Journal of The Korea Society of Computer and Information
Vol. 22 No. 4, pp. 25-32, April 2017
www.ksci.re.kr
https://doi.org/10.9708/jksci.2017.22.04.025

# Garbage Collection Technique for Balanced Wear-out and Durability Enhancement with Solid State Drive on Storage Systems

Sungho Kim*, Jong Wook Kwak**

## Abstract

Recently, the use of NAND flash memory is being increased as a secondary device to displace conventional magnetic disk. NAND flash memory, as one among non-volatile memories, has many advantages such as low power, high reliability, low access latency, and so on. However, NAND flash memory has disadvantages such as erase-before-write, unbalanced operation speed, and limited P/E cycles, unlike conventional magnetic disk. To solve these problems, NAND flash memory mainly adopted FTL (Flash Translation Layer). In particular, garbage collection technique in FTL tried to improve the system lifetime. However, previous garbage collection techniques have a sensitive property of the system lifetime according to write pattern. To solve this problem, we propose BSGC (Balanced Selection-based Garbage Collection) technique. BSGC efficiently selects a victim block using all intervals from the past information to the current information. In this work, SFL (Search First linked List), as the proposed block allocation policy, prolongs the system lifetime additionally. In our experiments, SFL and BSGC prolonged the system lifetime about 12.85% on average and reduced page migrations about 22.12% on average. Moreover, SFL and BSGC reduced the average response time of 16.88% on average.

▸ Keyword : Solid State Drive, NAND Flash Memory, Storage Systems, Durability, Garbage Collection

## I. Introduction

최근 들어 비휘발성 메모리(non-volatile memory)의 사용은 많은 시스템에서 점차적으로 증가하고 있다. 특히 낸드 플래시 메모리(NAND flash memory)는 비휘발성 메모리 중 하나로써 낮은 전력, 내 충격성, 빠른 동작 속도 등의 장점을 가지고 있어 기존의 하드 디스크를 대신할 보조 기억 장치로 사용이 증가하고 있다. 하지만 낸드 플래시 메모리는 기존의 하드디스크와 비교하여 연산 속도 불균형, 불가능한 제자리 덮어쓰기, 낮은 내구성 등의 문제점을 가지고 있다.

이와 같은 문제점을 해결하기 위해 낸드 플래시 메모리 기반의 전용 파일 시스템이나 B+ 트리 등을 활용하는 기법이 제안되었다. 낸드 플래시 메모리 기반의 대표적인 파일 시스템으로는 JFFS(Journalling Flash File System), YAFFS(Yet Another Flash File System) 등이 있다. JFFS는 리눅스 로그 구조 기반의 순환 로그(circular log)를 이용 하였다[1]. JFFS는 순환 로그 사용으로 인해 메모리 사용량이 증가하는데 반면에 YAFFS는 이와 같은 문제점을 해결함으로써 부팅 시 마운트 속도를 향상하였다[2].

한편, 하드 디스크 기반의 자료 구조인 B+ 트리를 변형한 형태를 비휘발성 시스템에 적용하기 위해 $\mu$ 트리, FlashDB, LA 트리 등이 제안되었다. $\mu$ 트리는 기존의 B+ 트리의 단점

• First Author: Sungho Kim, Corresponding Author: Jong Wook Kwak
*Sungho Kim (boca123@ynu.ac.kr), Dept. of Computer Engineering, Yeungnam University
**Jong Wook Kwak (kwak@yu.ac.kr), Dept. of Computer Engineering, Yeungnam University
• Received: 2016. 09. 24, Revised: 2016. 10. 31, Accepted: 2016. 12. 05.
• 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).

인 업데이트 파생 문제(update propagation problem)를 해결하기 위해 제안되었다. $\mu$ 트리는 하나의 페이지에 부모 노드와 자식 노드를 저장하는 방식으로 이를 해결하였다[3]. FlashDB는 로그 모드(log mode)와 디스크 모드(disk mode) 사이의 비용을 산정하고, 그에 따른 최적의 비용에 맞는 모드로 전환하여 업데이트 파생 문제를 해결하였다[4]. LA 트리는 부모 노드에서 자식 노드까지 다중 버퍼를 사용하여 버퍼의 유지비용이 쓰기 비용보다 클 경우 낸드 플래시 메모리에 쓰기 연산을 수행하는 방식으로 업데이트 파생 문제를 해결하였다[5].

최근에는 하드 디스크 기반의 파일 시스템 혹은 자료 구조를 그대로 적용하는 방법을 채택하고 있으며, 주로 FTL(Flash Translation Layer)을 채택하여 낸드 플래시 메모리의 문제점을 해결하고자 노력하였다[6-10]. 특히 FTL의 가비지 컬렉션은 낸드 플래시 메모리의 오버헤드와 응답 시간을 줄임과 동시에 수명까지 연장하는 기법이다. 기존의 많은 연구에서는 가비지 컬렉션을 수행할 때 과거의 모든 구간 정보를 이용하지 않았기 때문에 희생 블록을 선정하는데 작업량(workload)에 따라 민감한 문제점을 가지고 있다.

따라서 본 논문에서는 과거의 모든 구간 정보를 이용한 기법인 BSGC (Balanced Selection-based Garbage Collection)를 제안한다. BSGC는 과거의 모든 구간 정보를 측정하기 위해 소거 연산이 발생하였을 때 경과 시간과 무효 페이지의 개수를 활용함으로써 희생 블록을 선정하는데 효율성을 증대시킨다. 또한, BSGC의 쓰기 연산을 수행하기 위해 본 논문에서는 SFL (Search First linked List) 기법을 더불어 제안함으로서 추가적인 성능 향상을 도모한다. 이하 본 논문의 구성은 다음과 같다. 2장에서는 배경 지식과 관련 연구를 소개하며, 3장에서는 본 논문에서 제안하는 BSGC와 블록 할당 정책을 서술한다. 4장에서는 다른 기법과 비교하여 BSGC의 성능 평가를 진행할 것이며, 5장은 결론을 맺는다.

## II. Background and Related works

### 1. FTL

FTL은 설계 목적에 따라 파일 시스템 기반의 FTL과 낸드 플래시 메모리 기반의 FTL로 구분한다[11]. 파일 시스템 기반의 FTL은 기존의 파일 시스템을 사용하지 못하는 IoT, 임베디드 시스템 등의 낮은 사양을 가지는 시스템에 적용된다. 낸드 플래시 메모리 기반의 FTL은 기존의 파일 시스템을 유지하며 사용하는 PC, 서버 시스템 등에 주로 사용한다. 일반적으로, FTL의 구성 요소는 그림 1과 같이 크게 주소 변환 테이블, 가비지 컬렉션, 마모도 평준화로 구성된다[12].

주소 변환 테이블은 낸드 플래시 메모리에서 제자리 덮어쓰기 문제점을 해결하기 위한 기법으로 특정한 프로그램에 의해 연산이 발생한 주소와 낸드 플래시 메모리 내의 주소를 상호

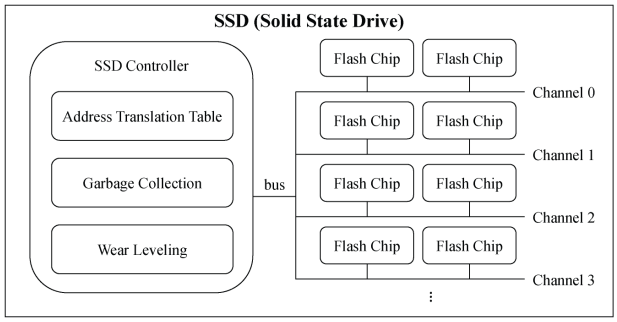


Fig. 1. System architecture based on SSD

변환해주는 역할을 수행한다. 주소 변환 테이블은 상호 변환을 수행하기 위해서 논리 주소와 물리 주소로 구분된다. 논리 주소는 프로그램에 의해서 연산이 발생하는 주소를 의미하며, 물리 주소는 낸드 플래시 메모리 내의 실질적인 데이터 주소를 의미한다. 따라서 주소 변환 테이블은 특정한 논리 주소에 쓰기 연산이 발생하였을 때, 낸드 플래시 메모리 내의 쓰기 연산이 발생하지 않은 물리 주소를 찾아 쓰기 연산을 수행한다. 이후 주소 변환 테이블은 쓰기 연산이 수행된 물리 주소를 해당하는 논리 주소로 매핑 하는 작업을 수행한다. 그 반대의 경우, 주소 변환 테이블은 논리 주소에 해당하는 물리 주소를 찾아 읽기 연산을 수행한다. 이와 같은 과정을 통해, 주소 변환 테이블은 제자리 덮어쓰기 문제점을 해결하였다. 주소 변환 테이블은 매핑 단위를 구성하는 방법에 따라 페이지 단위인 페이지 매핑, 블록 단위인 블록 매핑, 페이지와 블록 매핑의 장점을 가지는 하이브리드 매핑으로 구분된다.

가비지 컬렉션은 낸드 플래시 메모리 내에서 사용 가능한 블록(free block)이 부족할 경우 사용 가능한 공간을 확보하기 위해 수행된다. 이러한 과정을 수행하기 위해, 가비지 컬렉션은 사용 중인 블록(used block) 가운데 하나 혹은 다수의 블록을 가비지 컬렉션 대상 블록으로 선정하며, 이 블록들을 희생 블록(victim block)이라 부른다. 희생 블록은 다수의 유효 페이지(valid page)와 무효 페이지(invalid page)를 포함하고 있다. 유효 페이지는 현재 사용 중인 페이지를 의미하며, 무효 페이지는 과거에 사용하였지만 현재는 더 이상 사용하지 않은 페이지를 의미한다. 따라서 희생 블록 내에 존재하는 유효 페이지는 사용 가능한 페이지(free page)로의 이주가 필수적이고, 이후 소거 연산을 수행해야 한다. 이와 같은 가비지 컬렉션 과정을 통해, FTL은 낸드 플래시 메모리 내에 사용 가능한 블록을 확보한다.

마모도 평준화는 낸드 플래시 메모리의 낮은 내구성 문제를 해결하기 위한 기법으로 공간 지역성(spatial locality)이 발생하지 않는 커널 데이터, 프로그램 코드 등의 읽기 연산이 중점적으로 발생하는 블록이나 페이지를 이주하여 추가적인 오버헤드를 감수하더라도 수명을 연장시키는 기법이다. 마모도 평준화는 공간 지역성이 발생하는 영역에서만 이를 수행하는 동적 마모도 평준화(dynamic wear leveling)와 낸드 플래시 메모리의 모든 영역을 대상으로 균등화를 수행하는 정적 마모도 평준화(static wear leveling)로 구분한다.

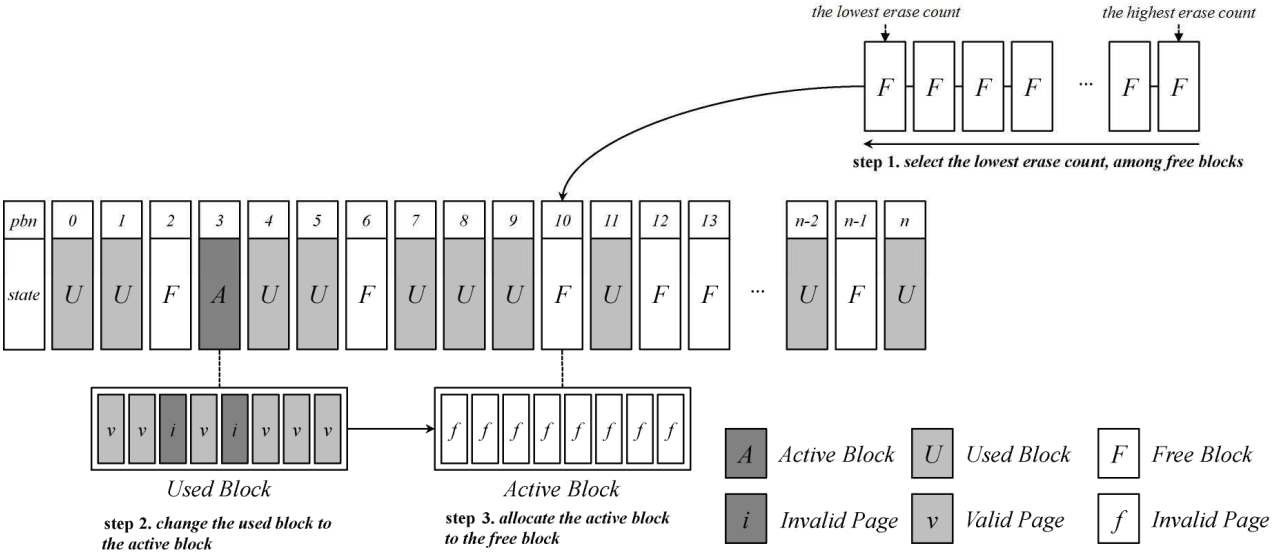


Fig. 2. Block allocation procedure based on SFL

이외에도 FTL은 설계 목적과 단위에 따라 낸드 플래시 메모리의 데이터를 보존하기 위한 암호화 기법, 데이터의 신뢰성을 보존하기 위한 ECC 기법, 전원이 차단되었을 때 데이터를 보존하는 전원 차단 기법, 칩 단위의 작업량 분배 기법, 공간 지역성을 고려한 디스크 캐시 정책 등의 다양한 기법들을 포함하여 설계한다.

## 2. Related Works

가비지 컬렉션은 낸드 플래시 메모리의 오버헤드와 응답 시간을 줄이면서 수명을 연장시키는 기법이며, 가비지 컬렉션을 통해서 수명을 연장하는 방법을 동적 마모도 평준화라 부른다. 기존의 많은 연구에서 가비지 컬렉션을 통해서 낸드 플래시 메모리에 수명을 연장하고자 노력하였다[13]. 구체적으로 살펴보면 다음과 같다.

GA(Greedy Algorithm)는 가장 대표적인 가비지 컬렉션 기법으로, 선택되었을 때 효율이 가장 높은 블록을 희생 블록으로 선정하며, 이는 무효 페이지의 개수가 가장 많은 블록을 의미한다. GA는 무효 페이지의 개수만을 고려하기 때문에 구현이 쉽고 간단하나 마모도 평준화를 수행하지 않기 때문에 높은 오버헤드와 응답 시간, 낮은 수명에 대한 단점이 존재한다[14].

CB(Cost-Benefit)는 GA의 단점을 보완하여 마모도 평준화를 수행하기 위해 블록의 마지막 무효 페이지가 발생한 시간과 무효 페이지의 개수를 이용한 식 $(a\times(1-u))/2u$에 의해서 비용을 산정하고 가장 높은 비용을 가지는 블록을 희생 블록으로 선정한다. 이 때, $u$는 유효 페이지의 비율, $a$는 마지막으로 무효 페이지가 발생 시간에서 현재까지의 경과 시간을 의미한다[15].

CAT(Cost-Age-Time)는 CB의 단점을 추가적으로 보완한 기법으로 경과 시간과 더불어 마모도 횟수를 고려한 기법이다. CAT는 식 $(u/(1-u)\times(1/age)\times EC)$에 의해서 비용을 산정하고 가장 낮은 비용을 가지는 블록을 희생 블록으로 선정한다. CAT의 식에서 $u$는 유효 페이지의 비율, $age$는 마지막으로 갱신한 페이지의 해당 시점에서 현재까지의 경과 시간, $EC$는 마모도 횟수를 의미한다. CAT는 CB과 비교하여 전반적인 수명은 연장되었으나, 희생 블록을 선정함에 있어서 유효 페이지가 많은 블록을 자주 선택하기 때문에 많은 페이지 이주를 동반하며 그로 인해 응답 시간이 증가하는 단점이 존재한다[16].

SAGC(Swap-Aware Garbage Collection)는 모든 페이지에 대하여 무효 페이지가 발생하는 시간을 고려한 기법으로 모든 페이지의 경과 시간($i\_age$)을 고려하여 가장 큰 비용을 가지는 블록을 희생 블록으로 선정한다. 또한, SAGC는 다른 기법과 다르게 블록 할당 정책을 활용함으로 성능 향상을 추구하고 있으며, 블록 할당 정책으로는 마모도 횟수가 가장 적은 블록을 우선 할당한다[17].

SCATA(Swap-aware Cost-Age-Time Age-sort)는 무효 페이지의 발생 시간과 유효 페이지의 발생 시간에 대한 편차를 고려함과 동시에 마모도 횟수도 함께 고려한 기법으로, 식 $max(age_i)\times(1/EC)\times min(est_i)\times((1-u)/(1+u))$에 의해서 가장 큰 비용을 가지는 블록을 희생 블록으로 선정한다. SCATA의 식에서 $u$는 유효 페이지의 비율, $age$는 무효 페이지가 발생한 시간에서 현재까지의 경과 시간, $est$는 유효 페이지가 발생한 시간에서 현재까지 경과 시간, $EC$는 마모도 횟수를 의미한다. SCATA는 유효 페이지와 무효 페이지의 경과 시간을 활용하여 성능 향상을 하였다[18].

EIGC(Erasure Interval-based Garbage Collection)는 소거 연산이 발생한 시점을 기준으로 수명을 계산하는 기법으로 식 $cost=(u/(1-u))\times((1-c_i_u)/time)\times EC$에 의해서 최소한의 비용을 가지는 블록을 선정한다. EIGC의 식에서 $u$는 유효 페이지의 비율, $time$는 소거 연산이 발생한 시점을 기준

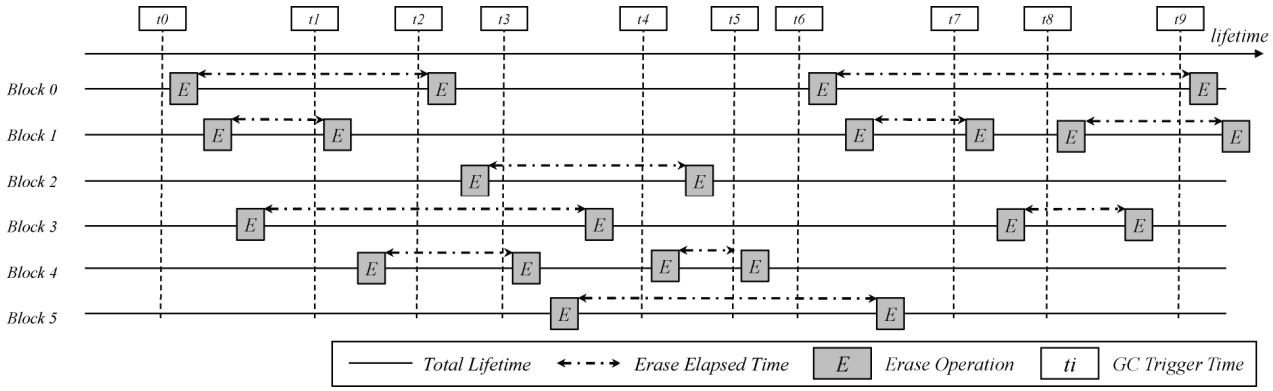


Fig. 3. Timing diagram with garbage collection

으로 경과된 시간, $c_i_u$는 소거 연산이 발생한 시점 간격에 해당하는 유효 페이지의 평균 변화량, $EC$는 마모도 횟수를 의미한다. EIGC는 소거 연산이 발생한 시점을 활용하여 낸드 플래시 메모리의 수명을 연장하였다[19].

### 3. Motivation

CB와 CAT는 페이지가 무효화 되었을 때 혹은 갱신되었을 때의 경과 시간(elapsed time)을 사용하여 성능을 향상시켰다. SAGC와 SCATA는 CB와 CAT의 단점을 해결하기 위해 모든 페이지를 대상으로 경과 시간을 측정하였다. 그러나 기존의 기법들은 직전의 가비지 컬렉션 발생 시간과 현재의 가비지 컬렉션 발생 시간만을 고려하기 때문에 작업량에 민감한 문제점을 가지고 있다. 이는 경과 시간에 대한 명확한 기준 시간이 없을 뿐만 아니라, 과거의 모든 구간 정보를 사용하지 않기 때문에 작업량의 변화에 따라 성능이 민감한 문제점이 존재한다.

## III. BSGC: Balanced Section-based Garbage Collection Technique

### 1. Overview

본 논문에서 제안하는 BSGC는 작업량의 변화에 따라 성능이 민감하다는 문제점을 인식하고 이를 해결하기 위해 다음 두 가지 방법을 제안한다. 첫 번째, BSGC와 이에 적합한 블록 할당 정책을 소개함으로써 검색 성능을 향상시킴과 동시에 블록의 사용 편차를 줄임으로써 수명, 오버헤드, 응답 시간을 향상시킨다. 이는 다음의 2절 블록 할당 정책에서 서술한다. 두 번째, BSGC는 블록의 소거 연산이 발생하였을 때를 기준 시간으로 정의하고 활용하여 이를 통해 경과 시간에 대한 명확한 기준을 마련한다. 또한 과거의 모든 구간 정보를 포함하여 희생 블록을 선정하므로 작업량에 민감한 문제점을 해결 할 수 있으며 이는 다음 3절 가비지 컬렉션 기법에서 서술한다.

### 2. Block Allocation Policy

기존의 블록 할당 정책은 가장 기본적인 알고리즘으로 FIFO(First In First Out)를 사용한다. FIFO는 구현이 쉽고 간편하다는 장점을 가지고 있지만, 블록의 마모도 횟수를 고려하지 않기 때문에 핫 블록만 자주 할당되는 문제점이 존재한다. 따라서 BSGC에서는 보다 효과적인 블록 할당 정책을 수행하기 위해 연결 리스트(linked list)를 사용한다. 하지만 연결 리스트는 기본적으로 자료의 비교 검색하기 위해선 최대 O(n)의 시간이 소요되는 단점이 존재한다. 본 논문에서는 느린 비교 검색 문제를 해결하기 위해, 연결 리스트에 삽입 연산을 수행할 때 마모도가 낮은 순서로 정렬하여 검색 성능을 향상시키며, 본 논문에서는 이를 SFL(Search First linked List)이라 명명한다.

SFL은 삽입 연산을 수행할 때 최악의 경우 O(n)의 시간이 소요 되지만, 검색이나 소거 연산을 수행할 때 O(1)의 시간만 소요된다는 장점을 가지고 있다. SFL의 삽입 연산은 가비지 컬렉션에서 소거 연산이 발생할 때, 즉 시스템의 유휴 시간(idle time)에 수행하므로 전반적인 시스템의 성능 저하를 가져오지 않는다. 반면에 검색이나 삭제 연산은 실행 시간(running time)에 수행하기 때문에 빠른 응답 시간을 보여준다.

그림 2는 본 논문에서는 제안하는 SFL 기반의 블록 할당 과정을 보여주고 있다. 먼저 활성화 블록이 가득 찼을 경우 SFL의 첫 번째 노드에 해당하는 사용 가능한 블록을 가져온다(step 1). 이때 검색 시간은 O(1)을 만족한다. 그 후 이전의 활성화 블록을 사용 중인 블록으로 변경한다(step 2). 마지막으로 SFL에 의해서 가져온 사용 가능한 블록을 활성화 블록으로 할당한다(step 3). 본 논문에서는 4장의 성능 평가를 통해, 모든 평가 지표에 대해서 BSGC의 FIFO와 SFL를 결합하는 성능 평가를 진행 할 것이며, 이를 통해서 SFL 방식과 결합된 BSGC가 보다 더 적합한 블록 할당 정책임을 입증할 것이다.

### 3. Garbage Collection Technique

이 절에서는 본 논문에서 제안하는 BSGC 기법을 서술한다. 그림 3은 BSGC의 특징인 과거의 모든 구간에 대해 경과 시간과 무효 페이지의 개수 정보를 측정하는 방법을 보여주고 있다. 그림 3에서 가비지 컬렉션이 발생할 때 특정 블록의 소거 연산

Table 1. The values of simulation environment

| Description | Value |
|---|---|
| Total capacity | 4Gb |
| Reserved free blocks | 15% |
| Garbage Collection Trigger | the number of free blocks performed under 5% |
| Flash chip elements | 1 |
| Planes per elements | 1 |
| Blocks per plane | 2048 |
| Pages per block | 64 |
| Page size | 4KB |
| Page read latency | 60$us$ |
| Page program latency | 800$us$ |
| Block erase latency | 1.5$ms$ |
| Blocks per lifetime | $10^4$ |

이 발생하고 이후 다시 소거 연산이 발생하는데 이와 같은 모든 구간을 “누적 소거 구간”이라 부른다. 낸드 플래시 메모리의 누적 소거 구간은 발생 빈도가 짧을수록 핫 블록(hot block)일 확률이 크다. 따라서 BSGC는 누적 소거 구간 정보에 대한 경과 시간을 측정할 경우 핫 블록과 콜드 블록을 식별 할 수 있을 뿐만 아니라 가비지 컬렉션이 발생한 시간을 기준으로 경과 시간을 측정하므로 기준 시간의 정확성이 높다. BSGC는 식(1)에 의해서 누적 소거 구간에 대한 경과 시간을 측정한다.

$$c\_i\_time = (e\_time + i\_time) \times \alpha \quad (1)$$

$e_time$은 소거 연산이 발생한 시간, $i_time$은 누적 소거 구간 동안의 경과 시간, $\alpha$는 가중치, $c_i_time$은 현재 누적 소거 구간의 경과 시간이다. BSGC는 블록의 쓰기 변화량을 측정하기 위해 누적 소거 구간 동안의 유효 페이지 개수의 변화량을 이용하며, 이는 블록의 쓰기 패턴을 측정할 수 있는 지표가 된다. BSGC는 식 (2)를 활용하여 쓰기 패턴의 변화량을 측정한다.

$$c\_i\_u = (u + i\_u) \times \beta \quad (2)$$

$u$는 유효 페이지 비율, $i_u$는 누적 소거 구간 동안에 유효 페이지 비율, $\beta$는 가중치, $c_i_u$는 현재 누적 소거 구간에 유효 페이지 비율이다. 본 논문에서는 누적 소거 구간 동안에서의 경과 시간과 유효 페이지 비율을 과거와 현재 모두 반영하기 때문에 $\alpha$와 $\beta$는 각각 0.5의 가중치를 가진다. 또한 BSGC는 식 (3)에 의해서 핫 블록과 콜드 블록을 식별하기 위한 경과 시간을 측정한다.

Table 2. the hot and cold ratios of the synthetic traces

| trace | hot ratio | cold ratio |
|---|---|---|
| trace1 | 15% | 85% |
| trace2 | 25% | 75% |
| trace3 | 35% | 65% |

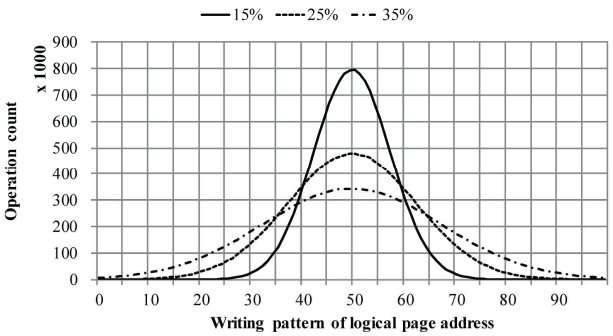


Fig. 4. Distribution of operation count in synthetic traces

$$time = c\_time + c\_i\_time \quad (3)$$

$c_i_time$은 현재 누적 소거 구간의 경과 시간, $c_time$은 가비지 컬렉션이 발생한 시간, $time$은 블록의 경과 시간이다. 추가로 BSGC는 블록의 마모도 횟수를 고려하여 상대적으로 큰 마모도 횟수를 가지는 블록을 희생 블록으로 선정하는데 제외시킨다. BSGC의 마모도 횟수는 식 (4)에 의해서 측정한다.

$$\epsilon = 1 - \frac{max\_ec - ec}{1 + (max\_ec - min\_ec)}$$
$$\tau = \frac{\epsilon}{max\_durability} \quad (4)$$

$ec$는 블록의 마모도 횟수, $min_ec$는 모든 블록 중 가장 작은 마모도 횟수, $max_ec$는 모든 블록 중 가장 큰 마모도 횟수, $max_durability$는 블록의 최대 마모도 횟수, $\epsilon$은 상대적인 블록 마모도 횟수의 비율, $\tau$는 블록의 마모도 횟수의 비율에 대한 남은 마모도 횟수이다. 이와 같은 정보를 기반으로, BSGC는 누적 소거 구간 동안의 경과 시간, 유효 페이지 비율, 그리고 블록의 남은 마모도 횟수의 비율을 이용하여 식 (5)에 의해서 희생 블록을 선정한다.

$$cost = \frac{u}{1-u} \times \frac{1 - c\_i\_u}{time} \times \tau \quad (5)$$

$u$는 유효 페이지 비율, $c_i_u$는 현재 누적 소거 구간의 유효 페이지 비율, $time$는 현재 누적 삭제 구간의 경과 시간, $\tau$는 블록의 상대적인 남은 마모도 횟수의 비율, $cost$는 계산된 희생 블록의 비용이다. BSGC는 가장 낮은 $cost$를 가지는 블록을 희생 블록으로 선정한다.

결과적으로 본 논문에서 제안하는 BSGC는 유효 페이지와 무효 페이지의 비율, 과거에서 현재까지의 누적 소거 구간 동안의 경과 시간과 유효 페이지의 비율, 블록의 남은 마모도 횟수 비율을 종합적으로 고려하여 작업량의 변화에 민감한 문제점을 해결함과 동시에 효과적으로 희생 블록을 선정한다.

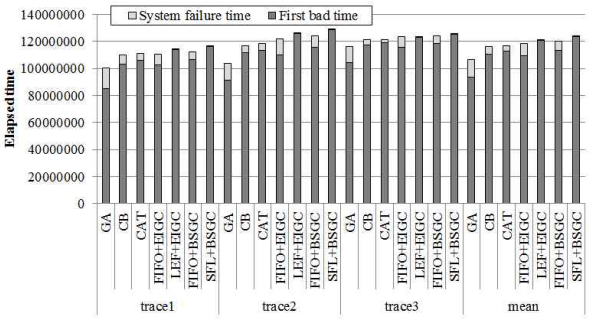


Fig. 5. System lifetime

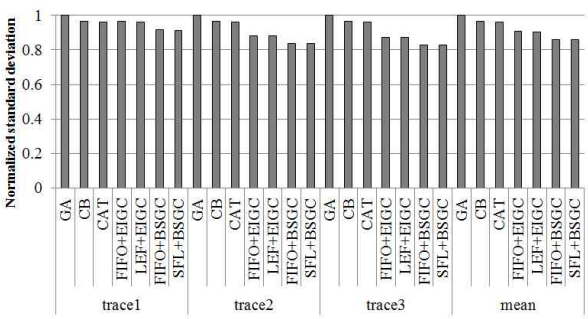


Fig. 6. Normalized standard deviation

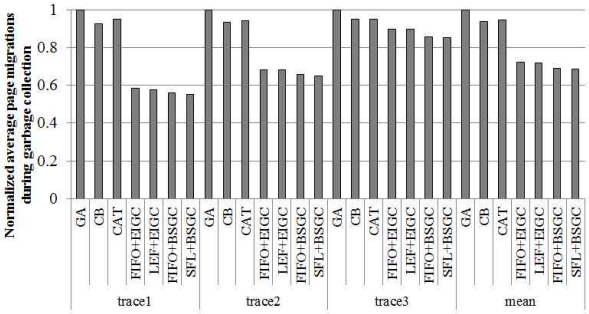


Fig. 7. Normalized average page migrations

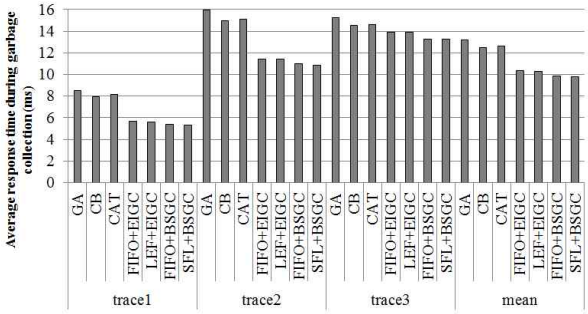


Fig. 8. Average response time

## IV. Performance Evaluation

### 1. Simulation Setup

본 논문에서 제안하는 BSGC의 성능 평가를 진행하기 위해 DiskSim Simulation Environment 4.0 기반의 SSD Extension for DiskSim Simulation Environment를 사용하여 성능 평가를 진행하였다[20]. 이는 SSD 모델을 확장한 마이크로소프트사의 시뮬레이터이다. 시뮬레이션을 진행하기 위하여, 삼성전자에서 개발한 MLC 낸드 플래시 메모리의 상세 명세를 이용하여 실험 환경을 구축하였다. 실험 환경에서 사용한 자세한 매개변수는 표 1과 같다. 시뮬레이션에 사용한 트레이스(trace)는 정규 분포를 사용하였고 표 2는 핫 블록과 콜드 블록의 쓰기 비율을 보여주고 있다. 그림 4는 표 2를 기반으로 각각의 트레이스에 대한 쓰기 분포를 보여주고 있다.

### 2. Experiment Results

이 절에서는 세 가지 지표를 통해 BSGC의 성능 평가를 진행하였다. 첫 번째, 가비지 컬렉션의 중요한 지표 중 하나인 첫 번째 배드 블록 발생 시간을 측정한다. 이는 저장 시스템에 있어서 첫 번째 배드 블록 발생 시간 이후부터 공간 효율성이 떨어지기 때문이다. 두 번째, 낸드 플래시 메모리의 블록 간 편차를 측정한다. 이는 블록 간 편차를 줄임으로써 시스템의 수명을 연장시킬 수 있기 때문에 중요한 지표가 된다. 세 번째, 가비지 컬렉션에 의해서 발생하는 오버헤드를 측정한다. 가비지 컬렉션의 오버헤드는 페이지 이주의 횟수를 의미하며, 페이지 이주가 많을수록 가비지 컬렉션의 응답 시간이 증가하기 때문에 오버헤드를 줄이는 것이 중요하다. 한편, 본 논문에서 제안하는 SFL의 성능 평가를 진행하기 위해 BSGC는 FIFO와 SFL로 구분하여 성능 평가를 진행한다. 먼저 첫 번째 배드 블록 발생 시간의 성능을 평가하기 위해, 103의 블록 당 마모도 횟수를 가정하여 시스템 수명을 측정하였다. 그림 5는 각 기법들의 시스템 수명을 보여주고 있다. SFL+BSGC는 GA, CB, CAT, FIFO+EIGC, LIF+EIGC, FIFO+BSGC와 비교하여 첫 번째 배드 블록 발생 시간이 최대 41.14%, 15.58%, 13.58%, 16.94%, 2.25%, 11.39%까지 연장되었고, 시스템 수명은 최대 24.03%, 10.45%, 9.15%, 5.78%, 2.25%, 3.87% 연장되었다.

다음으로 블록 간 마모도의 편차를 측정하였다. 이는 시스템의 수명 증가에 있어서 균등한 마모도가 중요한 성능 지표이기 때문이다. 블록 간 마모도의 편차를 측정하기 위해, 108의 쓰기 연산을 수행하였을 때 표준 편차를 측정하였다. 그림 6은 각 기법들의 표준 편차를 보여주고 있다. SFL+BSGC는 GA, CB, CAT, FIFO+EIGC, LIF+EIGC, FIFO+BSGC와 비교하여 표준 편차를 최대 20.81%, 16.66%, 16.45%, 5.57%, 5.26%, 0.29% 줄였다.

마지막으로 가비지 컬렉션에 의해서 발생하는 오버헤드를 측정하였다. 가비지 컬렉션의 오버헤드를 측정하기 위해, $10^8$의 쓰기 연산을 수행하였을 때의 페이지 이주 횟수를 오버헤드로 측정하였다. 그림 7은 각 기법들의 평균 페이지 이주비용을 정규화 하여 보여주고 있다. SFL+BSGC는 GA, CB, CAT, FIFO+EIGC, LIF+EIGC, FIFO+BSGC와 비교하여 평균 페이지 이주가 최대 81.69%, 68.41%, 72.5%, 6.3%, 5.0%, 1.56% 줄었다.

Table 4. Summary comparisons of experiment results

| Benchmark | | GA | CB | CAT | FIFO-EIGC | LEF-BSGC | FIFO-BSGC | SFL-BSGC |
|---|---|---|---|---|---|---|---|---|
| First Failure time (sec) | trace1 | 1421.9 | 1716.0 | 1766.0 | 1707.7 | 1896.4 | 1772.0 | 1938.9 |
| | trace2 | 1521.6 | 1858.0 | 1890.8 | 1836.5 | 2100.4 | 1927.9 | 2147.6 |
| | trace3 | 1741.6 | 1956.4 | 1979.2 | 1925.1 | 2049.0 | 1976.2 | 2091.0 |
| mean | | 1561.7 | 1843.5 | 1878.7 | 1823.1 | 2015.3 | 1892.1 | 2059.2 |
| System failure time (sec) | trace1 | 1669.7 | 1833.1 | 1853.2 | 1846.1 | 1898.2 | 1874.0 | 1940.6 |
| | trace2 | 1732.9 | 1946.0 | 1969.2 | 2031.9 | 2102.0 | 2069.2 | 2149.4 |
| | trace3 | 1933.8 | 2016.8 | 2020.8 | 2054.8 | 2050.5 | 2065.2 | 2092.5 |
| mean | | 1778.8 | 1932.0 | 1947.7 | 1977.6 | 2016.9 | 2002.8 | 2060.8 |
| Standard deviation (%) | trace1 | 100.0 | 96.6 | 96.4 | 96.6 | 96.3 | 91.8 | 91.5 |
| | trace2 | 100.0 | 96.6 | 96.4 | 88.2 | 88.0 | 83.8 | 83.6 |
| | trace3 | 100.0 | 96.6 | 96.4 | 87.1 | 87.1 | 82.8 | 82.8 |
| mean | | 100.0 | 96.6 | 96.4 | 90.7 | 90.5 | 86.1 | 86.0 |
| Average page migrations (%) | trace1 | 100.0 | 92.7 | 94.9 | 58.5 | 57.8 | 55.9 | 55.0 |
| | trace2 | 100.0 | 93.4 | 94.2 | 68.4 | 68.2 | 65.6 | 64.9 |
| | trace3 | 100.0 | 94.9 | 95.1 | 89.8 | 89.6 | 85.6 | 85.4 |
| mean | | 100.0 | 93.7 | 94.7 | 72.2 | 71.9 | 69.0 | 68.4 |
| Average response time (ms) | trace1 | 8.5 | 8.0 | 8.1 | 5.4 | 5.3 | 5.4 | 5.3 |
| | trace2 | 15.9 | 15.0 | 15.1 | 10.9 | 10.9 | 11.0 | 10.9 |
| | trace3 | 15.3 | 14.6 | 14.6 | 13.3 | 13.2 | 13.3 | 13.2 |
| mean | | 13.2 | 13.2 | 12.5 | 9.9 | 9.8 | 9.9 | 9.8 |

한편, 낸드 플래시 메모리는 페이지 이주가 증가할수록 평균 응답 시간도 더불어 증가하기 때문에 가비지 컬렉션의 평균 응답 시간을 관찰하는 것이 중요하다. 그림 8은 가비지 컬렉션의 평균 응답 시간을 보여주고 있다. SFL+BSGC는 GA, CB, CAT, FIFO+EIGC, LIF+EIGC, FIFO+BSGC와 비교하여 평균 응답 시간을 최대 58.74%, 49.19%, 52.13%, 5.93%, 5.0%, 1.12% 줄였다. 이상으로 제안하는BSGC와 SFL은 다른 기법과 비교하여 시스템 수명, 마모도의 편차, 오버헤드, 평균 응답 시간과 같은 전반적인 부분에서 우수한 성능을 보였음을 입증하였으며, 성능 평가에 대한 자세한 평가 지표는 표 3에서 상세하게 기술하였다.

## V. Conclusions

본 논문에서는 희생 블록의 선정에 있어서 작업량의 변화에 따라 민감하게 변화하는 시스템 성능의 문제점을 인식하고 이를 해결하기 위한 기법인 BSGC를 제안하였다. BSGC는 희생 블록을 선정함에 있어서 과거의 모든 구간 정보에 대하여 누적 소거 구간을 활용하여 경과 시간과 유효 페이지의 변화를 측정한다. 또한 본 논문에서는 블록 할당 정책에 있어서 검색 성능 향상과 블록 사용의 편차를 줄이기 위해서 BSGC에 적합한 블록 할당 정책인 SFL를 제안하였다.

SFL+BSGC는 다른 기법과 비교하여 첫 번째 배드 블록 발생 시간은 평균적으로 12.85% 연장시켰고, 시스템 수명은 6.27% 연장하였다. 또한 SFL+BSGC의 표준 편차는 평균적으로 8.62% 줄였고, 평균 페이지 이주는 22.12% 줄였다. 가비지 컬렉션의 평균 응답 시간도 16.88% 줄였다. 이상에서와 같이 BSGC는 기존 블록 할당 정책인 FIFO를 사용함에도 성능을 향상시켰지만, 추가로 제안한 SFL를 통해 전반적인 수명을 크게 연장하였고, 표준 편차와 오버헤드까지 줄임으로써 BSGC에 적합한 블록 할당 정책임을 성능 평가를 통해 입증하였다. 향후 SSD 기반의 가비지 컬렉션이 전체 시스템에 끼치는 영향을 분석하고, 이를 해결하기 위한 연구를 진행 할 예정이다.

## REFERENCES

[1] Woodhouse, David. "JFFS: The journalling flash file system." Ottawa linux symposium. Vol. 2001. 2001.

[2] One, Aleph. http://www. aleph1. co. uk/yaffs/index. html. Cambridge, UK, 2002.

[3] Kang, Dongwon, et al. "μ-tree: an ordered index structure for NAND flash memory." Proceedings of

the 7th ACM & IEEE international conference on Embedded software. ACM, 2007.
[4] Nath, Suman, and Aman Kansal. "FlashDB: dynamic self-tuning database for NAND flash." Proceedings of the 6th international conference on Information processing in sensor networks. ACM, 2007.
[5] Devesh, et al. "Lazy-adaptive tree: An optimized index structure for flash devices." Proceedings of the VLDB Endowment 2.1, 361-372, 2009.
[6] Wang, Chundong, and Weng-Fai Wong. "Observational wear leveling: an efficient algorithm for flash memory management." Design Automation Conference (DAC), 2012
[7] Lin, W. et al., "Swap time-aware garbage collection policy for NAND flash-based swap system." Electronics Letters 49.24, pp. 1525-1526, 2013.
[8] Jeong, Jaehyeong et al., "A technique to improve garbage collection performance for NAND flash based storage systems." Consumer Electronics, IEEE Transactions on 58.2, pp. 470-478, 2012.
[9] Chung, Ching-Che, Duo Sheng, and Ning-Mi Hsueh. "A high-performance wear-leveling algorithm for flash memory system." IEICE Electronics Express 9.24, pp. 1874-1880, 2012.
[10] Yang, Ming-Chang, et al. "New ERA: new efficient reliability-aware wear leveling for endurance enhancement of flash storage devices." Proceedings of the 50th Annual Design Automation Conference. ACM, p. 163, 2013.
[11] Ma, Dongzhe, et al., "A survey of address translation technologies for flash memories." ACM Computing Surveys (CSUR) 46.3, pp. 36, 2014.
[12] Gal, Eran, and Sivan Toledo. "Algorithms and data structures for flash memories." ACM Computing Surveys (CSUR) Vol. 37, No. 2, pp. 138-163, 2005.
[13] Yang, Ming-Chang, et al. "Garbage collection and wear leveling for flash memory: Past and future." Smart Computing (SMARTCOMP), IEEE, 2014
[14] Wu, Michael, and Willy Zwaenepoel. "eNVy: a non-volatile, main memory storage system." ACM SigPlan Notices, Vol. 29, No. 11, pp. 86-97, 1994.
[15] Kawaguchi, Atsuo et al., "A Flash-Memory Based File System." USENIX. 1995.
[16] Chiang, M-L., and R-C. Chang. "Cleaning policies in mobile computers using flash memory." Journal of Systems and Software 48.3, pp. 213-231, 1999.
[17] Kwon, Ohhoon et al., "Swap space management technique for portable consumer electronics with NAND flash memory." Consumer Electronics, IEEE Transactions on 56.3, pp. 1524-1531, 2010.
[18] Lin, M. W., et al. "Garbage collection policy for flash-aware Linux swap system." Electronics letters 47.22, pp. 1218-1220, 2011.
[19] Kim, Sung Ho, and Jong Wook Kwak. "Garbage Collection Technique using Erasure Interval for NAND Flash Memory-based Storage Systems." International Journal of Applied Engineering Research, Vol. 11, No. 7, pp. 5188-5194, 2016.
[20] V. Prabhakaran and T. Wobber, "SSD Extension for DiskSim Simulation Environment", 2010.

## Authors

Sungho Kim received a B.S. degree in Department of Computer Engineering from Yeungnam University College, Daegu, Korea in 2012. He is currently a Ph.D. candidate in Department of Computer Engineering from Yeungnam University. His current research interests include embedded systems and non-volatile memory systems.

Jong Wook Kwak received the B.S. degree in computer engineering from Kyungpook National University, Daegu, South Korea, in 1998, and the M.S. degree in computer engineering and the Ph.D. degree in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2001 and 2006, respectively. From 2006 to 2007, he was a Senior Engineer with the system-on-chip (SoC) Research and Development Center, Samsung Electronics Company, Ltd., Suwon, South Korea. During 2011-2012, he was a Guest Researcher at the Research Institute of Advanced Computer Technology, Seoul National University. During 2012-2013, he was a Visiting Scholar at the Georgia Institute of Technology, Atlanta, GA, USA. He is currently an Associate Professor with the Department of Computer Engineering, Yeungnam University, Gyeongsan, South Korea. His current research interests include advanced processor architecture, low-power mobile embedded system design, and high-performance parallel and distributed computing.