2013년 5월 전자공학회 논문지 제 50 권 제 5 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 5, May 2013 http://dx.doi.org/10.5573/ieek.2013.50.5.094

논문 2012-50-5-11

# 논리 블록의 접근경향을 활용한 이종 낸드 플래시 기반 저장장치를 위한 Flash Translation Layer

## ( Flash Translation Layer for Heterogeneous NAND Flash-based Storage Devices Based on Access Patterns of Logical Blocks )

방 관 후*, 박 상 훈*, 이 혁 준**, 정 의 영*

( Kwanhu Bang, Sang-Hoon Park, Hyuk-Jun Lee©, and Eui-Young Chung )

요 약

낸드 플래시 메모리에 기반 한 저장장치는 이미 여러 분야에서 기존 디스크 기반 저장장치를 대체하며 거대한 규모의 시장을 확보하고 있다. 이 중 집적도는 높지만 성능과 신뢰성이 상대적으로 낮은 multi-level cell (MLC) 낸드 플래시 메모리와 반대의 특성을 지니는 single-level cell (SLC) 낸드 플래시 메모리를 혼용하여 서로의 장점만을 얻고자 하는 이종 낸드 플래시 기반 저장장치에 관한 연구 또한 활발하게 이루어지고 있다. 이종 낸드 플래시 기반 저장장치에서는 SLC에 기록된 데이터가 MLC로 옮겨질 경우에 발생하는 마이그레이션 오버헤드와, 상대적으로 적은 용량의 SLC 내부에서 발생하는 가비지 컬렉션 오버헤드가 전체 저장장치의 성능을 악화시키는 문제가 있는데, 본 논문에서는 이를 완화하고자 논리 블록의 접근경향을 활용하여 SLC를 효율적으로 활용하는 이종 낸드 플래시 기반 저장장치용 flash translation layer (FTL)을 제안하고자 한다. 제안하는 FTL 은 논리 블록들의 접근 경향을 파악하여 SLC에 기록되었을 시 성능 향상을 가져올 것이라고 기대되는 논리 블록들만을 선별하여 SLC에 기록하게 된다. 실험 결과 본 논문에서 제안하는 FTL을 사용한 이종 낸드 플래시 기반 저장장치는 기존 FTL 대비 전체 실행 시간에서 35% 향상된 성능을 보여주었다.

Abstract

The market for NAND flash-based storage devices has grown significantly as they rapidly replace traditional disk-based storage devices. Heterogeneous NAND flash-based storage devices using both multi-level cell (MLC) and single-level cell (SLC) NAND flash memories are also actively researched since both types of memories complement each other. Heterogeneous NAND flash-based storage devices suffer from the overheads incurred by migration from SLC to MLC and garbage collection of SLC. This paper proposes a new flash translation layer (FTL) for heterogeneous NAND flash-based storage devices to reduce the overheads by utilizing SLC efficiently. The proposed FTL analyzes the access patterns of logical blocks and selects and stores only logical blocks expected to bring performance improvement in SLC. The experimental results show that the total execution time of heterogeneous NAND flash-based storage devices with our proposed FTL scheme is 35% shorter than that with the previously proposed best FTL scheme.

**Keywords** : 낸드 플래시 메모리, single-level cell, multi-level cell, solid-state drive, flash translation layer

* 정회원, 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Yonsei University)
** 정회원, 서강대학교 컴퓨터공학과
(Department of Computer Science and Engineering, Sogang University)
※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (2012-0007181, 2012-047670, 2011-0023798)
© Corresponding Author(E-mail:hyukjunl@sogang.ac.kr)
접수일자: 2013년2월12일, 수정완료일: 2013년4월18일

## Ⅰ. 서 론

낸드 플래시 메모리는 오래전부터 다양한 비휘발성 메모리들을 대체해 왔고, 낸드 플래시 메모리의 주 활용처였던 모바일 기기의 폭발적인 성장을 통해 이미 DRAM보다 큰 시장 규모를 가지는 메모리이다. 1개의 cell이 1개의 트랜지스터로 이루어져 현존하는 메모리

중 가장 높은 집적도를 가지고 있는 낸드 플래시 메모리의 집적도를 한 단계 진일보시켰던 것은 multi-level cell (MLC) 낸드 플래시 메모리의 등장이었다. 기존 한 개의 cell에 한 개의 bit만을 저장하던 single-level cell (SLC) 낸드 플래시 메모리와 같은 cell 구조를 가지면서도 한 개의 cell에 두 bit 이상의 데이터를 저장할 수 있게 된 MLC는 물리적인 변화 없이도 손쉽게 낸드 플래시 메모리의 집적도를 2배 가까이 증가시킬 수 있었다.

하지만 MLC의 높은 집적도를 가능하게 한 컨트롤 방법은 SLC 대비 느린 접근 속도라는 단점을 가지게 하였고, 이에 따라 SLC와 MLC를 동시에 사용하는 이종 낸드 플래시 기반 저장장치의 구조와 관리 방법들이 제안되었다[3~6]. 이 방법들은 주 메모리 시스템의 느린 속도를 보완하기 위해 사용되는 SRAM 캐시와 마찬가지로, MLC의 느린 속도를 보완하기 위하여 SLC를 캐시와 유사하게 사용하는 방법을 택하고 있다. 먼저 이종 낸드 플래시 기반 저장장치를 위한 파일시스템[3]의 경우 관리 방법 자체의 효율성과 관계없이 기존 파일 시스템과의 호환성을 유지하고자 개발된 flash translation layer (FTL)와는 서로 다른 특징을 보여 여러 시스템에 적용하기는 어렵다.

또한 마이그레이션 오버헤드를 줄이기 위해 독립적으로 동작할 수 있는 다수의 이종 낸드 플래시 메모리를 활용한 방법[4] 역시 반드시 독립적인 다수의 이종 낸드 플래시 메모리를 포함해야 한다는 점에서 유연성이 떨어진다.

본 논문에서 제안하고자 하는 방식과 유사하게 이종 낸드 플래시를 위해 제안된 FTL[5]의 경우는 다양한 이종 낸드 플래시 구성에 무리 없이 대응할 수 있다. 하지만 SLC의 활용을 위해 다수의 가비지 컬렉션을 거쳐서야 비로소 MLC로 데이터가 옮겨지도록 설계되어 있어, 비록 해당 가비지 컬렉션 횟수를 동적으로 적용한다고 하더라도, 일부 경우에 바로 MLC에 데이터를 기록하는 것 보다 큰 오버헤드를 발생시킬 수 있다.

따라서 본 논문에서는 SLC와 MLC의 특성을 기반으로, SLC를 활용할 시 성능상의 이점을 가져올 수 있는 데이터만을 선택적으로 SLC에 기록하도록 하여 이종 낸드 플래시 저장장치의 성능을 극대화 하는 FTL 기반의 관리 방법을 제안하고자 한다. 실험결과에 따르면, 본 논문에서 제안하는 방법은 전체 실행시간 측면에서 MLC만을 사용한 단일 낸드 플래시 기반 저장장치 대비 56%, 이전에 제안된 FTL[5] 대비 35%의 성능향상을 보임을 알 수 있었다.

## Ⅱ. 본 론

### 1. SLC와 MLC의 특성과 마이그레이션 오버헤드

MLC는 SLC와 같은 구조의 cell에 1-bit가 아닌 2-bit를 저장하게 된다. 이에 따라 데이터를 저장하는 프로그램 동작을 진행하는 과정에서 보다 세세하게 cell의 상태를 정해야 하고 이는 보다 긴 동작시간을 요구한다. 표 1은 SLC와 MLC의 동작시간을 간략하게 요약하고 있으며, MLC는 SLC 대비 프로그램 시 6배 이상, 읽기 시 3배 정도의 성능차이가 있음을 알 수 있다.

하지만 SLC의 빠른 속도를 사용하고자 모든 데이터를 우선적으로 SLC에 저장하는 경우 추후에 추가적인 마이그레이션 오버헤드를 발생시킬 수 있다. 그림 1은 같은 크기의 데이터를 각각 이종 낸드 플래시 기반 저장장치와 단일 낸드 플래시 기반 저장장치에 기록하는 상황을 가정하고 있다. 이종 낸드 플래시 기반 저장장치는 t1에서 데이터를 SLC에 기록하여 MLC만 사용한 저장장치 대비 짧은 시간에 기록을 마칠 수 있었다. 하지만, SLC의 상대적으로 낮은 집적도로 인해 저장장치가 제공하는 용량의 매우 일부분만을 담당할 수 있기에, 해당 데이터는 최종적으로는 MLC로 옮겨져야만 한다. 이에 따라 t2에서 이종 낸드 플래시 기반 저장장치는 데이터를 SLC에서 MLC로 마이그레이션 하여 MLC에 저장된다. 결론적으로 MLC에 옮겨질 때까지, 이종

표 1. SLC와 MLC 낸드 플래시 메모리의 성능 비교[1~2]

Table 1. Performance comparison of SLC and MLC NAND flash memories.[1~2]

| 항목 | SLC | MLC |
|---|---|---|
| 프로그램 ($t_{PROG}$) | 200us | 1350us |
| 읽기 ($t_R$) | 20us | 60us |
| 지우기 ($t_{BER}$) | 2ms | 3ms |
| 페이지 크기 | 4KB | 4KB |

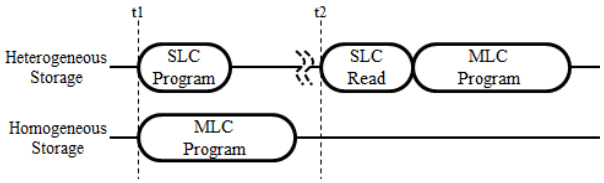



그림 1. 이종 낸드 플래시 기반 저장장치와 단일 낸드 플래시 기반 저장장치의 동작 방법

Fig. 1. Timing diagram of heterogeneous and homogeneous NAND flash-based storage devices.

낸드 플래시 기반 저장장치는 MLC 저장장치 대비 각각 한 번씩의 SLC 프로그램과 읽기 동작을 추가로 필요로 하게 되는 것이다. 또한 마이그레이션 전까지 (그림 1의 t1~t2 사이), SLC 내부적으로 가비지 컬렉션이 발생하는 경우 그 오버헤드는 증가하게 된다. 이는, 데이터가 SLC에 존재하는 동안 마이그레이션 및 SLC 가비지 컬렉션에 의해 발생할 수 있는 오버헤드 보다 큰 성능상의 이점을 가져오는 경우에만 SLC를 활용한 효과를 볼 수 있음을 의미한다.

## 2. 논리블록의 접근경향을 활용한 데이터 선별

앞선 문단에서 논의된 바와 같이, SLC에 기록되는 데이터는 후에 마이그레이션과 가비지 컬렉션 오버헤드를 발생시키게 되므로 세밀한 기준에 의해 선별되어야 할 필요가 있다. 이에 따라 제안하는 FTL은 쓰기 요청이 주어졌을 때 그림 2와 같은 방식으로 데이터를 선별하여 SLC에 기록한다.

이 방식은 어떤 쓰기 요청이 주어지면, 먼저 요청의 논리 섹터 주소로부터 논리 블록 주소를 계산한다. 여기서 논리 블록이란, 저장장치에 사용된 SLC 낸드 플래시 메모리의 블록과 같은 크기의 인접한 논리 데이터 그룹을 의미한다. 이 주소에 따라 논리 블록 접근 패턴 테이블 (LAPT)을 인덱싱 하여 논리 블록에 해당하는 값을 얻어내게 된다. LAPT의 모든 요소들은 값 이외에 포인터를 사용하여 값의 내림차순으로 정렬된 상태로 유지되는데, 바뀌거나 추가한 값을 삽입할 때 $O(n)$의 복잡도로 정렬하여 삽입하도록 구현되었다. FTL은 이 값의 순위를 확인하여 순위가 SLC 전체 블록 개수보다 작을 경우에만 데이터를 SLC로 보내고, 그렇지 않은 경우에는 MLC로 보내게 된다. 즉 값이

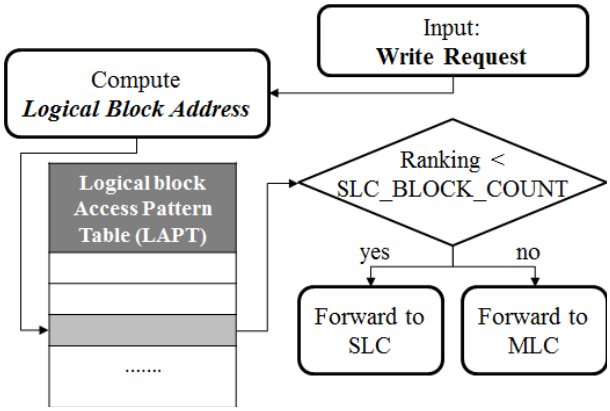



그림 2. 제안하는 SLC에 기록될 데이터 선별 방법
Fig. 2. Proposed selection method of data written to SLC.

큰 경우에는 SLC로 보내도 마이그레이션 오버헤드를 상쇄시킬 수 있을 것이라고 판단하는 것이다. LAPT의 각 요소들의 업데이트 방법에 대해서는 추후에 자세히 설명할 것이다.

위와 같은 방식으로 SLC에 데이터를 기록할 경우, 임의의 시각에서 제안하는 방법이 SLC에 기록할 수 있다고 판단하는 블록의 목록은 동적으로 변화하며 그 개수는 SLC 전체 블록의 개수를 넘어서지 않는다. 이는 SLC에 기록되는 데이터의 전체 양을 줄이되 반드시 필요한 데이터만 기록하도록 하여 마이그레이션 및 가비지 컬렉션 오버헤드를 줄일 수 있도록 한다.

## 3. LAPT의 관리 방법

그림 2와 같은 방법의 데이터 선별 방식은 결국 LAPT의 값들을 어떻게 업데이트 및 관리하느냐에 따라 그 효율성이 결정된다. 먼저 LAPT는 각 논리 블록별로 값을 지니고 있도록 설계되었으며 논리 블록 $BA$에 대한 값을 $LAPT(BA)$라 칭한다. 각 요소 단위의 크기는 주소 변환 테이블과 독립적으로 사용 가능하므로 다양한 수준의 주소 변환 테이블에 대해 일반성을 가지게 된다.

LAPT 요소의 단위를 블록으로 설정한 것은 크게 두 가지 이유 때문이며, 첫 번째는 메모리 사용량이다. LAPT의 한 요소의 크기가 작다고 가정하더라도 고용량의 저장장치의 경우 LAPT를 위하여 많은 메모리 공간이 필요하게 되며 임베디드 시스템인 낸드 플래시 기반 저장장치에 메모리 공간 추가는 부담으로 작용하기 때문이다.

두 번째로는 LAPT는 temporal locality 보다는 spatial locality를 판단하기 위해 사용하기 때문이다. 즉 어떤 한 주소의 특성을 판단하고 그 주변의 주소공간은 비슷한 특징을 가질 것으로 예측하는 것이다. 이러한 의도가 앞서 설명한 메모리 사용량과 맞물려 논리 블록을 LAPT의 관리 단위로 정할 수 있었다.

LAPT 각 요소는 값이 클수록 해당 블록은 자주 업데이트 되는 블록이며, 값이 작을수록 업데이트보다는 많이 읽히는 블록이라는 의미를 지니도록 정의되었다. 또한 자주 업데이트 되는 블록이 높은 순위를 가지도록 설계되어 SLC에 기록되게 된다. 이러한 설계는 MLC의 읽기 동작속도가 SLC보다 느려 자주 읽히는 데이터의 경우 SLC의 성능을 활용할 수 있음에도 불구하고 업데이트 시 얻을 수 있는 이점보다는 그 절대값이 작다는 점에 착안하였다. 결론적으로 한정적인 SLC의 공간에

읽기보다는 효과가 큰 업데이트를 배치하여 MLC 대신 SLC가 처리하게 하는 것이 효율적이라고 판단한 것이다. 만약 읽기와 쓰기가 교차로 반복되는 패턴이 주어지는 경우라고 하더라도 LAPT는 블록단위로 관리되므로 블록에 속한 다수의 페이지들의 전체 경향을 나타내게 되어 비교적 올바른 순위를 나타낼 수 있을 것이라고 예측하였다. 이러한 독특한 패턴에 경우 추후 개선을 통해 정확도를 향상시키는 것을 목표로 하였다.

LAPT 각 요소의 값이 위와 같은 의미를 지니게 하기 위해서 LAPT의 각 논리 블록별 항목 $LAPT(BA)$ 값은 매 요청이 처리된 후에 업데이트 된다. 만약 어떤 요청이 논리 블록 주소 $BA$에 대한 업데이트를 일으킨 경우에는

$$LAPT(BA) = \min(LAPT(BA) + 1, 30) \quad (1)$$

와 같은 식으로, 값을 증가시키되, 30 이하의 값을 가지도록 한다. 또한 어떤 요청이 해당 논리 블록 주소에 대해 읽기를 요청한 경우 위와는 반대로

$$LAPT(BA) = \max(LAPT(BA) - 1, -31) \quad (2)$$

와 같은 방식으로 값을 감소시키되, 최소값인 -31 미만의 값을 가지지 않도록 한다. 이에 따라, 각 값은 6-bit의 공간만을 가지고 표현할 수 있게 되며 exploration 결과 이 이상의 dynamic range는 성능향상을 가져오지 않았다.

## 4. SLC와 MLC 낸드 플래시 메모리 관리 방법

그림 2와 식 (1), (2)를 바탕으로 SLC 또는 MLC에 기록한 데이터는 각 낸드 플래시 메모리의 관리 방법에 따라 내부적으로 관리된다. 낸드 플래시 메모리의 관리 방법은 크게 주소 변환 테이블과 가비지 컬렉션 방법으로 나눌 수 있다. 기본적으로 두 종류의 낸드 플래시 메모리는 모두 페이지 수준의 주소 변환 테이블로 관리한다. 페이지 수준의 주소 변환 테이블은 그 크기가 크지만 유연한 관리가 가능하여 성능측면에서 이득이 있다. 이전에 제안된 FTL[5]은 SLC는 페이지 수준으로, MLC는 유닛 (여러개의 논리 블록을 묶은 그룹) 수준으로 관리하고 있으며 이 방법은 보다 이전에 제안된 단일 낸드 플래시 용 FTL[7]과 유사하다. 이 주소 변환 테이블 방식의 경우 페이지 수준과 유사한 성능을 목표로 하며 테이블의 크기 또한 페이지 수준과 유사하다. 다만, 캐싱 및 여러 단계의 인덱싱을 통해 메모리 오버헤드를 줄이게 되는데, 그에 따라 주소 테이블을 메모리에 적재하는 데까지의 오버헤드를 감수해야만 한다. 하지만 주소 변환 테이블을 캐싱하는 방법[8]을 사용한다면 작은 성능 오버헤드로 메모리 오버헤드를 크게 감소시킬 수 있음을 확인할 수 있다. 다만 본 논문에서는 이와 같은 테이블 캐싱 기법은 도입하지 않았으며 모든 주소 변환 테이블이 메모리에 적재되어 있다고 가정하였는데, 후에 실험을 진행할 시 비교하려는 다른 FTL의 경우에도 인덱싱 및 캐싱 없이 모든 주소 변환 테이블을 메모리에 적재하여 실험함으로써 공정한 실험결과를 보일 수 있도록 하였다.

표 2. 사용된 trace의 특징
Table 2. Characteristics of used traces.

| 이름 | 요청 횟수 | 쓰기 요청의 비율 | 평균 쓰기 섹터 개수 |
|---|---|---|---|
| Exchange[9] | 204571 | 79% | 24.83 |
| Photoshop | 23498 | 29% | 414.21 |
| Random | 33740 | 99% | 8.28 |
| General | 300000 | 23% | 27.67 |
| TPC_C[9] | 348644 | 37% | 18.28 |

가비지 컬렉션의 경우 SLC는 first-in, first-out (FIFO) 방식으로 설계되어, SLC의 사용량이 어느 수준을 넘어가 때 가장 먼저 사용된 블록이 가장 먼저 가비지 컬렉션의 대상으로 선택되도록 설계되었으며, MLC는 greedy 정책을 사용하도록 하였다. 고안한 LAPT와의 연동성이 없기 때문에 LAPT를 통해 SLC에 쓰이도록 판단되는 논리 블록 또한 MLC로 마이그레이션 될 수 있음을 알 수 있다. 이는 FIFO로 동작하는 SLC는 데이터의 temporal locality를 판단하는 기준이며 LAPT는 앞서 언급한 바와 같이 블록 단위의 spatial locality를 판단하는 기준으로 작용할 것이라고 가정했기 때문이다.

가비지 컬렉션이 발생하면 SLC에서 대상으로 선택된 블록에 존재하는 모든 유효한 페이지는 MLC로 마이그레이션 되고, 해당 블록은 지워져 새롭게 사용될 수 있게 한다. 이 과정에서 일부 페이지를 다시 SLC에 넣지 않는 이유는, 앞서 설명한 바와 같이 이미 temporal locality를 잃었다고 판단하여 앞으로 발생할 수 있는 마이그레이션 오버헤드를 줄이기 위함이다. MLC의 경우 가비지 컬렉션은 내부적으로만 일어나며, SLC로의 마이그레이션은 고려하지 않도록 설계되었다.

## Ⅲ. 실 험

### 1. 실험 환경

실험을 위하여 낸드 플래시 메모리의 성능을 모델링하고 FTL을 포함하는 trace-driven 시뮬레이터를 C언어로 작성하였다. 이 시뮬레이터는 입력으로 trace와 하드웨어 구성 등을 받아들이고 발생한 낸드 플래시 메모리의 동작을 출력해준다. 이에 포함된 FTL은 소프트웨어 시뮬레이션용으로 작성된 것이며 다양한 하드웨어 구성에 대응할 수는 있으나 실제 하드웨어에 적용하기 위해서는 일부 모듈의 수정 및 하드웨어와의 인터페이스를 갖추는 등의 추가적인 포팅 작업을 필요로 한다.

실험을 위해 사용한 낸드 플래시의 사양은 앞선 표 1과 동일하되, 모든 낸드 플래시는 4-plane을 사용하여 multi-plane 동작이 가능하도록 구성하였다. 또한 MLC는 2-way로 구성하여 interleaving을 통한 대역폭 향상 효과를 볼 수 있도록 구성하였다. 또한 MLC/SLC의 블록 당 페이지 개수는 128개와 64개로 설정[1~2] 하였으며, MLC 블록의 개수는 1024, SLC 블록의 개수는 64개가 기본으로 설정되어 있으나, SLC 블록 개수의 따른 성능 변화를 살펴보기 위하여 이 설정은 유동적으로 바뀔 수 있도록 구현되었다.

표 2는 실험에 사용된 trace 들의 특징를 나타내고 있다. Exchange 와 TPC_C를 제외한 3개의 trace는 Diskmon[10]을 활용하여 직접 Windows 기반 PC에서 추출한 것이며, Photoshop은 그래픽 관련 소프트웨어로 이미지를 편집과 저장을 반복하였는데, 이미지를 읽는 횟수가 많고 수정된 부분만 저장되는 특성으로 읽기 요청의 횟수가 많으며 큰 이미지의 크기로 인하여 연속적인 요청이 많음을 알 수 있다. Random은 디스크 벤치마크 프로그램으로 임의 쓰기 성능 테스트를 하였기에 평균적으로 길이가 짧은 쓰기 요청들이 반복적으로 들어오는 trace이다. General은 약 8시간 동안의 일반적인 PC 활용을 trace로 추출하여 Photoshop과 Random의 중간정도의 특성을 보인다. Exchange[9] trace는 서버의 활동을 기록한 것이며 TPC_C[9] 디스크 성능 검증을 위한 trace로 알려져 있다.

마지막으로 FTL 간의 성능 비교를 위하여 ComboFTL[5]과 SLC를 사용하지 않은 MLC_homo 방법, MLC_homo 방법에서 낸드 플래시 종류를 MLC에서 SLC로 바꾼 SLC_homo 방법을 구현하여 이용하였다. ComboFTL을 제외한 모든 방법은 페이지 수준 주소 변환 테이블을 사용하도록 구현되었으며, 제안되는 방법에서 사용하는 LAPT로 인한 메모리 오버헤드는 약 16KB이며, 이는 페이지 수준 주소 변환 테이블 대비 3% 정도의 크기이다.

### 2. 실험 결과

#### 가. 전체 실행 시간

그림 3은 모든 trace를 처리하는데 걸리는 시간을 MLC_homo에 표준화하여 보여준다. 평균적으로 MLC_homo 대비하여 ComboFTL이 성능을 30% 정도 개선한 데 반해 제안하는 방법 (Proposed) 은 56% 정도의 개선을 보여주고 있다.

특히 Photoshop trace에 대해서 ComboFTL은

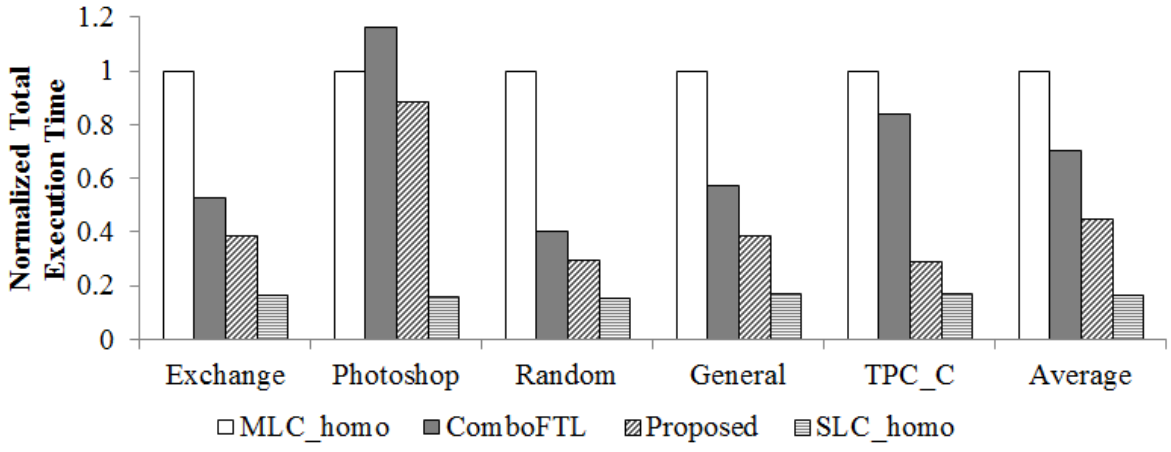



그림 3. MLC_homo 방식에 표준화된 전체 실행 시간
Fig. 3. Total execution time normalized to MLC_homo.

MLC_homo 보다도 좋지 않은 성능을 보여주고 있는데, 이는 SLC로 들어간 데이터가 SLC에서 지속적으로 가비지 컬렉션 되는 과정에서 발생한 오버헤드 때문인 것으로 분석되었다. 제안된 방법의 경우, ComboFTL과 MLC 접근 횟수는 비슷하였지만, SLC의 가비지 컬렉션 횟수를 절반 이하로 줄이는데 성공하여 MLC_homo 보다 좋은 성능을 달성하는데 성공하였다. 이는 SLC에 기록할 데이터를 보다 정확하게 선별했다고 해석할 수 있다. 또한 TPC_C trace와 Random trace의 경우 ComboFTL은 상이한 성능 향상 정도를 보인 반면, 제안된 방법의 경우 유사한 정도의 성능 향상을 보인 것을 확인할 수 있다. 이 또한 앞 문단에서 분석한 것과 유사하게 ComboFTL은 MLC 가비지 컬렉션을 일부 감소시켰지만 SLC 가비지 컬렉션 오버헤드를 줄이지 못했기 때문으로 분석된다.

### 나. SLC 크기에 따른 성능 변화 분석

이종 낸드 플래시 기반 저장장치에서 SLC의 크기는 가격적인 측면에서 매우 중요한 요소이다. 따라서 그림 4와 그림 5를 통하여 제안된 방법이 다양한 숫자의 SLC 블록과 사용되었을 시의 성능을 분석해 보았다. 먼저, 그림 4는 ComboFTL이 MLC_homo 보다 나쁜 성능을 보였던 Photoshop trace를 활용한 실험이다. 모든 실험결과는 MLC_homo에 같은 trace를 사용했을 때의 성능에 표준화되었다.

제안된 방법은 SLC 블록 개수에 큰 영향을 받지 않는 것을 확인할 수 있지만, ComboFTL의 경우 64개 이후부터는 빠른 속도로 성능이 개선되는 것을 확인할 수 있고, 256개의 블록을 사용한 경우에는 Proposed를 역

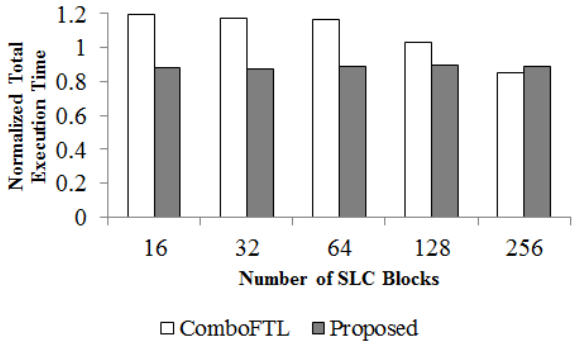



그림 4. Photoshop trace를 이용했을 때 SLC 블록 개수에 따른 표준화된 전체 실행 시간
Fig. 4. Normalized total execution time according to various number of SLC blocks with Photoshop trace.

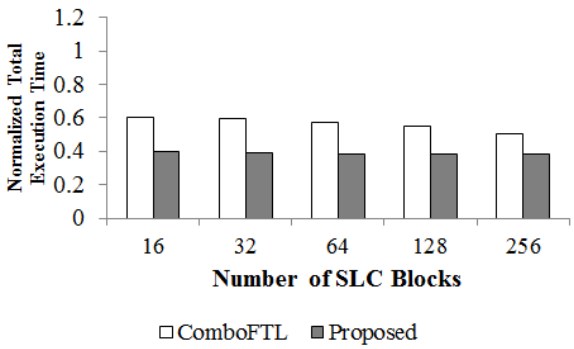



그림 5. General trace를 이용했을 때 SLC 블록 개수에 따른 표준화된 전체 실행 시간
Fig. 5. Normalized total execution time according to various number of SLC blocks with General trace.

전한 것을 확인할 수 있다. 분석 결과에 따르면 ComboFTL은 MLC 및 SLC의 가비지 컬렉션 횟수가 모두 줄어들었으나, Proposed의 경우 SLC 의 가비지 컬렉션 횟수가 늘어났기 때문인 것으로 분석되었다. SLC의 크기가 늘어남에 따라, SLC에 데이터를 많은 시간 보관하고 있는 ComboFTL이 이득을 본 예라고 할 수 있을 것이다.

하지만 그림 5에서 확인할 수 있듯, 모든 trace에서 그러한 현상이 발생하는 것은 아니다. General trace로 실험한 결과를 보여주는 그림 5에 따르면, SLC 용량이 큰 폭으로 늘어도 ComboFTL은 Proposed의 성능을 넘어서지 못했음을 알 수 있다. 이러한 경향은 Photoshop trace를 제외한 모든 trace에서 살펴볼 수 있으며, 크게 두 가지 원인으로 분석된다. 첫 번째는, 유닛 기반 주소 변환 테이블의 비효율성으로 인한 잦은 MLC 가비지 컬렉션의 발생이다. SLC 블록 개수에 따라 다소 차이는 있으나, ComboFTL은 Proposed 보다 많은 MLC 가비지 컬렉션을 발생시켰다. 두 번째로는, SLC에 기록하도록 선별된 데이터의 상이함이다. SLC 가비지 컬렉션은 Proposed 방법이 ComboFTL 보다 많은 것으로 나타났고, 이는 곧 Proposed 방법이 ComboFTL 보다 많은 데이터를 SLC에 기록했음을 의미한다. 하지만 그에 따라, SLC에서 업데이트 되어 마이그레이션 오버헤드를 상쇄시키고 성능향상을 가져온 SLC의 이점을 ComboFTL 보다 많이 취한 것으로 분석된다.

## Ⅳ. 결 론

본 논문은 이종 낸드 플래시 메모리 기반 저장장치를

위한 FTL을 제안했으며, 이 FTL은 기존 이종 낸드 플래시 메모리 기반 저장장치들이 구조 등에 제약이 있었던 것과는 다르게 다양한 구조와 구성에 유연하게 대응이 가능하다. 또한 기존 방법에서 제시된 것과 같이 요청의 길이에 기반 한 데이터 선별 방식이 아닌 접근 패턴을 기반으로 한 데이터 선별 방식을 제안하였다. 실험결과를 통해 제안한 FTL은 MLC 저장장치 대비 전체 실행시간 관점에서 평균 56% 개선된 성능을 나타내었고, 이는 기존 제안된 방법인 ComboFTL[5] 대비해서 약 35% 추가 개선을 보여준 것이다.

다만 MLC 관리를 위해 페이지 수준 주소 변환을 사용하였으므로 메모리 오버헤드를 줄이기 위한 작업이 필요할 것으로 보이며, 저장장치의 유휴 시간을 활용한 마이그레이션 오버헤드 감소 또한 큰 성능향상을 가져올 수 있을 것으로 보인다. 마지막으로 본 논문은 시뮬레이션을 통해 성능을 검증하였으나 앞서 언급한 바와 같이 하드웨어 포팅 가능한 코드로 바꾸어 적절한 하드웨어 플랫폼이나 실제 제품과 함께 실험한다면 보다 정확한 결과를 추출할 수 있을 것이다.

## 참 고 문 헌


[1] S.-H. Chang, S.-K. Lee, S.-J. Park, M.-J. Jung, J.-C. Han, I.-S. Wang, K.-H. Lim, J.-H. Lee, J.-H. Kim, W.-K. Kang, T.-K. Kang, H.-S. Byun, Y.-J. Noh, L.-H. Kwon, B.-K. Koo, M. Cho, J.-S. Yang, and Y.-H. Koh, “A 48nm 32gb 8-level nand flash memory with 5.5mb/s program throughput,” in *IEEE International Solid-State Circuits Conference*, pp.240-241, Feb. 2009.

[2] HY27UH08AG5M, Hynix Semiconductor Inc., www.hynix.com.

[3] S. Lee, K. Ha, K. Zhang, J. Kim, and J. Kim, “FlexFS: a flexible flash file system for MLC NAND flash memory, ” in Proc. of *the 2009 conference on USENIX Annual technical conference (USENIX'09)*, pp.9-9, Berkeley, CA, USA, 2009.

[4] S. Jung, Y. Song, "Hierarchical use of heterogeneous flash memories for high performance and durability," *Consumer Electronics, IEEE Transactions on* , vol.55, no.3, pp.1383-1391, August 2009

[5] S. Im, D. Shin, “ComboFTL: Improving performance and lifespan of MLC flash memory using SLC flash buffer, ” *Journal of Systems Architecture*, vol.56, no.12, pp.641-653, December 2010.

[6] 이성진, 김지홍, “SLC/MLC 이종 플래시 저장장치 기술”, *전자공학회지*, 제37권, 3호, 281-290쪽, 2010년 3월

[7] D. Jung, J.-U. Kang, H. Jo, J.-S. Kim, and J. Lee, “Superblock FTL: A superblock-based flash translation layer with a hybrid address translation scheme”, *ACM Trans. Embed. Comput. Syst.*, vol. 9, no.4, pp.1-41, April 2010.

[8] A. Gupta, Y. Kim, and B. Urgaonkar, “Dftl: a flash translation layer employing demand-based selective caching of page-level address mappings,” *SIGPLAN Not.*, vol. 44, pp. 229-240, March 2009

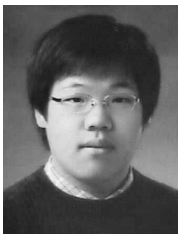
[9] SNIA, Advancing Storage and Information Technology, http://www.snia.org/

[10] M. Russinovich, DiskMon for Windows v2.01, http://technet.microsoft.com/en-us/sysinternals/bb896646.aspx, Nov. 2006.

## 저 자 소 개

**방 관 후**(정회원)
2006년 연세대학교 학사 졸업
2008년 연세대학교 석사 졸업
2008년~현재 연세대학교 박사과정
<주관심분야 : 저전력 설계기술, 플래시 메모리 응용, 바이오 응용>

**이 혁 준**(정회원)
1993년 University of Southern California 학사 졸업
1995년 Standford Unversity 석사 졸업
2001년 Stanford University 박사 졸업
<주관심 분야 : 임베디드 시스템, 바이오 컴퓨팅, 저전력 설계, 메모리 구조>



**박 상 훈**(정회원)
2009년 연세대학교 학사 졸업
2009년~현재 연세대학교 통합과정
<주관심분야 : 시스템 구조, 플래시 메모리 응용>

**정 의 영**(정회원)
1988년 고려대학교 학사 졸업
1990년 고려대학교 석사 졸업
2002년 Stanford University 박사 졸업
<주관심 분야 : 시스템 구조, VLSI 설계, 저전력 설계>