KM-평준화: NAND 플래시 메모리를 위한 레벨 기반 소거 횟수 평준화 기법*

김도윤° 박상원
한국외국어대학교 컴퓨터및정보통신공학과
dykim@dislab.hufs.ac.kr, swpark@dislab.hufs.ac.kr

KM-leveling : A Level-Based Wear Leveling Scheme for NAND Flash Memory

Do Yun Kim°, Sangwon Park
Department of Computer and Information Communication Engineering, Hankuk University of Foreign Studies

요 약

최근 휴대전화, 디지털 카메라, 랜 스위치, 디지털 셋톱박스, 휴대용 MP3 플레이어, 노트북용 PC 카드, 내장 기기의 펌웨어 등 플래시 메모리의 활용이 증가하고 있다. 하지만 기존 저장 장치와 달리 플래시 메모리는 특정 블록에 쓰기 연산을 하기 전에 해당 블록은 미리 소거(erase-before-write)되어 있어야 하는 제약이 있으며, 각 블록은 소거될 수 있는 횟수가 제한적이다. 이런 단점들은 플래시 메모리가 대용량화됨에 따라 중요한 문제로 대두되고 있다. 이런 각 블록에 대한 소거 횟수의 제한을 해결하기 위하여 소거 횟수 평준화(wear-leveling) 기법이 필수적이다. 본 논문에서는 블록의 소거 횟수의 한계를 극복하기 위한 새로운 소거 횟수 평준화 기법으로 전체 블록에 대한 소거 횟수 레벨을 두어 소거 횟수 평준화를 이루는 KM-평준화(KM-leveling)를 제안한다. KM-평준화는 소거 횟수 평준화를 위한 전체 블록의 계산 비용을 최소화하고 블록에 대한 소거 레벨을 두어 적은 공간을 사용하는 효율적인 기법이다. 본 논문은 M값 범위 이내에 각 블록의 소거 횟수들이 존재하도록 보장하는 KM-평준화를 제안한다.

1. 서 론

최근 디지털 카메라, MP3 플레이어, 핸드폰 등의 사용량이 급증해 짐에 따라 이동성이 중요한 요소로 차지하는 기기들이 많이 등장하였다. 이에 따라 소형화, 대용량화, 저 전력화, 비휘발성, 고속화 그리고 충격에 강한 메모리가 필요하게 되었다. 이런 요구사항들로 인하여 기존 디스크 저장 장치에 비해 빠른 데이터 접근 성능을 보장하게 됨으로써 많은 응용에서 플래시 메모리가 디스크를 대체할 것이라고 예상된다.

플래시 메모리는 특정 블록에 쓰기 연산을 하기 전에 해당 블록은 미리 소거(erase-before-write)되어 있어야 하는 제약이 있다. 이에 따라 플래시 메모리에 저장된 데이터를 직접 갱신하는 것은 불가능하기 때문에 빈 블록을 확보하고 변경된 데이터를 갱신하는 작업이 필요하다. 또한 각 블록은 소거될 수 있는 한계횟수(보통 10만 번)를 가지고 있기 때문에 각 블록에 대한 소거 횟수를 평준화하는 기법이 필수적이다. 이러한 플래시 메모리의 특성은 플래시 메모리가 기존의 하드디스크를 대체하는 것을 어렵게 만들고 플래시 메모리 시스템의 전체 성능의 저하를 가져온다. 플래시 메모리의 일반적인 특성[1]은 표 1과 같다.

* 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업[2006-S-040-01, Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술 개발]과 과학기술부 및 대구경북과학기술연구원의 연구개발사업의 일환으로 수행하였음.

플래시 메모리는 소거 연산이 특정 블록에 집중이 되면 일부 블록을 사용할 수 없게 되어 플래시 메모리의 수명이 오래 가지 못하게 된다. 따라서 플래시 메모리는 특정 블록에 있어서 소거 연산이 집중되지 않고 각 블록에 대하여 균등한 소거 횟수를 가질 수 있도록 소거 횟수 평준화(wear-leveling) 기법이 필수적이다. 이런 소거 횟수 평준화 기법은 플래시 메모리를 안정적으로 오래 사용할 수 있도록 하는 방법이 되며 플래시 메모리가 디스크를 대체할 때 매우 중요한 요소가 될 수 있다.

표 1. 플래시 메모리의 특성(Intel 28F640J3A)

Read access time	100~150ns
Writing time using buffer Erasing and writing a block	218μs /32bytes 0.8 sec/block
Erasing block time	1.0 sec/block
Erase limit	100,000 times
Power consumption	Standby power : 0~120μA Operating power : 5~70mA

기존의 소거 횟수 평준화 알고리즘은 일반적으로 각 블록의 소거 횟수를 기록하여 빈 블록을 할당하는 과정에서 소거 횟수가 적은 블록을 할당하는 방법으로 소거 횟수를 평준화하고 있다. 하지만 이런 방법은 복잡한 연산을 필요로 하며 빈 블록을 할당하기 위해 전체 블록을 검색해야 하는 오버 헤드가 발생하게 된다. 플래시 메모리의 대용량화로 인해서 이런 연산의 복잡도를 줄이면서 좀 더 효과적으로 소거 횟수를 평준화하는 기법이 필요하다.

본 논문에서는 블록의 소거 횟수에 대해서 M번이내의

K 소거 횟수 차이를 유지하며 많은 소거 연산이 들어오더라도 각 블록의 소거 횟수가 균등한 수준이 될 수 있도록 하는 기법을 제안한다. 이 기법에서는 각 블록의 소거 횟수를 적은 공간을 사용하고 기록하여 소거 횟수 평준화를 보장하며 빈 블록을 할당하는 기간 내에 평준화 작업을 위한 블록의 교체 작업이 이루어지므로 계산 비용을 줄인다. 또한 이 기법에서는 블록에 대한 교체 작업을 제한함으로써 교체 작업에 대한 비용을 최소화하고 효과적으로 소거 횟수 평준화 기능을 수행한다.

2. 관련 연구

플래시 메모리는 각 블록이 소거될 수 있는 한계 횟수를 가지고 있으므로 특정 블록이 집중적으로 기록 및 소거된다면 그 수명이 다하게 되어 저장 장치로써의 내구성이 떨어지게 된다. 따라서 이런 집중적인 소거를 줄이기 위해 소거 횟수 평준화 기능을 제공해야 한다. 이러한 소거 횟수 평준화를 통해 플래시 메모리 저장 장치의 내구성을 개선하려는 연구는 초기의 블록 관리 기법의 개발과 함께 있어 왔다.

임계값(threshold) 기반 소거 횟수 평준화[2]는 하나의 여유 소거 블록(spare block)을 설정하여 소거 횟수가 가장 많은 블록과 가장 적은 블록과 소거 횟수를 비교한다. 이렇게 소거 횟수를 비교하여 그 차이가 임계값을 초과하면 핫 블록(hot block)과 콜드 블록(cold block)을 교체한다. 교체 정책은 다음과 같다. 소거 횟수에 대한 정보는 섹터의 헤더에 기록하여 블록들의 소거 횟수를 비교하는데 사용한다.

step 1. 콜드 블록의 내용을 여유 블록으로 복사
step 2. 콜드 블록을 소거
step 3. 핫 블록의 내용을 소거된 콜드 블록으로 복사
step 4. 핫 블록은 새로운 여유 블록으로 사용

이 방법은 각 블록의 소거 횟수를 유지해야 하기 때문에 공간의 낭비가 있을 수 있으며 임계값과의 비교를 통해서 교체하기 위해 항상 여유 영역을 마련해야 하는 단점이 있다.

빈 블록 리스트(free block list)를 이용한 소거 횟수 평준화 기법[3]은 마이크로프로세서로의 접근을 허용하는 것을 전제로 한다. 섹터의 헤더에는 미터 필드(meter field)에 소거 횟수를 저장하고 소거 횟수에 따라서 빈 블록 리스트를 구성할 때 사용한다. 빈 블록 리스트는 원형 큐로 구현되어 있으며 소거된 빈 블록을 큐에 삽입하고 소거 횟수에 따라서 정렬하여 삽입하게 된다. 즉 큐의 처음은 소거 횟수가 가장 적은 블록이 위치하게 되고 큐의 마지막은 소거 횟수가 가장 많은 블록이 위치하게 된다. 만약 빈 블록이 필요할 경우 빈 블록 리스트의 첫 번째 블록을 사용하게 된다. 이 방법은 빈 블록에 대한 리스트를 유지해야 하는 오버헤드를 가지고 있으며 빈 블록 리스트에 정렬하여 삽입해야 하는 단점을 가지고 있다.

소거 시간 기반 소거 횟수 평준화 기법[4]은 우선 기존 플래시 메모리의 구조를 수정하였으며 소거 횟수를 사용하여 소거 횟수 평준화를 하는 것이 아니라 블록이 소거되는 시간을 사용한다. 블록의 소거 횟수가 증가될수록 소거 시 걸리는 시간이 증가한다는 전제로 실제 소거 경과 시간이 정해둔 소거 경과 시간보다 크면 블록을 교체하는 방법을 취하고 있다. 이 방법은 각 블록마다 소거 경과 시간을 유지해야 한다는 단점을 가지고 있다. 또한 이런 참조되는 소거 경과 시간은 플래시 메모리에 따라서 달라질 수 있어서 미리 결정해야 하며 소거 경과 시간의 결정이 중요하다.

점수(score) 기반 소거 횟수 평준화 방법[5]은 소거되는 블록을 선정할 때 점수로 선정하는 방법을 취하고 있다. 점수는 아래와 같은 식을 통해서 계산되고 최고 점수를 가진 블록을 선택한다.

$$\text{score}(j) = 0.8 \times \text{obsolete}(j) + 0.2 \times \left(\max_i\{\text{erasures}(i)\} - \text{erasures}(j)\right)$$

이 때 0.8과 0.2는 공간 측정으로 결정되며 0.8의 함수는 다른 블록으로 복사할 비용이 적은 블록을 의미하며 0.2의 함수는 소거 횟수가 가장 적은 블록을 선택한다. 이 때 핫 블록과 콜드 블록 소거 횟수의 차이가 500 이상이면 다음의 평준화-우선(wear-heavy) 정책을 사용한다.

$$\text{score}'(j) = 0.2 \times \text{obsolete}(j) + 0.8 \times \left(\max_i\{\text{erasures}(i)\} - \text{erasures}(j)\right)$$

점수 기반 소거 횟수 평준화 기법은 각 블록의 점수를 결정하기 위해서 전체 블록과 비교하므로 계산 비용이 많이 든다.

트랜잭션 기반 플래시 메모리 파일 시스템[6](Transactional Flash Memory File System, TFFS)을 위한 소거 횟수 평준화 기법은 상대적으로 사용 빈도가 높은 맵 영역과 교대 영역의 위치를 주기적으로 이동시킴으로써 특정한 블록들이 맵과 교대 영역으로서 집중적으로 사용되는 것을 방지한다. 이 때 영역 이동에 의해 변경되는 교대 영역과 데이터 영역의 위치는 맵 영역에 기록되면, 변경되는 맵 영역의 위치는 별도의 고정된 블록에 기록하는데 플래시 메모리가 대용량화되면 영역 이동 기법에 따른 비용이 커지게 되는 문제가 있다. 위와 같이 플래시 메모리 파일 시스템에서의 소거 횟수 평준화 기법은 플래시 메모리 파일 시스템을 구축해야 하는 단점을 가지고 있으며, 더욱이 이런 파일 시스템을 구축해야 하는 것은 플래시 메모리 파일 시스템에 종속적일 수밖에 없다는 것을 의미한다.

3. KM-평준화(KM-leveling) 기법

본 논문에서는 플래시 메모리 시스템에서 여러 FTL(Flash Translation Layer)[7] 알고리즘에 쉽고 효율적으로 적용 가능한 KM-평준화 기법을 제안하고 구현한다. 또한 각 블록에 대한 정보를 작은 저장 공간을 사용하여 소거 횟수 평준화를 할 수 있도록 한다. KM-평준화 기법은 다음의 5가지의 주안점을 가지고 설계하며 K-leveling을 개선한 효율적인 소거 횟수 평준화 기법이다. KM-평준화를 적용한 컴팩트 플래시 시스템의 전체 구조는 그림 1과 같다.

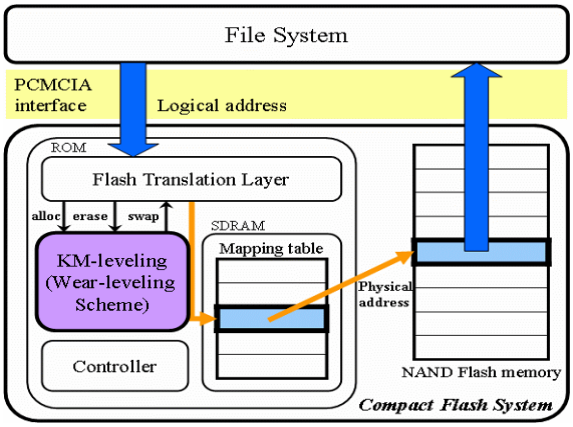

그림 1. KM-leveling Architecture

3.1 소거 레벨(erase level)

KM-평준화 기법은 각 블록에 대한 소거 횟수(age)를 가지는 것이 아니라 소거 레벨을 가지며 이러한 레벨을 사용하여 소거 횟수 평준화를 이룬다. KM-평준화는 전체 블록의 각 소거 횟수의 차이가 M값 이내이도록 하는 것이 목적이다. 따라서 M값은 각 블록의 소거 횟수가 가질 수 있는 차이의 최대값을 의미하고 K값은 M값 이내에 존재하는 임계(threshold)값을 의미한다. 이와 같이 소거 레벨을 둠으로써 블록에 대한 소거 횟수를 연산해야 하는 공간 오버헤드를 줄일 수 있다.

각 블록이 소거되면 레벨은 증가되며, K값으로 소거 연산이 잦은 블록(레벨이 높은 블록)과 그렇지 않은 블록을 판단하고 이 블록들을 교체해서 소거 횟수 평준화한다. 이런 K값이 M값 내에서 변하면서 소거 횟수 평준화를 하게 된다. 다음은 M값과 K값을 정의한 것이다.

K_0: 레벨 초기값
$K_0 \le K_i < M (0 \le i < n$인 정수$)$
$M = K_n$

M값 내에서 K값이 변경(variable K)되는 이유는 블록의 소거 레벨이 최대 레벨을 초과하면 소거 레벨에 대한 일관성을 잃는다. 따라서 K값은 고정된 M값 사이에서 가변적으로 변하면서 소거 횟수의 평준화를 이룬다.

3.2 업-레벨링(up-leveling)과 다운-레벨링 (down-leveling)

블록에 대한 소거 연산으로 인한 소거 레벨이 증가하면 어떤 블록은 K값을 초과할 수 있다. 따라서 K값을 초과할 경우 M값 범위 내에서 K값을 증가시켜야 한다. 또한 최소 레벨의 블록이 존재하지 않을 경우 K값은 감소해야 한다. 그림 2는 업-레벨링과 다운-레벨링에 대한 동작도를 나타낸 그림이다. 그림 2의 a)는 초기 KM-평준화가 진행되고 있는 상태이다. 그러다가 b)와 같이 업-레벨링이 필요한 시기에는 Max_Level을 증가시키고 K값을 증가시키게 된다. 그리고 다운-레벨링이 필요할 때에는 그림 2의 c)와 같이 Min_Level을 증가시키고 K값을 감소시키게 된다.

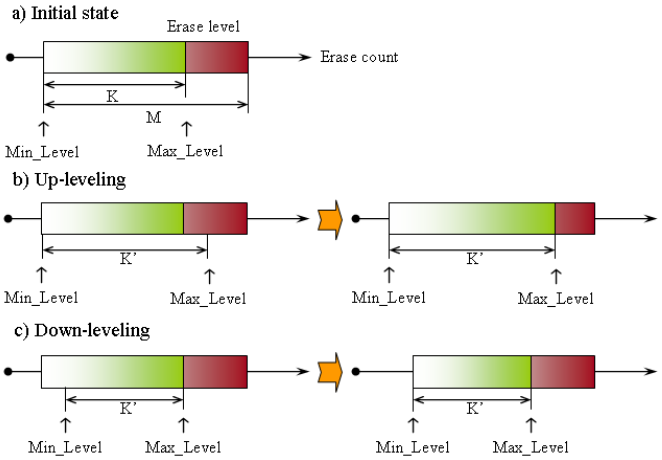

그림 2. 업-레벨링과 다운-레벨링 동작도

K값이 증가하게 되는 경우는 블록의 교체 시 콜드블록은 존재하나 핫블록이 없을 경우이다. 즉 최대 레벨(max level)인 블록, 맥스블록만 존재할 경우를 의미한다. 맥스 블록은 다음과 같이 정의한다.

X: 맥스 블록, $K-1$ 레벨인 블록

K값이 감소하게 되는 경우는 최소 레벨의 빈 블록과 콜드블록이 존재하지 않은 경우이다. 따라서 다운-레벨링은 업-레벨링으로 인해 증가된 K레벨을 줄여 레벨의 균형을 맞춘다. 다운-레벨링을 하는 이유는 앞에서 설명한 것과 같이 X레벨로 상승하였을 때 다음 레벨로 상승할 수 없게 되어 상태에 대한 명확성을 잃게 되는 것을 방지하기 위해서이다. 이런 업-레벨링과 다운-레벨링의 프로세싱은 전체 시스템에 처리 속도에 영향을 거의 주지 않는다.

Erase Level	The number of free block	The number of used block
M-1		
...		
K-1	n(X)	$n(R_{K-1})$
K-2	n(H)	$n(R_{K-2})$
K-3	$n(F_{K-3})$	$n(R_{K-3})$
...		
2	$n(F_2)$	$n(R_2)$
1	$n(F_1)$	$n(R_1)$
0	$n(F_0)$	n(C)

K_i

M

그림 3. KM-평준화를 위한 자료구조

그림 3은 KM-평준화를 위한 자료구조이다. 최대 M레벨이내에 K레벨이 업/다운-레벨링을 한다. 각 레벨에 따른 빈 블록과 할당된 블록의 개수를 유지하며 블록을 할당할 때 최소 레벨의 빈 블록을 할당하기 위해 사용된다. 또한 핫블록과 콜드블록을 교체할 때, 그림 2의 정보를 가지고 간단하게 판단하며 교체할 수 있다. 더욱이 이와 같은 정보를 활용하여 효율적으로 업/다운-레벨링을 할 수 있도록 한다.

3.3 공평한 빈 블록의 할당

KM-평준화 기법은 소거 횟수 평준화를 위해 FTL 알고리즘 중 빈 블록을 할당할 때 공평하게 빈 블록을 할 수 있도록 한다. 이와 같이 공평하게 빈 블록을 할당 하는 것이 중요한 이유는 소거 레벨을 고려하여 할당해서 각 블록이 비슷한 레벨에서 소거될 수 있도록 한다.

공평하게 빈 블록을 할당하기 위해서 순차적으로 할당하는 방식(wrap-around)을 사용한다. 순차적 할당 방식인 랩-어라운드 방식은 빈 블록을 순차적으로 할당하는 방식을 말하는데, 빈 블록을 찾을 때에는 블록의 처음부터 찾는 것이 아니라 이전에 할당된 블록의 다음 블록부터 순차적으로 찾아 빈 블록을 할당하는 방식을 말한다. 하지만, 이런 랩-어라운드 방식은 소거 레벨에 대한 전혀 고려하지 않기 때문에 소거 횟수 평준화를 위해서 각 블록의 소거 레벨을 고려하여 할당하는 방법이 필요하다. 따라서 빈 블록을 만나게 되면 바로 할당하는 것이 아니라 소거 레벨을 고려하여 할당하게 된다.

$$F_i : i \text{ 레벨의 빈 블록}$$
$$0 \le i < K-3$$
$$R_i : i \text{ 레벨의 할당된 블록}$$
$$0 \le i < K-3$$

빈 블록은 위와 같이 K-3의 값을 넘지 않는 블록을 의미하며 F_0부터 할당될 수 있도록 노력한다. 또한 이런 빈 블록을 빠르게 찾기 위해서 FreeQ 큐를 사용한다.

이와 같이 빈 블록을 공평하게 할당하는 것은 소거 레벨을 고려하면서 할당하여 차후에 K값을 넘어서는 블록을 할당하는 것을 최소화할 수 있다.

3.4 소거 레벨에 따른 블록의 교체

블록의 소거 레벨이 높으면서 빈 블록을 핫블록(hot block)이라고 하며 소거 레벨이 최소이며 할당된 블록을 콜드블록(cold block)이라고 한다. 이와 같은 핫블록과 콜드블록은 다음과 같이 정의된다.

H: 핫 블록, $K-2$ 레벨인 빈 블록
C: 콜드 블록, 0 레벨인 할당된 블록

핫블록은 소거 연산이 많이 이루어져 높은 레벨을 가지므로 빈 블록으로 할당하지 않고, 콜드블록과 교체하게 한다. 핫 블록은 교체 시 빠르게 찾기 위해서 HotQ 큐(Queue)를 사용한다. 콜드블록은 할당된 블록 중에서 소거 레벨이 0인 블록으로써 소거에 대한 연산이 드문 블록(read-only)이고 핫블록과 마찬가지로 ColdQ 큐에 쌓이게 된다. 교체작업을 하게 되면 강압적으로 소거 연산(forced-erase)이 이루어지므로 추가적인 비용을 초래한다. 하지만 블록에 대한 교체 작업은 특정 블록에 대한 소거 연산의 집중을 막아서 소거 횟수 평준화를 하게 한다.

알고리즘 1은 소거 레벨에 따른 블록의 교체의 프로시저를 나타낸 것이다. 핫블록과 콜드블록의 존재유무에 따라서 해당되는 작업을 하게 된다.

Algorithm 1: Swap operation

```
Output: F : free block

procedure Swap()
if hot and cold blocks exist then
    F ← KM_swap()
    return F
else if hot blocks exist
        and cold blocks not exist then
    down_leveling()
    if free blocks exist then
        F ← FreeQ.dequeue()
        return F
    end if
else if hot blocks not exist
        and cold blocks exist then
    K++      // up-leveling
    if hot blocks exist then
        F ← KM_swap()
        return F
    else
        return error      // memory full
    end if
else     // hot and cold blocks not exist
    down-leveling
    if free blocks exist then
        F ← FreeQ.dequeue()
        return F
    else
        if hot blocks exist then
            F ← KM_swap()
            return F
        else
            return error      // memory full
        end if
    end if
end if
end procedure
```

Algorithm 2: KM-swap operation

```
Output: CB: cold block
// HB: hot block

1: procedure KM_swap()
2: HB ← HotQ.dequeue()
3: CB ← ColdQ.dequeue()
4: // swap scheme depend on FTL algorithm
5: FTL_swap(HB, CB)
   return CB
   end procedure
```

그리고 실제 핫블록과 콜드블록을 교체는 알고리즘 2에서와 같이 동작하게 된다. 핫블록과 콜드블록을 선정하고 각 FTL에 해당하는 교체 정책에 따라서 교체한다.

3.5 정보의 유지

블록에 대한 상태와 레벨에 대한 정보의 유지를 위해 KM-평준화에서는 2가지 방법을 사용한다.

policy 1. 정보 블록(information block) 사용
policy 2. 여유 공간(spare area) 사용

첫 번째는 물리적인 블록 테이블에 대한 정보는 정보 블록(information block)에 상주시키는 방법을 이용하고

두 번째 방법으로 각 블록의 K 값은 그 블록의 헤더(header)에 저장한다. 이렇듯 두 가지 방법을 두는 이유는 정보의 안정성과 신뢰성을 높이기 위한 것이다. 각 블록에 대한 정보를 갱신하는 시기는 첫 번째 정책의 경우에는 다운-레벨링 시 갱신을 하게 되며 한 블록에만 정보를 쓰는 것을 막는다. 두 번째 정책의 경우에는 소거 연산이 수행될 때 소거 레벨을 기록한다. 두 경우에 대한 정보의 일관성(consistence) 문제는 시간정보(time-stamp)를 기록하여 두 정보 중 낮은 값을 선택하게 하여 최신 정보를 알아낸다.

정보의 일관성을 유지하는 것은 예기치 못한 상황이 발생했을 경우 복구(recovery)하기 위해서 중요하다. 즉 시스템의 예기치 못한 종료는 KM-평준화에서 소거 횟수에 대한 차이를 증가시켜 이후에는 KM-평준화의 안정성을 보장받지 못하는 결과를 낳을 수 있다. 정보의 일관성 유지는 중요한 요소 이지만 본 논문에서는 논의하지 않겠다.

4. 실험 및 성능 평가

4.1 실험 개요

앞서 설명한 대로 5가지를 주안점을 두고 KM-평준화를 제안하였다. 하지만 본 논문에서는 실험 결과에 영향을 미치는 변화 요인을 제한하고 다음과 같은 가정으로 실험하였다. 그 이유는 많은 변화 요인으로 실험 결과에 따른 성능 평가 결과에 대한 분석이 어려워질 수 있기 때문이다.

4.1.1 실험 제한 및 가정

(1) 공평한 블록 할당

빈 블록을 공평하게 할당하는 것은 앞서 설명하였듯이 소거 횟수 평준화를 이루는데 중요한 요인 중에 하나이다. 앞으로 실험 환경에서는 본 논문에서 제안하는 KM-평준화와 성능 비교를 위해 블록을 할당하는 방법으로 KM-평준화를 적용하지 않은 랩-어라운드와 비교한다. 그리고 FreeQ 큐의 크기는 10개로 두어서 실험하였다.

(2) 소거 레벨에 따른 블록의 교체

블록의 교체 전략은 FTL 알고리즘마다 블록의 특성 및 용도에 따라 다를 수 있다. 본 논문에는 FTL 알고리즘 중 로그 블록 기법[9](log block scheme)중 FAST(Full Associative Sector Translation)[10] 알고리즘을 가지고 성능 평가를 수행하였다. FAST 알고리즘의 블록 교체 정책은 4.1.2절에서 설명하도록 하겠다. 또한, 소거 레벨에 따른 블록의 교체에서 HotQ 큐와 ColdQ 큐의 크기를 3개로 두고 실험하였다.

(3) 정보의 유지

본 논문에서는 KM-평준화에 대한 성능평가에 초점을 두었으며, 예기치 못한 시스템 종료로 인한 정보의 일관성 유지에 대한 문제는 다루지 않는다.

4.1.2 로그 블록 기법(FAST)의 경우 교체 정책

FTL 알고리즘 중에서 로그 블록 기법중 하나인 FAST 알고리즘을 선택한 이유는 가장 성능이 우수하기 때문이다[11]. FAST는 블록의 특성 및 용도에 따라 3가지로 나뉜다. 데이터 블록(data block), 임의쓰기용 로그 블록(random log block), 그리고 순차쓰기용 로그 블록(sequential log block)으로 나뉜다. 따라서 각 블록의 특성과 용도에 따라서 교체해야 한다. FAST 뿐 아니라 다른 FTL 알고리즘에도 쉽게 적용이 가능하다.

4.2 실험 환경

소거 횟수 평준화 기법의 성능을 비교 평가하기 위하여 KM-평준화를 적용하지 않는 것과 K-평준화 기법, 그리고 KM-평준화 기법의 결과를 비교하였다. 또한 성능 평가를 위해 표 2과 같은 테스트 작업을 수행하였다. 플래시 메모리는 6,600개의 블록(약 100M)을 사용하였다. 로그 파일은 다중 프로세서가 JPG와 MP3파일을 복사 혹은 삭제에 대한 디스크의 랜덤 I/O를 추출한 로그 파일이다. 현재 실험에 사용한 로그파일은 NTFS 파일 시스템으로 임의의 데이터에 대한 I/O를 수행하여 실제 물리적인 플래시 메모리에 대해서 각 블록 당 소거 횟수를 살펴보고 성능의 차이를 시뮬레이션하였다.

표 2. 테스트 작업의 구성

로그파일 형태	NTFS file system 랜덤형 데이터(JPG/MP3)
테스트 FTL 알고리즘	FAST
플래시 메모리 크기	111,513,600Bytes(6,600 blocks) 95%이상의 워크로드
소거 레벨	K_0 : 16 / M : 128

위 실험 환경을 테스트하기 위해서 로그정보를 읽어 FTL 알고리즘의 성능을 평가하는 도구인 FTL APAT(FTL Algorithm Performance Analysis Tool)[11]를 사용하여 KM-평준화를 적용하지 않은 것과 K-평준화, 그리고 KM-평준화 기법을 FAST 알고리즘에 적용하여 실험하였다.

4.3 실험 결과

그림 3~5의 그래프는 물리적인 블록의 소거 횟수를 점으로 나타낸 그래프이다. 그림 3은 KM-평준화를 적용하지 않은 결과이며 그림 4는 K-평준화를 적용한 그래프이다. 그리고 그림 5는 KM-평준화를 적용한 그래프이다. 각 결과 그래프는 블록 당 소거 횟수를 표현한 그래프를 나타낸다.

4.4 성능 분석

표 3. KM-평준화 성능평가 결과

Wear-leveling	Log	TE	AE	σ
no KM-leveling	NTFS/랜덤	276,902	41.95	**9.04**
K-leveling	NTFS/랜덤	310,423	47.03	**3.92**
KM-leveling	NTFS/랜덤	281,503	42.65	**3.91**

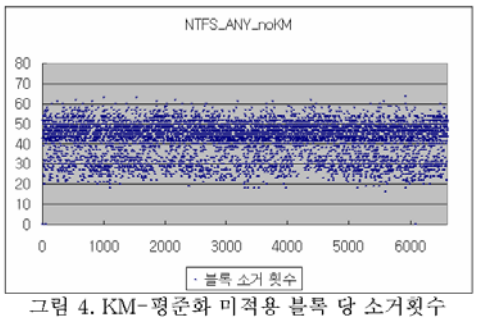

그림 4. KM-평준화 미적용 블록 당 소거횟수

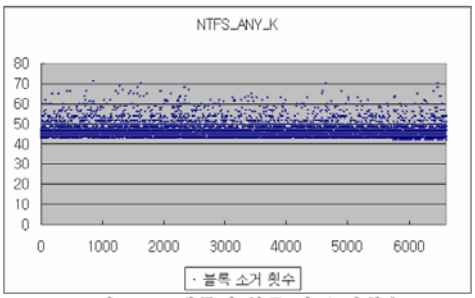

그림 5. K-평준화 블록 당 소거횟수

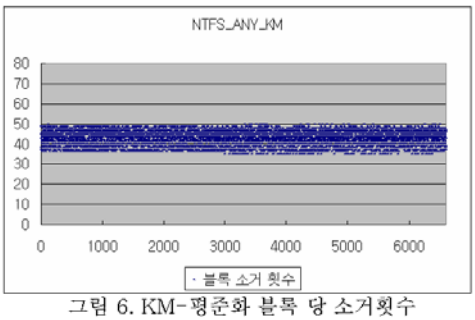

그림 6. KM-평준화 블록 당 소거횟수

표 3은 FAST 알고리즘에 KM-평준화 미적용한 것과 K-평준화, 그리고 KM-평준화 기법을 성능 평가한 결과를 표로 나타낸 것이다. 각 평준화 기법에 따른 결과를 나타낸 것인데 소거 총합(TE)은 전체 블록에 대한 전체 소거 횟수를 의미하고 소거 평균(AE)은 각 블록 당 소거 횟수의 평균을 의미한다. 표준편차(σ)는 각 블록의 소거 횟수에 대한 표준편차를 의미한다. 즉, 표준편차가 낮을수록 각 블록에 대한 소거 횟수의 차이가 나지 않는 다는 것을 의미한다.

현재 표 3의 결과로 비춰볼 때 KM-평준화를 적용하지 않은 것과 K-평준화 기법보다 KM-평준화 기법이 우수하다는 것을 알 수 있다. 특히 K-평준화보다 KM-평준화 기법의 성능이 우수한 이유는 K-평준화는 고정된 K값이기 때문에 TE를 증가시키고, 소거 레벨이 K값을 초과하게 되어 일관성을 잃기 때문이다. 표 3에서 보는 바와 같이 블록의 교체로 인한 강제적 소거로 플래시 메모리 시스템에 약 2%정도의 성능 저하가 나타나며 소거 횟수 평준화는 약 56%정도(표준편차 기준) 향상된 결과를 알 수 있다.

5. 결론 및 향후 연구 과제

플래시 메모리는 블록 당 소거 횟수의 한계를 가지고 있기 때문에 플래시 메모리에 대한 소거 횟수 평준화 기법이 필수적이다. 이런 소거 횟수 평준화 기법은 플래시 메모리의 안정성을 보장하게 한다. 플래시 메모리의 대용량화에 따른 전체 블록의 계산 비용을 최소화하고 소거 횟수에 대한 정보를 적은 공간을 이용하여 관리하는 기법이 필요하다. 본 논문은 전체 블록에 대한 레벨을 두어 공간을 적게 차지하며 M값 범위 이내에 각 블록의 소거 횟수들이 존재하도록 보장한다. 그리고 FreeQ, HotQ, ColdQ 큐를 두어 블록을 찾는데 걸리는 비용을 줄이며, 업/다운-레벨링의 비용을 최소화하였다.

향후 연구로는 M값을 결정하는 연구가 필요하다. M값은 전체 블록 중 사용하지 않는 빈 블록의 개수와 관계가 있다. KM-평준화는 M값 이내에 각 블록의 소거 횟수들이 존재함을 보장해야 하기 때문에 M값을 산출하기 위해서는 KM-평준화를 위한 여유 블록을 얼마만큼 두어야 하는지를 연구되어야 할 것이다. 또한 시스템의 예기치 못한 종료로 인하여 정보의 일관성 유지를 위한 구현에 대해서도 고려해 보아야 하며, FTL 알고리즘 중 FAST 알고리즘뿐 아니라 다른 알고리즘에 적용하여 성능 평가해봐야 할 것이다.

참고문헌

[1] 이태훈, 이상기, 정기동, "임베디드 플래시 파일 시스템을 위한 플레인 지움 정책", 제32회 추계학술발표회 논문집, pp 778-780

[2] Lofgren et al. "Wear Leveling Techniques for Flash Memory EEPROM Systems", United States Patent, no. 6,594,183, 2003

[3] Jou et al. Flash Memory wear leveling system providing immediate direct access to micoroprocessor, United States Patent, no. 5,568,423, 1996

[4] Han. Flash Memory Wear Leveling System and Method, United States Patent, no. 6,016,275, 2000

[5] Wells. Method for Wear Leveling in a Flash EEPROM Memory, United States Patent, no. 5,341,339, 1994

[6] 배영현, 최종무, 이동희, 노삼혁, 민상렬, "플래시 메모리 파일 시스템을 위한 효율적인 소거 횟수 평준화 기법", 한국정보과학회 가을 학술발표논문집, pp580-582, 2004

[7] Tae-Sun Chung, Dong-Joo Park, Sangwon Park, Dong-Ho Lee, Sang-Won Lee, and Ha-Joo Song, System Software for Flash Memory: A Survey, EUC 2006, LNCS 4096, Aug. 2006

[8] 김도윤, 유현석, 박성환, 박원주, 박상원, K-평준화: 플래시 메모리의 효율적인 소거 횟수 평준화 기법, 한국정보과학회 학술대회, 2005년 7월

[9] Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min, and Yookun Cho. A space-efficient flash translation layer for compact flash systems. IEEE Transactions on Consumer Electronics, 48(2), 2002

[10] Sang-Won Lee, Dong-Joo Park, Tae-Sun Chung, Dong-Ho Lee, Sangwon Park and Ha-Joo Song, A Log Buffer based Flash Translation Layer using Fully Associative Sector Translation, ACM Transactions on Embedded Computing Systems, Vol.6, No. 1, Feb. 2007

[11] 박원주, 유현석, 박성환, 김도윤, 박상원, 윈도우즈 파일 시스템에서 플래시 메모리의 FTL 알고리즘 성능 분석, 한국정보과학회 학술대회, 2005년 7월