한국지능시스템학회 논문지 2010, Vol. 20, No. 6, pp. 761-767
DOI : 10.5391/JKIIS.2010.20.6.761

데이터 갱신 패턴 기반의 낸드 플래시 메모리의 블록 사용 균일화 기법

A Wear-leveling Scheme for NAND Flash Memory based on Update Patterns of Data

신효정* · 최돈정** · 김보경* · 윤태복** · 이지형**†
Hyojoung Shin, Don-Jung Choi, Bo-Keong Kim, Taebok Yoon and Jee-Hyong Lee

* 성균관대학교 임베디드소프트웨어학과
** 성균관대학교 전자전기컴퓨터공학과

요 약

낸드 플래시 메모리는 블록에 새로운 데이터를 쓰고자 할 때 삭제 연산이 선행되어야 하며 일정 횟수 이상 지움 연산이 반복된 블록은 더 이상 사용이 불가능하다. 데이터의 갱신이 빈번한 핫 데이터는 블록을 빠르게 사용 불가능한 상태에 도달하게 만들 수 있고 이로써 낸드 플래시 메모리의 용량은 시간이 지남에 따라 감소할 수 있다. 본 논문에서는 데이터의 접근 패턴을 고려해 핫 데이터와 콜드 데이터를 분류하는 알고리즘을 제시한다. 이렇게 분류된 데이터 정보를 이용해 삭제 횟수가 많은 블록에 갱신 확률이 적은 콜드 데이터를, 삭제 횟수가 상대적으로 적은 블록에 갱신 확률이 높은 핫 데이터를 맵핑한다. 입력 데이터 패턴을 이용한 핫/콜드 데이터 분류 기법이 기존의 분류 기법을 사용했을 때보다 플래시 메모리의 블록 사용이 균일한 것을 실험을 통해 확인하였다.

키워드 : 낸드 플래시 메모리, Cold 데이터, Hot 데이터, Wear-leveling, FTL 알고리즘.

Abstract

In the case of NAND flash memory, a whole block needs to be erased for update operations because update-in- place operations are not supported in NAND flash memory. Blocks of NAND flash memory have the limited erasure cycles, so frequently updated data (hot data) easily makes blocks worn out. As the result, the capacity of NAND flash memory will be reduced by hot data. In this paper, we propose a wear-leveling algorithm by discriminating hot and cold data based on the update patterns of data. When we applied this scheme to NAND flash memory, we confirmed that the erase counts of blocks became more uniform by mapping hot data to a block with a low erase count and cold data to block with a high erase count.

Key Words : Flash Memory, Cold data, Hot data, Wear-leveling, FTL Algorithm

1. 서 론

낸드 플래시 메모리는 연속적인 블록의 집합으로 이루어져 있고, 하나의 블록은 연속적인 페이지의 집합으로 이루어져 있다. 그리고 전원에 관계없이 데이터가 저장되는 비휘발성 메모리로써 휴대용 기기의 저장장치로 적합하다. 현대인의 생활 및 업무 스타일이 정적으로 실내에서 이루어지던 것에서 동적으로 이동하는 실외로 변화함에 따라 스마트폰, PMP, 노트북과 같은 휴대기기의 사용이 증가하고 있다. 이로 인하여 작은 크기로 큰 저장 공간을 제공하는 낸드 플래시 메모리의 사용도 함께 증가하고 있다. 낸드 플래시 메모리는 블록에 데이터가 쓰인 후 해당 위치에 데이터가 다시 쓰이려면 블록 전체를 지우는 연산이 선행되어야 하는데 쓰기 및 읽기 연산(페이지 단위)과 지우기 연산(블록 단위)의 단위가 다르다. 또한 블록 마다 일정 횟수 이상 지우기 연산을 하면 해당 블록을 다시 사용할 수 없게 된다.

Flash Translation Layer(FTL)은 위에서 언급한 것과 같은 낸드 플래시 메모리의 물리적 성질에 의해 발생하는 성능저하를 소프트웨어적으로 보완하기 위해 플래시 메모리와 파일 시스템 사이에서 사용되는 알고리즘의 일종이다. 그림 1은 파일 시스템과 플래시 메모리 사이에 존재하는 FTL을 나타낸다.

접수일자 : 2010년 3월 24일
완료일자 : 2010년 11월 15일
본 논문은 본 학회 2010년도 춘계학술대회에서 선정된 우수논문입니다.
+교신저자
본 연구는 문화체육관광부 및 한국콘텐츠진흥원의 2010년도 문화콘텐츠산업기술지원사업의 연구결과로 수행되었음

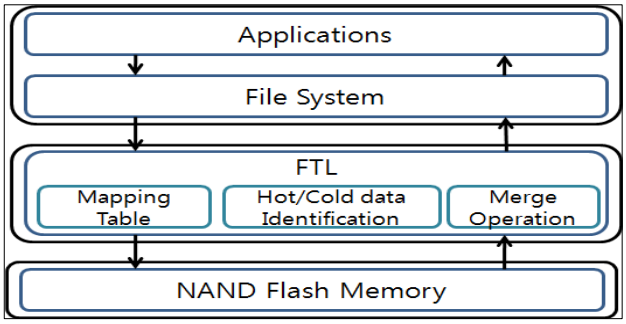


그림 1. NAND flash memory 시스템 구조

Fig. 1. The architecture of the NAND flash memory system

FTL의 주된 기능은 파일 시스템의 논리적 주소와 낸드 플래시 메모리의 물리적 주소를 맵핑하는 것이다. 맵핑 테이블을 사용하는 경우, 논리적 주소와 물리적 주소를 1대 1로 고정적 맵핑한 경우보다 시스템의 성능이 월등히 향상된다. 예를 들어 빈번하게 갱신되는 논리적 주소와 1대 1로 고정적 맵핑을 이룬 물리적 주소(블록)는 데이터를 갱신하기 위해 해당 블록을 지우고 다시 써야 한다. 이러한 상황은 해당 블록이 빠르게 최대 지움 횟수에 도달하게 만들고, 이렇게 최대 지움 횟수에 도달한 블록은 데이터의 저장을 할 수 없는 상태가 된다. 따라서 FTL은 맵핑 테이블을 제공하여 논리적 주소에 다수의 물리적 주소가 연결되게 하여 파일 시스템으로 하여금 블록을 고르게 사용할 수 있는 환경을 제공한다.

기존의 연구에서는 논리적 주소 값에 비어있는 물리적 블록 접근하여 맵핑하는 방식으로 처리하고 있다[1][2]. 하지만 빈번하게 갱신되는 핫 데이터와 갱신 횟수가 거의 없는 콜드 데이터는 메모리 블록의 균일한 사용을 불가능하게 만든다. 따라서 데이터의 특성에 따라 해당 데이터에 가장 적합한 블록을 할당해 주는 알고리즘을 사용한다면 보다 효과적으로 플래시 메모리를 사용할 수 있다. 기존의 데이터 분류에서는 블록의 접근 횟수를 통해 일정한 횟수 이상 접근되는 경우를 핫 데이터로 분류하는 방법을 사용하고 있다[3][4].

본 논문에서는 낸드 플래시 메모리에 저장되는 데이터의 패턴을 분석하고 특성을 분류함으로써 메모리를 효과적으로 사용할 수 있는 FTL을 제안한다. FTL의 논리적 주소와 물리적 주소를 맵핑하는 단계에서 지움 횟수가 적은 블록에 핫 데이터를, 많은 블록에는 콜드 데이터를 맵핑시킴으로써 전체적인 메모리가 균일하게 사용될 수 있는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구의 기반 연구로서 FTL의 정의 및 대표적 기법에 대해 설명하고 있다. 3장에서는 본 논문에서 제안하는 핫/콜드 데이터 분류를 통한 메모리 사용 균일화 알고리즘의 구조를 설명한다. 4장에서는 제안된 알고리즘의 성능을 기존의 연구와 비교하여 평가한 후 5장에서 결론을 도출하여 향후 연구 계획을 설명한다.

2. 기반 연구

2.1 NAND Flash Memory

낸드 플래시 메모리는 기존의 자기 디스크를 대신하는 차세대 데이터 저장 장치로 각광받고 있다. 노어 플래시 메모리와 더불어 휴대용 기기에서 많이 사용되는 비휘발성 메모리로서 값이 저렴하고 크기 대비 저장 용량이 다른 저장 매체보다 뛰어나다는 장점이 있다. 또한 내구성이 뛰어나 데이터 센터에서 사용되는 기존의 자기 디스크의 세대교체용 저장 장치로 여겨지고 있다.

낸드 플래시 메모리는 공정 방식에 따라 Small Block NAND와 Large Block NAND로 나뉘며 셀에 대한 저장 방식에 따라 Single Level Cell과 Multi Level Cell로 나뉜다. 일반적으로 많이 사용되는 낸드 플래시 메모리 구조는 그림 2와 같이 나타낼 수 있다.[5].

낸드 플래시 메모리는 여러 블록의 집합으로 이루어져 있고 하나의 블록은 다수개의 페이지로 구성되어 있다. 이 페이지 또한 셀의 집합으로 이루어져 있으며 페이지는 사용자가 데이터를 저장할 수 있는 부분과 스페어 영역이라고 하는 다양한 정보를 저장하는 부분으로 나뉘어져 있다. 스페어 영역에는 블록의 지움 횟수 및 불량 셀과 같은 정보를 저장하고 있다.

낸드 플래시 메모리의 쓰기 읽기 단위는 페이지 단위로 이루어지며 지움 연산은 블록 단위로 수행되며 가장 큰 특징 중 하나로 낸드 플래시 메모리는 한번 데이터 쓰인 곳에 새로운 데이터를 덮어쓰기 할 수 없다. 모든 갱신 연산은 지움 연산을 선행한 쓰기 연산으로 간주될 수 있다.

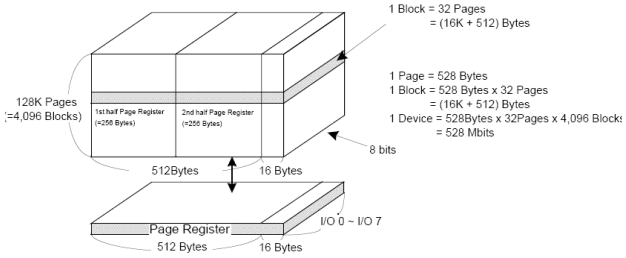


그림 2. NAND Flash Memory 구조

Fig. 2. The structure of NAND Flash Memory

2.2 Flash Translation Layer

Flash Translation Layer(FTL)의 역할은 파일 시스템에서 사용되는 논리적 주소와 플래시 메모리에서 사용되는 물리적 주소를 맵핑하는데 있다. 이때 FTL은 낸드 플래시 메모리가 가진 단점을 보완하기 위해 단순 맵핑이 아닌 다양한 맵핑 알고리즘 및 추가 기능을 사용하여 메모리의 성능을 개선시키고 있다.

기존의 연구에서는 페이지 단위의 맵핑, 블록 단위의 맵핑, 로그 블록을 사용한 블록 단위의 맵핑 등 다양한 맵핑 알고리즘을 제안하고 있다. 본 논문에서 사용한 맵핑 방법은 그간의 연구 중 가장 뛰어난 성능을 보인 로그 블록을 사용한 블록 단위의 맵핑이다(FAST)[2].

FAST는 기본적으로 블록 단위의 맵핑을 사용하고 있다. 블록 단위 맵핑은 페이지 단위 맵핑보다 맵핑 테이블의 크기가 작기 때문에 추가적으로 사용되는 자원이 적다는 장점을 가지고 있다.

낸드 플래시 메모리의 경우 앞서 설명한 것과 같이 쓰기 연산의 단위와 지움 연산의 단위가 다르다. 이러한 물리적 특징은 메모리 시스템 상에서 큰 단점으로 작용한다. 예를 들어 블록 내에서 하나의 페이지가 새로운 데이터로 갱신되어야 하는 경우, 쓰기 전에 지움 연산이 먼저 수행되어야 한다. 지움 연산은 블록 단위로 수행되기 때문에 갱신되어야 할 페이지를 제외한 모든 페이지들은 불필요한 데이터

복사 연산과 지움을 수행하게 된다. 결과적으로 블록의 지움 횟수를 증가시켜 빠르게 사용 불가능한 단계에 도달하게 하며 쓰기 연산에 대한 수행 완료 시간을 길게 만든다.

이러한 문제를 해결하기 위해 FAST는 낸드 플래시 메모리를 데이터 블록과 로그 블록으로 나누어 관리하고 있다. 데이터 블록은 일반적인 블록으로 입력되는 데이터를 저장하는 부분이고, 로그 블록은 갱신되는 페이지들을 따로 관리하는 부분으로서 하나의 페이지 갱신으로 인한 블록의 지움 연산을 피하게 해준다. 갱신되는 페이지는 로그 블록에 의해 페이지 단위 맵핑으로 관리되어지고 데이터 블록에 저장되어 있는 본래 데이터는 무효화시킨다. FAST는 모든 데이터 블록이 모든 로그 블록을 공유하는 방식을 사용하고 있어 로그 블록을 효율적으로 관리할 수 있는 환경을 제공한다. 즉, 새로이 갱신되는 모든 페이지를 로그 블록에 순차적으로 기입하여 로그 블록을 비는 페이지 없이 처음부터 끝까지 완전히 사용하는 방식을 사용하고 있다. 모든 로그 블록이 갱신되는 데이터를 위해 사용되면 FAST는 합병 연산을 사용하여 모든 로그 블록을 합병하여 자유 블록으로 만든다. 합병 연산은 그림 3과 같이 두 가지 방식으로 나누어 나타낼 수 있다.

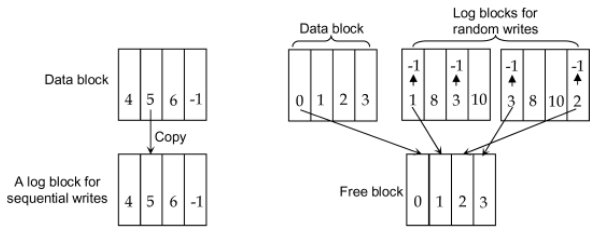


그림 3. 합병 연산
Fig. 3. The merge operation

FAST의 로그 블록은 순차 쓰기(SW)를 위한 블록과 랜덤 쓰기(RW)를 위한 블록으로 나누어져 관리된다. 새로이 갱신되는 데이터의 물리적 주소 오프셋이 페이지의 가장 첫 부분이면 순차 쓰기 로그 블록에, 그렇지 않다면 랜덤 쓰기 로그 블록에 저장 된다. 위와 같은 방식으로 로그 블록이 모두 사용된 후 합병 연산이 이루어지는데 순차 쓰기 로그 블록의 경우 그림 3의 (a)와 같이 로그 블록에 데이터 블록의 나머지 페이지(갱신되지 않은 페이지)들을 복사한 후 데이터 블록을 삭제하고 로그 블록을 데이터 블록으로 변환시킨다. 랜덤 쓰기 로그 블록을 합병하는 경우에는 그림 3의 (b)와 같이 논리적 주소를 위한 새로운 자유 블록을 해당 주소에 할당한다. 이때 데이터 블록에 존재하는 유효한 페이지들과 로그 블록에 존재하는 최신 페이지를 새로 할당된 자유 블록에 복사하여 모두 유효한 데이터를 가진 블록을 만들어 낸 다. 그리고 합병 연산에 사용된 데이터 블록을 삭제하고, 로그 블록의 경우 모든 유효한 페이지가 새로운 블록에 복사된 경우에만 삭제한다.

3. 핫/콜드 데이터 분류를 통한 메모리 사용 균일화 정책

3.1 핫 데이터와 콜드 데이터 정의 및 관리 정책

FTL에서 정의하는 핫 데이터는 다른 데이터들에 비해 갱신 횟수가 매우 빈번한 데이터를 말하고 콜드 데이터는 한번 메모리에 저장된 후 갱신이 거의 없는 데이터를 말한다. 핫 데이터에 의해 사용 불가능한 블록이 계속해서 생성되는 동안 콜드 데이터는 지움 횟수가 적은 블록을 장시간 차지함으로써 전체적인 플래시 메모리의 수명에 영향을 미친다. 따라서 이러한 핫/콜드 데이터를 판단하여 갱신 확률이 적은 콜드 데이터를 지움 횟수가 높은 블록에, 갱신 확률이 높은 핫 데이터를 지움 횟수가 낮은 블록에 배치하는 것으로 블록의 평균 수명을 연장할 수 있다.

기존의 핫/콜드 데이터 분류방법은 블록의 지움 횟수나 [3][4], 블록 삭제 시 걸리는 시간으로[6] 분류한 뒤 블록 내 데이터 교체를 수행하는 방식을 사용하고 있다. 이 기법은 블록의 지움 횟수가 일정 횟수 이상인 경우 해당 블록의 데이터를 지움 횟수가 적은 블록으로 이동시킨다. 하지만 데이터의 교체가 한 시점에서 집중적으로 발생하는 현상을 야기한다. 또한 특정한 쓰기 연산에서 수행 시간이 과도하게 지체되는 현상을 발생시킬 수 있는 단점을 가지고 있다. 그리고 한번 핫 데이터로 분류된 후 다시 핫 데이터로 분류되거나 일반 데이터로 분류하는 데에 정의가 명확하지 않다. 또한 횟수만을 고려하는 핫 데이터 검출 기법은 동일한 갱신 횟수를 가진 두 개의 논리적 주소에 대해서 동일한 데이터 특성을 부여하기 때문에 정확성에 문제 있다.

본 논문에서는 데이터의 접근 패턴을 분석하여 특성을 결정한다. 이 방법은 블록의 지움 횟수만 고려하는 기존의 방법에 비해 두 가지 측면에서 나은 성능을 보인다. 먼저 일정한 시간 간격 내에서의 접근 패턴을 고려하기 때문에 데이터의 특성을 좀 더 유동적으로 결정지을 수 있고 패턴을 분석함으로써 정확도가 증가하였다. 예를 들어 동일한 갱신 횟수를 가지더라도 최근에 갱신 횟수가 증가하는지, 감소하는지에 따라 두 논리적 주소는 다른 데이터 특성을 부여받게 된다. 이처럼 각각의 논리적 주소에 대해 갱신 횟수라는 하나의 특징뿐만 아니라 접근 패턴도 함께 고려함으로써 핫/콜드 데이터의 분류 정확도가 향상될 것으로 기대된다.

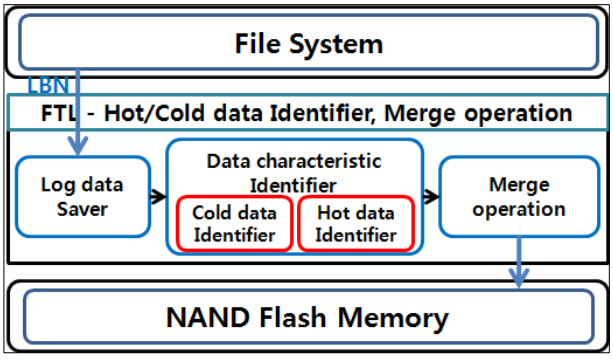


그림 4. 데이터 특성 분류 시스템
Fig. 4. The system of data characteristic identification

3.2 전체 프로세스

본 논문에서 제안하는 핫/콜드 데이터 분류는 그림 4와 같다. 파일 시스템에서 논리적 블록 주소를 FTL로 보내면 FTL은 해당 논리적 블록 주소의 접근 패턴을 이용해 핫 데이터와 콜드 데이터를 분류한다. 분류된 데이터 특성에 따라 합병 연산 시 합병될 블록 데이터가 핫 데이터라면 지움 횟수가 적은 자유 블록을 할당하고, 콜드 데이터라면 횟수가 많은 자유 블록을 할당한다. 그림 4에서 보는 것과 같이 분류기는 로그 데이터를 저장하는 부분과 핫/콜드 데이터를 분류하는 부분으로 나뉜다.

FTL 내부에는 맵핑 테이블 외에 핫/콜드 데이터 분류기가 추가적으로 적재되어 있다. 해당 모듈은 파일 시스템에서 입력되는 논리적 블록 주소를 일정한 시간 단위(Time Slot) 내에서 존재하는 가산 카운터를 이용해 모든 논리적 블록 주소마다 접근 패턴에 대한 로그 데이터를 만들어낸다. FTL은 이 로그 데이터를 이용해 논리적 블록 주소 별로 데이터 접근 패턴을 분석한 후에 해당 블록이 핫 데이터인지 콜드 데이터인지 판별하고 이 정보를 이용해 낸드 플래시 메모리의 성능을 개선하기 위한 추가 기능을 수행한다.

그림 5처럼 로그 데이터는 논리적 블록 번호 당 3개의 좌표 쌍으로써, 각 좌표 쌍은 카운터와 데이터 접근 횟수로 구성된다. 카운터는 일정한 시간 단위 내에 존재하는 내부 카운터이다. 접근 횟수는 해당 논리 블록이 접근될 때마다 하나씩 증가하는 연산을 수행하여 얻는 수치이다. 3개의 좌표 쌍은 3구간으로 나누어진 시간 구간에서의 마지막 카운터 값과 시간 단위 내에서의 데이터 접근 횟수를 나타낸다.

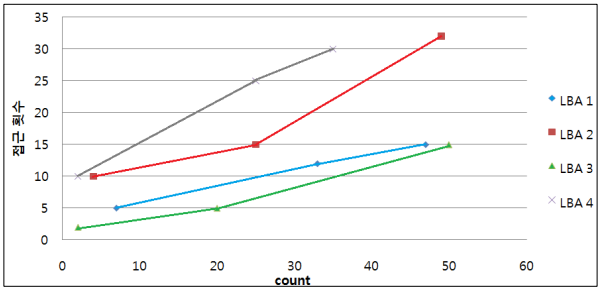


그림 5. 로그 데이터를 이용한 패턴 분석
Fig. 5. The analysis of log data patterns

블록의 지움 횟수를 균일하게 맞춰주기 위해, 데이터 특성 분류기를 통해 분류된 데이터의 특성에 맞는 자유 블록을 합병 연산 시 해당 데이터에 할당하였다. 이로써 추가되는 데이터 복사, 이동 및 지움 연산 없이 메모리의 특정 블록이 사용 불가능한 상태에 도달하는 것을 방지한다. 핫 데이터에는 지움 횟수가 비교적 적은 자유 블록을, 콜드 데이터에는 비교적 지움 횟수가 많은 자유 블록을 할당한다.

```
hot_identifier(lbn)
input : logical block number
output : 0(not Hot data) or 1(Hot data)
  1 thV : threshold gradient for Hot data
  2 hot_blk : list of hot block number
  3 lbn : logical block number
  4 if gradient of 2nd and 3rd is bigger than
    gradient of 1st and 2nd
  5   if gradient of 2nd and 3rd is bigger than thV
  6     for i=1 to k
  7       if hot_blk[i] is empty
  8         hot_blk[i] = lbn
  9         break
  10    end for
  11  end if
  12 end if
  13 end
```

그림 6. 핫 데이터 판단 알고리즘
Fig. 6. The algorithm of hot data identification

핫 데이터 분류기는 로그 데이터 저장 장치로부터 입력받은 3개의 좌표 쌍으로 구성된 로그 데이터를 분석하여 해당 논리적 블록이 핫 데이터인지 판별해 낸다. 3개의 좌표 쌍을 잇는 선분의 기울기를 이용하여 해당 논리적 블록이 향후에 핫 데이터가 될 것인지 예측한다. 그림 6은 데이터의 논리적 블록 주소에 대한 접근 패턴을 기반으로 획득된 로그 데이터를 이용해 핫 데이터를 판단하는 알고리즘을 나타낸다. 과거의 구간보다 기울기가 증가하면 해당 논리적 주소는 향후에도 빈번하게 갱신되는 핫 데이터로 분류될 확률이 증가하게 되고 이 변량이 미리 지정한 임계값 이상이라면 해당 데이터는 핫 데이터가 된다.

콜드 데이터는 데이터의 접근 패턴에 대한 로그 데이터가 수집되는 일정한 시간 단위(Time Slot)내에 한번이라도 접근했던 논리적 블록 주소가 다음 시간 단위에서 접근되지 않는 경우 해당 논리적 블록 주소의 데이터를 콜드 데이터로 간주하고 콜드 블록 주소를 저장하는 리스트에 해당 논리적 블록 주소를 저장함으로써 콜드 데이터를 관리하고 있다.

4. 실험 및 평가

4.1 실험 환경

본 논문에서 제시하는 알고리즘의 성능을 측정하기 위해 낸드 플래시 메모리 환경을 가상으로 구현하여 해당 알고리즘을 분석하였다. 실험을 위해 사용된 FTL은 블록 단위 맵핑이며 로그 블록을 사용하는 FAST이다. FAST 알고리즘은 로그 블록을 위한 맵핑 테이블의 크기가 작고 구현이 간단하면서 효과적인 성능을 보인다. 제안한 알고리즘은 지움 횟수 기반 방식[3] 보다 나은 성능을 보였다. 본 실험에서 사용한 낸드 플래시 메모리의 구성은 표 1과 같다.

표 1. 낸드 플래시 메모리 구성
Table 1. The structure of NAND flash memory

항목	크기
페이지 크기	4k
블록 내 페이지 개수	64
전체 블록 개수	4k

위와 같이 구성된 가상의 낸드 플래시 메모리에서, 가상의 트레이스를 수행한 후, 각각의 알고리즘에 대해 블록 전체의 지움 횟수 분산을 비교하였다. 비교된 알고리즘은 FAST 알고리즘, 지움 횟수 기반의 데이터 특성 분류기를 포함한 FAST 알고리즘, 그리고 본 논문에서 제안하는 분류기를 포함한 FAST 알고리즘이다. 지움 횟수를 비교하는 실험에서 사용한 트레이스[7]는 표2와 같다. 이 트레이스는 Pennsylvania state university의 PSU 연구실에서 제공하는 것이며 FTL을 테스트하기 위해 실제의 메모리 사용과 유사하게 만들어낸 트레이스이다.

표 2. 트레이스의 특징
Table 2. The characteristic of trace

항목	크기
논리적 주소의 범위	0 ~ 2641
총 연산 횟수	968625
쓰기 연산 횟수	876681
읽기 연산 횟수	91944
핫 데이터의 비율	1%
콜드 데이터의 비율	0.6%

4.2 실험 결과

실험은 제안한 알고리즘이 부가적인 지움 연산 없이 낸드 플래시 메모리의 블록 사용을 균일하게 하는데 성능 평가의 기준을 두고 실험되었다.

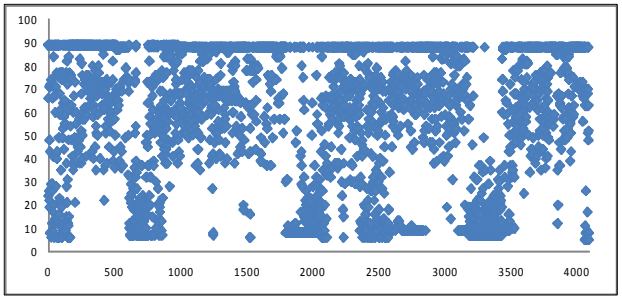


그림 7. 메모리에서의 지움 횟수
Fig. 7. The result of erase

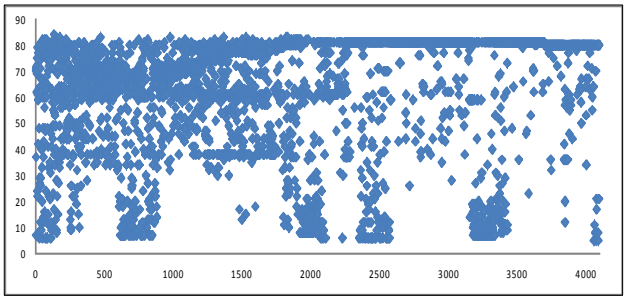


그림 8. 갱신 횟수 기반의 블록 사용 균일화 기법 적용 결과
Fig. 8. The result of wear-leveling with block counts

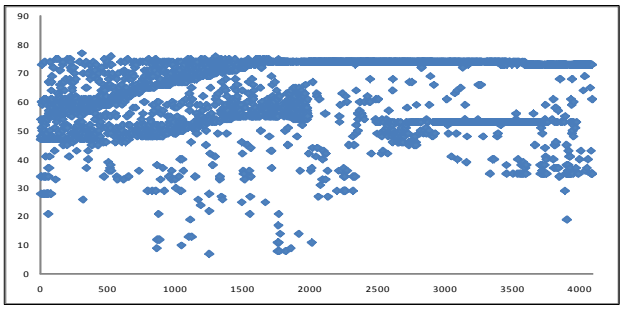


그림 9. 제안하는 알고리즘 기반의 블록 사용 균일화 기법 적용 결과
Fig. 9. The result of wear-leveling with the proposed scheme

그림 7은 표 2의 트레이스를 수행한 후에 낸드 플래시 메모리의 모든 블록에 대한 지움 횟수를 그래프로 나타낸 것이다. 그래프의 X축은 블록의 물리적 주소를 의미하고 Y축은 해당 블록의 지움 횟수를 의미한다. 그림 7을 보면 블록의 지움 횟수가 5회에서 90회 사이에 넓게 분포된 것을 확인 할 수 있다. 이것은 전체적인 낸드 플래시 메모리의 블록이 고르게 사용되지 못하고 블록 간의 사용 정도 차가 매우 큰 것을 보여주고 있다.

그림 8은 데이터의 갱신 횟수만을 고려하여 핫/콜드 데이터를 분류하는 알고리즘[3]을 적용하였을 때의 블록 별 지움 횟수를 나타낸 그림이다. 핫/콜드 분류를 적용하지 않은 그림 7과 비교해 보면 지움 최대 지움 횟수가 90회에서 85회 이하로 하향되었고 블록간의 지움 횟수 격차가 데이터 분류기를 사용하지 않을 때에 비해 좁아진 것을 확인 할 수 있다.

그림 9는 본 연구에서 제안하는 핫 콜드 데이터 분류 기법을 적용하고, 표 2의 트레이스를 수행한 후에 동일한 낸드 플래시 메모리의 모든 블록에 대한 지움 횟수를 그래프로 나타낸 것이다. 그래프의 X축과 Y축은 그림 7과 동일하다. 블록의 지움 횟수는 5회에서 75회 사이에 분포되었고 그림 7이나 8보다 지움 횟수가 좁게 분포한 것을 확인 할 수 있다.

실험 결과로부터 지움 횟수의 표준편차가 기존의 FAST 알고리즘은 29.65, 횟수만을 고려한 분류의 경우 23.2, 제안된 알고리즘의 경우 11.72로 기존의 두 방법 보다 향상 되었다.

그림 10은 위 실험 결과를 하나의 그래프로 표현한 것이다. X축은 지움 횟수가 적은 순서대로 정렬된 블록의 인덱스 번호이고 Y축은 지움 횟수이다. 점선으로 표현된 실선은 가장 이상적인 상태를 나타내고 있다. 이상적인 실선에 가까울수록 블록의 사용정도가 고르다고 할 수 있다. 제안하는 알고리즘을 사용하였을 때에 낸드 플래시 메모리의 전체적인 지움 횟수가 다른 것에 비해 평준화된 것을 확인할 수 있다. 또한 지움 횟수가 다른 블록에 비해 현저히 낮은 부분도 기존의 FAST 알고리즘 보다 비중이 줄어든 것을 확인 할 수 있다.

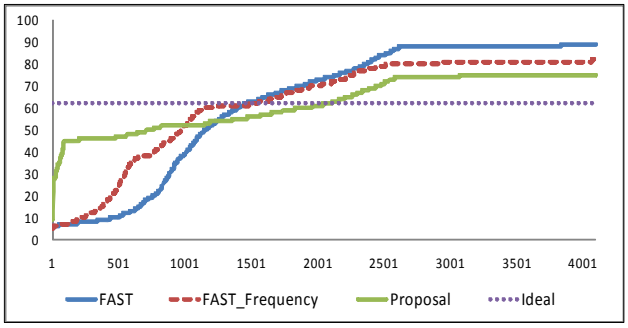
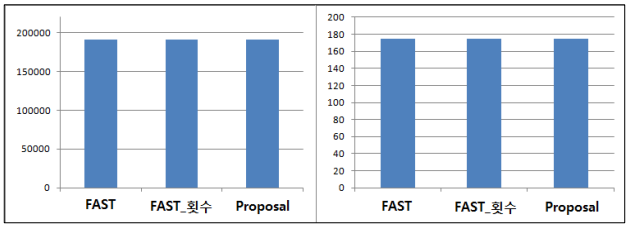


그림 10. 실험 결과
Fig. 10. The result of experiments

제안하는 알고리즘의 경우 합병연산의 단계에서 핫/콜드 데이터에 적합한 자유 블록을 할당함으로써 블록 사용 평준화를 위해 부가적인 지움 연산이나 데이터 복사 연산이 발생하지 않는다. 그림 11은 합병 연산 횟수를 그래프로 나타낸 것이다.



(a)순차 쓰기 (b)랜덤 쓰기

그림 11. 순차쓰기/랜덤쓰기 합병 횟수
Fig. 11. The number of merging operation

그림 11의 (a)는 순차 쓰기를 위해 할당된 로그 블록의 합병 횟수를 나타내고 (b)는 랜덤 쓰기를 위해 할당된 로그 블록의 합병 횟수를 나타낸다. 두 개를 비교해 보면 추가적인 합병 연산 없이 블록 사용이 평준화되었음을 확인할 수 있다. 따라서 블록 사용의 균일화를 위한 작업이 낸드 플래시 메모리상에 초과되는 합병 연산을 만들어 내지 않으며 효과적으로 수행 되었다는 것을 확인할 수 있었다.

5. 결론 및 향후 연구

데이터의 논리적 주소 접근 패턴을 이용해 핫 데이터와 콜드 데이터를 분류하고 해당 정보를 합병연산에서 사용함으로써 기존 Wear-Leveling 기법에서 발생하는 페이지 교체 비용을 0으로 감소시킬 수 있었다. 블록의 지움 횟수를 전체적으로 하향평준화 함으로써 낸드 플래시 메모리의 수명이 연장되는 것을 확인할 수 있었다. 블록의 최대 지움 횟수를 기존의 알고리즘보다 하향시켰고, 지움 횟수 분산도 또한 기존의 알고리즘 보다 작아진 것을 확인할 수 있었다. 본 연구에서는 기존의 연구 대비 추가적으로 발생하는 오버헤드를 줄이기 위해 핫 데이터를 위한 지움 횟수가 적은 블록, 콜드 데이터를 위한 지움 횟수가 많은 자유 블록을 검색 할 때에 모든 블록을 검색하지 않고 일정 수준 이상, 이하의 지움 횟수를 갖는 블록을 만날 때 검색 연산을 멈추는 방식을 사용하고 있다. 향후에는 이러한 오버헤드를 효과적으로 줄이는 알고리즘을 연구하여 최소한의 오버헤드로 낸드 플래시 메모리의 성능을 개선할 계획이다.

참 고 문 헌

[1] J. Kim, J. Kim, S. H. Noh, Y. Cho, "A Space- efficient Flash Translation Layer for Compactflash Systems", *IEEE Transaction on Consumer Electronics*, vol. 48, no. 2, pp. 366-375, 2002.
[2] S. W. Lee, D. J. Park, T. S. Chung, D. H. Lee, S. W. Park, H. J. Song, "A Log Buffer-based Flash Translation Layer using Fully-associative Sector Translation", *ACM Embedded Computing Systems*, vol. 6, no. 3, article 18, 2007.
[3] L. Chang, "On Efficient Wear Leveling for Large -scale Flash-memory Storage Systems", *Symposium on Applied Computing*, pp. 1126 - 1130, 2007.
[4] M. L. Chiang, Paul C. H. Lee, and R. C. Chang, "Using Data Clustering to Improve Cleaning Performance for Flash Memory", *Software Practice & Experience*, vol. 29, no.3, pp. 267-290, 1999.
[5] Samsung Electronics. "K9F1208U0C Flash Memory datasheet", Available: http://www.samsung.com, 2006, [Accessed: November 1, 2010]
[6] S. W. Han, "Flash memory wear leveling system and method", *United States Patent*, No.6016275, 2000.
[7] John Bucy, Greg Ganger, "DiskSim Storage Subsystem Simulation Environment (Version 3.0)", Available: http://csl.cse.psu.edu, 2003, [Accessed: November 1, 2010]

저 자 소 개

신효정 (Hyojoung Shin)
2010년 : 충북대 정보통신공학과 학사
2010년 ~ 현재 : 성균관대 임베디드 소프트웨어학과 석사과정

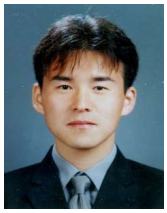
관심분야 : 임베디드소프트웨어, 지능 시스템, Flash translation layer
Phone : +82-31-290-7987
Fax : +82-31-299-4637
E-mail : shinhj0728@skku.edu

최돈정 (Don-Jung Choi)
2010년 : 아주대 정보 및 컴퓨터공학부학사
2010년 ~ 현재 : 성균관대 컴퓨터공학과 석사과정

관심분야 : 지능 시스템, 데이터 마이닝, 기계 학습
Phone : +82-31-290-7987
Fax : +82-31-299-4637
E-mail : tenosis@skku.edu

김보경 (Bo-Keong Kim)
2009년 : 숙명여대 컴퓨터과학과 학사
2009년 ~ 현재 : 성균관대 임베디드 소프트웨어학과 석사과정

관심분야 : 지능시스템, 상황인식, 임베디드소프트웨어
Phone : +82-31-290-7987
Fax : +82-31-299-4637
E-mail : kbk1225@skku.edu



윤태복 (Taebok Yoon)
2001년 : 공주대 전자계산학과 학사
2005년 : 성균관대 컴퓨터공학 석사
2010년 : 성균관대 컴퓨터공학 박사

관심분야 : 사용자 모델링, 게임 인공지능
Phone : +82-31-290-7987
Fax : +82-31-299-4637
E-mail : tbyoon@skku.edu

이지형 (Jee-Hyong Lee)
1993년 : 한국과학기술원 전산학과 학사
1995년 : 한국과학기술원 전산학과 석사
1999년 : 한국과학기술원 전산학과 박사
2002년 ~ 현재 : 성균관대 정보통신공학부 부교수

관심분야 : 지능시스템, 기계학습, 사용자 모델링
Phone : +82-31-290-7154
Fax : +82-31-299-4637
E-mail : jhlee@ece.skku.ac.kr