논문 2017-12-14

실시간 시스템에서의 플래시 메모리 저장 장치를 위한 적응적 가비지 컬렉션 정책

(A Adaptive Garbage Collection Policy for Flash-Memory Storage System in Embedded Systems)

박 송 화*, 이 정 훈, 이 원 오, 김 희 언
(Song-Hwa Park, Jung-Hoon Lee, Won-Oh Lee, Hee-Earn Kim)

Abstract : NAND flash memory has advantages of non-volatility, little power consumption and fast access time. However, it suffers from inability that does not provide to update-in-place and the erase cycle is limited. Moreover, the unit of read/write operation is a page and the unit of erase operation is a block. Therefore, erase operation is slower than other operations. The AGC, the proposed garbage collection policy focuses on not only garbage collection time reduction for real-time guarantee but also wear-leveling for a flash memory lifetime. In order to achieve above goals, we define three garbage collection operating modes: Fast Mode, Smart Mode, and Wear-leveling Mode. The proposed policy decides the garbage collection mode depending on system CPU usage rate. Fast Mode selects the dirtiest block as victim block to minimize the erase operation time. However, Smart Mode selects the victim block by reflecting the invalid page number and block erase count to minimizing the erase operation time and deviation of block erase count. Wear-leveling Mode operates similar to Smart Mode and it makes groups and relocates the pages which has the similar update time. We implemented the proposed policy and measured the performance compare with the existing policies. Simulation results show that the proposed policy performs better than Cost-benefit policy with the 55% reduction in the operation time. Also, it performs better than Greedy policy with the 87% reduction in the deviation of erase count. Most of all, the proposed policy works adaptively according to the CPU usage rate, and guarantees the real-time performance of the system.

Keywords : NAND Flash memory, Garbage collection, Wear-leveling

Ⅰ. 서 론

플래시 메모리는 가볍고 충격에 강하며 저전력의 특성을 갖기 때문에, 모바일 장치, 디지털 카메라와 같은 임베디드 시스템에서 사용이 증가하고 있다. NAND 플래시 메모리는 제품의 용량 향상과 단가 하락으로 인하여 SSD (Solid State Drive)와 같은 고성능, 저전력 저장장치에 장착되어 노트북 시장뿐만 아니라 데스크톱 PC 시장에서도 사용이 확대되었다.

플래시 메모리를 저장장치로 사용하기 위해서는 극복해야 할 두 가지 단점이 있다. 첫째, 플래시 메모리는 데이터 수정 시 본래 주소에 덮어쓰기 (update-in-place)가 불가능하다. 플래시 메모리의 각 비트가 단방향으로만 토글링 (toggling)되기 때문에 쓰기 연산 시 지움 연산을 선행해야 한다. 즉, 쓰기 연산 전에 초기화된 메모리 공간의 확보가 필요하다. 둘째, 플래시 메모리의 각 블록은 지움 연산의 횟수가 제한되어 있기 때문에 플래시 메모리의 전체 공간이 균등하게 사용되지 못하면 사용가능한 메모리 공간이 급격하게 줄어들게 된다 [1].

플래시 메모리 기반의 임베디드 시스템은 성능과 신뢰성을 만족시키기 위하여 앞에서 언급된 두

*Corresponding Author (shpark25@lignex1.com)
Received: May 12 2017, Revised: May 23 2017, Accepted: May 24 2017.
S. Park, J. Lee, W. Lee, H. Kim: LIG Nex1

 2017 June 12(3) 121-130
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2017.12.3.121

가지 단점을 극복하는 것이 필요하다. 첫 번째 단점을 극복하기 위해서는 일반적으로 갱신 연산이 발생했을 때 다른 공간에 새로운 데이터를 쓰고 이전의 데이터는 무효화시키는 방법을 사용한다. 그리고 새로운 공간 확보를 위해서는 무효화된 데이터를 가지고 있는 블록들에 대해서 가비지 컬렉션 (Garbage Collection)을 수행한다. 두 번째 단점을 극복하기 위해서는 블록의 지움 횟수를 고려하여 블록을 관리한다. 지움 횟수가 지나치게 높은 블록의 활용을 줄이고 비교적 낮은 지움 횟수를 가지는 블록의 활용도를 높임으로써 전체 블록의 지움 횟수에 대해 균등화 정책 (Wear-leveling policy)이 필요하다.

가비지 컬렉션을 수행하면 지움 연산뿐만 아니라 대상 블록의 유효 페이지들을 새로운 영역에 복사하는 부하 (overhead)가 발생한다. 따라서 가비지 컬렉션의 빠른 수행을 위해서는 유효 페이지 복사 과정을 최소화해야 하고, 플래시 메모리의 수명을 고려하기 위해서는 전체 블록이 균등하게 사용되도록 지움 대상 블록을 적절히 선택해야 한다. 특히, 임베디드 시스템의 저장장치로 플래시 메모리를 사용하는 경우에는 가비지 컬렉션 수행으로 인한 시스템 연산 속도의 저하가 발생하지 않아야 한다.

이에 본 논문에서는 NAND 플래시 메모리를 저장장치로 사용하는 임베디드 시스템에서 실시간성을 보장하고 효율적인 지움 연산 및 지움 균등을 고려한 적응적 가비지 컬렉션 정책 (AGC: Adaptive Garbage Collection)을 제안한다. 제안한 정책은 임베디드 시스템의 CPU 사용률에 따라 3가지 모드 (Fast Mode, Smart Mode, Wear-leveling Mode)로 동작하며, 각 모드는 CPU 사용률에 따라 지움 연산 속도와 지움 균등을 고려한 가비지 컬렉션을 수행한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 플래시 메모리와 기존의 가비지 컬렉션 정책 연구에 대해 기술하고, Ⅲ장에서는 본 논문에서 제안하는 가비지 컬렉션 정책에 대해 설명한다. Ⅳ장에서는 제안하는 정책의 성능을 실험을 통해 평가하고, Ⅴ장에서는 결론 및 향후 과제에 대해 제시한다.

Ⅱ. 관련 연구

1. 플래시 메모리

플래시 메모리는 일종의 EEPROM (Electrically Erasable and Programmable ROM)으로 전기적으로 데이터를 지우고 다시 기록할 수 있는 비휘발성 (non-volatile) 컴퓨터 기억 장치이다. 부피가 작고 가벼우며 소비 전력이 낮고 입출력 속도가 빠를 뿐만 아니라 EEPROM보다 가격이 싼 장점으로 인하여 저장 매체로서 사용이 증가하고 있다 [1].

플래시 메모리는 게이트 (gate) 타입과 셀 (cell)을 구성하는 구조에 따라 NOR 플래시 메모리와 NAND 플래시 메모리로 구분된다. NOR 플래시 메모리는 버스 형태의 외부 인터페이스를 가지고 있어 바이트 단위의 임의 접근이 가능하고, CPU에 직접 연결되어 CPU가 수행하는 코드를 직접 실행할 수 있다. 또한 바이트 단위의 프로그래밍이 가능하여 주로 코드를 저장하고 실행하는 용도로 사용된다. NAND 플래시 메모리는 NOR 플래시 메모리에 비해 블록 단위의 지움 연산 속도와 페이지 단위의 쓰기 연산 속도가 빠르다. 그러나 임의 접근 속도가 느리며 읽기와 쓰기 연산이 페이지 단위로 이루어진다. NAND 플래시 메모리는 느린 임의 접근 속도 때문에 음악 파일이나 이미지 파일 등 대용량의 데이터를 저장하는 용도로 주로 사용된다.

NAND 플래시 메모리의 최소 읽기/쓰기 단위인 페이지는 메모리 소자의 집적도를 높이기 위해 같은 워드라인을 공유하도록 설계되었기 때문에, 메모리 디바이스의 사용에 있어 제약사항이 존재한다. NAND 플래시의 가장 주요한 약점 중 하나는 비트 단위로 데이터를 저장하는 셀의 플로팅 게이트에 전하가 인가된 후에는 페이지의 집합인 블록 지움 연산을 통해서만 전하의 제거가 가능하다는 것이다. 이것은 한 셀에 데이터의 덮어쓰기가 불가능하다는 것을 의미하며, 따라서 NAND 플래시는 '쓰기-전-지움 (erase-before-write)' 연산이 필수적인 특성을 갖게 된다. 또한, NAND 플래시는 셀의 플로팅 게이트와 트랜지스터 게이트 사이에 존재하는 산화막의 마모에 의해 불규칙적인 전하의 누수가 발생하며, 그로 인한 셀 단위 비트 에러율 (BER: Bit Error Rate) 증가로 블록 단위로 수명이 제한되는 문제가 발생한다. 즉, 일정 횟수의 쓰기/지움 연산이 수행된 후에는 플로팅 게이트의 전하 저장이 원활하게 수행되지 않으므로 페이지의 BER이 급격히 상승하며, 이로 인해 NAND 플래시의 수명이 단축된다.

플래시 메모리를 저장 장치로 사용하는 임베디드 시스템은 성능과 신뢰성을 만족시키기 위하여 앞에서 언급된 두 가지 단점을 극복하는 것이 필요하며, 이를 위하여 많은 가비지 컬렉션 정책이 연구되어 왔다.

2. 가비지 컬렉션

과거 플래시 메모리 기반 시스템에서의 가비지 컬렉션 연구는 가비지 컬렉션 수행 횟수를 최소화하고 플래시 메모리의 수명을 연장시키는데 목표가 있었다.

Greedy 정책 [2, 3]은 유효하지 않은 자료가 가장 많은 블록을 선택하여 지움 연산을 수행하는 방법으로서, 삭제 횟수는 가능한 최소화하면서 많은 공간을 확보한다. 지움 연산의 효율성은 좋으나 블록의 지움 횟수가 고려되지 않아 플래시 메모리의 수명을 단축시킬 수 있는 단점이 있다.

Cost-benefit 정책 [4]은 특정 블록에 집중된 지움 연산으로 인하여 플래시 메모리의 수명이 단축되는 문제점을 개선하기 위하여 연구된 정책이다. Cost-benefit 정책은 삭제의 효율성뿐만 아니라 특정 블록에 대한 지움 연산 집중을 피하기 위해, 블록이 사용된 최근 시간을 고려하여 삭제 할 블록을 선택한다.

이 외에도 지움 블록은 Greedy 정책과 동일하게 선정하고 블록 균등 사용을 위한 할당 기법을 제안한 SWAP 기법 [5], 가비지 컬렉션에 영향을 미치는 요소들을 정의하고 가비지 컬렉션 연산 횟수를 낮추기 위한 기법을 제안한 MODA [6], 가비지 컬렉션에서 발생할 수 있는 오버헤드를 줄여 응답시간 및 수명을 증가시키는 기법을 제안한 CAPi [7], BIT와 대리블록을 이용한 가비지 컬렉션 기법인 PBGC [8] 등이 제안되었다.

하지만 기존에 제안된 정책들은 지움 연산의 효율성 또는 플래시 메모리의 수명 중 하나만을 고려하였거나 임베디드 시스템의 실시간성을 고려하지 않았다. Greedy 정책은 지움 연산을 효율적으로 수행하여 시스템의 실시간성은 보장하지만 블록의 지움 균등을 고려하지 않아 플래시 메모리의 수명이 단축될 수 있다. 반면 Cost-benefit 정책은 플래시 메모리의 수명 단축은 고려하였으나 Greedy 정책에 비하여 지움 연산에 부하가 발생하므로 시스템의 CPU 사용률이 높은 경우에는 플래시 처리 응답이 늦어질 수 있다. 플래시 처리 응답이 늦어지는 것은 시스템의 실시간성에 영향을 미칠 수 있음을 의미한다.

이에 본 논문에서는 플래시 메모리를 저장장치로 사용하는 임베디드 시스템에서 지움 연산의 효율성과 플래시 메모리의 수명뿐만 아니라 실시간성 보장을 위하여 CPU 사용률에 따라 적응적 가비지 컬렉션을 수행하는 기법을 제안한다.

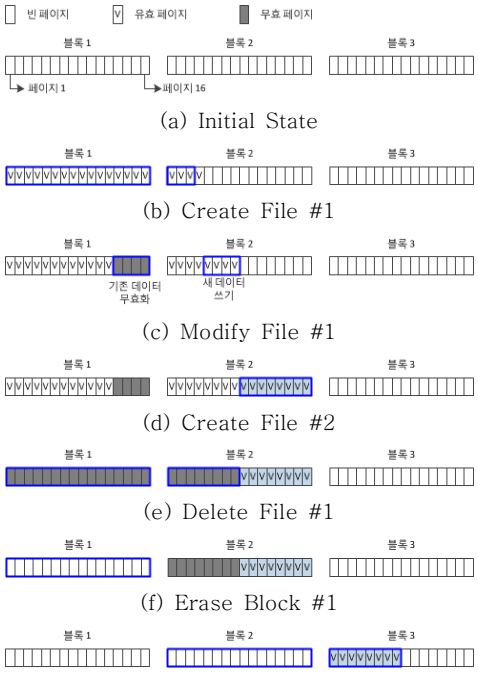


그림 1. 파일 연산에 따른 플래시 메모리 상태 변화
Fig. 1 Status Change of Flash Memory on File Operation

Ⅲ. 제안하는 기법

1. 플래시 메모리 상태 분석

파일 연산 시, 플래시 메모리의 상태는 그림 1과 같이 변화한다. 초기 플래시의 상태는 (a)와 같이 모든 블록이 초기화되어 있다. 파일을 생성하면, (b)와 같이 데이터 쓰기를 위하여 페이지 할당 및 데이터 쓰기 연산이 발생한다. 생성한 파일의 일부분이 수정되면 갱신된 데이터를 쓰기 위하여 (c)와 같이 페이지 할당 및 쓰기 연산이 발생하고 기존 데이터가 저장된 페이지는 무효 처리된다. 파일 생성 및 삭제 연산이 발생하면 (e)와 같이 블록 내의 모든 페이지 또는 일부 페이지가 무효 처리된다. (e)의 상태에서 블록 1을 지울 경우에는 모든 페이지가 무효하므로 (f)와 같이 지움 연산 시 유효 페이지 복사가 발생하지 않지만, 블록 2를 지울 경우에는 (g)와 같이 유효 페이지의 복사가 발생하게 된다. 즉, 지움 연산을 수행할 블록의 상태에 따라 유효 페이지 복사로 인한 추가적인 연산이 발생하는 것을 알 수 있으며 이는 곧 플래시 처리 응답 시간에 영향을 주게 된다.

표 1. 블록 상태 정보
Table 1. Block Status Information

Item	Description
Block Number	• Based on the physical location of the block • Sequential increase from 1
Degree of invalidation	• The sum of the number of empty pages and the number of invalid pages in the block
Erase count	• Block erase operation execution count
Erase Deviation	• A value obtained by subtracting the erase count of block from the erase count average of the entire block
Score	• Score to be selected as a erase(victim) block • Score = Invalidity×W_1 + ED×W_2, (W_1 + W_2 = 1.0) - Invalidity : Degree of invalidation - W_1 : Weight of the Invalidity - ED : Erase Deviation - W_2 : Weight of the ED • The greater the efficiency of erase operations and wear-leveling effect, the higher the score

Block Number	1	2	3	4	5	6	···	2048
Degree of invalidation	0	16	3	2	9	0	···	2
Erase count	56	2	7	34	18	10	···	33
Erase Deviation	-32.5	21.5	16.5	-10.5	5.5	13.5	···	-9.5
Score	-16.25	18.75	9.75	-4.25	7.25	6.75	···	-3.75

그림 2. 블록 상태 정보의 예
Fig. 2 Example of Block Status Information

2. 플래시 메모리 상태 관리

본 논문에서 제안하는 적응적 가비지 컬렉션 정책은 지움 연산의 효율성과 균등 지움을 고려하기 위하여 블록 상태 정보, 페이지 상태 정보 및 자유 블록 정보를 관리한다.

2.1 블록 상태 정보

지움 블록 선택 시, 지움 연산 속도와 균등 지움을 고려하기 위하여 표 1, 그림2와 같이 플래시 메모리를 구성하는 블록의 상태 정보를 관리한다.

블록 무효도는 블록에 포함된 유효 페이지의 개수와 반비례하므로, 블록 무효도가 높은 블록일수록

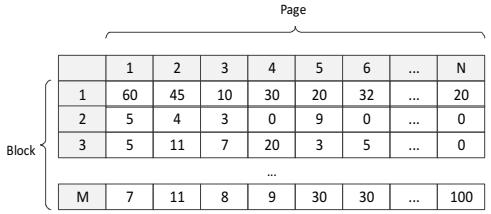


그림 3. 페이지 상태 정보의 예
Fig. 3 Example of Page Status Information

지움 연산에 소요되는 시간이 적다. 블록의 지움 편차는 전체 블록의 지움 횟수 평균에서 해당 블록의 지움 횟수를 뺀 값으로, 지움 횟수 편차가 음수인 경우에는 해당 블록의 지움 횟수가 상대적으로 높은 것을 의미하고, 양수인 경우에는 지움 횟수가 상대적으로 낮은 것을 의미한다. 블록 Score는 각 블록의 무효도와 지움 편차를 반영하므로 블록 Score를 사용하여 지움 블록을 선택하는 경우, 무효도를 사용하여 지움 블록을 선택한 경우보다 지움 연산 속도는 느리나 균등 지움 효과가 발생한다.

2.2 페이지 상태 정보

Ⅲ.1 절에서 살펴본 바와 같이, 블록을 구성하는 페이지들의 갱신 경향이 유사할수록 해당 블록의 지움 연산을 효율적으로 처리할 수 있다.

블록을 구성하는 페이지들의 갱신이 빈번한 경우, 해당 블록의 무효도는 높으며 지움 연산이 빈번하게 발생한다. 하지만 지움 연산 발생 시 유효 페이지 개수가 적으므로 유효 페이지 복사 연산은 적게 발생한다. 반면, 블록을 구성하는 페이지들의 갱신 빈도가 낮은 경우, 해당 블록의 무효도는 낮으며 상대적으로 지움 블록으로 선택될 확률이 적다. 블록을 구성하는 페이지의 갱신 정도가 다양한 경우, 상대적으로 지움 블록 연산 시 유효 페이지 복사가 많이 발생하고 지움 블록으로 선택될 확률도 높을 수 있다. 따라서 블록 내에 갱신 경향이 유사한 페이지들을 배치하게 되면 지움 연산 속도의 향상 효과를 기대할 수 있다.

페이지의 갱신 정도를 관리 및 예측하기 위하여 그림 3과 같이 페이지의 Age 정보를 관리한다. Age는 페이지의 갱신 이후 경과된 시간으로, Age가 낮을수록 갱신 빈도가 높으므로 향후에도 갱신이 많이 발생할 것으로 예측하며, Age가 높을수록 갱신 빈도가 낮으므로 향후에도 갱신이 적게 발생할 것으로 예측한다.

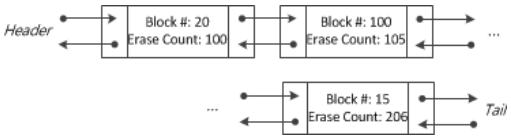


그림 4. 자유 블록 정보 이중 연결 리스트의 예
Fig. 4 Example of Free Block Information Doubly Linked List

2.3 자유 블록 정보

블록 할당을 빠르게 수행하기 위하여 그림 4와 같이 초기화된 자유 블록을 이중 연결 리스트로 관리한다. 리스트를 구성하는 노드는 블록의 번호와 지움 횟수 정보로 구성되며, 지움 횟수를 기준으로 오름차순으로 정렬된다. 지움 균등을 고려한 쓰기 블록 할당 시에는 리스트의 헤더 (header)가 가리키는 블록을 할당한다. 지움 연산 수행 중 Age가 낮은 페이지들을 백업하기 위하여 블록을 할당할 경우에는 테일 (tail)이 가리키는 블록, 즉 지움 횟수가 높은 블록을 할당하여 해당 블록에 대해 지움 연산이 적게 발생할 것을 기대한다.

3. 가비지 컬렉션의 동작 모드

본 논문에서는 표 2, 그림 5와 같이 시스템의 CPU 사용률에 따라 3가지 모드로 동작하는 적응적 가비지 컬렉션 (AGC) 정책을 제안한다.

3.1 Fast Mode

Fast Mode는 CPU 사용률이 높은 상황에서의 동작 모드이다. 시스템의 CPU 사용률이 높을 경우에 시스템의 실시간성을 만족하기 위해서는 가비지 컬렉션 수행에 소요되는 CPU 사용률이 적어야 하며 플래시 처리 응답이 빨라야 한다. Fast Mode는 CPU 사용률의 최소화와 빠른 플래시 처리 응답을 위하여 지움 연산의 부하가 최소화하도록 동작한다. 이를 위하여 Fast Mode는 가비지 컬렉션 수행 시, 무효도가 높은 블록을 지움 블록으로 선택한다. 그림 6에서 Fast Mode는 무효도가 가장 높은 블록 2를 지움 블록으로 선택하며, 자유 블록 중 지움 횟수가 가장 적은 블록 20을 쓰기 블록으로 할당한다.

3.2 Smart Mode

Smart Mode는 지움 연산의 효율성과 균등 지움을 동시에 고려한 모드로, 제안하는 가비지 컬렉션의 기본 모드이다. Smart Mode는 지움 블록

표 2. 제안하는 적응적 가비지 컬렉션의 동작 모드
Table 2. Operation Mode of Proposed Adaptive Garbage Collection

Category		Fast Mode	Smart Mode	Wear-leveling Mode
CPU Usage		High	Medium	Low
Consider erase operation speed		O	O	O
Consider Wear-leveling	Selection of a erase block	X	O	O
	Select of a backup block	X	X	O

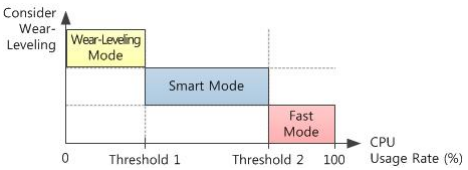


그림 5. CPU 사용률에 따른 적응적 가비지 컬렉션 동작 모드
Fig. 5 Operation Mode of Adaptive Garbage Collection on CPU Usage Rate

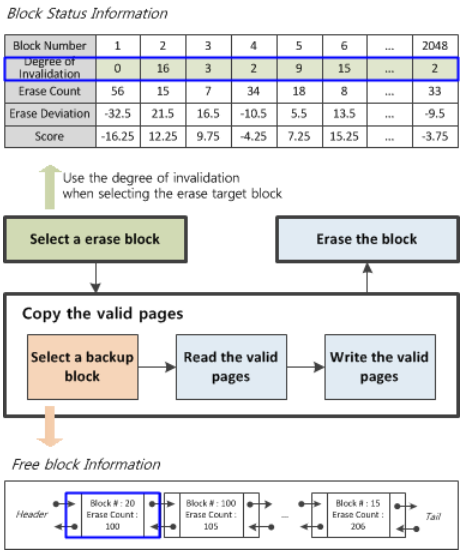


그림 6. 제안한 가비지 컬렉션의 Fast Mode 동작
Fig. 6 Fast Mode Operation of Proposed Garbage Collection

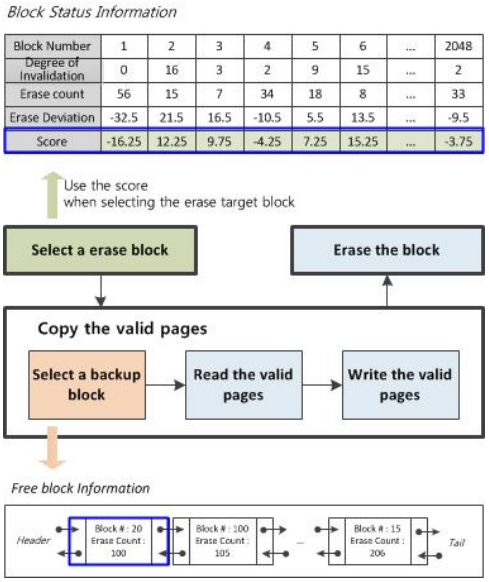


그림 7. 제안한 가비지 컬렉션의 Smart Mode 동작

Fig. 7 Smart Mode Operation of Proposed Garbage Collection

선택 시, 블록의 무효도와 지움 횟수를 고려하기 위하여 블록의 Score 정보를 사용하며, Score가 높은 블록을 지움 블록으로 선택한다. 그림 7에서 Smart Mode는 Score가 가장 높은 블록 6을 지움 블록으로 선택하며, 자유 블록 중 지움 횟수가 가장 적은 블록 20을 쓰기 블록으로 할당한다.

3.3 Wear-leveling Mode

Wear-leveling Mode의 기본 동작은 Smart Mode와 동일하며, 다른 모드와 달리 유효 페이지 복사 이전에 유효 페이지의 재그룹화 과정을 수행하여 유효 페이지 복사 과정에서도 지움 균등을 고려한다. 그림 8에서 Wear-leveling Mode는 Smart Mode와 동일하게 Score가 가장 높은 블록 6을 지움 블록으로 선택한다. Wear-leveling Mode는 유효 페이지 복사 전에 지움 블록으로 선택된 블록들의 유효 페이지들을 페이지 상태 정보를 사용하여 Age가 유사한 페이지들을 그룹화 한다. 다른 모드에서 쓰기 블록은 항상 지움 횟수가 가장 적은 블록이 할당되었으나, Wear-leveling Mode에서는 Age가 낮은 페이지들을 동일 블록에 저장하고, 쓰기 블록으로 지움 횟수가 낮은 블록을 할당한다. 즉, 지움 횟수가 낮은 블록에 갱신 빈도가 높은

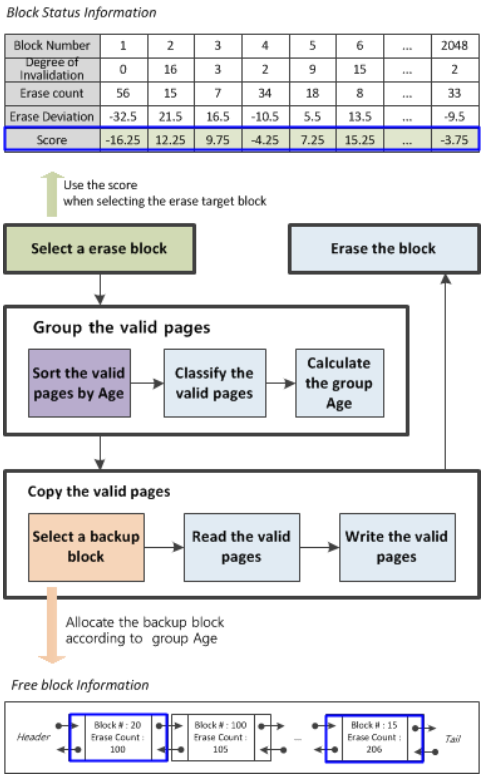


그림 8. 제안한 가비지 컬렉션의 Wear-leveling Mode 동작

Fig. 8 Wear-leveling Mode Operation of Proposed Garbage Collection

데이터를 저장하여 블록들의 지움 횟수 분산이 적게 되도록 한다. 갱신 빈도가 낮아 Age가 높은 페이지들도 동일 블록에 저장하며, 쓰기 블록으로 지움 횟수가 높은 블록을 할당한다. 즉, 지움 횟수가 높은 블록에 갱신 빈도가 낮은 데이터를 저장하여 해당 블록이 지움 블록으로 선택될 확률을 낮추도록 한다. 그림 8에서 Wear-leveling Mode는 Age가 높은 페이지들의 복사를 위해서 자유 블록 중 지움 횟수가 가장 적은 블록 20을 쓰기 블록으로 할당하고, Age가 낮은 페이지들의 복사를 위해서는 지움 횟수가 가장 높은 블록 15를 쓰기 블록으로 할당한다.

Ⅳ. 성능 평가

1. 실험 환경

본 논문에서 제안한 적응적 가비지 컬렉션 정책

의 성능을 평가하기 위하여 SSD Extension for DiskSim Simulation을 참고하였으며 [9, 10], 시뮬레이션 환경은 표 3과 같다.

제안한 기법에 적용한 파라미터 값은 표 4와 같으며, 실험에 사용된 시스템의 CPU 사용률과 제안한 가비지 컬렉션의 동작모드는 그림 9와 같다.

제안한 기법의 성능 평가를 위하여 표 5의 연산이 랜덤하게 발생하도록 하고, 연산 수행 및 가비지 컬렉션에 소요된 시간을 측정하였다.

2. 기존 기법과의 성능 비교

제안한 기법의 성능을 평가하기 위하여 기존의 Greedy 정책, Cost-benefit 정책의 성능과 비교하였다. 3 가지 기법에 대해 동일한 환경에서 동일한 파일 연산을 수행하도록 하였으며 가비지 컬렉션 수행 시점도 동일하도록 하였다. 각 기법에서 소요된 연산 시간과 가비지 컬렉션 소요시간은 각각 그림 10~12와 같다.

Greedy 정책 (그림 10)은 유효하지 않은 자료가 가장 많은 블록을 선택하여 지움 연산을 수행한다. 따라서 세 가지 기법 중 지움 연산의 부하가 가장 적으므로 연산 및 가비지 컬렉션에 소요된 시간이 가장 짧다. 또한 다른 기법과 비교하였을 때, 가비지 컬렉션이 발생하는 시점에서 플래시 처리 응답이 지연되는 정도가 적다.

Cost-benefit 정책 (그림 11)은 지움 연산의 효율성뿐만 아니라 블록이 사용된 최근 시간을 고려하여 지움 블록을 선택한다. 따라서 세 가지 기법 중

표 3. 시뮬레이션 파라미터
Table 3. Parameter of Simulation

Item	Value
Condition of Triggering	# (Empty Block) < 5 %
#blocks/chip	2,048
#pages/block	64
Page Read/Write Latency	60 us / 800 us
Block Erase Latency	1.5 ms
Program-Erase (P/E) Cycle	10,000

표 4. 제안한 가비지 컬렉션 정책의 파라미터
Table 4. Parameter of Proposed Garbage Collection Policy

Category	Item	Value
Operation Mode	$Threshold_1$	30.0
	$Threshold_2$	70.0
Score	W_1	0.5
	W_2	0.5

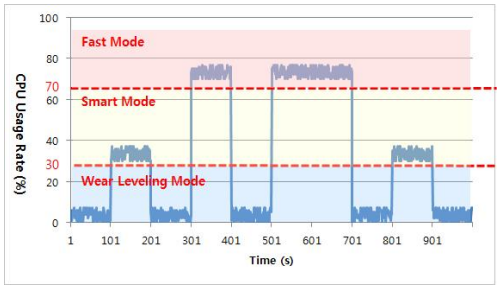
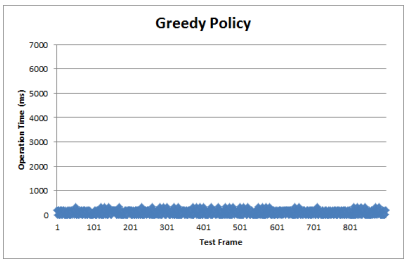


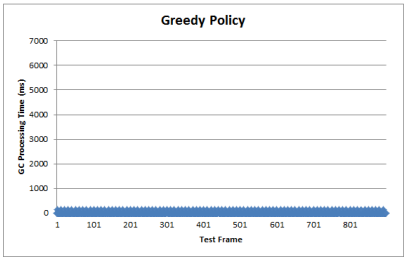
그림 9. 시험에 사용된 CPU 사용률 및 가비지 컬렉션의 동작 모드
Fig. 9 CPU Usage Rate and Operation Mode of Proposed Garbage Collection for the Test

표 5. 실험에 사용된 파일 연산
Table 5. File Operation used at Experiment

File Operation	Description
Creation (Write)	• File Size: 1 MB
Read	• File Selection: Random • Read Size: Random
Modification	• File Selection: Random • Modification Size: Random
Deletion	• File Selection: Random

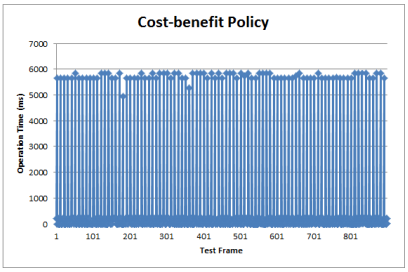


(a) Operation Time

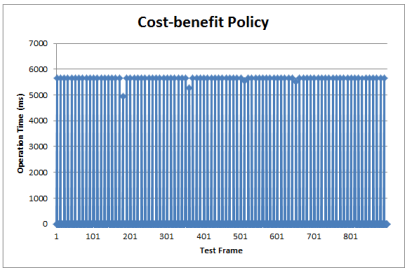


(b) Garbage Collection Processing Time

그림 10. Greedy 정책의 연산 및 가비지 컬렉션 소요 시간
Fig. 10 Operation and Garbage Collection Processing Time of Greedy Policy



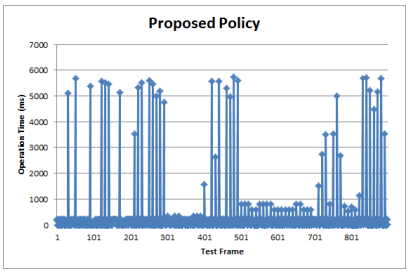
(a) Operation Time



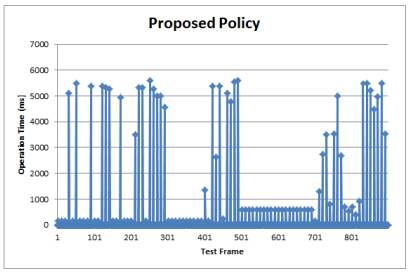
(b) Garbage Collection Processing Time

그림 11. Cost-benefit 정책의 연산 및 가비지 컬렉션 소요 시간

Fig. 11 Operation and Garbage Collection Processing Time of Cost-benefit Policy



(a) Operation Time



(b) Garbage Collection Processing Time

그림 12. 제안한 정책의 연산 및 가비지 컬렉션 소요 시간

Fig. 12 Operation and Garbage Collection Processing Time of the Proposed Policy

지움 연산의 부하가 가장 크며 연산시간 및 가비지 컬렉션에 소요된 시간이 가장 길다. 또한 다른 기법과 비교하였을 때, 가비지 컬렉션이 발생하는 시점에서 플래시 처리 응답 지연이 가장 크다.

제안한 정책 (그림 12)은 CPU 사용률에 따라 적응적으로 동작한다. CPU 사용률이 높은 경우에는 지움 연산의 효율성만을 고려하고 CPU 사용률이 낮은 경우에는 지움 연산의 효율성 및 지움 균등을 고려하여 가비지 컬렉션을 수행한다. 따라서 세 가지 기법 중 유일하게 CPU 사용률에 따라 연산 및 가비지 컬렉션에 소요된 시간이 변화하며, CPU 사용률의 경향과 반대의 경향을 보인다. 즉, CPU 사용률이 높은 경우에는 Fast Mode로 동작하여 연산 및 가비지 컬렉션에 소요된 시간이 짧고, CPU 사용률이 낮은 경우에는 Wear-leveling Mode로 동작하여 연산 및 가비지 컬렉션에 소요된 시간이 길다. Smart Mode로 동작한 경우에는 유효 페이지 그룹화 과정이 수행되지 않으므로 Wear-leveling Mode보다 연산에 소요된 시간이 짧았다. 다른 기법과 비교하면, 표 6과 같이 연산시간은 Greedy 정책에 비해 약 2.8배 느리고 Cost-benefit 정책에 비해 약 2.2배 빨랐다.

표 6. 가비지 컬렉션 수행 시간 및 지움 횟수 분산 비교

Table 6. Comparison of Garbage Collection Processing Time and Erase Count Variance

Item	AGC (Proposed)	Greedy	Cost-benefit
Operation Time(s)	185.7	13.4	507.5
Erase Count Variance	48.46	375.34	5.18

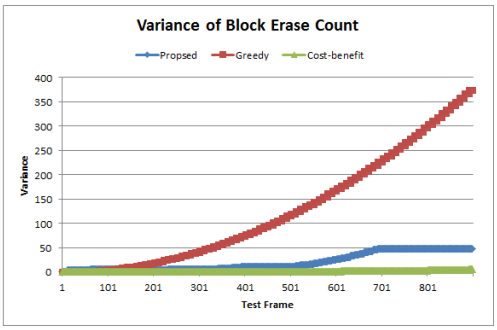


그림 13. 블록 지움 횟수 분산

Fig. 13 Variance of Block Erase Count

각 기법의 지움 균등 효과를 비교하기 위하여 그림 13과 같이 블록 횟수 분산을 비교하였다. Greedy 정책은 연산 처리 속도는 가장 빨랐으나 지움 균등을 고려하지 않아 연산이 수행될수록 블록 지움 횟수 분산도 증가하였다. Cost-benefit 정책은 연산처리 속도는 가장 느렸으나 연산 수행 횟수가 많아지더라도 블록 지움 편차가 적게 증가하였다. 제안한 정책은 CPU 사용률에 따라 지움 균등의 고려도가 달라지며, Greedy 정책과 비교하여 지움 횟수의 분산이 약 87% 감소되었다. 하지만 CPU 사용률이 높은 경우에는 지움 연산의 효율성만을 고려하므로 Cost-benefit 정책에 비해서는 지움 분산이 약 9배 크게 발생하였다. 제안한 정책이 Cost-benefit 정책에 비하여 지움 균등 효과는 낮으나, 시스템의 실시간성을 보장하는 장점이 있다.

실험 결과를 통해 제안한 기법이 CPU 사용률에 따른 적응적 가비지 컬렉션을 수행하여 시스템의 실시간성 및 지움 균등을 보장한 것을 확인할 수 있다. 하지만 CPU 사용률에 따라 동작 모드가 결정되므로, 동작 모드를 결정하는 문턱값과 지움 연산의 효율성과 균등 지움의 반영 정도를 결정하는 가중치의 설정이 중요한 요소가 된다. 즉, 문턱값 및 가중치가 시스템의 특징에 맞게 설정된 경우에는 지움 연산의 효율성과 균등 지움을 동시에 만족시킬 수 있지만, 적절하게 설정되지 못했을 경우에는 제안된 기법의 성능이 낮아질 수 있는 단점이 있다.

V. 결 론

본 논문에서는 실시간 임베디드 시스템에서 저장 장치로 사용되는 NAND 플래시 메모리를 위한 AGC 기법을 제안하였다.

제안한 AGC 기법은 3 가지 모드의 가비지 컬렉션을 가지며, 시스템의 실시간성을 보장하기 위하여 시스템의 CPU 사용률에 따라 적응적으로 동작 모드를 선택한다. CPU 사용률이 높은 경우에는 Fast Mode로 동작하여 지움 연산의 효율성만을 고려하며, CPU 사용률이 낮은 경우에는 Wear-leveling Mode로 동작하여 지움 연산의 효율성 및 지움 균등을 고려하고 이후에 수행될 지움 연산의 효율성을 위하여 유효 페이지의 그룹화를 통해 갱신 경향이 유사한 페이지들을 재배치하도록 하였다. Smart Mode는 지움 연산의 효율성 및 지움 균등을 고려하지만 유효 페이지의 그룹화 과정은 수행하지 않으므로 Wear-leveling Mode 보다 플래시 처리 속도를 빠르게 하였다.

성능 측정 결과, CPU 사용률에 따른 적응적 가비지 컬렉션 수행으로 CPU 사용률이 높은 경우에는 플래시 처리 시간의 지연을 최소화하고, CPU 사용률이 낮은 경우에는 지움 균등을 고려한 가비지 컬렉션을 수행하여 블록의 균등 사용성이 향상되는 것을 확인하였다. Greedy 정책에 비해 약 2.8배 연산 시간이 더 소요되었으나 블록들의 지움 횟수 편차는 약 87% 감소되었다. Cost-benefit 정책에 비해서는 블록 지움 횟수 편차가 약 9배 크게 발생하였으나 플래시 처리 시간은 2.2배 향상되었음을 확인하였다. 하지만 본 연구에서 제안한 기법은 CPU 사용률에 의해서만 동작 모드를 결정하며 시스템 및 연산의 특성은 고려하지 못하였다.

향후 본 연구는 CPU 사용률 외의 시스템 특성 및 발생하는 연산의 특성을 고려하고, 가비지 컬렉션 모드 설정을 위한 문턱값과 가중치를 적응적으로 변화시켜 지움 연산의 효율성 및 지움 균등을 더 향상시킬 수 있는 방향으로 나아가야 할 것이다.

References

[1] F. Douglis, R. Caceres, M.F. Kasho, P. Krishnan, K. Li, B. Marsh, J. Tauber, “Storage Alternative for Mobile Computers,” Proceedings of the 1st USENIX Symposium on Operating System Design and Implementation, pp. 25-37, 1994.

[2] M.L. Chiang, P.C.H. Lee, R.C. Chang, “Managing Flash Memory in Personal Communication Devices,” Proceedings of IEEE Symposium on Consumer Electronics, pp. 177-182, 1997.

[3] M.L. Chiang, R.C .Chang, “Cleaning Policies in Mobile Computers Using Flash Memory,” Journal of System and Software, Vol. 48, No. 3, pp. 213-231, 1999.

[4] A. Kawaguchi, S. Nishioka, H. Motoda “A Flash-Memory Based File System,” Proceedings of 1995 USENIX Technical Conference, pp. 155-164, 1995.

[5] O. Kwon, K. Koh, “Swap-aware Garbage Collection for NAND Flash Memory Based Embedded Systems,” Proceedings of IEEE 7th

International Conference on Computer and Information Technology, pp. 787-292, 2007.
[6] S.J. Baek, J.M. Choi, "Design and Implementation of MODA Allocation Scheme Based on Analysis of Block Cleaning Cost," Journal of KISE, Vol. 34, No. 11, pp. 599-609, 2007 (in Korean).
[7] S.H. Hwang, J.W. Kwak, "Garbage Collection Techniques for Reduction of Migration Overhead and Lifetime Prolongment of NAND Flash Memory," IEMEK J. Embed. Sys. Appl., Vol. 11, No. 2, pp. 125-134, 2016 (in Korean).
[8] S.H. Kim, J.W. 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, 2015 (in Korean).
[9] V. Prabhakaran, T. Wobber, "SSD Extension for DiskSim Simulation Environment." Microsoft Research 2009.
[10] K9GAG08U0M 2G x 8bit MLC NAND Flash Memory Data Sheet, Samsung Electronics, https://www.samsung.com, 2007.

Song-Hwa Park (박 송 화)

Park received a B.S. and M.S. degrees in Computer Engineering from Pusan National University in 2005 and 2007, respectively. She is currently a research engineer at LIG Nex1. Her research interests include flash file system, embedded system and track fusion.
Email: shpark25@lignex1.com

Won-Oh Lee (이 원 오)

Lee received a B.S. and M.S. degrees in Electrical Engineering and Computer Science from Kyungpook National University in 2008 and 2011, respectively. Currently, he is a research engineer at LIG Nex1. His research interests include computer vision and pattern recognition in embedded system.
Email: wonoh.lee@lignex1.com

Jung-Hoon Lee (이 정 훈)

Lee received a B.S., M.S. and Ph.D. degrees in Electronic Engineering from Kyungpook National University in 1997, 1999 and 2004, respectively. He is now a research engineer at LIG Nex1. His research interests include SONAR system, embedded system, and signal processing.
Email: junghoon.lee04@lignex1.com

Hee-Earn Kim (김 희 언)

Kim received a B.S. and M.S. degrees in Computer Engineering from AJOU University in 1997 and 2012, respectively. He is now a research engineer at LIG Nex1. His research interests include SONAR system, FPGA, DSP and embedded system.
Email: hekim70@lignex1.com