매핑 테이블을 제공하는 것이다.

• First Author: Seung-Woo Lee, Corresponding Author: Kwan-Woo Ryu
*Seung-Woo Lee (zpa007@knu.ac.kr), Dept. of Computer Science, Kyungpook National University
*Kwan-Woo Ryu (kwryu@knu.ac.kr), Dept. of Computer Science, Kyungpook National University
• Received: 2019. 07. 29, Revised: 2019. 08. 21, Accepted: 2019. 08. 26.

FTL은 주소 매핑 테이블을 이용하여 데이터 덮어쓰기 불가 문제를 극복하며 낸드 플래시 메모리의 물리적인 구조로 인해 발생하는 추가적인 특징들을 해결한다.

지금까지 알려진 주소 매핑 기법으로 페이지단위 매핑기법과 블록단위 매핑기법이 있으며 이 두 가지 기법의 장점을 모두 가지는 하이브리드 매핑기법이 있다. 특히 하이브리드 매핑기법의 경우 로그블록을 이용하여 데이터 수정 작업에 대한 처리효율을 극대화 한다. 하지만 수정된 데이터를 기록한 로그 블록과 기존 데이터가 기록된 데이터 블록과의 병합작업 시 발생하는 블록단위 삭제 작업 및 빈번한 페이지 복사는 많은 시스템 자원을 소모하는 단점으로 알려져 있다.

최근 이러한 단점은 더욱 크게 부각되고 있는데 현재 모바일 기기 분야는 더 빠른 통신 속도를 기반으로 더 다양한 멀티미디어 중심의 서비스 이용이 점차 증가하고 있으며 모바일기기에서 처리되는 데이터 파일들의 크기 역시 점차 대용량화 되고 있는 추세이다[2,3]. 이러한 크고 많은 양의 데이터 파일들을 저장하기 위해 낸드 플래시 메모리 역시 대용량화되고 있으며 대용량 낸드 플래시 메모리는 이전과 비교해 상대적으로 블록 내 페이지의 수가 더 많으며 페이지의 크기 또한 이전과 비교해 더 크다. 이러한 대용량 낸드 플래시 메모리를 대상으로 한 블록단위 삭제 작업 시 소모되는 시스템 자원은 비례하여 증가할 것이다. 이러한 대용량 낸드 플래시 메모리를 효율적으로 사용하기 위해서는 최대한 블록단위 삭제 작업을 최소화는 것이 필요하며 가능한 블록 내 페이지를 최대한 활용할 수 있는 매핑기법이 필요하다.

본 논문에서는 점차 대용량화 되는 낸드 플래시 메모리의 효율적인 사용을 위해 블록 매핑기법에 기반 한 새로운 매핑기법을 제안한다. 제안기법의 경우 데이터 수정 작업에 대해 추가적인 블록 사용 없이 기존 블록 내 빈 페이지를 이용하여 데이터 수정 작업을 처리함으로써 병합작업 시 발생하는 블록단위 삭제작업을 최소화 하였으며 순차 쓰기요청과 무작위 쓰기요청의 발생 빈도에 따라 블록 내 데이터를 기록하는 페이지와 수정요청된 데이터를 기록하기 위한 페이지의 비율을 가변적으로 조정함으로 블록 내 페이지의 활용도를 최대한 높여 순차쓰기와 무작위쓰기에 최적화된 블록 운영이 가능하다.

## II. Preliminaries

### 1. Related works

#### 1.1 Structure of NAND flash memory

플래시 메모리는 전기적으로 데이터를 기록, 삭제할 수 있는 비휘발성 기억 장치이며 데이터를 저장하는 최소 단위는 셀이다. 셀은 플로팅 게이트 트랜지스터로 구성되며 플로팅 게이트에 전자를 채우고 비우는 방식으로 데이터를 기록한다[4,5]. 플로팅 게이트 트랜지스터는 컨트롤 게이트로 충분히 큰 전압을 인가할 경우 전기장의 영향으로 전자가 절연체의 막을 통과하여 플로팅 게이트로 들어오게 되며 전압을 끊을 경우 들어온 전자들이 절연체의 막에 의해 플로팅 게이트에 갇히게 된다. 이러한 물리적인 구조의 셀은 기본적으로 1비트의 정보를 저장하며 낸드 플래시 메모리는 이러한 셀들이 직렬 구조로 연결되어 페이지 단위로 그룹화 되고 페이지는 다시 블록 단위로 그룹화 된다. 최근 일부 대용량 낸드 플래시 메모리는 하나의 셀에 4비트의 정보를 저장하는 QLC(Quad Level Cell) 방식으로 동작하며 이러한 QLC 방식은 집적률을 높여 낸드 플래시 메모리의 용량 증가에 도움이 되지만 플로팅 게이트에 전자 충전과 방전이 4배 증가함으로 절연체의 가용성을 심각히 저하시킬 수 있다[5]. 이러한 QLC(Quad Level Cell) 방식의 낸드 플래시 메모리를 효과적으로 운용하기 위해서는 반드시 이를 고려한 FTL이 필수적으로 요구된다.

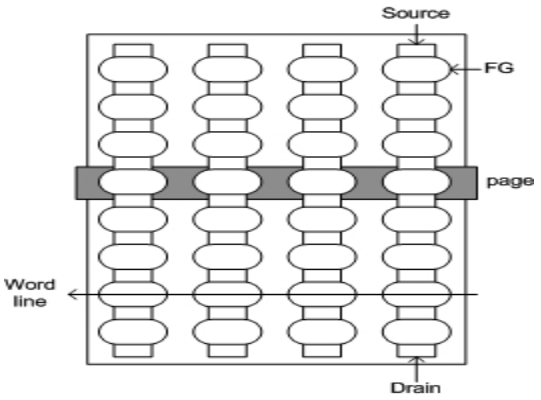


Fig. 1. Block structure of NAND flash memory

#### 1.2 Characteristics of NAND flash memory

낸드 플래시 메모리는 셀들이 직렬 구조로 연결되어 있으며 별도에 주소라인이 없으므로 이러한 낸드 플래시 메모리의 물리적인 구조를 고려하여 효율적으로 사용하기 위해 반드시 낸드 플래시 메모리의 읽기와 쓰기 그리고 삭제 작업에 대한 이해가 필요하다.

첫째 읽기와 쓰기 작업은 페이지 단위로 실행되며 읽기 작업의 경우 몇 바이트의 데이터를 읽는 경우에도 해당 데이터를 저장한 셀을 포함한 직렬구조로 연결된 페이지 전체의 셀을 모두 읽은 후 필요한 데이터를 따로 추출한 뒤 불필요한 데이터는 버려진다. 또한 쓰기 작업 역시 페이지 단위로 실행되며 몇 바이트의 데이터를 쓰고자 할 경우에도 해당 데이터를 저장한 셀을 포함하여 직렬구조로 연결된 페이지 전체의 셀에 데이터가 기록된다. 이러한 쓰기 작업 시 페이지 사이즈에 맞지 않는 쓰기 작업은 부가적인 쓰기를 발생시킴으로 이를 최소화하기 위해서는 가능한 페이지 크기에 맞춘 쓰기작업이 필요하다.

둘째 데이터는 덮어 쓰기 될 수 없다. 이는 셀의 물리적인 구조로 인해 데이터가 이미 기록된 셀은 절연체의 막 안쪽에 전자가 채워져 있는 상태임으로 전자를 비우지 않고 다시 전자를 채워 넣을 수 없기 때문이다. 이 때문에 페이지를 이루는 모든

셀들이 전자를 가지지 않는 즉 빈 페이지인 상태에서만 쓰기 작업이 가능하다.

셋째 하나의 페이지를 삭제할 경우에도 블록 전체 페이지를 삭제해야한다. 이는 주소라인이 없음으로 인해 직렬로 연결된 셀 전체를 대상으로 전자를 방출해야하기 때문이다.

### 1.3 FTL(Flash Translation Layer)

앞서 언급한 것처럼 낸드 플래시 메모리의 하드웨어 구조는 하드디스크와는 크게 다름으로 낸드 플래시 메모리를 기존 하드디스크와 동일한 방식으로 이용하기 위해서는 필수적으로 기존 파일시스템과 낸드 플래시 메모리 사이에서 동작하는 시스템 소프트웨어인 FTL(Flash Translation Layer)이 요구된다[6]. 이러한 FTL은 주소매핑과 가비지 컬렉션 기법을 제공하며 주소매핑 기법이란 기본적으로 파일시스템으로부터 발생하는 논리적인 섹터 단위의 입출력 명령에 대응하는 낸드 플래시 메모리의 물리 페이지 주소를 서로 기록한 매핑 테이블을 말한다. 또한 매핑 테이블을 이용하여 낸드 플래시 메모리의 특성 중 하나인 제자리 덮어쓰기 불가현상을 해결할 수 있으며 이는 특정 페이지의 데이터를 수정하고자 할 경우 수정 요청된 데이터를 기존 페이지가 아닌 빈 페이지를 이용해 기록한 뒤 이전 페이지를 무효화시키며 매핑 테이블에 매핑 된 기존 논리 주소와 기존 페이지의 주소를 수정 요청 데이터를 기록한 페이지의 주소로 단순 변경함으로 해결할 수 있다. 이를 통해 무효화 된 페이지를 즉시 삭제할 필요가 없으며 해당 페이지를 포함한 블록의 삭제를 최대한 연기할 수 있다. 이러한 매핑기법은 운용방식에 따라 다양한 기법들이 알려졌다. 또한 FTL은 가비지 컬렉션 기법을 제공한다. 이는 제자리 덮어쓰기 불가현상으로 인해 블록 내 무효화된 페이지 비율이 증가하여 점차 낸드 플래시 메모리에 빈 페이지가 부족할 경우 특정 시점에 이러한 블록을 대상으로 발생하는 블록단위 삭제 작업을 말한다.

### 1.4 Page-level mapping techniques

페이지 매핑 기법은 호스트로부터 요청되는 논리 페이지 주소에 대응하는 모든 물리 페이지에 대한 주소정보를 매핑 테이블로 관리하는 기법이다. 매핑 테이블은 논리 페이지 주소에 대응하는 물리 페이지 주소로 구성되며 호스트로부터 특정 논리 페이지 주소에 대한 읽기 또는 쓰기 요청 시 이에 대응하는 물리 페이지 주소를 매핑 테이블로부터 찾을 수 있다. 또한 특정 논리 페이지 주소에 대한 데이터 수정 요청 시 이에 대응하는 기존 물리 페이지는 즉시 무효화되며 새로운 빈 페이지를 이용해 수정 요청 데이터를 기록한 뒤 해당 물리 페이지 주소를 기존 논리 페이지 주소와 매핑하기 위해 매핑테이블을 수정한다.

이러한 페이지단위 매핑 기법의 장점은 쓰기 요청 시 데이터를 즉시 새로운 빈 페이지를 이용해 쓸 수 있다는 점이다. 또한 데이터 수정 작업되었을 때에도 낸드 플래시 메모리 내에 사용가능한 페이지가 존재한다면 데이터 복사 및 삭제 작업 없이 바로 쓰기가 가능하여 무작위 쓰기 패턴에 높은 성능을 보이게 된다.

하지만 페이지단위 매핑은 쓰기 작업이 계속되어 빈 페이지가 부족해질 경우 무효화된 페이지를 삭제하기 위해 블록단위 삭제가 필요하며 이는 데이터 복사 및 삭제 동작이 일시에 요구된다는 것이다. 이러한 집중적인 블록 삭제 작업은 시스템의 성능을 매우 떨어뜨리는 것이며 더불어 전체 물리 페이지에 대한 주소 매핑 정보를 기록하여야 함으로 낸드 플래시 내 페이지의 수가 많을수록 매핑 테이블 자체의 크기가 커져 많은 메모리를 소모하는 점이다.

이러한 페이지단위 매핑 기법으로는 DFTL이 있으며 이는 전체 페이지 사상 정보를 플래시에 구축하고 그 중 자주 참조되는 일부를 메모리에서 관리하도록 하는 다중 수준 사상으로 동작한다[7]. 하지만 DFTL은 갑작스러운 전원 차단이 발생할 경우 낸드 플래시 메모리에 존재하는 모든 페이지를 읽어야만 장치의 일관성을 보장할 수 있다. 이러한 DFTL의 단점을 보완한 것이 LazyFTL이며 LazyFTL 또한 다중 매핑 정보를 이용하여 페이지 매핑을 위한 메모리의 양을 줄이는 페이지단위 매핑기법이다[8]. LazyFTL은 장치 초기화 시간을 줄이기 위해 낸드 플래시 메모리에 존재하는 모든 블록의 첫 페이지에 블록의 종류를 기록해 놓는 방법을 이용하며 이를 통해 부팅 시 DFTL에 비해 장치 초기화 시간을 줄일 수 있는 장점이 있다.

한편 최근 대용량화 되는 낸드 플래시 메모리의 효율적인 사용을 위해 페이지단위 메핑 기법이 주목받고 있다. 이는 캐싱기법을 이용한 페이지 매핑기법을 뜻하며 캐싱기법을 이용하여 매핑테이블에 소모되는 메모리 사용량을 크게 줄일 수 있기 때문이다. 하지만 이러한 페이지 매핑기법 역시 캐싱기법 운용에 있어 Map Hit 확률을 증대하는 별도에 운용이 필요하며 Mis-aligned 상황에 대한 대비 또한 필요함으로 부가적인 자원 소모가 발생한다[9].

### 1.5 Block unit mapping technique

블록단위 매핑 기법은 페이지단위 매핑 기법에 단점인 매핑 테이블에 필요한 메모리 사용량을 줄이기 위한 매핑기법이며 낸드 플래시 메모리의 삭제 동작단위인 블록 단위로 매핑테이블을 관리하는 방법이다. 블록 매핑 테이블의 엔트리는 논리 블록 주소와 물리 블록 주소로 구성되며 논리 페이지 주소로부터 논리 블록 주소와 페이지 오프셋을 계산한다.

논리 블록 주소는 논리 페이지 주소를 블록 내 페이지의 개수로 나눈 값의 몫을 의미하며 나머지 값은 페이지의 위치를 뜻하는 오프셋을 의미한다.

블록단위 매핑 기법은 페이지단위 매핑 기법에 비해 메핑 테이블의 크기가 작고 블록 내 페이지에 접근하기 위해 사용하는 오프셋 값이 항상 고정되어 있으므로 연속 쓰기 요청 시 이를 기록하고 읽는데 좋은 성능이 좋다. 연속 쓰기 패턴에 성능이 좋은 이유는 갱신하고자 하는 데이터가 모두 연속적인 물리 블록 주소 및 오프셋을 가지므로 연속 쓰기 패턴의 데이터 수정 요청 시 빈 블록을 대상으로 순차적으로 데이터를 기록함으로 별도에 블록 내 페이지 복사가 일어나지 않기 때문이다.

이러한 블록단위 매핑기법의 단점은 무작위 쓰기 요청 시 발생하는데 무작위 쓰기 패턴의 경우 쓰기 또는 읽기 시 요청되는 논리 페이지의 주소가 연속하지 않음으로 블록 내 연속하여 기록된 페이지 중 하나의 페이지를 대상으로 한 데이터 수정 요청 시 에도 블록 내 고정 오프셋을 유지하기 위해 블록 내 전체 페이지를 별도에 빈 블록에 복사해야하는 부가적인 동작 필요하다.

### 1.6 Hybrid mapping technique

하이브리드 매핑기법은 페이지단위 매핑기법과 블록단위 매핑기법의 장점을 혼합한 것으로 매핑테이블에 필요한 메모리 사용량을 줄이고 무작위 쓰기 패턴에 대응하는 별도에 블록인 로그 블록을 통해 블록 단위 삭제를 최소화한다.

이러한 하이브리드 매핑기법은 최근 낸드 플래시 메모리를 대상으로 가장 활발히 연구되어온 매핑기법으로 알려진 기법으로는 BAST, FAST, Super Block 등 로그 블록을 활용하는 방식에 따라 다양한 매핑기법이 소개되었다[10,11,12,13,14].

하지만 무작위 쓰기 요청 시 이에 대응하는 별도에 블록인 로그 블록을 활용함으로 순차 쓰기 요청에 대응하는 블록과 무작위 쓰기 요청에 대응하는 블록을 동시에 관리해야하며 최종적으로 로그 블록에 무효 페이지가 많아질 경우 두 블록을 서로 병합해야하는 작업이 필요하다. 블록 병합은 두 블록 간에 유효페이지를 별도에 블록에 합치는 것으로 이를 위해 기존 블록 내에 유효페이지를 별도에 블록에 복사해야하며 병합 이후 기존 데이터블록과 로그 블록은 삭제해야하는 부가 작업이 필요하다. 또한 최근 제안되는 하이브리드 매핑기법은 로그 블록을 이용하여 무작위 쓰기 요청 시 데이터를 기록할 때 차후 블록 병합을 고려하여 페이지의 기록 순서를 블록 병합에 알맞게 조정하는 여러 기법들을 이용하고 있다.

하지만 순차 쓰기 요청에 대응하는 데이터 블록과 무작위 쓰기 요청에 대응하는 로그 블록 내 페이지의 활용도 역시 고려하여야 한다. 이는 특정 쓰기 요청 시 무작위 쓰기 패턴이 많을 경우 로그 블록 내 페이지의 활용도는 높지만 데이터 블록 내 페이지는 활용도가 높지 않게 되며 점차 블록 내 페이지 수가 많은 대용량 낸드 플래시 메모리의 경우 블록 내 페이지 활용도를 최대한 높일 수 있는 새로운 매핑 기법이 필요하다.

## III. The Proposed Scheme

### 3.1 Operation structure and characteristics of proposed method

제안기법은 기존 매핑기법들의 단점을 보완하고 대용량 낸드 플래시 메모리에 적용 가능한 새로운 주소매핑 기법이다. 제안 기법은 페이지 단위 매핑기법의 단점인 주소매핑 테이블에 필요한 메모리 사용량 문제와 블록 매핑 기법의 단점인 블록 내 페이지 접근을 위해 고정 오프셋을 사용함으로 발생하는 빈번한 페이지복사 및 블록 삭제문제 그리고 하이브리드 매핑 기법의 단점인 로그블록과 데이터블록 간에 병합 작업 시 발생하는 페이지복사 및 블록 삭제문제를 효과적으로 줄일 수 있다.

이를 위해 제안기법은 블록 내에 페이지를 7대 3비율로 나누어 순차 쓰기용도에 페이지와 데이터 수정 용도에 페이지로 사용하며 페이지 접근 시 오프셋을 이용한다.

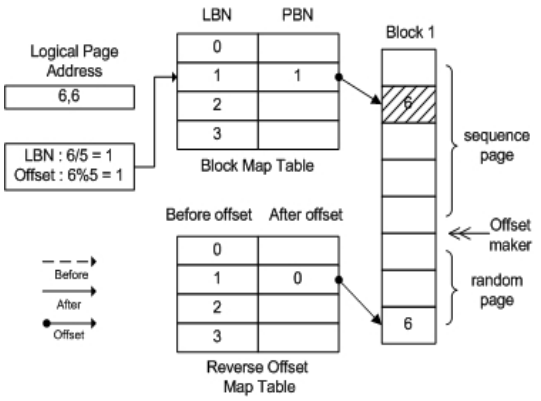


Fig. 2. Operational structure of proposed scheme

순차 쓰기용도에 페이지는 호스트로부터 발생하는 순차쓰기 요청에 대응하는 페이지이다. 연속하는 논리 페이지주소에 연속하는 물리 페이지를 대응시킴으로 별도에 주소매핑이 필요 없으며 블록 내 순차 쓰기용도에 페이지에 접근하기 위해 오프셋 개념을 이용함으로 주소매핑 테이블에 필요한 메모리 사용량을 줄일 수 있다. 순차 쓰기요청 시 주소매핑 테이블에 필요한 메모리 사용량은 기존 페이지단위 매핑기법과 비교했을 때 블록 당 페이지의 개수가 4개일 경우 최대 75% 줄일 수 있다.

또한 데이터 수정 용도에 페이지는 호스트로부터 발생하는 무작위쓰기 요청에 대응하는 페이지이다. 무작위쓰기 요청이란 이전 쓰기 요청에 논리 페이지주소와 직후 쓰기 요청에 논리 페이지주소가 연속하지 않음으로 오프셋을 이용해 페이지에 접근할 수 없다. 기존 블록단위 매핑기법의 경우 이미 데이터가 기록되어있는 페이지를 대상으로 데이터 수정 요청이 발생할 경우 빈 블록에 동일 오프셋 위치에 수정 요청된 데이터를 기록하고 이전 블록에 유효페이지들 역시 수정 요청된 데이터를 기록한 빈 블록에 동일 오프셋 위치에 모두 복사할 수밖에 없다. 이는 기존 블록단위 매핑기법에 최대 단점이며 제안기법에서는 이를 해결하기 위해 블록 내 일정 페이지를 데이터 수정 용도로 사용한다. 이러한 데이터 수정 용도에 페이지는 블록 내 가장 마지막 페이지부터 역순으로 대응시킨다.

제안기법은 블록 내 순차 쓰기용도에 페이지에 접근 시 순차 오프셋을 이용하며 데이터 수정 용도에 페이지에 접근 시 별도에 오프셋 개념인 역순 오프셋을 이용한다. 결론적으로 제안기법은 블록 내 용도에 따라 나뉜 페이지에 접근하기 위해 순차 오프셋과 역순 오프셋을 이용하며 이를 다중 오프셋이라 한다.

### 3.2 Multiple offsets

낸드 플래시 메모리의 경우 물리적 특성으로 인해 데이터 덮어쓰기가 불가능하며 기존 블록단위 매핑기법의 경우 데이터 수정 요청 시 빈 블록에 동일 오프셋에 수정 요청된 데이터를 기록하며 블록 내 페이지 접근을 위한 고정된 오프셋을 유지하기 위해 이전 블록에 유효 페이지 또한 수정 데이터가 기록된 블록에 동일 오프셋에 모두 복사해야만 한다. 즉 기존 블록단위 매핑기법은 고정된 오프셋을 이용함으로 블록 내 단 한 번에 데이터 수정 요청이 발생하여도 앞서 언급한 단점이 발생한다. 고정된 오프셋 사용은 주소매핑에 필요한 메모리 사용량을 줄일 수 있지만 부가적인 페이지 복사 및 블록 삭제 증가라는 더 큰 단점이 발생한다.

제안기법은 이러한 단점을 보완하기 위해 최초 블록 내 페이지를 순차 쓰기용도에 페이지와 데이터 수정 용도에 페이지로 나누어 7대 3의 비율로 운용함으로 순차 쓰기용도에 페이지에 대한 주소매핑 테이블에 필요한 메모리 사용량을 크게 낮추고 데이터 수정 요청 시 발생하는 페이지 복사 문제를 해결하였다.

또한 제안기법은 블록 내 페이지에 접근하기 위해 블록 내 첫 페이지를 기준으로 하는 순차 오프셋과 블록 내 마지막 페이지를 기준으로 하는 역순 오프셋을 사용한다. 블록 내 첫 페이지를 기준으로 하는 순차 오프셋은 쓰기용도에 페이지에 접근하기 위해 사용되며 이는 기존 블록 매핑 기법에서 사용되는 오프셋 개념과 동일하다.

역순 오프셋은 데이터 수정 용도의 페이지에 접근하기 위해 사용되는 오프셋을 말하며 제안기법은 블록 내 페이지를 7대 3 비율로 나누어 쓰기용도에 페이지와 데이터 수정 용도의 페이지로 사용한다. 데이터 수정 용도의 페이지는 호스트로부터 발생한 데이터 수정 요청에 대응하는 페이지로 제안기법은 블록 내 가장 마지막 페이지를 기준으로 대응한다. 이러한 데이터 수정 용도의 페이지에 접근하기 위해서는 별도에 오프셋이 필요하며 이를 역순 오프셋이라고 한다.

제안기법은 순차 오프셋을 이용하여 순차 쓰기용도에 페이지에 접근하는 경우 별도에 주소매핑 테이블이 필요하지 않지만 역순 오프셋의 경우 별도에 오프셋매핑 테이블이 필요하다.

이는 데이터 수정 요청 시 데이터가 기록된 기존 페이지를 무효화하고 동일 블록 내에 가장 마지막 페이지를 이용하여 데이터 수정 요청을 처리함으로 무효화된 페이지의 오프셋과 수정 요청된 데이터를 기록한 페이지의 오프셋을 매핑 하는 것이며 해당 페이지에 접근하기 위해 별도에 오프셋을 기억한 오프셋매핑 테이블이다.

제안기법은 이러한 데이터 수정작업에 사용되는 페이지가 많아질수록 메모리 사용량이 증가할 수 있다. 하지만 동일 블록 내에 페이지를 이용하여 데이터 수정작업을 처리함으로써 해당 블록에 공간 활용도를 최대한 높일 수 있으며 이를 통해 전체 블록에 삭제 횟수를 감소시키는 장점이 있다. 제안기법의 역순 오프셋 테이블에 필요한 메모리 사용량은 기존 블록단위 매핑 기법을 제외하고 로그 블록 기반의 매핑기법들과 비슷하거나 더 적다. 제안기법은 데이터 수정 용도에 페이지를 블록 내 가장 마지막 페이지에 대응시킨다. 이는 블록 운용 중 순차 쓰기 용도에 페이지와 데이터 수정 용도에 페이지의 비율이 가변적으로 변하기 때문이며 이에 대해서 3.3절에서 설명한다.

### 3.3 Offset Marker

오프셋을 이용하는 매핑기법의 최대 장점은 논리페이지 주소와 물리페이지 주소 간에 주소매핑을 생략할 수 있다는 점과 항상 연속하는 물리 페이지에 연속하는 데이터가 기록되어있으므로 순차 쓰기요청에 좋은 성능을 보인다는 점이다. 하지만 이러한 장점은 곧 단점으로 작용한다. 왜냐하면 블록 내 페이지에 접근할 수 있는 유일한 방법은 고정된 오프셋이며 이러한 고정된 오프셋을 이용한 블록 내 페이지 접근이 가능하기 위해서는 항상 순차적으로 데이터가 기록되어 있어야하기 때문이다. 이를 해결하기 위해 제안기법은 앞서 언급한 것처럼 다중 오프셋을 사용한다. 더불어 제안기법은 운용 중 용도별 페이지의 비율을 가변적으로 변화시킴으로 최대한 데이터 수정 요청을 동일 블록 내에서 처리하도록 한다. 이러한 방법을 통해 블록 공간 활용도를 최대한 높일 수 있으며 이는 곧 전체 블록 삭제 횟수에 감소를 의미한다. 제안기법은 블록 내 페이지의 가변적인 운용을 위해 오프셋 마커를 이용한다. 오프셋 마커는 현재 블록 내 페이지의 비율을 확인하기 위한 값이며 최조 블록 내 순차 쓰기용도에 페이지 중 가장 마지막 페이지의 오프셋 값을 뜻한다.

제안기법은 호스트로부터 발생하는 다양한 쓰기요청 패턴에 능동적으로 대체하기 위해 데이터 수정 요청에 대응하는 페이지의 개수를 고정하지 않으며 특정 블록 내 페이지에 대한 데이터 수정 요청이 빈번히 발생할 경우 이에 대응하는 데이터 수정 용도에 페이지의 비율을 증가시킨다. 제안기법은 최초 정한 데이터 수정 용도에 페이지 비율을 넘어서는 경우 블록 내 가장 마지막 순차 쓰기용도에 페이지부터 순차적으로 데이터 수정 용도에 대응하는 페이지로 전환되며 해당 블록에 블록 경계포인터 값은 갱신된다.

제안기법의 오프셋 마커는 순차 쓰기용도에 페이지와 데이터 수정 용도에 페이지의 경계를 나타내며 이를 통해 호스트로부터 발생하는 다양한 쓰기 패턴에 대해 효과적인 대응할 수 있다. 호스트로부터 발생하는 순차 쓰기요청 시 순차 쓰기용도에 페이지와 대응시킴으로 순차 오프셋을 이용한 주소매핑의 메모리 사용량을 줄일 수 있으며 또한 무작위 쓰기 요청 시 블록 내 정해진 페이지 비율을 가변적으로 조정함으로 최대한 해당 블록 내 페이지에 수정 데이터를 기록할 수 있으므로 블록 공간 활용도를 높일 수 있다. 또한 이를 통해 블록병합 시 최소한에 페이지 복사 및 블록 삭제가 가능하다.

제안기법에서 오프셋 마커는 쓰기, 읽기 요청에 모두 사용되며 해당 블록에 현재 오프셋 마커 값은 블록 내 별도에 페이지에 기록되어 유지된다.

### 3.4 Block metapages

앞서 제안기법의 특징에 대해 설명하였다. 이러한 특징을 가지는 제안기법을 운용하기 위해서는 필수적으로 특정 블록에 대한 역순 오프셋 테이블과 오프셋 마커 값 그리고 블록 내 유효 페이지 개수와 장치 초기화 및 복구를 위한 체크포인터 등의 운용을 위한 정보들을 기록 유지하여야한다.

제안기법은 이를 위해 최초 블록 내 가장 마지막 페이지를 블록 메타 페이지로 이용하여 해당 페이지는 데이터 쓰기요청과 읽기요청에 제외된다. 본 논문은 새로운 주소매핑 기법에 대한 것이므로 초기화 및 복구를 위한 체크포인터 설명은 생략하며 향후 논문에서 설명한다.

### 3.5 Writing of the proposed technique

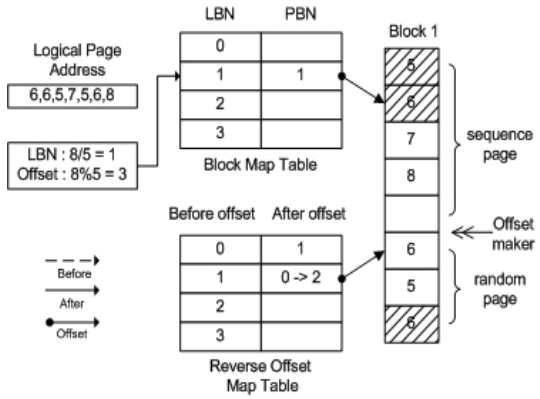


Fig. 3. Writing of the proposed technique

제안기법의 쓰기 요청 발생 시 동작과정에 대해 설명한다. Fig. 3.과 같이 현재 블록 내 페이지는 블록 메타 페이지를 제외한 8개이며 그중 5개는 순차 쓰기용도로 사용되고 나머지 3개는 데이터 수정 용도로 사용된다. 또한 현재 오프셋 마커의 값은 5이다. 제안기법은 호스트로부터 쓰기요청 발생 시 요청된 논리 페이지 주소를 블록 내 페이지의 개수로 나누어 몫을 논리 블록 주소로 사용하며 나머지 값은 해당 논리 블록의 페이지 순차 오프셋 값으로 사용한다.

Fig. 3.과 같이 최초 빈 블록을 대상으로 첫 번째 쓰기 요청 시 연산을 통해 논리 블록 주소 1과 오프셋 1값을 얻어 해당 위치에 데이터를 기록한다. 이어 두 번째 쓰기 요청 시 동일한 연산과정을 진행하며 논리 블록 주소 1과 오프셋 1값을 얻는다. 하지만 해당 블록 내 페이지는 이미 데이터가 기록된 상태이며 이때 제안기법은 해당 페이지를 무효화하고 블록 내 가장 마지막 페이지에 수정 요청 데이터를 기록하며 역순 오프셋 테이블에 무효 페이지와 유효페이지의 오프셋을 기록한다. 이는 오프셋을 이용하여 블록 내 페이지에 접근하는 기존 매핑기법과는 다른 동작과정으로 데이터 수정 요청 시 별도에 블록에 데이터를 기록할 필요 없이 동일 블록 내에 기록함으로 페이지 복사 및 블록 삭제 횟수 감소에 장점이 있다. 제안기법은 이후 동일한 과정으로 쓰기 요청을 처리한다.

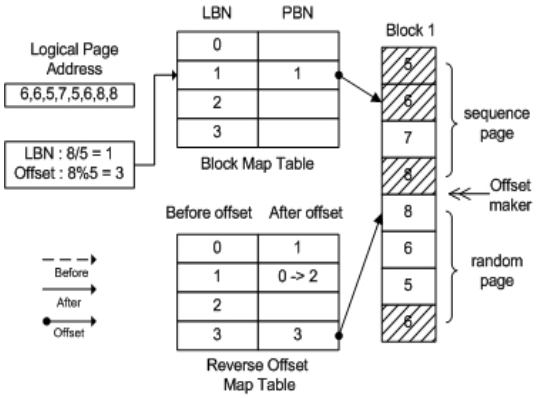


Fig. 4. Variable Utilization of Pages

Fig. 4.는 Fig. 3.에 마지막 쓰기요청 이후 8번째 쓰기 요청에 대한 것이다. Fig. 4.과 같이 쓰기 요청 된 논리 페이지의 주소 8에 대해 연산을 통해 블록 주소 1과 오프셋 주소 3을 얻는다. 하지만 해당 페이지는 이미 데이터가 기록된 페이지이며 또한 데이터 수정 용도에 페이지는 모두 사용된 상황이다. 제안기법은 이러한 경우 순차 쓰기 용도에 페이지와 데이터 수정 용도에 페이지 수를 가변적으로 조정한다.

이는 해당 블록에 오프셋 마커 값을 통해 가능하다. 현재 오프셋 마커 값에 해당하는 페이지가 비어있는 경우 해당 페이지를 데이터 수정 용도에 페이지로 전환시킨다. 블록 내 페이지를 가변적으로 활용함으로 호스트로부터 발생하는 다양한 쓰기 요청의 패턴에 최대한 능동적으로 페이지 이용할 수 있으며 이를 통해 블록 내 페이지를 최대한 활용함으로 블록 내 공간 활용도를 높여 블록 삭제 횟수의 감소시킬 수 있다.

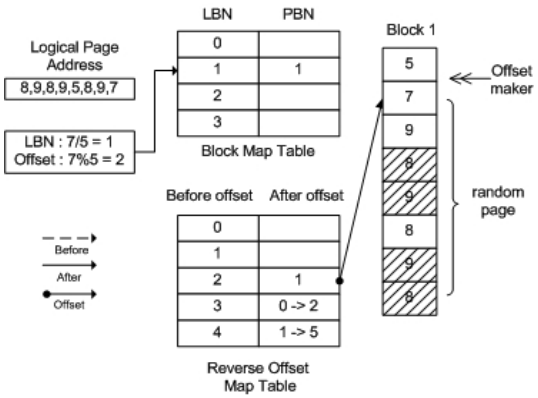


Fig. 5. Random write pattern

Fig. 5.의 경우 이전 쓰기요청과 달리 무작위쓰기 요청이 더 빈번히 발생하는 상황을 보여준다. Fig. 5.와 같이 마지막 쓰기 요청 시 논리 페이지의 주소 7에 대해 연산으로 블록 주소 1과 오프셋 주소 2를 얻는다. 하지만 해당 페이지는 이미 데이터 수정 용도의 페이지로 변경되었으며 이미 직전 쓰기 요청에 의해 데이터가 기록되어있다. 이는 오프셋 마커 값을 통해 알 수 있다. Fig. 5.와 같이 쓰기 요청된 논리 페이지의 순차오프셋 값이 현재 오프셋 마커 값은 2와 같거나 더 클 경우 이는 해당 순

차 오프셋 위치에 페이지가 이미 데이터 수정 용도의 페이지로 전환되었음을 의미하기 때문이다. 제안기법은 이러한 순차 쓰기 용도로 사용되는 페이지가 데이터 수정 용도에 페이지로 전환됨으로 인해 순차 오프셋을 적용할 수 없는 경우에 즉시 해당 페이지를 즉시 데이터 수정 용도에 페이지로 변경하며 역순 오프셋에 이전 오프셋과 이후 오프셋을 기록한다. Fig. 5.와 같은 경우는 블록 내 순차 쓰기 용도의 페이지와 데이터 수정 용도의 페이지가 많음을 의미하며 블록 내 페이지의 수가 점차 줄어들어 오프셋이 가까워지거나 역전되는 경우는 곧 블록병합 작업이 일어남을 의미한다. 제안기법의 블록병합은 3.3절에서 설명한다.

### 3.6 Reading of the proposed technique

제안기법의 읽기 요청 발생 시 동작과정에 대해 설명한다. 제안기법은 순차 쓰기 용도에 페이지 접근 시 순차 오프셋을 사용하며 데이터 수정 용도에 페이지 접근 시 역순 오프셋을 사용한다. 제안기법은 먼저 순차 오프셋을 이용하여 페이지에 접근하며 해당 페이지가 유효페이지일 경우 추가적으로 오프셋 마커 값과 비교해 오프셋 더 작은 경우 오프셋에 해당하는 페이지에 접근할 수 있으며 더 큰 경우 역순 오프셋을 사용한다. 해당 페이지가 무효페이지일 경우에는 역순 오프셋을 이용하여 역순오프셋 매핑 테이블에 해당 물리 페이지 오프셋에 매핑 된 오프셋에 해당하는 페이지에 접근할 수 있다.

Fig. 5.와 같이 읽기 요청 된 논리 페이지의 주소 7에 대해 연산을 통해 블록 주소 1과 오프셋 주소 2를 얻는다. 이후 해당 블록에 순차 오프셋 값에 유효페이지가 존재하지만 오프셋 마커 값이 더 큼으로 역순오프셋 테이블을 이용하여 데이터가 기록된 페이지 오프셋인 1값을 알 수 있다. 그림4의 경우는 해당 논리 페이지의 주소 7에 대한 오프셋 위치의 페이지는 페이지 가변 운용으로 인해 용도가 변경된 경우이며 이러한 경우를 위해 오프셋 마커 값과 비교하는 것이다.

Fig. 5.의 경우 블록 내 페이지 상황은 제안기법의 쓰기 동작 과정을 설명하기 위해 무작위 쓰기가 과도하게 적용된 측면이 있으며 Fig. 5.의 경우 블록 내 페이지가 모두 사용된 상황이며 이후 쓰기 요청발생 시 제안기법은 해당 블록을 병합작업 함으로 이러한 읽기 동작과정이 지속되지 않는다. 오히려 제안기법은 페이지 접근 시 오프셋 방식을 이용함으로 블록단위 매핑 기법에 장점인 순차 읽기 요청에 대한 장점을 유지한다.

### 3.7 Copy of the proposed technique

제안기법의 블록 병합 동작과정에 대해 설명한다.

제안기법은 블록 내 유효 페이지를 모두 사용한 블록을 대상으로 블록 복사를 실행하며 블록 복사는 기존 블록과 빈 블록 간 1대 1로 일어난다.

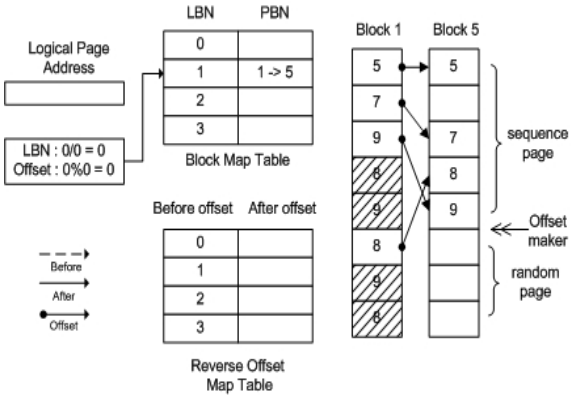


Fig. 6. Block copy of the proposed technique

블록 복사는 기존 블록의 순차 오프셋을 참고하여 순차 쓰기 용도의 페이지를 빈 블록에 복사하며 이후 역순 오프셋을 이용하여 데이터 수정 용도에 페이지를 빈 블록에 복사한다. 기존 블록은 해당 블록 메타 페이지에 기존 블록을 삭제 대기상태로 기록하되 이전 데이터를 복사한 블록의 오프셋 마커 값 역시 초기상태 값으로 해당 블록 메타페이지에 각각 기록된다.

제안 기법은 블록 내 페이지를 가변적으로 운용함으로 블록 내 페이지 활용도가 좋으며 항상 삭제 블록의 수는 1개이다. 또한 페이지 접근 시 오프셋을 이용함으로 기존 블록 내 순차 쓰기 용도에 페이지가 다수인 경우 블록 복사 시 추가 동작을 줄일 수 있다.

## IV. Experimental

이장에서는 실험을 통해 제안기법 적용 시 주소매핑에 필요한 메모리 사용량과 블록 삭제횟수를 분석함으로써 제안기법이 상대적으로 기존 매핑기법들 보다 효율적임을 검증한다.

### 4.1 Experimental environment and Trace file spec

실험에 사용된 시스템은 인텔 i5-6500, DDR3 1600MHz 4GB 메모리로 구성되며 Centos 7.3-1611 버전의 운영체제를 사용한다. 실험에서 시뮬레이션 한 낸드 플래시 메모리의 상세 성능은 아래 표1.과 같다.

Table 1. Parameter of Nand flash memory used for simulation.

| | | |
|---|---|---|
| Nand flash memory | Pages per block | 512 |
| | Page size | 16KB |
| | Block size | 8192KB |

실험은 낸드 플래시 메모리 사용 환경을 가상으로 조성한 트레이스 구동 시뮬레이션 Flashsim을 이용하였으며 실험에 사용된 무작위 쓰기 패턴의 워크로드와 순차쓰기 패턴의 워크로드의 생성은 리눅스의 blktrace를 이용하여 읽기/쓰기 명령을 추출하였다[12,13].

### 4.2 Mapping table total memory usage

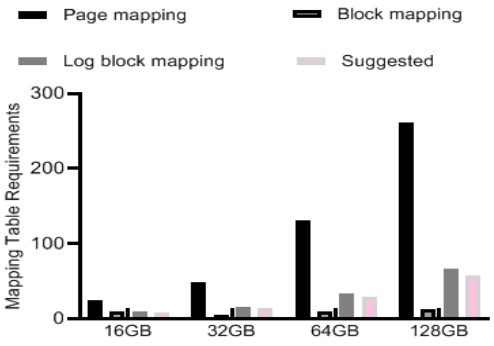


Fig. 7. Mapping capacity to memory table memory usage

Fig. 7.은 기존 매핑기법과 제안기법 적용 시 낸드 플래시 메모리의 용량 증가에 따른 주소 매핑테이블에 필요한 메모리양을 나타낸 것이다. 낸드 플래시 메모리의 용량이 증가할 수 록 페이지단위 매핑 기법의 경우 주소 매핑에 필요한 메모리 사용량은 기하급수적으로 증가함을 볼 수 있다. 이는 논리 페이지 주소와 물리 페이지 주소를 항상 1대1로 매핑하기 때문이며 블록단위 매핑 기법의 경우 낸드 플래시 메모리의 용량 증가와 상관없이 주소매핑 테이블에 필요한 메모리 사용량은 사실상 0에 가깝다. 이는 특정 논리페이지 주소를 연산하여 얻은 물리 블록주소와 오프셋 값을 이용하여 특정 블록 내 페이지에 접근하기 때문이다. 이러한 블록단위 매핑 기법은 SRAM 사용량이 가장 낮은 기법이지만 블록 내 공간 활용도 문제와 데이터 수정 요청 시 페이지 복사 및 블록단위 삭제의 빈번한 발생으로 현실적으로 사용할 수 없다. 하이브리드 기법의 경우 낸드 플래시 메모리 용량 증가에 따라 주소매핑 테이블의 메모리 사용량 역시 점차 증가함을 알 수 있다. 이는 데이터 수정 요청 시 수정 요청된 데이터를 별도에 로그 블록에 기록하기 때문이며 로그 블록의 경우 고정 오프셋 방식이 아님으로 반드시 주소매핑이 필요하기 때문이다. 제안기법의 경우 하이브리드 기법과 비교 시 주소매핑에 사용되는 메모리의 양은 크게 다르지 않다. 하지만 제안기법의 경우 하이브리드 기법과 다르게 데이터 블록과 대응하는 로그블록 간에 주소를 기억할 필요가 없으며 이는 부가적인 메모리사용을 줄여준다.

### 4.3 Number of deletions per block

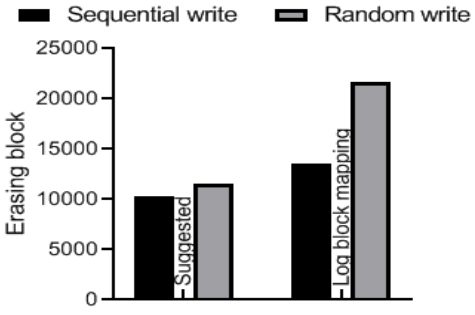


Fig. 8. Proposed technique and number of block deletion of log block

FTL 성능의 핵심은 블록 단위 삭제를 최대한 줄이는 것이며 이는 블록단위 삭제 요청 시 발생하는 블록 내 유효페이지의 복사 오버헤드를 최대한 줄이는 것이다. Fig. 8.에서처럼 제안기법의 블록 삭제 횟수는 하이브리드 기법과 비교했을 경우 더 낮다. 제안기법과 하이브리드 기법은 기본적으로 오프셋 개념을 사용하지만 특정 쓰기 패턴 중 무작위 쓰기 패턴이 지속되는 경우 하이브리드 기법은 데이터 블록 내 페이지의 활용도가 현저히 떨어지고 수정 요청 데이터를 별도에 로그 블록에 기록함으로 블록 삭제 시 해당 로그 블록을 추가로 삭제해야한다. 제안기법의 경우 블록 내 페이지를 순차 쓰기 용도에 페이지와 데이터 수정 용도에 페이지로 구분함으로 무작위 쓰기 패턴이 지속되는 경우에도 블록 내 페이지의 활용을 가변적으로 결정하여 이를 통해 블록 내 페이지의 공간 활용도를 최대한 활용할 수 있으며 이는 곧 블록 삭제 감소를 의미한다.

### 4.4 Number of deletions according to the number of pages per block

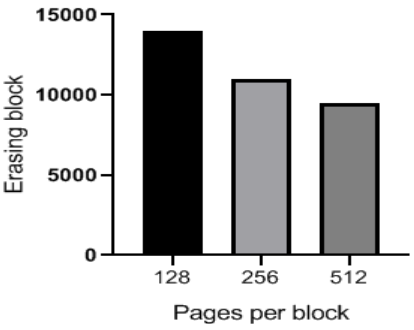


Fig. 9. Number of deletions according to the number of pages in the block

Fig. 9.는 제안기법 적용 시 블록 당 페이지의 개수에 따른 블록 삭제 횟수를 나타낸다. 제안기법의 경우 블록 내 페이지 개수가 증가할 수 록 더 많은 페이지를 가변운용할 수 있으며 이는 다양한 쓰기 패턴에 최적화 된 운용이 가능함을 의미한다. 특히 무작위 쓰기 패턴이 지속적으로 연속하여 발생하는 경우 블록 내 순차 데이터에 대응하는 페이지를 최대한 줄이고 수정 데이터를 기록하는 페이지를 최대치로 늘림으로 별도에 로그 블록을 사용하는 하이브리드 기법과 비교 시 블록 내 페이지 수 증가에 따른 블록 삭제 횟수 감소 효과를 극대화 할 수 있다. 이러한 제안기법의 무작위 쓰기 패턴에 좋음 성능을 보이는 것은 역순 오프셋을 이용하기 때문이며 이는 블록 내 페이지의 능동적인 운용을 가능하게 한다.

## V. Conclusions

본 논문에서는 점차 대용량화되는 낸드 플래시 메모리에 적용 가능한 새로운 매핑기법을 제안하였다. 본 논문은 호스트로부터 발생하는 순차쓰기요청과 무작위쓰기요청에 대응하는 페이지를 동일 블록 내에서 처리하기 위한 역순 오프셋과 운용 중 페이지의

용도를 가변적으로 변화시키기 위한 오프셋마커를 제안하였으며 이를 통해 무작위 쓰기패턴 발생 시 역순오프셋을 이용하여 동일 블록 내 페이지를 이용함으로써 블록 내 페이지 활용도를 최대한 높여 블록 삭제 횟수에 감소 효과를 보였다. 또한 오프셋마커를 이용하여 순차쓰기 패턴과 무작위쓰기 패턴 발생에 따라 블록 내 페이지 활용을 가변적으로 조정함으로써 블록 삭제 횟수에 감소 효과를 보였다. 결론적으로 제안기법은 쓰기요청 시 발생하는 다양한 쓰기요청에 대한 최적화된 페이지 운용이 가능하며 별도에 추가적인 블록 없이 쓰기요청을 처리함으로써 FTL 성능에 중요한 블록 단위 삭제를 최소화하였다. 향후 이러한 제안기법을 바탕으로 장치 초기화 시간을 줄이며 메모리 사용량을 좀 더 줄일 수 있는 연구를 지속할 예정이다.

## REFERENCES


[1] Gartner, https://www.gartner.com/en/newsroom/press-releases/2017-10-17-gartner-says-worldwide-device-shipments-will-increase-2-percent-in-2018

[2] Sang Oh Park, and Sung Jo Kim, "An efficient file system for large-capacity storage with multiple NAND flash memories," IEEE International Conference on Consumer Electronics (ICCE), March 2011.

[3] Yongju Lee, Hyunwoo Kim, Huijeong Kim, Taeyeong Huh, Sanghyuk Jung, and Yong Ho Song, “Adaptive Mapping Information Management Scheme for High Performance Large Sale Flash Memory Storages,” Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 3, March 2013

[4] Novotný R, Kadlec J and Kuchta R, "NAND Flash Memory Organization and Operations," JITSE Volume 5 Issue 1, January 2015.

[5] Yoshiki Takai, Mamoru Fukuchi, Reika Kinoshita, Chihiro Matsui, Ken Takeuchi, "Analysis on Heterogeneous SSD Configuration with Quadruple-Level Cell (QLC) NAND Flash Memory," IEEE 11th International Memory Workshop (IMW), June 2019

[6] Yoshiki Takai, Mamoru Fukuchi, Reika Kinoshita, Chihiro Matsui, Ken Takeuchi, “Analysis on Heterogeneous SSD Configuration with Quadruple-Level Cell (QLC) NAND Flash Memory,” 2019 IEEE 11th International Memory Workshop (IMW), June 2019.

[7] Feng Chen, Tong Zhang, Xiaodong Zhang, "Software Support Inside and Outside Solid-State Devices for High Performance and High Efficiency," Proceedings of the IEEE, Vol. 105, Issue 9, Sept 2017.

[8] A. Gupta, Y. Kim, and B. Urgaonkar, "DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-Level Address Mappings," Proc. of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems, Mar. 2009.

[9] D. Ma, J. Feng, and G. Li, "LazyFTL: A Page-level Flash Translation Layer Optimized for NAND Flash Memory," Proc. of the 2011 ACM SIGMOD International Conference on Management of Data, Jun. 2011.

[10] Jiang, and Song, "S-ftl: An efficient address translation for flash memory by exploiting spatial locality," Mass Storage Systems and Technologies (MSST), IEEE 27th Symposium on. IEEE, July 2011.

[11] J. Kim, J. M. Kim, S. Noh, S. L. Min, and Y. Cho, "A space-efficient flash translation layer for compact flash systems," Consumer Electronics, IEEE Transactions on, vol.48, pp.366-375, May 2002.

[12] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, and H.-J. Song, "A log buffer-based flash translation layer using fully-associative sector translation," ACM Trans. Embed. Comput. Syst. vol.6, no.3, p.18, July 2007.

[13] Dawoon Jung, Jeong-UK Kang, Heeseung Jo, Jin-Soo Kim, Joonwon Lee, "Superblock FTL: A superblock-based flash translation layer with a hybrid address translation scheme." ACM Transactions on Embedded Computing Systems (TECS) TECS Homepage archive Volume 9 Issue 4, March 2010.

[14] Youngjae Kim, Brendan Tauras, Aayush Gupta, Bhuvan Urgaonkar, "FlashSim: A Simulator for NAND Flash-Based Solid-State Drives," IEEE Xplore 09, October 2009.

[15] Giuliano Casale, Stephan Kraft, Diwakar Krishnamurthy, "A Model of Storage I/O Performance Interference in Virtualized Systems," IEEE Xplore 25, July 2011.


## Authors


Seung-Woo Lee received the B.S. degree from Kyungil University and M.S. degree from Kyungpook National University, South Korea, in 2010 and 2013, respectively. He is currently enrolled for PhD degree in degital media lab. His current interests include embedded and flash memory based storage system.



Kwan-Woo Ryu received the B.S. degree from Kyungpook National University. He received M.S. degree from KAIST, and PhD degree in University of MARYLAND, USA, in 1980, 1982, 1990, respectively. He is currently an professor in school of computer science and engineering of Kyungpook National University. His research interests include multi-paradigm algorithm, parallel computing.