논문 2016-11-15

낸드 플래시 메모리의 이주 오버헤드 감소 및 수명연장을 위한 가비지 컬렉션 기법

(Garbage Collection Technique for Reduction of Migration Overhead and Lifetime Prolongment of NAND Flash Memory)

황 상 호, 곽 종 욱*
(Sang-Ho Hwang, Jong Wook Kwak)

Abstract : NAND flash memory has unique characteristics like as 'out-place-update' and limited lifetime compared with traditional storage systems. According to out-of-place update scheme, a number of invalid (or called dead) pages can be generated. In this case, garbage collection is needed to reclaim invalid pages. Because garbage collection results in not only erase operations but also copy operations of valid (or called live) pages to other blocks, many garbage collection techniques have proposed to reduce the overhead and to increase the lifetime of NAND Flash systems. This techniques sometimes select victim blocks including cold data for the wear leveling. However, most of them overlook the cost of selecting victim blocks including cold data. In this paper, we propose a garbage collection technique named CAPi (Cost Age with Proportion of invalid pages). Considering the additional overhead of what to select victim blocks including cold data, CAPi improves the response time in garbage collection and increase the lifetime in memory systems. Additionally, the proposed scheme also improves the efficiency of garbage collection by separating cold data from hot data in valid pages. In experimental evaluation, we showed that CAPi yields up to, at maximum, 73% improvement in lifetime compared with existing garbage collections.

Keywords : NAND flash memory, Garbage Collection, Wear Leveling, Cost Age

Ⅰ. 서 론

저전력, 내충격성, 빠른 접근 속도 및 무소음 등의 장점을 가진 낸드 플래시 메모리는 차세대 사물인터넷 시스템의 중요한 메모리 구조 가운데 하나이며, 차세대 저장 매체로 널리 사용되고 있다. 하지만 낸드 플래시 메모리는 기존의 하드 디스크와 다르게 덮어쓰기가 불가능하고 수명이 존재한다. 덮어쓰기가 불가능한 낸드 플래시에서 저장된 데이터에 업데이트가 이루어지면 기존의 데이터는 무효화가 되고 다른 영역에 데이터가 저장된다. 낸드 플래시 공간을 점유하고 있는 무효화된 영역은 가비지 컬렉션(Garbage Collection)을 통해 빈 공간으로 확보된다. 가비지 컬렉션이 이루어지는 블록에는 유효 페이지들이 남아 있을 수 있으며 이 페이지들은 다른 블록으로 이주되어야 한다. 이것은 가비지 컬렉션 시에 발생할 수 있는 오버헤드이며, 페이지 이주를 최소화하는 일반적인 방법은 유효 페이지가 가장 적은 블록을 선택하는 것이다. 하지만 유효 페이지가 가장 적은 블록만을 선택하게 된다면 블록 간에 마모도 불균형이 일어나게 된다. 낸드 플래시 블록에는 수명이 존재하기 때문에, 이러한 마모도 불균형은 배드 블록을 빠르게 생성시키고 결국 시스템 성능을 저하시키게 된다. 따라서 지금까지 수명을 증가시키고 효율적으로 가비지 컬렉션을 수행

*Corresponding Author (kwak@yu.ac.kr)
Received: 13 Jan. 2016, Revised: 9 Mar. 2016, Accepted: 22 Mar. 2016.
Sang-Ho Hwang, Jong Wook Kwak: Yeungnam University
※ 논문은 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. NRF-2014R1A1A2057146).
 2016 Apr. 11(2) 125-134
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2016.11.2.125

표 1. 가비지 컬렉션 기술들의 요약
Table 1. A summary of various garbage collection techniques

Algorithm	Garbage Collection	Migration policy for valid pages	Allocation policy
GA	select the block with the least valid pages	N/A	FIFO
CB	select the block with the largest value $alue = age \times \frac{1-u}{2u}$	N/A	FIFO
CAT	select the block with the least value $value = \frac{1}{age_a} \times \frac{u}{1-u} \times C$	Redistribution for hot and cold data	FIFO
SCATA	select the block with the largest value $value = \max(age_i) \times \frac{1}{EC} \times \min(est_j) \times \frac{1-u}{1+u}$ $age_i = cTime - iTime_i$ and $est_j = cTime - oTime_j$	Redistribution for hot and cold data	FIFO
FeGC	select the block with the largest value $value = \sum_{n=1} age_n$ $age_n = cTime - iTime_n$	Redistribution for hot and cold data	Adaptive youngest Blocks

하기 위해 CB(Cost Benefit), CAT(Cost Age Time), SCATA(Swap-aware Cost-Age Time with Age-sort), FeGC(Fast and Endurant Garbage Collection)와 같은 기법들이 제안되었다 [1-4]. 이러한 기법들의 특징은 공통적으로 업데이트 빈도가 낮아 유효 페이지가 많은 블록도 가비지 컬렉션의 대상으로 선정하기 위해 경과 시간을 사용한다. 하지만 기존의 가비지 컬렉션 기법들은 콜드 데이터를 가진 블록을 희생블록으로 선정하는 것에 대한 추가 비용을 고려하지 않아 많은 오버헤드가 발생하였다.

본 논문에서는 가비지 컬렉션에서 발생할 수 있는 오버헤드를 줄여 응답시간 및 수명을 증가시키는 CAPi(Cost Age with Proportion of invalid page) 가비지 컬렉션을 제안한다. 또한 제안하는 기법은 희생 블록 내에 있는 유효 페이지들을 추가적인 오버헤드를 발생시키지 않는 수준에서 Cold/Hot 데이터를 분리해서 저장하고 있다.

이하 논문의 구성은 다음과 같다. 2장에서는 낸드 플래시 메모리에 대한 자세한 설명과 함께 기존의 가비지 컬렉션 기법에 대하여 소개한다. 3장에서는 본 논문에서 제안하는 가비지 컬렉션 기법을 소개하고, 4장에서는 실험을 통해 기존의 기법들과 비교하고, 5장에서는 결론 및 향후 연구과제에 대하여 기술한다.

II. 관련 연구

1. 배경 지식

낸드 플래시 메모리는 기존의 저장매체인 하드디스크와는 다른 몇 가지 제약사항을 가지고 있다. 첫 번째로 낸드 플래시는 쓰기 및 읽기 연산 외에 지움 연산이 있다. 낸드 플래시는 플로팅 게이트에 전자들을 저장하는 방법으로 데이터를 저장하게 된다. 저장되는 데이터는 플로팅 게이트 내에 머무르는 전자들의 양에 따라 결정되며, 1개 셀이 1bit를 저장할 수 있는 것을 SLC(Single Level Cell)라 하고 2bit 이상을 저장할 수 있는 것을 MLC(Multi Level Cell)라 한다. 쓰기 연산이 일어나면 전자들이 셀 내부의 산화막을 통과하여 플로팅 게이트 내에 머물게 된다. 쓰기가 이루어진 셀에 다른 데이터를 저장하기 위해서는 플로팅 게이트 내의 전자들을 모두 제거된 상태여야 하는데, 플로팅 게이트 내의 전자들을 비우는 것을 지움 연산이라 한다. 두 번째로 쓰기, 읽기 및 지움 연산의 속도와 단위가 다르다. 낸드 플래시 칩마다 차이는 있지만 읽기 연산은 25 μs ~ 60 μs, 쓰기 연산은 250 μs ~ 900 μs, 지움 연산은 700 μs ~ 3500 μs의 동작 속도를 가진다 [5]. 또한 읽기 연산과 쓰기 연산은 페이지 단위로 이루어지며, 지움 연산은 블록 단위로 이루

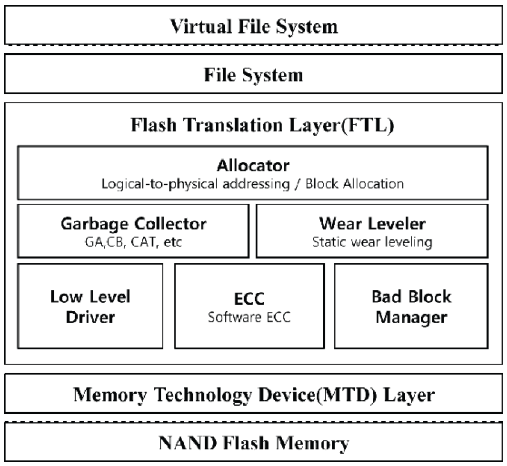

그림 1. 낸드 플래시 파일 시스템의 구조
Fig. 1 The architecture of NAND flash file systems

어진다. 블록은 일반적으로 64 ~ 256개의 페이지들로 구성되어 있다. 세 번째로 제자리 갱신이 불가능하다. 앞서 설명한 것처럼 데이터를 갱신하기 위해서는 우선 지움 연산을 수행해야한다. 하지만 지움 연산은 블록 단위로 수행되기 때문에 1개 페이지를 수정할 때마다 지움 연산을 수행하는 것은 너무나 큰 오버헤드를 초래한다. 따라서 일반적으로 제자리 갱신이 일어나면 이전의 데이터를 바로 삭제하지 않고, 대신 빈 페이지에 데이터를 저장하고 사상 테이블(Mapping Table)을 수정하는 방법을 이용한다. 네 번째로 낸드 플래시의 블록들은 수명이 존재한다. 쓰기 연산 및 지움 연산은 플로팅 게이트 내에 전자들을 채우거나 제거하는 동작이다. 이때 전자들은 산화막을 통과하게 되는데 이 산화막이 연산 횟수에 따라 손상이 점점 누적되게 된다. 이 손상이 일정 수준에 도달하게 되면 셀에 정보를 저장할 수 없게 된다. 블록 당 최대 지움 연산수는 약 100,000회(SLC) ~ 10,000회(MLC)이다.

기존의 하드 디스크와 다른 낸드 플래시의 특징 및 단점을 극복하기 위해 저장시스템은 별도의 플래시 전환 계층(FTL: Flash Translation Layer)을 사용한다. 그림 1은 일반적인 플래시 전환 계층의 구조를 보여주고 있다. 플래시 전환 계층의 주요 기능 중에 가비지 컬랙터(Garbage Collector)는 데이터 업데이트 및 삭제에 의해 무효화된 영역을 수집하여 빈 공간을 확보하는 작업을 수행한다. 지움 연산이 블록 단위로 이루어지기 때문에 가비지 컬랙터는 빈 공간을 확보하기에 적절한 블록을 선정하는데 이 블록을 희생 블록(Victim Block)이라 한다. 선택되는 희생블록은 내부에 유효한 데이터를 가지고 있을 수 있으며, 이 페이지들은 다른 블록으로 이주되어야 한다. 가비지 컬렉션은 오버헤드를 최대한 줄이기 위해 블록 내에 유효 페이지가 적은 블록을 선정한다. 하지만 유효 페이지가 적은 블록만 가비지 컬렉션 대상으로 선정하는 것은 지움 연산이 업데이트 빈도가 높은 블록들에 집중되어 발생할 수 있다. 특정 블록에 편중되어 사용하게 되면 이 블록들의 수명은 빠르게 줄어든다. 수명을 초과하여 배드 블록이 발생하게 되면 낸드 플래시 저장 시스템의 가용 공간이 줄어들어 나머지 블록들의 수명 또한 빠르게 소모시키게 된다. 따라서 지금까지 수명을 증가시키고 효율적으로 가비지 컬렉션을 수행하기 위해 많은 기법들이 연구되어 왔다 [1-4, 6-14].

2. 관련 연구

낸드 플래시 메모리의 가용 영역을 확보하기 위한 가비지 컬렉션 기법은 지금까지 많은 연구가 이루어져 왔다. 가장 기본적인 가비지 컬렉션은 GA(Greedy Algorithm)이다 [1]. GA는 가비지 컬렉션에서 희생블록을 선정할 때 유효 페이지가 가장 적은 블록을 선정한다. 알고리즘이 간단하여 오버헤드가 적은 장점이 있지만, 콜드 데이터를 포함하는 블록의 경우 희생블록으로 거의 선택되지 않아 마모도 불균형이 빠르게 발생하는 단점이 있다.

가비지 컬렉션 수행 시에 콜드 데이터가 있는 경우에 희생 블록으로 선정되지 않는 단점을 해결하기 위해 CB(Cost Benefit)기법이 제안되었다 [2]. CB는 희생블록을 선정할 때 블록 내에 페이지들 중에 가장 마지막으로 무효화된 것의 시간을 평가 값에 포함시켜 콜드 블록도 희생블록으로 선정될 수 있도록 하였다. 표 1에서 ge 는 블록내의 무효화된 페이지 중에 가장 최근에 이루어진 것의 경과 시간을 의미한다. u는 블록내의 유효 페이지 비율을 나타내며, (1 - u)는 무효 페이지 비율을 나타내며, $2u$는 가비지 컬렉션에서 선정된 희생 블록의 유효 페이지를 복사하는 비용을 나타낸다. CB는 경과 시간을 이용하여 가비지 컬랙션의 대상이 되는 블록들을 골고루 사용하도록 유도하는 장점이 있지만 기존 블록들의 마모도를 고려하지 못한 단점이 있다.

CAT(Cost Age Time)은 가비지 컬렉션에서 희생 블록 선정 시에 블록의 삭제 연산횟수도 고려한

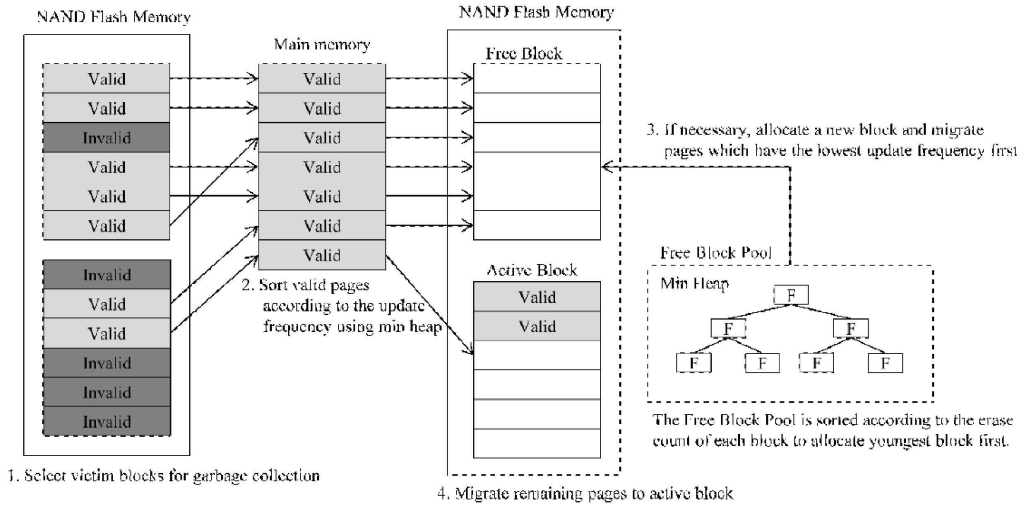

그림 2. CAPi의 구조

Fig. 2 The structure of CAPi

다 [3]. 또한 가비지 컬렉션 수행 시에 블록 내에 남아 있는 유효 페이지들은 Cold/Hot 의 구별을 통하여 각각 다른 빈 블록으로 이주가 이루어지며, 빈 블록 할당은 삭제 연산 횟수가 가장 적은 블록부터 이루어진다. 표 1에서 ge 는 블록이 사용을 위해 할당된 이후 지금까지 경과된 시간을 의미한다. 이 기법은 삭제 횟수가 많은 블록의 경우 희생 블록으로 선정될 확률을 줄임으로써 낸드 플래시 메모리의 수명을 증가시켰지만, 동시에 유효 페이지 이주에 대한 오버헤드도 같이 증가하는 단점이 있다.

SCATA(Swap-aware Cost-Age Time with Age-sort)는 CAT을 기반으로 하여 가장 최근에 무효화된 페이지의 경과시간을 고려하고 있다 [8]. 희생블록내의 무효 페이지들은 무효화된 시간을 기준으로 Cold/Hot 데이터로 구별되어져 각각 다른 빈 블록으로 이주가 이루어진다.

FeGC(Fast and Endurant Garbage Collection)는 가비지컬렉션 수행 시에 블록 내에 무효 페이지들의 경과 시간의 합을 기준으로 희생 블록을 선정한다. 선정된 희생 블록의 유효 페이지들은 Cold/Hot 데이터로 구별되어 각각 다른 빈 블록으로 이주가 되며, 빈 블록 할당 시에 Hot 데이터가 저장되는 블록은 삭제 연산 횟수가 가장 적은 블록이 선택되고, 콜드 데이터는 삭제 연산 횟수가 평균보다 많은 블록이 선택된다 [4]. 하지만 이 방식은 무효화 페이지들의 경과시간의 총합을 이용하는 방법을 통하여 희생블록 선택의 효율을 높였지만, 여전히 콜드 블록이 자주 선택되어 가비지 컬랙션의 오버헤드가 큰 단점이 있다.

그 외에도 CAT에서 파생된 CATA, CBA 기법들이 제안되었으며, 최근에는 스왑 시스템 기반의 SAGC, LFGC, NSAGC 등의 가비지 컬렉션 기법들이 제안되었다 [6-14].

본 논문에서는 콜드 데이터가 많은 블록을 희생블록으로 선택하는 것을 늦추어 가비지 컬랙션 수행 시 발생되는 유효 페이지 이주 오버헤드를 줄이는 CAPi 가비지 컬렉션 기법을 제안한다. 제안하는 기법은 희생 블록 내에 있는 유효 페이지들을 Cold/Hot 데이터로 분리해서 저장하는 방법을 통하여 가비지 컬랙션에서의 오버헤드를 추가적으로 감소시킨다.

Ⅲ. CAPi 가비지 컬렉션

제안하는 가비지 컬렉션 기법은 3가지 요소로 구별된다. 표 1을 기준으로 각각의 요소에 대해 관련 연구대비 본 논문에서 제안하는 기법의 차별성 및 특징을 분석하면 다음과 같다.

첫 번째는 낸드 플래시의 유효 페이지를 확보하기 위해 수행되는 가비지 컬렉션 정책이다. 가비지 컬렉션은 수행 시 발생되는 오버헤드를 줄이기 위해 아래 식 (1)의 값이 가장 큰 블록부터 희생블록으로 선정한다.

$$alue = age \times \frac{1-u}{(1+u)^2}$$
$$age_a = ime_c - Time_a \quad (1)$$

식 (1)에서 $Time_c$는 현재 시간을 의미하며, $Time_a$는 블록이 활성 블록(active block)으로 할당된 시간을 의미한다. 따라서 age_a는 블록이 할당된 이후 경과된 시간을 의미한다. 활성 블록이란 쓰기 연산이나 업데이트가 발생했을 때 데이터를 저장할 목적으로 선정된 블록이다. 이러한 활성 블록의 주소는 FTL에서 관리되고 있으며, 제안하는 기법은 이 활성 블록의 주소 정보를 활용한다. 는 블록 내에 유효 페이지의 비율을 의미한다. 대부분의 시간기반 가비지 컬렉션과 같이 제안한 기법 역시 식 (1)에 의해서 유효 페이지를 이용할 때의 비용대비 이득이 가장 큰 블록이 희생블록으로 선정된다. 하지만 대부분의 시간기반 가비지 컬렉션은 유효 페이지 이주에 따른 비용을 너무 낮게 적용하는 문제점을 가지고 있었다. 적용하고 있는 식에 의해 대부분의 시간기반의 가비지 컬렉션은 유효 페이지가 많은 콜드 블록이라도 경과된 시간에 따라 희생블록으로 선정될 수 있도록 유도하고 있다. 하지만 이렇게 선택된 콜드 블록은 유효 페이지가 많아서 쓰기 연산을 많이 발생시킬 뿐만 아니라 활성 블록내의 가용 영역보다 많은 페이지를 이주시켜 결과적으로 빈 블록을 소모하게 만든다. 가비지 컬렉션은 빈 블록의 수가 시스템이 필요로 하는 최소 빈 블록 수보다 클 때까지 이루어지기 때문에, 이는 결국 더 많은 블록의 삭제 및 유효 페이지 이주를 유발시키게 된다. 따라서 제안하는 기법에서는 식 (1)에 $(1+u)^2$를 적용하여 시간에 대한 가중치를 줄임으로써 유효 페이지가 많이 존재할 수 있는 콜드 블록들이 빠르게 희생블록으로 선택되는 것을 방지하고 있다. 이를 통하여 무효 페이지가 많은 블록들을 희생블록으로 선택하여 빈 공간 확보에 대한 효율을 높이고, 실제 가비지 컬렉션에서 추가로 발생할 수 있는 유효 페이지 이주에 대한 오버헤드를 함께 줄이고 있다. 이 과정은 그림 2에서 첫 번째 순서에 해당하며, 이 과정에서 빈 블록으로 만들기 위해 식 (1)을 이용하여 1개 이상의 희생블록을 선택한다.

두 번째는 유효 페이지 이주 정책이다. 가비지 컬렉션에서 선택된 희생블록들에는 유효 페이지가 존재할 수 있으며, 이 유효 페이지들은 다른 공간으로 이동되어야 한다. 유효 페이지가 이동될 공간 중에 가장 효율이 좋은 곳은 활성 블록이다. 이 활성

Algorithm 1: CAPi_GarbageCollector

```
1: for n = 1 to N do
2:   blk_value ← (Time_current − Time_n) × (1−u)/(1+u)^2
3:   PushMaxHeap(blks, blk)
4: end
5: while FreeBlock_count > FreeBlock_MIN
6:   blk ← PopMaxHeap(blks)
7:   for v = 1 to Page_MAX do
8:     if Page_v is valid then
         /* UFT = update frequency Table */
9:       Page_v_UF ← UFT[page_lpn]
10:      PutMinHeap(page_v, pages)
11:    end if
12:  end
13: end
14: while pages_count > 0
15:   page ← PopMinHeap(pages)
16:   if pages_count ≥ Page_MAX then
17:     move the page to youngest free blocks
18:   else then
19:     move the page to active block
20:   end if
21: end
22: erase all blks
23: add blks in the free block pool
```

알고리즘 1. CAPi의 의사코드
Algorithm 1. Pseudo code of CAPi

블록에는 데이터가 아직 입력되지 않은 가용 영역들이 존재할 수 있는데, 추가적인 빈 블록을 사용하지 않도록 이 영역을 최대한 활용하는 것이 유리하다. 하지만 활성 블록은 상대적으로 다른 블록에 비해 핫 데이터의 비율이 높고, 가비지 컬렉션에 의해 희생블록으로 선정된 블록에 있는 데이터들은 식 (1)에 의해 콜드 데이터를 가지고 있을 확률이 높다. 따라서 희생블록의 유효 페이지들을 분별하지 않고 활성 블록으로 이동하게 되면, 이 유효 페이지들은 차후에 가비지 컬렉션의 오버헤드가 될 확률이 높다. 따라서 활성 블록에 최대한 핫에 가까운 데이터가 저장될 수 있도록 유도해야 한다. 이를 위해, 제안하는 알고리즘은 업데이트 빈도에 따른 정렬을 수행하고 있고 추가적인 빈 블록 사용을 유발하지 않는 선에서 핫 데이터와 콜드 데이터를 분리해서 저장하고 있다. 이는 그림 2의 두 번째 과정에 해당하며, 선택된 희생블록 내에 존재하는 유효 페이지들은 업데이트 빈도에 따라 최소 힙(min-heap) 구조를 이용하여 메모리상에서 정렬이 된다. 이렇게 정렬이 되면, 유효 페이지들은 상대적으로 업데이트

표 2. 실험 환경
Table 2. Experimental environment

Chip	1 chip
Page Size(byte)	8192+640(Spare)
Block Size(pages)	256
Chip Size(blocks)	1024+84(Extended)
Random Read Time	60μs
Page Program Time	1500μs
Block Erase Time	5.0ms
P/E cycles	1000
Free Block trigger	5%

빈도가 비슷한 페이지들끼리 묶여서 블록에 저장된다. 만약 유효 페이지들의 갯수가 1개 이상의 빈 블록을 채울 정도로 많은 경우에는 업데이트 빈도가 낮은 유효 페이지들부터 빈 블록으로 옮긴다. 즉, 상대적으로 콜드에 가까운 페이지들이 빈 블록에 함께 저장된다. 빈 블록을 채우고 남은 유효 페이지들은 활성 블록으로 옮겨지며, 이 페이지들은 데이터 성격이 상대적으로 핫에 가깝다. 만약 가비지 컬렉션에 따라 발생한 유효 페이지들의 수가 빈 블록을 가득 채울 수 없을 정도로 적은 경우에는 유효 페이지들을 모두 활성 블록으로 옮긴다. 데이터가 모두 이동된 희생블록들은 삭제 연산이 이루어지고, 빈 블록은 할당되기 전에 대기하는 곳인 빈 블록 풀(Free Block Pool)에 넣어진다. 이는 그림 2의 세 번째와 네 번째 과정에 해당한다. 그림 2의 세 번째 과정에서 보는 것과 같이, 유효 페이지의 수가 1개 블록 내에서의 최대 페이지 수보다 큰 경우에 상대적으로 콜드 페이지에 해당하는 유효 페이지들 묶어서 빈 블록으로 먼저 이주를 시킨다. 이를 통하여 상대적으로 콜드에 해당하는 페이지들을 분리하여 저장할 수 있어 차후에 이루어지는 가비지 컬렉션의 오버헤드를 줄일 수 있다. 그림 2의 네 번째 과정은 남겨진 유효 페이지들을 기존의 활성 블록으로 이주시키는 것을 보여주고 있다. 남겨진 유효 페이지들은 상대적으로 업데이트가 자주 일어나 가비지 컬렉션의 오버헤드를 크게 유발시키지 않는다. 이러한 페이지들은 활성 블록으로 이주시키는 것이 공간 활용 측면에 있어 유리하다.

세 번째는 블록 할당 정책이다. 낸드 플래시 시스템에서 데이터는 할당된 활성 블록 내에 순차적으로 저장된다. 앞서 설명한 것처럼 활성 블록에 저장되는 데이터는 핫 데이터일 확률이 높아 한번 활성 블록으로 선정된 블록들은 빠른 시일 내에 또다시 희생블록으로 선정될 확률이 높다. 이러한 이유로 가장 마모도가 적은 블록을 활성 블록으로 선정하는 것이 가장 합리적이기 때문에 대부분의 가비지 컬렉션 기법에 활용하고 있다. 제안하는 기법 또한 기존의 기법들과 동일하게 가장 마모도가 적은 블록부터 할당에 사용할 수 있도록 최소 힙 구조를 사용하여 빈 블록 풀을 구성한다. 알고리즘 1은 제안한 기법의 의사코드를 보여주고 있다.

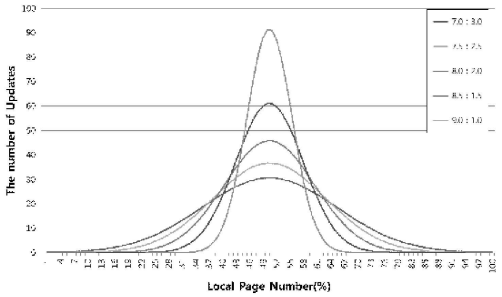

그림 3. 데이터 모음의 특징(콜드:핫 비율)
Fig. 3 Characteristics of data set(Cold:Hot ratio)

Ⅳ. 실 험

1. 환경 설정

본 논문에서 제안한 가비지 컬렉션 기법의 성능을 평가하기 위해 마이크로소프트사에서 개발한 SSD Extension for DiskSim을 사용하였다 [15]. 자세한 실험환경은 표 2와 같다.

Cold/Hot 비율에 따른 성능 평가를 위해, 실험에서는 그림 3과 같이 Cold/Hot 비율을 가지는 트레이스 파일을 제작하여 사용하였다. 데이터의 Cold/Hot은 접근 횟수에 따라 결정되며, 데이터의 접근횟수가 평균 접근횟수보다 낮은 데이터를 Cold라 하며 평균 접근횟수보다 높은 데이터는 Hot이라 한다. Cold 데이터의 비율이 낮은 경우에는 대부분의 블록들이 가진 데이터가 고르게 업데이트되어 가비지 컬렉션에 따른 오버헤드를 비교하기 힘들어 그림 3과 같은 Cold/Hot 비율로 성능 평가를 수행하였다.

실험은 LPN(Logical Page Number)에 해당하는 데이터를 업데이트 하는 방식으로 진행되었으며, 인접한 LPN의 경우 비슷한 접근 빈도를 가질 수 있도록 가우스 분포를 적용하였다. 시스템에 필요한 최소한의 빈 블록 수는 전체의 5%로 설정하였으며, 전체 용량의 85%에 데이터를 채운 상태에서 실험을 진행하였다. 제안한 기법의 성능을 비교하기 위

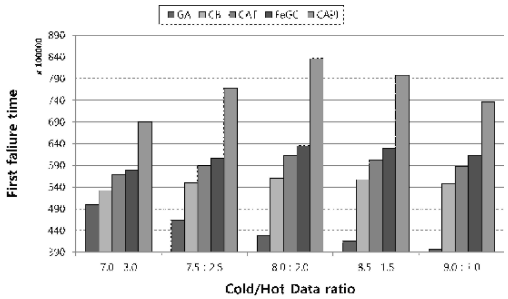

그림 4. 첫 번째 베드 블록 발생 시점
Fig. 4 First failure time

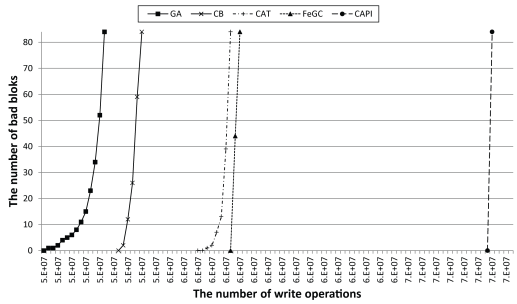

그림 5. 쓰기 연산에 따른 베드 블록의 수
Fig. 5 The number of bad blocks as write operations

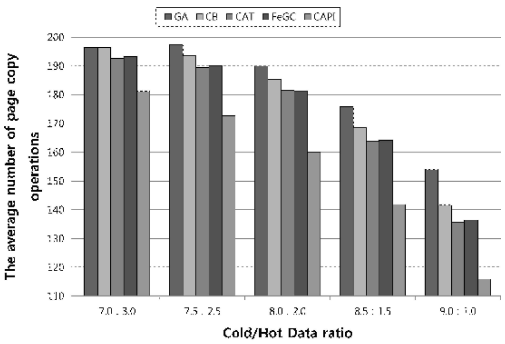

그림 6. 지움 연산에 따른 평균 페이지 복사 수
Fig. 6 The average number of page copies per erase operation

해 기존의 시간 기반 가비지 컬렉션 기법인 GA, CB, CAT, FeGC와 비교하였다.

2. 성능 평가

그림 4는 Cold/Hot 데이터 비율에 따라 각 기법들의 첫 번째 배드 블록이 발생하는 시점을 쓰기 연산 횟수를 기준으로 보여주고 있다. GA 알고리즘의 경우 Cold 비율이 높아질수록 희생블록으로 선정되는 블록의 수가 줄어들어 결국 전체 수명이 줄어드는 것을 확인할 수 있다. 제안한 기법은 GA에 비해 평균적으로 73%, CB에 비해 39%, CAT에 비해 29% 그리고 FeGC에 비해 24% 수명이 더 긴 것을 확인할 수 있다. 다만 Cold/Hot 데이터의 비율이 8.5:1.5와 9.0:1.0인 데이터 모음에서 실험 대상 기법들의 수명이 점점 줄어드는 것을 확인할 수 있는데, 이는 블록 내에 한 개의 무효 페이지도 발생하지 않는 완전 콜드 블록이 많기 때문이다. 가비지 컬렉션 정책상 이러한 블록들은 희생블록으로 선정되지 않게 된다. 이로 인하여 전체 블록의 수명은 불균형을 이루게 되고 저장 시스템의 수명이 줄어든다. 이는 정적 마모도 평준화를 통하여 해결할 수 있으나 본 논문의 범위를 벗어난다.

낸드 플래시는 제자리 갱신이 불가능한 특징으로 인하여 1개 블록을 사용하지 못해도 다른 블록들의 오버헤드가 증가한다. 왜냐하면 낸드 플래시는 업데이트가 발생하게 되면 빈 페이지에 쓰기 연산이 일어나는데, 사용할 공간이 줄어들면 그만큼 가비지 컬렉션이 빨리 발생하기 때문이다. 따라서 저장 시스템의 수명 및 오버헤드를 줄이기 위해서 대부분의 블록들이 동시에 배드 블록이 되도록 유도하는 것이 좋다.

그림 5는 Cold/Hot 데이터 비율이 7.0:3.0인 데이터 모음에서 쓰기 연산에 따라 연속적으로 발생되는 배드 블록 수를 보여주고 있다. FeGC와 제안한 기법의 경우에는 1개의 배드 블록이 발생한 이후에는 가파르게 배드 블록들이 발생하는 것을 볼 수 있는데, 이는 FeGC 기법과 CAPi 기법이 마모도가 가장 적은 블록을 할당하는 방법으로 블록들을 최대한 고르게 사용하기 때문이다. 그 결과로 첫 번째 배드 블록이 발생하는 시점을 크게 늦추고 있으며, 낸드 플래시 수명이 늘어났다.

그림 6은 희생블록에서 이주되는 평균 페이지 수를 Cold/Hot 데이터 비율에 따라 보여주고 있다. 콜드 데이터가 많다는 것은 상대적으로 적은 영역에 많은 업데이트가 이루어지고 있는 것이므로, 콜드 데이터가 많을수록 무효 페이지가 많은 블록이 희생블록으로 선정될 확률이 높아져 평가한 모든 가비지 컬렉션 기법들이 오버헤드가 줄어드는 것을 볼 수 있다. FeGC와 제안한 기법은 유효 페이지를 이주할 때, Cold/Hot 데이터를 분리하여 저장하는 방법으로 다음에 발생될 가비지 컬렉션 오버헤드를

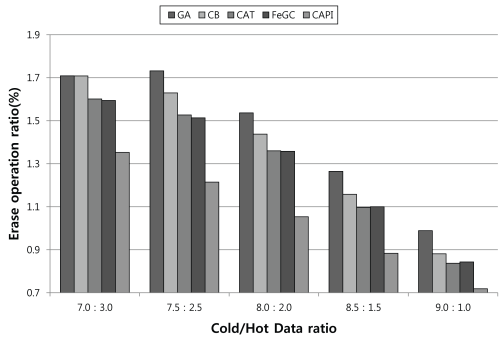

그림 7. 쓰기 연산 당 지움 연산 발생 빈도
Fig. 7 Erase operation ratio per write operation

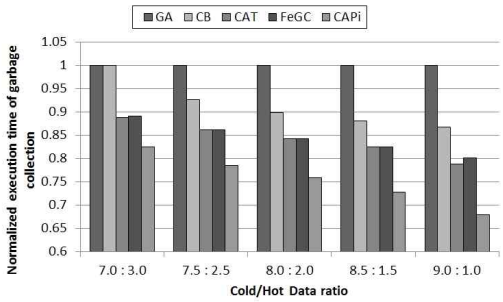

그림 8. 정규화한 가비지 컬렉션의 수행시간
Fig. 8 Normalized execution time of garbage collection

줄이고 있다. 다만 FeGC는 Cold/Hot 데이터를 따로 저장함에 있어 활성 블록을 활용하지 않아 추가적인 블록 사용에 의한 오버헤드가 발생하여, 전체 오버헤드를 크게 줄이지는 못했다. 제안한 기법은 GA에 비해 평균적으로 18%, CB에 비해 14%, CAT에 비해 11% 그리고 FeGC에 비해 12% 데이터 이주가 적게 발생하였다.

그림 7은 각 알고리즘들의 한 개의 쓰기 연산 시에 발생할 수 있는 삭제 연산 확률을 Cold/Hot 비율에 따라 보여주고 있다. 삭제 연산은 읽기 및 쓰기 연산에 비해 많이 느리므로 자주 발생하게 되면 가비지 컬렉션 수행 시 응답시간이 느려진다. 제안한 기법은 평균적으로 GA에 비해 38%, CB에 비해 30%, CAT에 비해 22% 그리고 FeGC에 비해 22% 삭제 연산이 적게 발생하였다.

그림 8은 각 알고리즘별로 정규화된 가비지 컬렉션 수행시간을 보여주고 있다. 제안된 기법은 GA, CB, CAT와 다르게 정렬을 수행하고 있어 정렬 수행시간에 대한 오버헤드를 가진다. 하지만 그림 6에서 설명한 바와 같이, 제안한 기법은 가비지 컬렉션을 수행함에 있어 가장 큰 오버헤드에 해당하는 I/O를 최대 18% 줄였다. 즉, 제안하는 기법은 정렬을 수행함에 따라 발생되는 추가적인 오버헤드보다 더 큰 I/O 연산의 빈도를 줄임으로써 다른 기법들에 비해 더 적은 가비지 컬렉션 수행시간을 가진다. 결론적으로 제안한 기법은 GA에 비해 평균적으로 31%, CB에 비해 20%, CAT에 비해 11% 그리고 FeGC에 비해 11% 적은 가비지 컬렉션 수행시간을 가진다.

V. 결 론

본 논문에서는 차세대 사물 인터넷 기반 시스템에서 각광받고 있는 낸드 플래시 메모리 환경에서 적용 될 수 있는 새로운 형태의 시간기반 가비지 컬렉션 기법으로 CAPi를 제안하였다. 제안한 기법은 가비지 컬렉션에서 발생할 수 있는 비용을 계산하여 전체 오버헤드를 줄이고 있다. 또한 가비지 컬렉션에서 발생하는 유효 페이지 이주에서 Cold/Hot 데이터를 분리 저장하는 기법을 통하여 차후에 발생할 수 있는 오버헤드를 줄이고 있다. 이를 통하여, 제안한 기법은 GA에 비해 평균적으로 73%, CB에 비해 39%, CAT에 비해 29% 그리고 FeGC에 비해 24% 수명이 더 긴 것을 확인할 수 있었고, 가비지 컬렉션에서 발생할 수 있는 유효 페이지 쓰기 연산 또한 기존의 기법들과 비교하여 평균적으로 18%, CB에 비해 14%, CAT에 비해 11% 그리고 FeGC에 비해서는 12% 감소하는 것을 확인하였다. 차후 연구로 가비지 컬렉션에서 대상으로 선정되지 않는 완전 콜드 블록도 고려하는 복합적인 가비지 컬렉션 및 마모도 평준화 기법에 대한 연구를 진행할 예정이다.

References

[1] M. Wu, W. Zwaenepoel, “eNVy: a non-volatile, main memory storage system,” Proceedings of the 6th international conference on Architectural support for programming languages and operating systems, Vol. 29, No. 11, pp. 86-97, 1994.

[2] A. Kawaguchi, N. Shingo, M. Hiroshi, "A Flash-Memory Based File System," USENIX,

pp.155-164, 1995.

[3] M.L. Chiang., R.C. Chang, "Cleaning policies in mobile computers using flash memory," Journal of Systems and Software Vol. 48, No. 3, pp. 213-231, 1999.

[4] O. Kwon, K. Koh, J. Lee, H. Bahn, "FeGC: An efficient garbage collection scheme for flash memory based storage systems," Journal of Systems and Software, vol. 84, no. 9, pp. 1507-1523, 2011.

[5] F. Chen, D.A. Koufaty, X. Zhang, "Understanding intrinsic characteristics and system implications of flash memory based solid state drives," Proceedings of the 11th international joint conference on Measurement and modeling of computer systems, Vol. 37. No. 1, pp. 181-192, 2009.

[6] L. Han, Y. Ryu, K. Yim, "CATA: a garbage collection scheme for flash memory file systems," Proceedings of 3rd International Conference on Ubiquitous Intelligence and Computing, pp. 103-112, 2006.

[7] L.Z. Han, Y. Rye, T.S. Chung, M. Lee, S. Hong, "An intelligent garbage collection algorithm for flash memory storages," Proceedings of International Conference on Computational Science and Its Applications, pp.1019-1027, 2006.

[8] M.W. Lin, S.Y. Chen, Y. Lu, Z. Zhou, "Garbage collection policy for flash-aware Linux swap system," IEEE Electronics letters, vol. 47 No. 22, pp. 1218-1220, 2011.

[9] O. Kwon, K. Koh, "Swap space management technique for portable consumer electronics with NAND flash memory," IEEE Transactions on Consumer Electronics, Vol. 56, No. 3, pp. 1524-1531, 2010.

[10] G. Xu, Y. Liu, X. Zhang, M. Lin, "Garbage collection policy to improve durability for flash memory," IEEE Transactions on Consumer Electronics, Vol. 58, No. 4, pp. 1232-1236, 2012.

[11] G. Xu, M. Wang, Y. Liu, "Swap-aware garbage collection algorithm for NAND flash-based consumer electronics," IEEE Transactions on Consumer Electronics, Vol. 60, No. 1, pp. 60-65, 2014.

[12] S. Jung, H.S. Yong, "LINK-GC: a preemptive approach for garbage collection in NAND flash storages," Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, pp. 1478-1484, 2013.

[13] M.C. Yang, Y.M. Chang, C.W. Tsao, P.C. Huang, Y.H. Chang, T.W. Kuo, "Garbage collection and wear leveling for flash memory: Past and future," Proceedings of International Conference on Smart Computing, pp. 66-73, 2014.

[14] S. Lee, D. Shin, J. Kim, "Bagc: Buffer-aware garbage collection for flash-based storage systems," IEEE Transactions on Computers, Vol. 62, No. 11, pp. 2141-2154, 2013.

[15] V. Prabhakaran, T. Wobber, "SSD extension for DiskSim simulation environment," Microsoft Research, 2009.

Sang-Ho Hwang (황 상 호)

He received the B.S. and M.S. degrees in Computer Engineering from Yeungnam University, Korea, in 2009 and 2013 respectively. His current research interests include embedded systems and non-volatile memory systems.

Email: snailcom@ynu.ac.kr

Jong Wook Kwak (곽 종 욱)

He 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.
Email: kwak@yu.ac.kr