Journal of The Korea Society of Computer and Information
Vol. 20 No. 11, pp. 1-8, November 2015

www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.11.001

Wear Leveling Technique using Bit Array and Bit Set Threshold for Flash Memory

Seon Hwan Kim*, Jong Wook Kwak**, Chang-Hyeon Park***

Abstract

Flash memory has advantages in that it is fast access speed, low-power, and low-price. Therefore, they are widely used in electronics industry sectors. However, the flash memory has weak points, which are the limited number of erase operations and non-in-place update problem. To overcome the limited number of erase operations, many wear leveling techniques are studied. They use many tables storing information such as erase count of blocks, hot and cold block indicators, reference count of pages, and so on. These tables occupy some space of main memory for the wear leveling techniques. Accordingly, they are not appropriate for low-power devices limited main memory. In order to resolve it, a wear leveling technique using bit array and Bit Set Threshold (BST) for flash memory. The proposing technique reduces the used space of main memory using a bit array table, which saves the history of block erase operations. To enhance accuracy of cold block information, we use BST, which is calculated by using the number of invalid pages of the blocks in a one-to-many mode, where one bit is related to many blocks. The performance results illustrate that the proposed wear leveling technique improve life time of flash memory to about 6%, compared with previous wear leveling techniques using a bit array table in our experiment.

▸ Keyword : Flash Memory, Wear Leveling, Storage Device, Block Erase Table, Bit Set Threshold, Bit Array

I. Introduction

NAND 플래시 메모리는 빠른 접근 속도를 보장하고 가격이 저렴하여 저장장치로 널리 사용되고 있다. 하지만 플래시 메모리는 제자리 덮어쓰기가 되지 않고(non-in-place update) 지우기 횟수에 제한이 있다는 단점이 있다. 제자리 덮어쓰기가 되지 않는 문제점은 플래시 메모리의 쓰기 단위와 지우기 단위가 다르고 쓰기 전에 지우기(erase before write)를 수행해야 하는 특징 때문에 발생하는 문제이다. NAND 플래시 메모리를 기준으로 쓰기는 페이지 단위로 되어 있고 지우기는 블록 단위로 되어 있으며 하나의 블록은 128~256개의 페이지로 이루어져 있기 때문에, 한 페이지를 덮어쓰기 위해 지우기 연산을 수행하면 같은 블록에 있는 다른 페이지들도 같이 지워져야 한다. 이런 비효율적인 방법은 기존의 하드 디스크를 기반으로 하는 파일 시스템과 응용 프로그램들을 제대로 구동할 수 없게 한다[1].

NAND 플래시 메모리에서 덮어쓰기를 구현하기 위해 데이터를 갱신할 때 갱신된 데이터는 다른 주소에 저장하고 기존의 데이터는 무효로 설정하여 사용한다. 그러면 데이터의 주소가 변경되기 때문에 갱신된 데이터의 주소를 테이블에 기록하여 추후 데이터를 접근할 때 참조한다. 이러한 주소 매핑 방식은 매핑 단위의 크기에 따라 페이지 매핑, 블록 매핑, 하이브리드 매핑 등의 3가지로 나누어진다. 이런 작업을 통해 하드 디스크 기반 파일 시스템과 응용 프로그램을 구동할 수 있게 하는 소프트웨어 계층을 FTL(Flash Translation Layer)이라고 한다.

• First Author: Seon Hwan Kim, Corresponding Author: Chang-Hyeon Park
*Seon Hwan Kim (amexist@ynu.ac.kr), Dept. of Computer Engineering, Yeungnam University
**Jong Wook Kwak (kwak@yu.ac.kr), Dept. of Computer Engineering, Yeungnam University
***Chang-Hyeon Park (park@yu.ac.kr). Dept. of Computer Engineering, Yeungnam University
• Received: 2015. 09. 07, Revised: 2015. 09. 24, Accepted: 2015. 10. 28.
• This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(No. NRF-2014R1A1A2057146).

FTL은 덮어쓰기를 지원하기 위해 사용되는 주소 매핑 기법이외에도 갑작스러운 시스템 고장에 대비한 복구 기능, 무효 페이지(invalid page)들을 수집하여 가용 페이지(free page)로 전환하는 가비지 컬렉션(garbage collection), 각 블록에 제한되어 있는 지우기 횟수를 평준화 하여 전체 플래시 메모리의 수명을 연장시키는 마모도 평준화(wear leveling) 등의 여러 기능들을 포함하고 있다[2-6].

마모도 평준화는 각 블록의 지우기 횟수, 콜드(cold) 및 핫(hot) 블록을 구분하는 정보, 각 페이지들의 참소 횟수, 참조 및 지우기 경과 시간 등의 여러 정보를 테이블에 저장하고, 이를 활용하여 플래시 메모리의 수명을 연장한다[7]. 하지만 여러 정보를 메모리에 저장하면 메모리의 사용량이 늘어나 저사양의 임베디드 시스템에 구현하기 힘들다는 단점이 있다. 그래서 비트의 배열로 구성되어 있어 메모리의 사용량을 줄일 수 있는 BET(Block Erase Table)을 이용한 마모도 평준화 기법이 연구되었다[8]. 또한 BET는 'k' 값이 커짐에 따라 여러 블록들을 하나의 그룹으로 묶어 각 비트에 대응시켜 사용할 수 있다. 그래서 'k' 값이 증가하면 메모리 사용량을 더욱 줄일 수 있다는 장점이 있다. 하지만 'k' 값이 증가할수록 마모도 평준화의 성능이 급격히 저하되는 단점이 있다.

이런 BET 기법의 단점을 해결하기 위해 본 논문에서는 성능 저하의 원인을 숨겨진 콜드 블록 문제(hidden cold block problem)로 지적하고, 이를 해결하기 위해 비트 배열과 비트 설정 임계값(BST: Bit Set Threshold)을 이용한 마모도 평준화 기법을 제안한다. 나머지 논문의 구성은 다음과 같다. 2장에서 마모도 평준화 관련연구들을 설명하고, 3장에서 제안하는 BST 기법을 서술한다. 4장에서 실험을 통해 BST 기법을 평가하고 5장에서 결론을 맺는다.

II. Related Works

1. Wear leveling techniques

플래시 메모리의 수명을 연장하기 위해 여러 마모도 평준화 기법들이 연구되었다. Dual-Pool은 콜드 블록과 핫 블록에 따라 2개의 그룹으로 나누고, 콜드 그룹의 첫 번째 블록과 핫 그룹의 마지막 블록을 서로 비교하여 임계값 이상이면 교체한다[9]. 지연 마모도 평준화 기법(Lazy Wear Leveling)은 블록의 지우기 횟수가 임계값 이상이면 바로 삭제를 하지 않고 추후 콜드 데이터를 해당 블록으로 옮긴다[10]. Rejuvenator는 각 페이지 별로 참조 횟수를 저장하여 콜드 페이지와 핫 페이지로 나누고, 각각 지우기 횟수가 많은 블록(old block)과 지우기 횟수가 적은 블록(young block)으로 이동시킨다[11]. OWL(Observational Wear Leveling)은 최근 참조된 블록들을 리스트로 저장하고 주기적으로 감시하여 자주 참조된 블록의 데이터를 가용 블록 중 가장 지우가 횟수가 적은 블록에 저장한다[12]. FeGC(Fast and endurant Garbage Collection)는 각 블록의 지워진 시간을 기록하여, 경과된 시간에 따라 각 블록의 비용을 계산하고 이를 이용하여 마모도 평준화를 수행한다[13].

Dual-Pool, 지연 마모도 평준화, Rejuvenator, OWL, FeGC는 각 블록의 지우기 횟수 제한이 동일하다는 점을 기준으로 하고 있다. 하지만 ERA(Efficient Reliability-Aware) 기법은 블록들이 모두 동일한 수명 기간을 갖는 것이 아니라 BER(Bit Error Rate)에 따라 실제 수명이 다르다는 것에 착안하여 BER를 기반으로 마모도 평준화를 수행한다[14].

이상의 기법들은 마모도 평준화를 수행하기 위해 블록 지우기 횟수 테이블, 참조 횟수 테이블, BER 테이블 등의 여러 테이블들을 사용한다. 따라서 일정 이상의 메모리 사용량이 요구되어 저사양의 임베디드 장치나 센서 노드에서는 구현 비용이 많이 든다는 단점이 있다.

2. BET

BET 기법은 메모리의 사용량을 효율적으로 줄여 저사양의 임베디드 장치에 사용될 수 있도록 제안되었다[8]. BET 기법은 정적 마모도 평준화(static wear leveling)를 수행하며, 이를 위해 각 블록의 지우기 연산 이력을 비트 배열로 구성되어 있는 테이블에 저장하고 이용한다. 'T'의 값으로 정해진 일정 기간 동안 각 비트에 대응하는 블록들이 지워지면 해당 비트를 '1'로 설정한다. 정해진 기간이 끝나면 '0' 비트로 설정된 블록들은 콜드 블록으로 인식하고, 이들을 강제로 이주하여 마모도 평준화를 수행한다. 또한 'k' 값에 따라 2^k 개의 블록을 하나의 비트에 대응시켜 테이블에 소요되는 메모리를 줄일 수 있다. 하지만 'k' 값이 커짐에 따라 여러 개의 블록을 하나의 비트에 대응시키면 그 그룹 내에 존재하는 블록들 중에서 콜드 블록을 구별하기 힘들어진다는 단점이 있다. 결국 누락되는 콜드 블록의 정보로 인하여 마모도 평준화의 성능은 급격히 떨어진다. 본 논문에서는 이 문제를 숨겨진 콜드 블록 문제로 정의한다.

III. Wear Leveling using Bit Array and Bit Set Threshold

1. Motivation

숨겨진 콜드 블록 문제는 하나의 비트에 대응하는 블록들을 하나의 그룹이라고 할 때, 그룹에 소속되어 있는 블록들 중 하나라도 삭제되면 관련 비트가 '1'로 설정되어 발생한다. 그러면 같은 그룹 내에 콜드 블록이 존재함에도 불구하고, 다른 블록의 삭제로 인해 콜드 블록이 없다고 인식하게 된다. 최악의 경우 모든 그룹에 핫 블록과 콜드 블록이 같이 존재하게 되면, 자주 지워지게 되는 핫 블록으로 인하여 모든 그룹들은 콜드 블록이 없는 것으로 인식되어 마모도 평준화가 수행되지 않는다.

그림 1은 숨겨진 콜드 블록 문제의 예를 나타낸 것이다. 플래시 메모리의 블록 개수가 16개이고, 'k'의 값이 2일 때 BET의 비트 개수는 4개가 된다. 그래서 4개의 블록이 하나의 비트

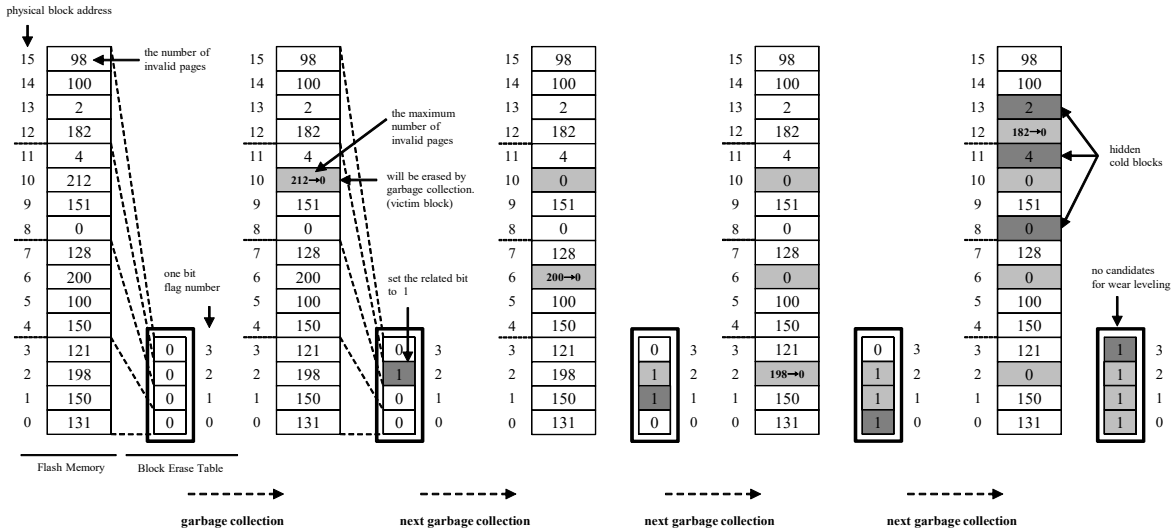


Fig. 1. Hidden cold block problem in BET technique. (k=2)

에 대응된다. 각 블록마다 무효 페이지의 개수가 기록되어 있으며, 이를 이용하여 가비지 컬렉션의 대상이 되는 희생블록을 선정한다. 일반적으로 탐욕 알고리즘(greedy algorithm)을 사용하여 희생블록을 선정한다. 그림 1의 초기 상태에서 첫 번째 가비지 컬렉션이 발생하면 전체 블록들 중에 가장 많은 무효 페이지를 가지고 있는 10번 블록이 희생블록으로 선정된다. 그런 후 가비지 컬렉션에 의해 희생블록은 지워지고 가용 블록으로 전환된다. 지워진 블록은 다시 무효 페이지 개수가 '0'으로 초기화 된다. 그리고 지워진 블록에 대응되는 BET의 2번 비트가 '1'로 설정된다. 두 번째 가비지 컬렉션이 발생하면 6번 블록이 희생 블록으로 선정되고 BET의 1번 비트가 '1로 설정된다. 결국 이와 같은 방식으로 나머지 가비지 컬렉션이 모두 수행되면 BET의 모든 비트는 '1'로 설정된다. 하지만 8번, 11번, 13번 블록은 무효 페이지의 개수가 작아 콜드 블록이 될 가능성이 높음에도 불구하고, BET의 모든 비트가 '1'로 설정되어 마모도 평준화를 수행할 대상이 없게 된다. 이렇게 콜드 블록이 존재함에도 불구하고 BET의 정보를 볼 때 콜드 블록이 없다고 인식하게 되어 마모도 평준화가 수행 대상이 없다면 마모도 평준화의 성능이 떨어진다.

특히 운영체제 데이터와 같이 갱신 빈도가 매우 낮은 데이터로 채워진 콜드 블록들이 대부분의 시스템에 존재하기 때문에 숨겨진 콜드 블록 문제가 발생할 가능성이 높다. 만약 콜드 블록으로만 구성되어 있는 그룹이 있다고 가정할 때 이 그룹은 BET의 비트가 '0'으로 유지될 수 있기 때문에 마모도 평준화의 대상이 된다. 하지만 이 그룹에서 하나의 블록을 마모도 평준화를 위해 옮기게 되면, 그 블록은 다른 그룹의 핫 블록들과 같이 존재하게 되어 숨겨진 콜드 블록이 된다. 또한 옮겨졌던 블록 위치는 가용 블록이 되고 그 블록에 핫 데이터가 저장되면 남겨진 콜드 블록들도 숨겨진 콜드 블록이 되어 성능이 저하된다.

이 문제를 해결하기 위해서는 그룹 내에 콜드 블록이 존재하면 BET의 해당 비트가 '1'로 설정되는 것을 막고 마모도 평준화에 해당 그룹을 참여할 수 있도록 유도해야 한다. 본 논문에서는 비트 설정 임계값(BST)을 두어 이를 해결하고자 한다.

2. Bit Set Threshold

BST는 다음의 식 (1)과 식 (2)에 의해서 계산된다. 각 블록(b_{index})은 'k'에 따라 특정 그룹($gn = (b_{index}/2^k)$)에 소속된다. BST는 가비지 컬렉션이 수행되어 희생 블록이 지워질 때 계산된다. 희생 블록이 소속된 그룹의 BST(BST_{gn})는 식 (1)에서 해당 그룹의 평균 무효 페이지 개수($AVG_{invalid}$)를 구하고 식 (2)에서 희생 블록의 무효 페이지 개수($VB_{invalid}$)에 그 값의 차이를 통해 구한다. 식 (1)에서 '$BN_{invalid}$ '는 그룹에 소속되어 있는 블록들의 무효 페이지 개수를 의미한다. 그리고 각 변수들이 가지는 값은 양의 정수로 한정한다.

$$AVG_{invalid}(n) = \frac{1}{2^k} \sum_{BN=2^k n}^{2^k(n+1)} BN_{invalid} \quad (1)$$

$$BST(gn) = VB_{invalid} - AVG_{invalid}(gn) \quad (2)$$

희생 블록의 무효 페이지 개수는 그룹에서 가장 많은 무효 페이지를 갖는 블록이 된다. 그래서 그룹의 평균 무효 페이지 개수와 희생 블록의 무효 페이지 개수의 차이가 크면 해당 그룹은 무효 페이지 개수의 분산이 클 가능성이 높다. 따라서 BST의 값이 커지면 핫 블록과 콜드 블록이 같이 존재할 가능성도 커진다.

3. Wear leveling operations

'k' 값에 따라 각 그룹에 대응되는 BET의 비트($BET[gn]$)는 식 (3)에 따라서 결정된다.

Algorithm 1: BST-GC
Input: k, f_{index}, b_{index}, BST, $BST_{prevent}$, $VB_{invalid}$, $MIN_{invalid}$, and $AVG_{invalid}$.
Output: A block related to b_{index} is cleaned, and $BST_{prevent}$ is updated by BST based on b_{index} and k.
/* if Garbage Collection (GC) is requested from SWL-Procedure which is SWL(Static Wear Leveling) trigger using BET, BST procedure is skipped */
1 $BST_{prevent}$ ← false;
/* a victim block is elected from one block in a group based on f_{index} and k. */
2 b_{index} ← GetVictimBlock(f_{index} , k); /* in this, greedy algorithm uses to elect a victim block.*/
3 **if** *GC is requested from SWL* **then** /* normal GC and BST procedure */
4 $VB_{invalid}$ ← GetInvalidCount(b_{index}); /* get the number of invalid pages of victim block */
5 $MIN_{invalid}$ ← GetMINInvalidCount(k); /* get the minimum number of invalid pages from a group based k */
6 $AVG_{invalid}$ ← GetAVGInvalidCount(k); /* get the average number of invalid pages from a group based k */
7 BST ← $MAX_{invalid} - AVG_{invalid}$; /* calculate BST*/
8 **if** $MIN_{invalid} < BST$ **then**
9 $BST_{prevent}$ ← true;
10 **end**
11 **end**
12 EraseBlock(b_{index});
13 **SWL-BETUpdateForBST( )**;

Fig. 2. Garbage collection algorithm using BST.

$$BET[gn] = \begin{cases} \text{not changed} & \text{if } BST(gn) > MIN_{invalid}(gn) \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

해당 그룹의 최소 무효 페이지 개수($MIN_{invalid}$)가 BST 보다 작다면 콜드 블록이 존재 한다고 인식하여, 가비지 컬렉션에 의해 그룹 내의 블록이 지워지더라도 BET의 비트가 '1'이 되는 것을 방지한다. 그래서 추후 마모도 평준화의 대상이 되도록 '0' 비트를 유지할 수 있도록 한다.

그림 2는 BST를 이용한 가비지 컬렉션의 알고리즘 *BST-GC*이다. 가비지 컬렉션이 발생하는 조건은 2가지로 나눌 수 있다. 무효 페이지를 회수하기 위해 호출되는 일반적인 상황과 콜드 블록을 이주시키기 위해 호출되는 마모도 평준화 수행 상황이다. 3번째 줄에서 BST는 일반적인 상황에서 사용되고

Algorithm 2: SWL-BETUpdateForBST
Input: e_{cnt}, f_{cnt}, k, b_{index}, $BST_{prevent}$, and BET
Output: e_{cnt}, f_{cnt}, and BET are updated based on the erased block address b_{index} and k in the BET mapping.
1 e_{cnt} ← e_{cnt} + 1 /* Increase the total erase count */
/* if BET updating is prevented by BST, it skips a operation*/
2 **if** $BST_{prevent}$ = false **then**
/* Update the BET if needed. */
3 **if** $BET[\ b_{index}/2^k\ ] = 0$ **then**
4 $BET[\ b_{index}/2^k\ ] \leftarrow 1$;
5 $f_{cnt} \leftarrow f_{cnt} + 1$;
6 **end**
7 **end**

Fig. 3. BET update algorithm for BST.

마모도 평준화가 수행될 때는 사용되지 않는다. 4~11번째 줄은 BST를 구하고 희생 블록이 속한 그룹에서 BST보다 낮은 무효 페이지 개수를 가지는 블록이 존재한다면, 해당 그룹의 비트가 '1'로 설정되는 것을 방지하게 위해 '$BST_{prevent}$' 변수에 'TRUE' 값을 저장한다. 그림 3은 각 블록이 지워질 때 해당 블록의 BET 비트를 설정하는 알고리즘 *SWL-BETUpdateForBST*이다. 알고리즘 *BST-GC*에서 설정된 '$BST_{prevent}$' 변수는 *SWL-BETUpdateForBST*에서 사용되며 2번째 줄에서 그 값이 'FALSE'일 경우만 BET의 비트를 '1'로 설정한다. 추후 마모도 평준화가 수행되면 BET에서 '0' 비트로 설정된 각 그룹 내에서 무효 페이지가 개수가 가장 작은 블록을 강제로 이주한다. 이 때 마모도 평준화가 수행되는 조건은 기존 BET 기법과 동일하다[8]. 마모도 평준화를 위해 선택된 희생 블록에 무효 페이지가 존재하지 않으면 가용 블록을 하나 선택하여 블록 복사로 이주하고, 무효 페이지가 존재하면 가용 블록 중에 하나를 SWL 전용 블록으로 설정하여 유효 페이지들을 이주한다. SWL 전용 블록은 다음 페이지가 써질 활성 블록(active block)과는 다른 것으로 오직 마모도 평준화에서 선택된 희생 블록의 유효 페이지만 이주해 올 수 있다.

그림 4는 BST를 이용하여 숨겨진 콜드 블록들을 마모도 평

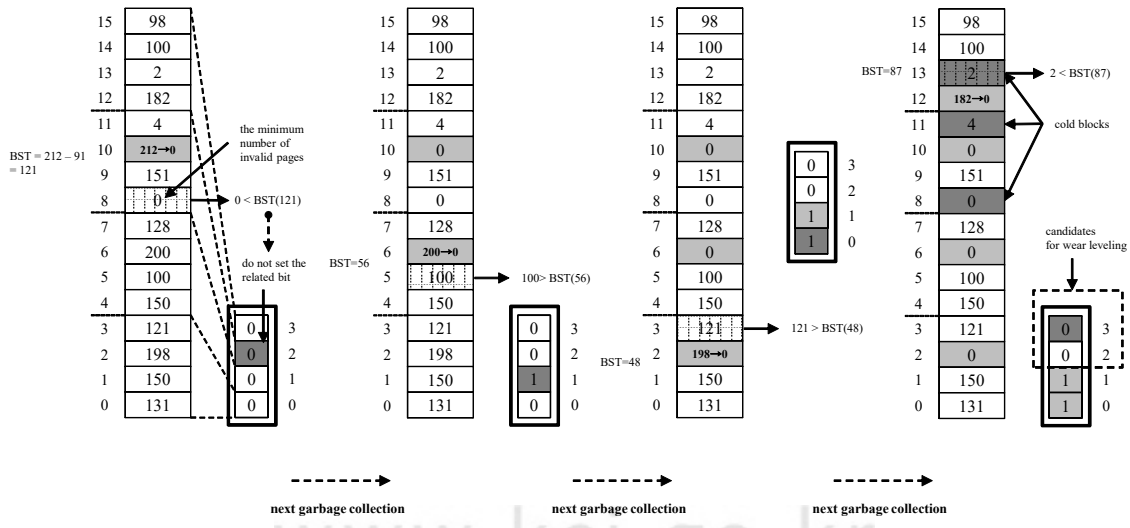


Fig. 4. BST to solve the hidden cold problem in BET technique. (k=2)

준화에 참여시키는 사례를 표시한 것이다. 첫 번째 가비지 컬렉션에서 10번 블록이 희생블록으로 선정되면 지워지기 전에 해당 그룹의 BST 값을 계산한다. BST 값은 소수점을 버리면 121이 된다. 8번~11번 블록 중에 무효 페이지가 가장 적은 블록은 8번 블록이며 무효 페이지의 개수는 0이기 때문에, BST를 비교할 때 BST보다 작아 관련된 BET의 비트를 '1'으로 설정하지 않는다. 두 번째와 세 번째 가비지 컬렉션에서는 계산된 BST보다 작은 무효 페이지 개수를 가지는 블록이 각각 존재하지 않기 때문에 각 BET의 관련 비트를 '1'로 설정하게 된다. 네 번째 가비지 컬렉션에서는 계산된 BST의 값이 87이고, 해당 그룹에서 가장 작은 무효 페이지 개수가 0이기 때문에 관련 BET의 비트는 '1'로 설정하지 않는다. 그림 4의 가비지 컬렉션들이 모두 완료되면 숨겨진 콜드 블록인 8번, 11번, 13번 블록들이 포함된 BET의 2번과 3번 비트가 '0' 비트로 남겨지게 된다. 이후 마모도 평준화가 수행되면 8번과 13번 블록이 대상이 되어 이주된다. 11번 블록은 앞서 설명한 방식처럼 수행될 경우, 다음 마모도 평준화에서 이주된다.

IV. Experiment and Evaluation

BST를 평가하기 위해 시뮬레이터를 구현하고 기존의 BET 기법과 비교하였다. 시뮬레이션에서 사용한 주소 매핑 기법은 페이지 매핑 기법을 사용하였고 5개의 시나리오 상황으로 나누어 BST를 평가하였다. 각 상황별로 총 40개의 파일 중에서 32개의 파일에 정규분포를 이용하여 갱신확률을 적용하였다. 나머지 8개의 파일은 운영체제와 같은 콜드 데이터를 반영하기 위해 갱신확률을 0%로 적용하였다. 그래서 파일들을 고려할 때 각 시나리오에서 콜드 데이터를 가지는 파일의 개수는 최소 8개 이상이 된다. 그리고 갱신확률의 최대가 100%로 되도록 보정하여 적용하였다. 정규분포로 계산된 갱신확률 'p'와 이를 보정한 갱신확률 'rp'는 각각 식 (4), (5)와 같다.

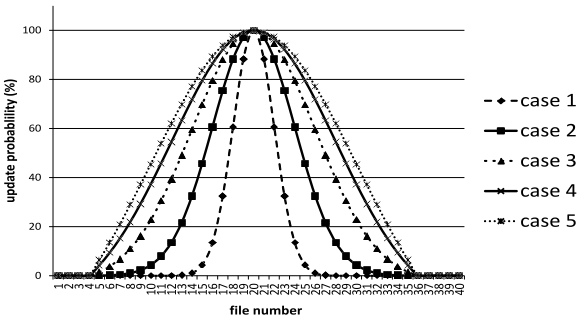


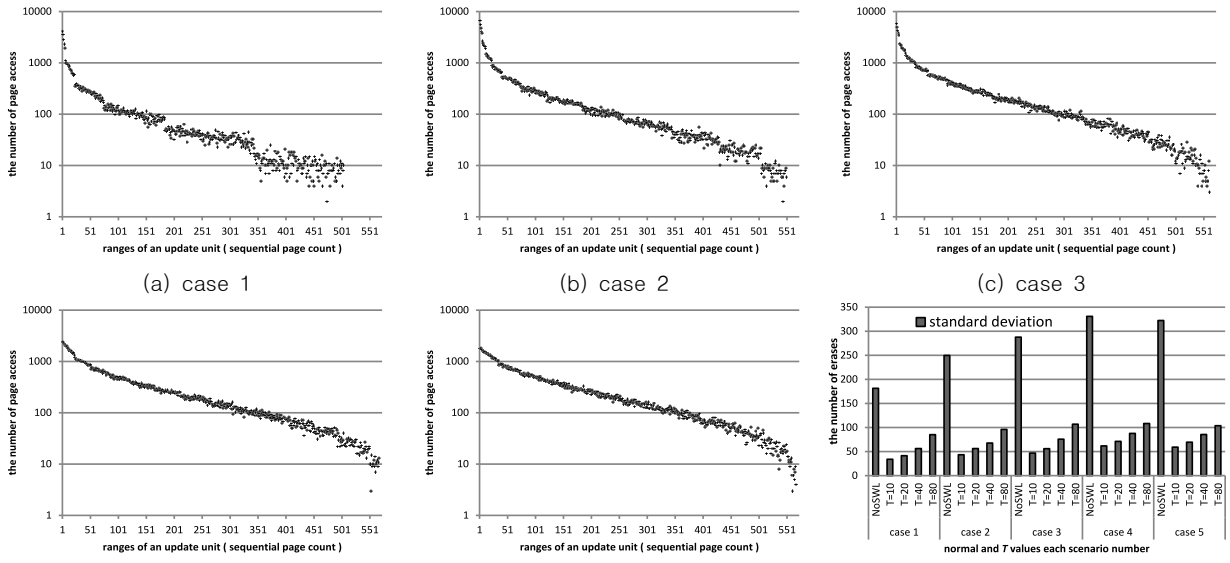
Fig. 5. Update probabilities of files by each case number for simulation.

$$p = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{4}$$

$$rp_x = p_x \frac{MAX(p) - MIN(p)}{100},\ p_x \in p \tag{5}$$

식 (4)에서 'x'는 1~32로 그리고 'u'는 16으로 설정하였다. 확률밀도 'σ'는 각 상황별로 2~10의 값을 적용하였다. 그림 5는 각 상황별로 40개의 파일에 적용된 갱신확률을 그래프로 표시한 것이다.

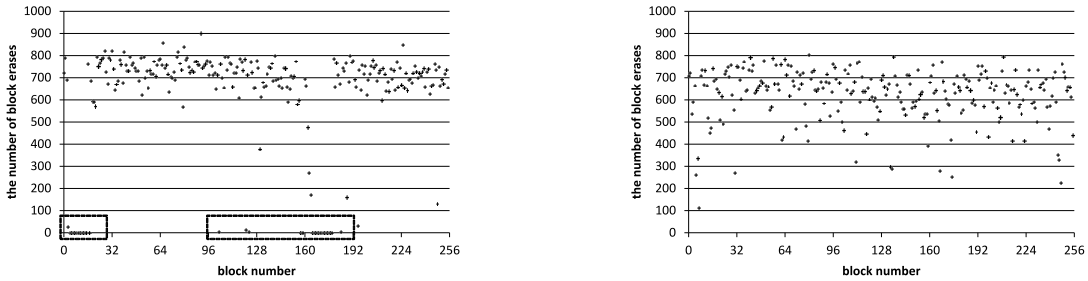
표 1은 실험환경의 상세사양을 나타낸 것이다. 각 파일의 크기는 모두 동일한 크기로 설정하였고 가비지 컬렉션은 가용 블



(a) case 1 (b) case 2 (c) case 3

(d) case 4 (e) case 5 (f) standard deviation (k=0)

Fig. 6. Distribution of update ranges composed sequential page units in each scenario number ((a)~(e)), and standard deviation of NoSWL and BET technique with different T values in each scenario number ((f)).



(a) BET (b) BST

Fig. 7. Distribution of the number of block erases by each technique in case 3. Dot line boxes indicate cold blocks, which do not participate in wear leveling. (k=1, T=40)

록이 약 2% 미만일 때 수행되도록 하였다.

Table 1. Details of the experimental environment

items	values and description
block count	256
page count per block	128
page size	4KB
total capacity	128MB
each file size	573 page (about 2.3MB)
file count	40
garbage collection trigger	the number of free blocks is under 5. (2%)

표 2는 실험에서 각 상황별로 마모도 평준화를 수행하지 않았을 때(NoSWL: No Static Wear Leveling) 나오는 각 수치를 보여주고 있다.

Table 2. Results of NoSWL by each scenario number the experiment.

	total EC	SD	avg. EC	total WS(MB)	total MPC
case 1	48071	181.4	187.7	22237.9	492800
case 2	105687	249.8	412.8	44222.3	2239638
case 3	146732	299.2	573.1	63722.3	2613327
case 4	168285	342.7	657.3	76019.2	2428173
case 5	189189	322.3	739	82425.4	3147831

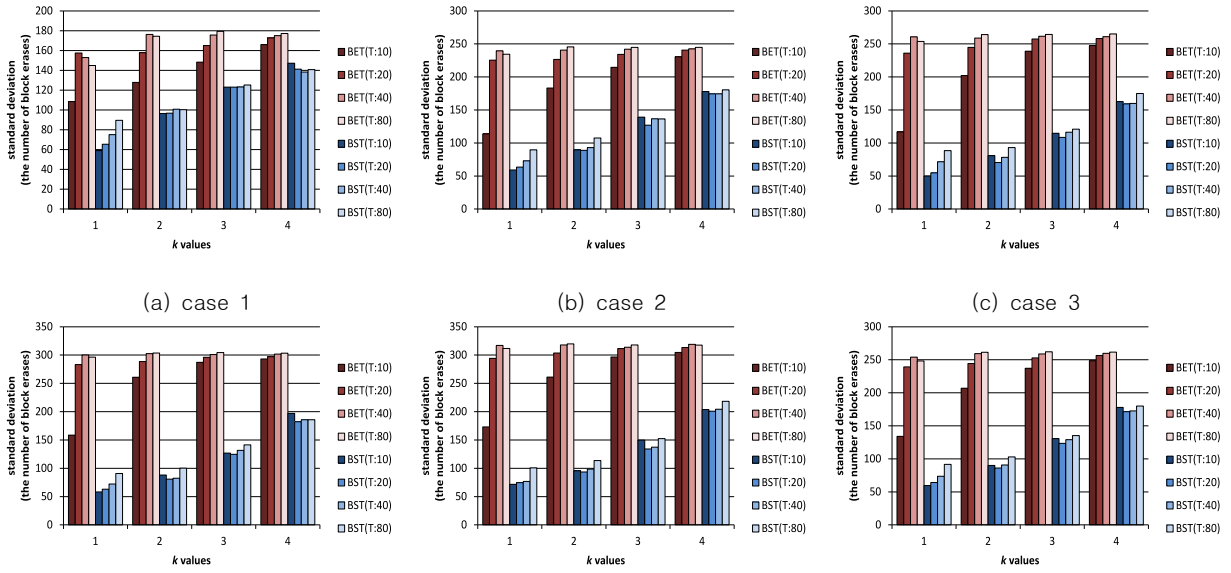
EC: Erase Count WS: Written Size
SD: Standard Deviation MPC: Migrated Page Count

표 2에서 보면, 시나리오 상황 1에서 5로 갈수록 전체 누적된 갱신확률이 높기 때문에, 갱신된 데이터의 용량이 많아진다. 한편, 상황 4와 상황 5를 비교할 때 상황 4의 표준편차가 더 크다는 것을 알 수 있다. 그 이유를 설명하면 상황 5가 상황 4보다 갱신확률이 전체적으로 높고 써진 데이터양이 많지만 콜드 블록의 개수는 동일하다. 그리고 이 2가지의 상황에서 콜드 블록들을 제외하고 나머지만 그림 5에서 비교할 때 갱신확률이 상황 4가 더 불균형한 상태임으로 상황 4의 표준편차가 상황 5의 표준편차보다 크게 된다.

그림 6은 각 상황별로 갱신 시 순차적으로 써진 페이지 단위의 분포와 'k'의 값이 0인 BET가 적용되었을 때의 표준편차를 보여주고 있다. 그림 6에서 (a)~(e)까지를 상호 비교 분석하였을 때, 순차적으로 써진 페이지 단위가 큰 것은 (a)에서 가장 적었으며 (e)로 갈수록 많다. 반대로 순차적으로 써진 페이지 단위가 작은 것은 (a)에서 (e)로 갈수록 적다. (f)를 따로 제시한 이유는 제안한 BST는 'k'의 값이 1이상일 때 적용이 가능하기 때문에 'k'의 값이 0일 때의 표준편차와 마모도 평준화를 적용하지 않은 일반적인 상황을 통해 각 시나리오 상황별 차이를 파악할 수 있고 이후에 나오는 실험결과와 비교하기 위해서다.

그림 7은 3번 상황에서 'k'의 값은 1이고 'T'의 값은 40일 때 BET와 BST에서 각 블록의 지우기 횟수 분포를 나타낸 것이다. BET만 적용된 (a)는 점선의 직사각형으로 표시된 부분과 같이 일부 블록들이 마모도 평준화에 포함되지 않아 지우기 횟수가 0인 블록들이 존재하였다. 하지만 BST가 적용된 (b)에서는 콜드 블록들이 마모도 평준화의 대상이 되어 대부분의 블록들의 지우기 횟수가 모두 0을 초과하였다. BET 기법에서는 (a)에서 지우기 횟수가 0인 콜드 블록들이 언젠가는 마모도 평준화에 대상이 될 수 있다고 언급한다[8]. 그러나 각 기법별로 최대 지우기 횟수를 기록하는 블록을 비교할 때 BET만 적용된 기법은 약 900회의 지우기가 발생하였으나 BST 기법은 약 800회의 지우기가 발생하여 BST 기법이 지우기 횟수가 약 100회 적었다. 따라서 콜드 블록들을 마모도 평준화 대상에 빠르게 포함시키는 BST가 전체적인 블록의 지우기 횟수의 차이를 더 줄이고 최대 지우기 횟수를 낮추어 플래시 메모리의 전체 수명을 연장시킬 수 있다는 것을 알 수 있다 .

그림 8은 각 상황별로 'k'의 값이 1이상일 때 각 기법별로 측정한 블록 지우기 횟수의 표준편차를 보여주고 있다. 'T'의 값이 커짐에 따라 표준편차가 커지는 것은 BET가 'T'의 값이 증가할수록 마모도 평준화의 수행 빈도가 낮아지기 때문이다. 각 상황에서 BST를 적용하였을 때, 기존의 BET보다 블록 지우기 횟수의 표준편차를 최대 약 75% 줄일 수 있었다. 그리고 BET는 동일한 'k'값에서 'T'값이 증가함에 따라 상대적으로 표준편차의 차이가 커지나 BST는 BET보다 표준편차의 차이가 작다는 것을 알 수 있다. 그래서 BST는 'T'값이 커지더라도 즉, 마모도 평준화 수행의 빈도수를 낮추더라도 상대적으로 균일한 성능을 얻을 수 있다. 또한 그림 8의 (c), (d)에서 'k'가 1이고



(a) case 1 (b) case 2 (c) case 3

(d) case 4 (e) case 5 (f) average of all cases

Fig. 8. Standard deviation of the number of block erases by each wear leveling technique.

'*T*'가 10일 때의 BST와 그림 6의 (f)에서 '*K*'가 0이고 '*T*'가 *10*일 때 BET의 표준편차가 비슷하다는 것을 알 수 있다. 결국 표준편차만을 고려할 때, 최선의 상황에서 BST가 BET의 메모리 사용량을 1/2만 사용하고도 비슷한 성능을 낼 수 있었다.

그림 9는 각 상황별로 각 기법에서 플래시 메모리의 수명 시간을 나타낸 것이다. 불량 블록이 생기는 시점에서 완료된 페이지 쓰기의 횟수를 시뮬레이션의 수명 시간으로 환산하여 측정하였다. 그리고 불량 블록은 지우기 횟수가 만 번을 넘은 블록으로 선정하였다. 그 결과 BST는 BET보다 수명을 최대 약 6%까지 늘릴 수 있었다. 각 상황을 비교하였을 때 갱신 확률이 0%인 콜드 블록이 많을수록 BET와 BST가 각각 성능이 좋아진다는 것을 알 수 있다. 그러나 BST가 BET보다 상대적으로 각 상황에 따라 성능향상이 균일하다는 것을 알 수 있다.

V. Conclusions

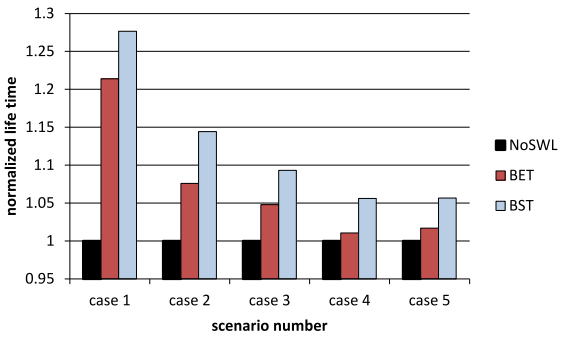


Fig. 9. Normalized elapsed time of BET and BST against the NoSWL (no static wear leveling) in each scenario number.(*T*=10, *k*=2)

본 논문에서는 플래시 메모리의 수명을 연장하기 위해 제안된 마모도 평준화 기법 중에 하나인 BET에서 '*K*'의 값이 1이상일 때, 콜드 블록 정보의 누락으로 성능이 급격히 낮아진다는 단점을 설명하였고, 그 원인으로 숨겨진 콜드 블록 문제를 지적하였다. 이를 해결하기 위해 비트 배열과 BST(Bit Set Threshold)를 이용한 마모도 평준화 기법을 제안하였다. 실험 결과 BST를 BET에 적용하였을 때 기존보다 블록 지우기 횟수의 표준편차를 최대 약 75% 줄일 수 있었으며, 수명을 약 6% 늘릴 수 있었다. BST는 테이블에 소모되는 메모리 사용량이 적고 동일한 환경에서 BET보다 플래시 메모리의 수명을 연장할 수 있어 메모리 사용량에 제한이 있는 임베디드 시스템과 센서 노드 등에 적합하다. BST 기법은 그룹을 구성하는 블록의 개수가 많을수록 BST의 계산 오버헤드가 많아진다는 단점이 있다. 따라서 이를 해결하는 방법을 향후 연구과제로 남긴다.

REFERENCE

[1] Lee, Sungjin, and Jihong Kim. "Improving Performance and Capacity of Flash Storage Devices by Exploiting Heterogeneity of MLC Flash Memory." Computers, IEEE Transactions, Vol. 63, No. 10, pp. 2445-2458, Oct. 2014.

[2] Chung, Tae-Sun, et al. "A survey of flash translation layer." Journal of Systems Architecture, Vol. 55, No. 5, pp. 332-343, May 2009.

[3] Kwon, Se Jin, et al. "FTL algorithms for NAND-type

flash memories." Design Automation for Embedded Systems, Vol. 15, No. 3-4, pp. 191-224, Dec. 2011.
[4] Ma, Dongzhe, Jianhua Feng, and Guoliang Li. "A survey of address translation technologies for flash memories." ACM Computing Surveys (CSUR), Vol. 46, No. 3, pp. 1-39, Jan. 2014.
[5] Jung-Hoon Lee. "Index block mapping for flash memory system", Journal of KSCI, Vol. 25, No. 8, pp. 23-30, Aug. 2010.(in Korean)
[6] Seon Hwan Kim, Jong Wook Kwak. "Garbage Collection Method using Proxy Block considering Index Data Structure based on Flash Memory", Journal of KSCI, Vol. 20, No. 6, pp. 1-11, Jun. 2015.(in Korean)
[7] Yang, Ming-Chang, et al. "Garbage Collection and Wear Leveling for Flash Memory: Past and Future." Smart Computing (SMARTCOMP), 2014 International Conference on. pp. 66-73, Nov. 2014.
[8] Chang, Yuan-Hao, Jen-Wei Hsieh, and Tei-Wei Kuo. "Improving flash wear-leveling by proactively moving static data." Computers, IEEE Transactions, Vol. 59, No. 1, pp. 53-65, Jan. 2010.
[9] Chang, Li-Pin. "On efficient wear leveling for large-scale flash-memory storage systems." Proceedings of the 2007 ACM symposium on Applied computing. ACM, pp. 1126-1130, Mar. 2007.
[10] Chang, Li-Pin, and Li-Chun Huang. "A low-cost wear-leveling algorithm for block-mapping solid-state disks." ACM SIGPLAN Notices, Vol. 46, No. 5, pp. 31-40, Apr. 2011.
[11] Murugan, Muthukumar, and David HC Du. "Rejuvenator: A static wear leveling algorithm for NAND flash memory with minimized overhead." Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on. IEEE, pp. 1-12, May 2011.
[12] Wang, Chundong, and Weng-Fai Wong. "Observational wear leveling: an efficient algorithm for flash memory management." Design Automation Conference (DAC), pp. 235-242, Jun. 2012.
[13] Kwon, Ohhoon, et al. "FeGC: An efficient garbage collection scheme for flash memory based storage systems." Journal of Systems and Software, Vol. 84, No. 9, pp. 1507-1523, Sep. 2011.
[14] Yang, Ming-Chang, et al. "New ERA: new efficient reliability-aware wear leveling for endurance enhancement of flash storage devices." Design Automation Conference (DAC), Mar. 2013.

Authors

Seon Hwan Kim received the B.S. and M.S. degrees in Computer Engineering from Yeungnam University, Korea, in 2006 and 2009, respectively. He is interested in NAND flash memory system, embedded system, and wireless sensor network.

Jong Wook Kwak received a B.S. degree in Computer Engineering from Kyungpook National University, Taegu, Korea in 1998, a M.S. degree in Computer Engineering from Seoul National University, Seoul, Korea in 2001, and a Ph.D. degree in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea in 2006. From 2006 to 2007, he worked as a senior engineer in the SoC R&D Center, at Samsung Electronics Co., Ltd. He is currently an associate professor in the Department of Computer Engineering, Yeungnam University. His research interests include advanced processor architecture, low-power mobile embedded system, and high performance parallel computing.

Chang-Hyeon Park received the B.S. degree in Electronics Engineering from Kyungpook University, Korea, in 1986 and M.S. and Ph.D degrees in Computer Science from Seoul University, Korea, in 1988 and 1992, respectively. Dr. Park joined the faculty of the Department of Computer Engineering at Yeungnam University, Gyeongsan, Korea, in 1993. He is currently a Professor in the Department of Computer Engineering at Yeungnam University. He is interested in artificial intelligence, data mining, and embedded system.