낸드 플래시 메모리를 위한 적응형 가비지 컬렉션 정책

(An Adaptive Garbage Collection Policy for NAND Flash Memory)

한 규 태 † 김 성 조 ††
(Gyu Tae Han) (Sung Jo Kim)

요 약 제자리 덮어쓰기가 불가능하고 블록의 지움 횟수가 제한되는 낸드 플래시 메모리를 저장매체로 사용하기 위해서 지움 횟수 평준화를 지원하는 다양한 가비지 컬렉션 정책들이 연구되고 있다. 기존 정책들은 지움 횟수 평준화를 지원하기 위해 가비지 컬렉션이 수행될 때마다 전체 블록에 대해 지움 대상 블록을 선정하기 위한 클리닉 지표를 구하는 연산을 수행하여야 하고 이 연산들은 시스템의 성능을 저하시킨다. 본 논문에서 제안하는 가비지 컬렉션 정책은 지움 횟수의 분산(variance)과 블록들의 최대 지움 횟수에 따라 변경되는 임계값을 이용하여 전체 블록에 대한 클리닉 지표를 구하는 연산을 수행하지 않으면서 지움 횟수 평준화를 제공한다. 가비지 컬렉션 시 분산이 임계값 보다 작을 때는 Greedy 정책을 이용하여 지움 비용을 최소화하고, 분산이 임계값 보다 클 때는 최대 지움 횟수를 가진 블록을 지움 대상에서 제외하여 지움 횟수를 평준화한다. 본 논문에서 제안하는 방법으로 가비지 컬렉션을 수행 하였을 때, 블록들의 지움 횟수가 지움 횟수 상한에 가까워질수록 블록들의 지움 횟수 표준 편차가 0에 근접하며, 기존의 지움 횟수 평준화를 지원하는 알고리즘과 비교하여 두 배 이상 빠른 가비지 컬렉션 속도를 보였다.

키워드 : 낸드 플래시 메모리, 파일 시스템, 가비지 컬렉션 정책

Abstract In order to utilize NAND flash memory as storage media which does not allow update-in-place and limits the number of block erase count, various garbage collection policies supporting wear-leveling have been investigated. Conventional garbage collection policies require cleaning-index calculation for the entire blocks to choose a block to be garbage-collected to support wear-leveling whenever a garbage collection is required, which results in performance degradation of system. This paper proposes a garbage collection policy which supports wear-leveling using a threshold value, which is in fact a variance of erase counts and by the maximum erase count of all blocks, without calculating the cleaning-index. During garbage collection, the erase cost is minimized by using the Greedy Policy if the variance is less than the threshold value. It achieves wear-leveling by excluding the block with the largest erase count from erase target blocks if the variance is larger than threshold value. The proposed scheme shows that a standard deviation approaches to zero as the erase count of blocks approaches to its upper limit and the measured speed of garbage collection is two times faster than the conventional schemes.

Key words : NAND Flash Memory, File System, Garbage Collection Policy

· 본 연구는 지식경제부 및 정보통신 연구진흥원의 대학 IT연구센터(홈네트워크 연구센터) 육성, 지원사업의 연구결과로 수행되었음
· 본 연구는 서울시 산학연 협력사업(CR070019) 지원으로 수행되었음

† 학생회원 : 중앙대학교 컴퓨터공학부
hellogt@konan.cse.cau.ac.kr
†† 종신회원 : 중앙대학교 컴퓨터공학부 교수
sjkim@cau.ac.kr

논문접수 : 2008년 10월 21일
심사완료 : 2009년 3월 21일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제5호(2009.5)

1. 서 론

최근 낸드 플래시 메모리는 전자 산업의 발전으로 가전, 통신기기, 휴대기기 등 다양한 장치의 저장매체로서 사용되고 있다. 낸드 플래시 메모리는 전기적으로 입출력을 수행하기 때문에 읽기 성능은 DRAM과 거의 같은 수준이며, 쓰기 속도도 하드디스크에 비해 수백 배 이상 빠르며, 가볍고 소비전력도 적게 든다[1,2].

하지만 낸드 플래시 메모리를 저장 매체로 사용하기 위해서는 2가지 제약 조건이 있다[3,4]. 첫째, 낸드 플래시 메모리는 각 비트의 토글링을 한쪽 방향으로만 허락하기 때문에 쓰기 연산을 하기 전에 반드시 해당 블록을 미리 지워야 한다. 이러한 이유로 낸드 플래시 메모리는 같은 주소에 덮어쓰기가 불가능하기 때문에 낸드 플래시 메모리에 데이터를 저장할 때 미리 그 영역을 0이나 1로 초기화 하여 데이터를 쓰기 위한 공간을 확보하고 있어야 한다. 둘째, 낸드 플래시 메모리를 구성하고 있는 각 블록들은 지움 횟수에 제한이 있다. 허용되는 지움 횟수를 초과한 플래시메모리 블록에 대해서는 쓰기 오류가 발생하기 때문에, 전체 플래시 메모리 공간이 균등하게 활용되지 못하는 경우에는 사용 가능한 저장 공간이 급격히 줄어드는 현상이 발생할 수 있다.

낸드 플래시 메모리를 관리하는 파일 시스템들은 쓰기 공간을 확보하기 위해, 가비지 컬렉션을 수행한다. 일반적으로 가비지 컬렉션 과정에서 블록을 지울 때 유효한 데이터는 다른 블록으로 복사한 뒤 지워야 하기 때문에 유효 데이터가 가장 적은 블록을 지우는 것이 쓰기 공간을 확보하는 가장 빠른 방법이다. 하지만 유효 데이터가 가장 적은 블록을 계속해서 삭제할 경우, 특정 블록에 지움 연산이 집중될 수 있다. 이로 인하여 해당 블록의 지움 횟수가 지움 횟수 상한(upper limit)을 초과하게 되면, 낸드 플래시 메모리의 사용 공간이 급격히 줄어든다. 즉, 지움 연산 비용 최소화에 치중할 경우, 지움 횟수 평준화가 급격히 악화된다. 반대로 지움 횟수 평준화를 위해 블록의 지움 횟수가 가장 적은 블록을 지운다면, 유효 데이터가 많이 존재하는 블록을 지울 수 있어 지움 연산 비용이 늘어 날 수 있다. 이와 같이 상충되는 두 가지 요구사항을 조화시키기 위하여 블록이 포함하고 있는 유효한 페이지 수와 블록의 지움 횟수를 근거로 블록을 지웠을 때 지움 비용이 적게 들면서 지움 횟수 평준화를 지원하는 지움 대상 블록을 결정해야 한다. 기존 연구[5-7]에서는 지움 대상 블록을 찾기 위하여 낸드 플래시 메모리에 존재하는 모든 블록들의 지움 횟수에 대한 클리닝 지표를 연산하여 지움 대상 블록을 선정하였고, 이 과정에서 가비지 컬렉션 수행에 많은 시간이 요구된다. 클리닝 지표란 각 블록들의 지움 우선 순위를 말하며, 가비지 컬렉션 시 블록마다 클리닝 지표를 구한 다음 정책에 따라 클리닝 지표가 가장 작거나 또는 가장 큰 블록이 지움 대상이 된다.

본 논문은 가비지 컬렉션 시 지움 비용과 지움 횟수 평준화를 동시에 고려하고, 이 과정에서 지움 대상 블록을 찾는 연산 횟수를 감소시킨 적응적 가비지 컬렉션 정책인 AGCP(Adaptive Garbage Collection Policy)를 제안한다. AGCP는 지움 횟수 평준화를 위해 유효 페이지수와 지움 횟수를 기준으로 데이터 블록들을 힙(Data Block Heap: DBH)으로 구성하여 블록 지움 요청 시 지움 대상 블록을 신속하게 검색한다. 또한, 비어 있는 블록들은 지금까지의 지움 횟수를 기준으로 힙(Clean Block Heap: CBH)을 구성하고, 이 힙을 통해 쓰기 연산 시 최소 지움 횟수를 가지는 블록을 제공함으로써 지움 횟수 평준화를 지원한다. 그리고 힙 만으로 해결할 수 없는 콜드(Cold) 데이터는 블록당 지움 횟수의 분산(Variance)을 활용한다. 낸드 플래시 메모리를 구성하고 있는 블록들 중 최대 지움 횟수, 지움 횟수 상한, 블록들의 개수에 따라 변하는 분산의 임계값을 설정하고 지움 횟수의 분산이 임계값을 넘지 않도록 한다. 분산이 임계값보다 작을 때에는 지움 횟수 평준화를 고려하지 않고 지움 비용이 최소인 블록이 선택되며, 지움 횟수가 특정 블록에 집중되어 분산이 임계값보다 큰 경우에는 최대 지움 횟수를 가지는 블록을 지움 대상에서 제외시켜 지움 횟수를 평준화한다.

본 논문의 구성은 다음과 같다. 2장은 기존 낸드 플래시 메모리 전용 파일 시스템에서 사용하는 가비지 컬렉션의 문제점에 대해 알아보고, 3장은 본 논문에서 제안하는 가비지 컬렉션 정책에 대하여 설명한다. 4장에서는 본 논문이 제안하는 AGCP의 성능을 지움 대상 블록 선정을 위한 연산 횟수와 지움 횟수 평준화 관점에서 평가하고, 마지막으로 5장은 결론과 함께 향후 연구과제에 대해 기술한다.

2. 관련 연구

Greedy 정책[5]은 무효 페이지가 많은 블록을 선택하여 삭제하는 방법이다. 이 정책을 이용하면 유효한 페이지를 다른 비어 있는 블록에 저장하는 시간이 줄어 들지만 특정 블록에 지움 연산이 집중해서 일어 날 수 있기 때문에 낸드 플래시 메모리의 사용 공간이 급격히 줄어드는 현상이 발생할 수 있다. 이 정책은 식 (1)을 이용하여 데이터가 저장되어 있는 블록i(i=1,…,n)를 대상으로 G_i값을 구한 뒤, 그 중 제일 큰 G_i값을 가지는 블록 i가 지움 대상이 된다. 여기서 n은 낸드 플래시 메모리에 존재하는 블록의 개수이고, u_i는 블록 내에 존재하는 유효데이터의 비율을 나타낸다.

$$G_i = \frac{1-u_i}{u_i}, \ i=1,\cdots,n \quad (1)$$

Cost-Benefit 정책[5]은 낸드 플래시 메모리의 사용 공간이 급격히 줄어드는 문제점을 개선하기 위하여 고안된 정책이다. 지움의 효율성뿐만 아니라 특정 블록에 집중되는 것을 피하기 위해 블록의 최근 지워진 시간을 고려하여, 지울 블록을 선택한다. 식 (2)는 이 정책에서 Cost-Benefit 계산 과정을 식으로 나타낸 것이다. 식 (2)에서 u_i는 Greedy 정책과 같이 블록의 이용률을 나타내고, age_i는 가장 최근에 수정된 후 경과된 시간을 나타낸다. 이 정책은 Cost-$Benefit_i$가 가장 큰 블록 i가 지움 대상이 된다.

$$\text{Cost-Benefit}_i = \frac{age_i * (1 - u_i)}{2u_i}, \ i=1,\cdots,n \quad (2)$$

그러나 age_i가 클수록 지움 횟수가 적다는 것을 의미하지는 않기 때문에, Cost-Benefit 정책은 정확한 지움 횟수 평준화를 지원하기 어렵다. 이러한 문제점을 해결하기 위해 RCP(Ranking Cleaning Policy)[6]와 PCP (Plain Cleaning Policy)[7]는 지움 횟수를 고려한다. RCP는 빠른 블록 지움과 지움 횟수 평준화를 위해 지움 비용 비율(c_i)과 지움 횟수 비율(e_i)를 각각 사용한다. 식 (3),(4)는 각각 c_i와 e_i를 구하는 식이며, Ce_i는 지움 비용, Cmax는 유효한 페이지로만 이루어진 블록을 지우는 비용(최대 지움 비용), Er_i는 블록의 남은 지움 횟수, Emax는 지움 횟수 상한을 의미한다. 이 정책은 식 (5)에 의해 계산된 $Rrcp_i$가 작은 순서대로 블록을 지운다. 이 식에서 a_i는 지움 비용에 중점을 둘 것인지 지움 횟수에 비중을 둘 것인지를 결정하는 가중치이고, 이를 결정하는 방법은 여러 가지 방법이 있을 수 있으며, 한 가지 예로 지움 횟수가 최대값에 가까워지면, $Rrcp_i$을 높여 지움 평준화를 하기 위해 $1/e_i$로 결정할 수 있다.

$$c_i = \frac{Ce_i}{Cmax}, \ i=1,\cdots,n \quad (3)$$

$$e_i = \frac{Er_i}{Emax}, \ i=1,\cdots,n \quad (4)$$

$$Rrcp_i = a_i \cdot c_i - \frac{1}{a_i} e_i, \ i=1,\cdots,n \quad (5)$$

PCP는 RCP가 요구하는 연산을 줄이면서 지움 횟수 평준화를 지원한다. 식 (6)은 지움 비용을 나타내며, V_i는 블록 내에 존재하는 유효한 데이터의 비율, F_i는 블록 내에 비어 있는 공간의 비율을 나타낸다. 식 (7)은 PCP의 계산 과정을 식으로 나타낸 것이다. EA_i(Erasable)는 플래시 메모리의 지움 횟수 한계치와 현재 지움 횟수의 차이 값으로 지움 횟수 평준화를 지원하기 위해 사용된다. 이 정책에서 한 블록의 클리닝 지표를 구하는 계산 복잡도는 O(6n + 1)로 RCP의 O(12n)과 비교하여 약 1/2로 줄었다.

$$C_i = (2 * V_i) + F_i, \ i=1,\cdots,n \quad (6)$$

$$Rpcp_i = \frac{EA_i}{C_i} \quad (7)$$

위의 정책들은 가비지 컬렉션이 요구될 때마다 지울 대상 블록을 찾기 위해 모든 블록에 대한 클리닝 지표(G, Cost-Benefit, Rrcp, Rpcp 등)를 구해야 하기 때문에 전체 시스템 성능이 크게 저하된다.

3. 적응적 가비지 컬렉션 정책

AGCP 정책은 DBH와 CBH 등 두 개의 힙을 이용하여 쓰기 대상이 되는 블록과 지움 대상이 되는 블록을 빠르게 선택하고, 분산과 분산의 임계값을 이용하여 특별히 오랫동안 변경되지 않은 콜드 데이터에 대해서도 지움 횟수 평준화를 지원하며, 쓰기 시간뿐만 아니라 Idle한 시간에 가비지 컬렉션을 수행함으로써 가비지 컬렉션으로 인한 쓰기시간 오버헤드를 줄인다. 여기서 말하는 콜드 데이터는 정확히 어느 시간만큼 변경되지 않는 데이터라 정의하지 않으며, 지움 횟수 평준화에 영향을 미치는 데이터라 정의한다. OS나 응용프로그램 같은 데이터가 그 예라 할 수 있다.

3.1 힙 구성

낸드 플래시 메모리에서 빠른 쓰기가 지원되기 위해서는 가능한 한 빨리 빈 블록을 찾을 수 있어야 한다. AGCP는 빈 블록들을 대상으로 지움 횟수에 따른 최소 힙(CBH)을 구성한다. 새로운 데이터를 낸드 플래시 메모리에 저장할 때 CBH의 루트 노드에 있는 블록을 할당함으로써 특별한 연산 없이 신속하게 지움 횟수 평준화를 지원할 수 있다. CBH를 사용할 경우, 콜드 데이터가 존재하지 않는다면, 다른 가비지 컬렉션 정책을 사용하지 않아도 지움 횟수 평준화가 지원될 수 있다. 하지만 CBH만 사용하게 된다면 콜드 데이터가 저장된 블록의 지움 횟수가 다른 블록들에 비해 급격히 작아지는 것을 방지할 수 없다. 이에 대한 해결책은 3.2절에서 제시한다.

블록을 지우기 위해서는 블록들이 가지고 있는 유효 데이터를 다른 블록에 복사하는 작업이 필요하다. 이 작업은 데이터를 읽고, 그 데이터를 다시 쓰는 작업이 필요 하므로 많은 시간이 소비된다. 따라서 가비지 컬렉션 시 가능한 한 유효 데이터를 최소로 가지는 블록을 지워야 한다. 기존의 정책들은 전체 블록들에 대한 지움 횟수와 지움 비용을 토대로 빠른 가비지 컬렉션과 지움 횟수 평준화를 위한 클리닉 지표를 구하고, 최적의 클리

닉 지표를 가진 블록을 찾는다. AGCP는 전체 블록을 검색하지 않고서도 지움 대상이 되는 블록을 찾기 위해서 유효 데이터수를 기준으로 최소 힙인 DBH를 구성한다. 이 DBH를 구성할 경우, 클리닉 지표들을 구하는 연산과정은 필요 없으며, 지움 대상이 되는 블록을 빠르게 찾을 수 있다.

그림 1은 AGCP의 힙 구조와 힙내에서의 블록 이동 과정을 나타낸다. 낸드 플래시 메모리의 블록들은 초기에 모두 CBH에 위치하게 되며, 파일 시스템에 쓰기 요청이 들어오게 되면, CBH의 루트에 존재하는 블록에 데이터를 저장하게 되고 CBH에서 삭제된다. 그 블록의 모든 페이지가 사용되면 그 블록은 DBH에 저장된다. 이후 DBH의 블록Bi에서 무효 페이지가 발생하면 Bi의 부모 노드에 있는 블록의 무효 페이지 수와 비교하여 적을 경우 힙에서의 위치가 부모 노드와 교환된다. 파일 시스템 실행 중, 가비지 컬렉터로부터 블록 지움 요청이 들어오면 DBH의 루트에 있는 블록에 대해 지움 연산을 수행하고, 지워진 블록은 다시 CBH로 이동하게 된다.

CBH와 DBH는 마운트(Mount)시 새로 구성한다. 하지만 이 힙 구조를 언마운트(Unmount)시 낸드 플래시 메모리에 저장하면, 힙은 마운트시 낸드 플래시 메모리에서 한번 읽어오고 언마운트시 낸드 플래시 메모리에 한번 저장된다. 그 사이의 시간에는 힙은 RAM상에 존재하게 되며, 플래시 메모리에 저장되지 않는다. 이러한 경우 파일 시스템을 최초로 구성할 때와 불안정하게 종료(언마운트가 수행되지 않고 종료)된 후 다시 마운트할 때 힙을 새로 구성한다. 이 힙은 파일 시스템이 마운트되어 있는 동안 RAM상에 유지되므로, 낸드 플래시 메모리에 대한 부가적으로 읽기, 쓰기가 발생하지 않는다.

3.2 분산과 분산의 임계값

CBH 힙을 통해 빈 블록을 신속하게 찾을 수는 있으나, 콜드 데이터가 존재할 경우 지움 횟수 평준화가 지원되지 못한다. 가비지 컬렉터의 블록 지움 요청시 DBH의 루트 노드를 지우기 때문에, 콜드 데이터만으로 구성된 블록은 지워지지 않는다. 이를 해결하기 위하여 본 논문에서는 낸드 내의 블록 당 지움 횟수의 분산(variance)을 계산하고, 분산의 임계값을 설정하여 콜드 데이터 발생에 의하여 지움 횟수가 한쪽 블록에 편중되는 현상이 발생 할 때, 최대 지움 횟수를 가지는 블록을 지움 대상 블록에서 제외함으로써 지움 횟수 평준화를 지원한다.

3.2.1 분산

블록들의 지움 횟수를 기준으로 한 표준편차 값은 블록들의 지움 횟수가 얼마나 평준화 되었는지를 보다 정확히 보여 주는 지표가 되나, 실제 연산 과정에서는 표준 편차 대신 분산을 사용한다. 표준 편차를 구하기 위해서는 분산의 제곱근을 구해야 하고, 이 제곱근 연산이 반복 수행될 경우 연산 오버헤드로 인하여 블록 지움이 지연될 수 있기 때문이다.

분산 Var은 식 (8)과 같이 계산되는데, NB는 낸드 플래시 메모리를 구성하는 전체 블록의 수, E_i는 i번째 블록의 지움 횟수, ave는 블록들의 평균 지움 횟수를

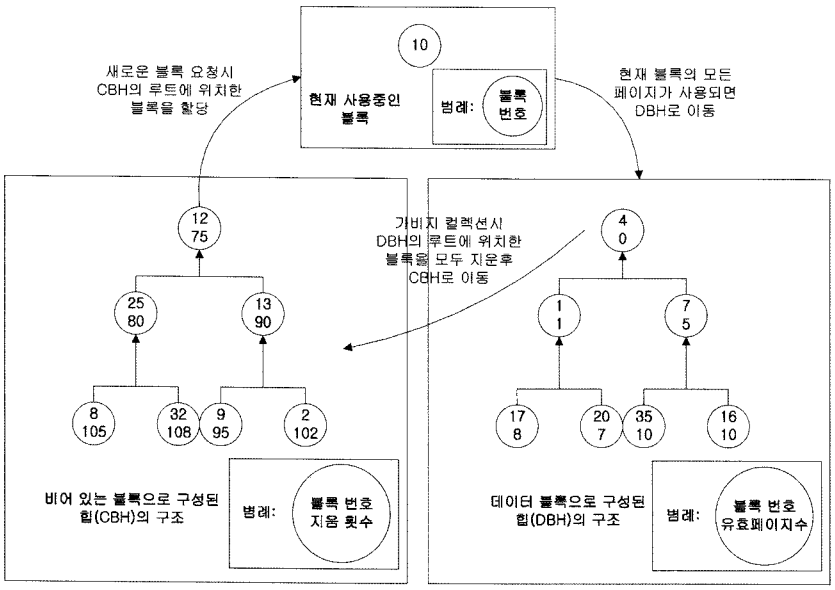


그림 1 AGCP의 힙 구조과 블록의 이동

나타낸다.

$$\mathrm{Var} = \frac{\sum_{i=1}^{NB}(E_i - ave)^2}{NB} = \frac{1}{NB}\sum_{i=1}^{NB}(E_i - ave)^2$$

$$= \frac{1}{NB}\sum_{i=1}^{NB}(E_i^2 - 2E_i ave + ave^2)$$

$$= \frac{1}{NB}\left\{\sum_{i=1}^{NB}E_i^2 - 2ave\sum_{i=1}^{NB}E_i + \sum_{i=1}^{NB}ave^2\right\}$$

$$= \frac{1}{NB}\left\{\sum_{i=1}^{NB}E_i^2 - 2ave^2 \cdot NB + NB \cdot ave^2\right\}$$

$$= \frac{1}{NB}\left\{\sum_{i=1}^{NB}E_i^2 - NB \cdot ave^2\right\} \quad (8)$$

블록 하나의 지움 횟수가 증가할 때 마다 분산을 다시 구해야 하는데 이 때 식 (8)을 이용하여 분산을 구하게 되면 모든 블록들의 지움 횟수의 합과 평균을 다시 구해야 한다. 이 연산은 블록의 개수가 NB개일 때, O(NB)의 연산 시간이 필요하다. 하지만 한번에 하나의 블록이 지워진다는 점을 이용하면 평균과 분산을 각각 식 (9), 식 (10)과 같은 방법으로 구할 수 있다. 식 (11)에서 Ek는 지워지는 블록의 지움 횟수를 나타낸다.

$$ave_{new} = ave_{old} + \frac{1}{NB} \quad (9)$$

$$Var_{new} = Var_{old} + \left(\frac{-E_k^2}{NB} + \frac{(E_k+1)^2}{NB}\right) + (ave_{old}^2 - ave_{new}^2)$$

$$= Var_{old} + \frac{(2E_k+1)}{NB} - \frac{2ave_{old} + \frac{1}{NB}}{NB} \quad (10)$$

3.2.2 분산의 임계값

낸드 플래시 메모리의 각 블록은 지움 횟수가 제한된 횟수를 넘어가게 되면 더 이상 사용이 불가능해진다. 따라서 낸드 플래시 메모리를 구성하는 블록들의 최대 지움 횟수가 증가할수록 수명이 줄어든다. 낸드 플래시 메모리의 수명을 최대한 보장하기 위해서는 블록들의 지움이 특정 블록에 집중되지 않고, 모든 블록들에 고르게 분포 되어야 한다. Var가 작아질수록 낸드 플래시 메모리를 구성하고 있는 블록들의 지움 횟수의 평준화가 잘 이루어지고 있다는 것을 의미하기 때문에, 블록들의 지움 횟수의 분산이 0에 가까워져야 낸드 플래시 메모리의 수명이 최대한 보장된다. 이를 위해 분산을 조절하는 임계값은 낸드 플래시 메모리의 수명이 다해 갈수록 0에 가깝게 설정 되어야 한다. 그리고 분산은 전체 블록의 개수와 블록당 최대로 지울 수 있는 횟수에 따라 달라지기 때문에 이 점들을 반영하여 분산의 임계값을 설정해야 한다. 블록당 최대 지울 수 있는 횟수를 LN, 현재 블록들 중 최대 지움 횟수를 MN이라 했을 때, 낸드 플래시 메모리의 남은 수명은 (LN-MN)/LN으로 연산하여 0과 1사이의 숫자로 표현할 수 있다. 분산은 블록들의 지움 횟수가 절반이 0, 나머지 반이 LN과 같을 때 최대 분산값이 $(LN/2)^2$이 된다. 위에서 계산한 지움 횟수의 비율과 최대 분산값을 가지고 수명이 최대 일 때, 최대의 분산이 되고 수명이 다해 갈수록 최소의 분산이 되도록 하기 위해 분산 임계값을 (LN-MN)*LN/4로 표현 할 수 있다. 이 식은 MN이 0일 때 최대, MN이 LN과 같을 때 0이 된다.

AGCP에서 분산을 사용하는 목적은 블록들의 지움 횟수를 고르게 분포 하도록 하기 위함이다. 분산은 블록들의 지움 횟수가 얼마나 고르게 분포되어 있는지 보여주는 지표가 되긴 하지만 그 대상이 많아질수록 최대값과 최소 값의 차이를 반영하지 못한다. 이는 분산을 구하는 과정에 전체블록의 수 NB로 나누어 주는 부분이 있기 때문이다. 이로 인해 블록의 수가 많아지게 되면 소수의 특정 블록의 지움 횟수가 다른 블록들과 차이가 많이 난다고 하더라도 분산이 커지지 않기 때문에 분산의 임계값은 NB의 크기 따라 다르게 적용해야 한다. AGCP에서는 위에서 구한 임계값이 낸드 플래시 메모리를 구성하는 블록의 개수에 상관없이 최소값과 최대값의 차이를 조절할 수 있도록 임계값 또한 NB로 나누었다. 이와 같은 기준으로 정한 임계값 Th_v는 식 (11)와 같이 계산된다. 식 (11)에서 NB, LN은 변하지 않는 상수 값이므로 MN이 변경될 때에만 임계값이 변경된다.

$$Th_v = \frac{(LN\text{-}MN)*LN}{4*NB} \quad (11)$$

초기에 블록들의 최대 지움 횟수가 0인 경우, 분산 임계값은 가장 크며, 이때는 지움 비용만 고려될 뿐, 지움 횟수 평준화는 고려되지 않는다. 하지만 블록들의 최대 지움 횟수가 지움 횟수 상한에 가까워지게 되면 분산 임계값은 0에 가깝게 설정되어, 지움 횟수 평준화가 엄격하게 고려된다.

3.3 가비지 컬렉션 수행 과정

낸드 플래시 메모리는 각 비트의 토글링을 한쪽 방향으로만 허락하기 때문에 쓰기 연산을 하기 전에 반드시 해당 블록을 미리 지워야 한다. 이러한 이유로 낸드 플래시 메모리는 제자리 덮어쓰기가 불가능하기 때문에 낸드 플래시 메모리에 데이터를 저장할 때 미리 그 영역을 0이나 1로 초기화하여 데이터를 쓰기 위한 공간을 확보하고 있어야 한다. 이와 같이 다시 사용 가능한 블록으로 변경시키는 작업을 가비지 컬렉션이라 하며, 가비지 컬렉션을 수행하는 가비지 컬렉터는 지움 대상 블록을 선정하고 삭제하는 역할을 한다. 지움 대상으로 선

택된 블록에 유효페이지가 하나 이상 존재한다면, 이 유효페이지들은 다른 비어있는 블록(Free Block)에 저장되고 지움 대상 블록이 지워진다.

가비지 컬렉션은 보통 쓰기 연산 시 빈 블록이 부족할 경우 실행되며, 전체 수행 과정은 그림 2와 같다. 본 논문에서는 쓰기 연산 시뿐만 아니라 낸드 플래시 메모리가 Idle할 때도 가비지 컬렉션을 수행함으로써 가비지 컬렉션 오버헤드를 줄일 수 있다.

쓰기 시간에는 현재 블록 지움 횟수들의 분산이 임계값을 넘었는지 여부를 체크한다. 넘었을 경우, 블록들의 지움 횟수를 평준화하기 위하여 최대 지움 횟수를 가지는 블록을 지움 대상 블록에서 제외한다. 만약 최대 지움 횟수를 가지는 블록이 DBH의 루트에 존재할 경우 그 블록은 지움 대상 블록으로 선정되지 않고, DBH를 차례대로 검색하여 최대 지움 횟수를 가지지 않는 블록을 지움 대상 블록으로 선정한다.

반대로 분산이 임계값을 넘지 않았을 경우, 지움 횟수의 평준화를 고려하지 않아도 되므로 무효페이지가 가장 많은 블록을 삭제하여 빠르게 가비지 컬렉션을 수행한다.

낸드 플래시 메모리가 Idle할 경우 가비지 컬렉션을 수행 하게 되면 쓰기 시간에 가비지 컬렉션 발생을 줄일 수 있다. 낸드 플래시 메모리의 Idle 상태는 낸드 플래시 메모리가 2초 동안 Idle할 경우, 다시 4초간 Idle할 확률이 95%이상인 점[8]을 참고하여 낸드 플래시 메모리가 2초 동안 Idle할 경우, Idle한 상태에 있다고 판단한다. 파일 시스템은 읽기, 쓰기, 삭제 연산을 위해 낸드 플래시 메모리에 접근하는데, 이 경우들을 제외하면 낸드 플래시메모리에 대한 접근은 없으므로 Idle하다고 판단한다. Idle하다고 판단되면, Idle 플래그가 세트되고, 2초 후에도 Idle 플래그가 변경되지 않았다면, 다시 4초간 Idle하다고 판단하여 가비지 컬렉션을 시작한다. Idle 시간에도 블록 지움 횟수들의 분산이 임계값을 넘었는지 여부를 파악한다. 먼저 넘었을 경우, 쓰기 시간에서

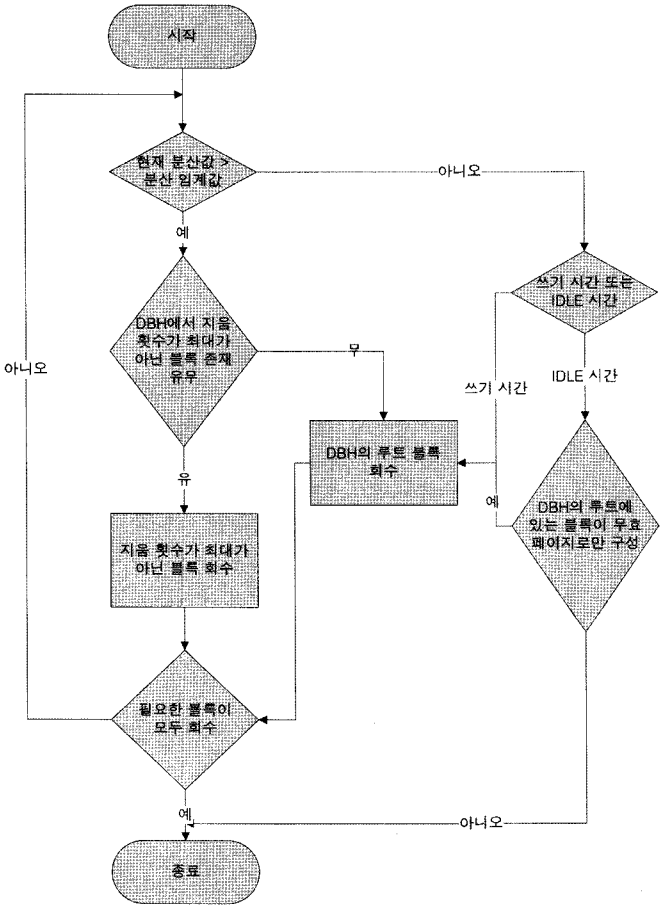


그림 2 가비지 컬렉션 수행 과정

와 같이 최대 지움 횟수를 가지는 블록을 지움 대상 블록에서 제외하고, 블록 지움 연산을 수행하여 블록들의 지움 횟수를 평준화한다. 그렇지 않은 경우는 무효 페이지들로만 구성된 블록들을 지우며, 무효 페이들로만 구성된 블록이 존재하지 않을 경우는 가비지 컬렉션을 종료하게 된다.

4. 성능평가

본 논문에서 제안한 가비지 컬렉션 정책의 성능 평가를 위해 리눅스 상에서 제공되는 NANDSIM(NAND Flash Memory Simulator)[9]를 사용하여 STMicro-electronics의 낸드 플래시 메모리(64Mib)를 시뮬레이션 하였으며, 테스트 환경은 표 1과 같다.

표 1 테스트 환경

항목	명세
CPU	Intel® Core™2 Duo CPU
RAM	2GB
LINUX	Fedora Core8(2.6.23.9)

4.1 지움 대상 블록 선정을 위한 연산 횟수

실제 소스 코드 상에서 가비지 컬렉션 때 지움 대상이 되는 블록을 선택하기 위한 연산 수행 횟수는 표 2와 같다. Greedy 알고리즘의 경우, 전체 블록들에 대한 ((1-블록의 이용률)/블록의 이용률)값을 구하고, 이 값이 최소인 블록을 찾아야 하므로 3NB의 연산 횟수가 필요하다. 이와 같은 방법으로 Cost-Benefit, PCP, RCP의 연산 횟수를 구했다.

AGCP정책은 현재까지 제안된 다른 정책들과 비교할 때 가비지 컬렉션시 연산 횟수를 O(NB)에서 O(log(NB))

표 2 가비지 컬렉션 정책들의 지움 대상 블록을 찾기 위한 연산 횟수(1블록 기준)

정책		가비지 컬렉션 연산 횟수
Greedy		$3NB$
Cost-benefit		$4NB$
PCP		$6NB+1$
RCP		$12NB$
AGCP	Best	11
	Worst	$(4OCH+(PPB/2))*\mathrm{Log}(NB)+10$

NB : 낸드 플래시 메모리를 구성하고 있는 전체 블록의 수
PPB : 블록을 구성하고 있는 페이지의 수
OCH : 힙에서의 연산 횟수

로 낮추었다. 특히 이 정책은 낸드 플래시 메모리 읽기, 쓰기 과정이 많이 발생되고, 마모도 평준화 연산이 집중되는 가비지 컬렉션 시간의 연산 횟수를 파일 삭제 시간으로 분배함으로써 파일 쓰기 시간에 집중 되는 연산을 줄였다.

가비지 컬렉션 테스트에서 사용된 낸드 플래시 메모리의 전체 블록 수는 512개이고, 블록을 구성하고 있는 페이지 수는 64개로 이를 대입해서 비교해보면 NAFS에서 사용한 가비지 컬렉션 정책의 블록당 연산 횟수는 최악의 경우에도 총 730회로서 AGCP정책의 가비지 컬렉션을 제외한 가장 적은 연산 횟수를 가지는 Greedy 정책(1536회)과 비교해서도 연산 횟수가 두 배 이상 감소 한 것을 확인 할 수 있다. 또한 현재까지 개발된 낸드 플래시 메모리의 전체 블록 수는 218으로(MLC-large block, 128Gbit)[10], 이 낸드 플래시 메모리에 AGCP를 적용할 경우 그 속도는 다른 가비지 컬렉션 정책을 적용 하였을 때보다 연산 횟수를 수 백배 이상 줄일 수 있다.

4.2 지움 횟수 평준화 평가

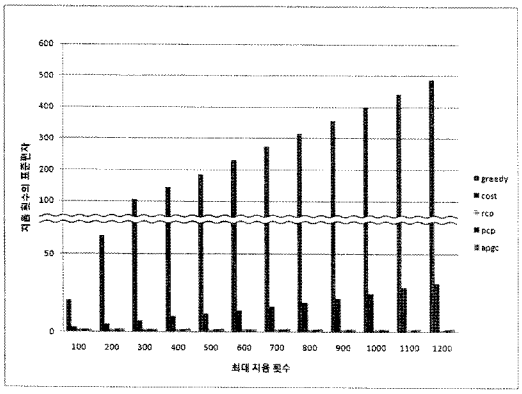
초기 데이터 양이 적을 때에는 임의로 파일을 갱신할 경우 콜드 데이터가 존재할 가능성이 적으나, 데이터 양이 증가할수록 콜드 데이터가 증가할 가능성이 커진다. 실제 초기 데이터 양을 10%에서 90%까지 파일을 생성하고, 낸드 플래시 메모리의 전체 용량만큼 파일 갱신을 수행하면 표 3과 같이 콜드 데이터가 생성 되는 것을 확인할 수 있었다.

즉, 실제 테스트 결과 초기 데이터 양이 10%에서 20%까지는 콜드 데이터가 나타나지 않았으며, 30%부터 콜드 데이터가 생성되기 시작하여 초기 데이터 양이 늘어날수록 콜드 데이터 양이 증가하였다. 이 결과를 바탕으로 지움 횟수 평준화에 대한 성능 평가는 데이터 양을 초기 10%에서 90%까지 증가시키면서, 임의의 파일을 갱신하는 형태로 수행하였다. 가비지 컬렉션 시 지움 횟수 평준화를 평가하는데 테스트 시간을 고려하여 실제 낸드 플래시 메모리의 최대 허용되는 지움 횟수는 2000번으로 가정하였으며, 파일 크기는 2KByte(1 페이지 크기)부터 128KByte(1 블록 크기)까지 임의로 생성하였다.

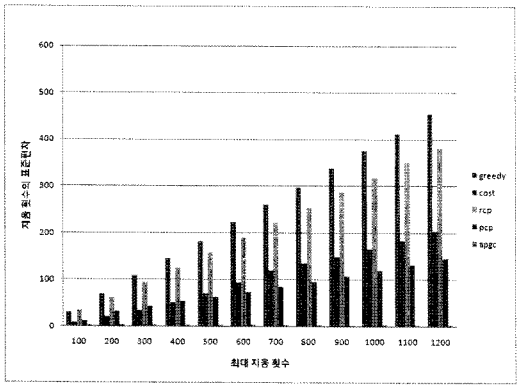
그림 3은 블록들의 지움 횟수에 대한 표준 편차의 변화를 보여 주는 그래프로 (a),(b),(c)는 각각 초기 데이터 양이 10%, 50%, 90%일 경우, 즉 콜드 데이터가 존재하지 않을 때, 콜드 데이터가 적을 때, 콜드 데이터가 많을 때의 블록 지움 횟수에 대한 표준 편차 변화를 나

표 3 초기 데이터 양에 따른 콜드 데이터 생성 비율

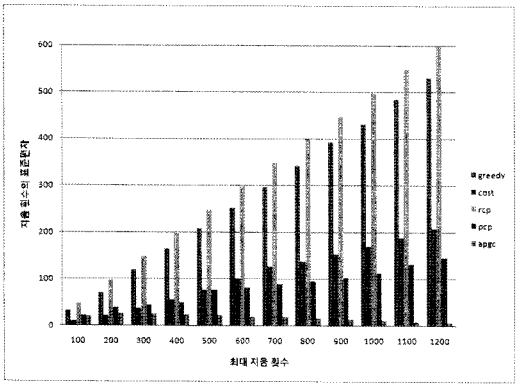
초기 데이터 양	10%	20%	30%	40%	50%	60%	70%	80%	90%
콜드 데이터 비율	0%	0%	3%	8%	14%	18%	23%	29%	32%



(a) 초기 데이터 양이 10%일 경우



(b) 초기 데이터 양이 50%일 경우



(c) 초기 데이터 양이 90%일 경우

그림 3 블록들의 지움 횟수에 대한 표준 편차의 변화

타낸다. Greedy 정책의 경우 모든 그래프에서 지움 횟수의 표준편차가 시간이 지날수록 큰 폭으로 증가하는 것을 볼 수 있다. Cost-Benefit 정책은 초기 데이터 량에 따라 큰 폭으로 지움 횟수의 표준 편차가 증가하지는 않으나, 최대 지움 횟수가 증가 할 때마다 지움 횟수의 표준편차가 계속해서 증가되는 것을 알 수 있다. RCP와 PCP는 초기 데이터 양이 10%일 경우 지움 횟수의 표준 편차가 0에 가깝게 유지 되지만 콜드 데이터가 나타나는 시점부터는 지움 횟수의 평준화가 이루어지지 않는 것으로 나타났다. RCP는 초기 데이터 양이 증가함에 따라 지움 횟수의 표준 편차가 급속히 커지데 반해, PCP는 콜드 데이터의 존재 유무에 따라 지움 횟수의 표준 편차가 증가하였다. 전체적으로 APGC를 제외한 다른 가비지 컬렉션 정책들은 콜드 데이터에 대한 처리를 해주지 못하고 있다. 이에 반해, APGC는 초기 데이터 양이 증가함에 따라 지움 횟수의 표준 편차가 증가하지만, 최대 지움 횟수가 지움 횟수 상한에 근접할수록 0에 가까워지는 것을 알 수 있다. AGCP의 가비지 컬렉션 정책은 마모도를 평준화 측면에서 다른 정책들에 비해 우수한 성능을 보였다.

5. 결론 및 향후 연구 과제

기존의 지움 횟수 평준화 기법들은 블록을 지우기 전 모든 블록들의 클리닉 지표를 구해 지워야 할 블록을 찾기 때문에 가비지 컬렉션 시간이 낸드 플래시 메모리를 구성하는 블록의 수에 비례하여 증가한다. 또한, 콜드 데이터가 존재할 때, 지움 횟수 평준화를 지원하지 못한다. 본 논문은 CBH, DBH 등 두 개의 힙을 사용하여 지움 대상이 되는 블록을 찾는 연산 오버헤드를 줄였으며, 낸드 플래시 메모리를 구성하고 있는 블록들의 지움 횟수에 대한 분산이 임계값을 넘지 않게 하여 낸드 플래시 메모리 내에 콜드 데이터의 존재 여부와 상관없이 효율적인 지움 횟수 평준화를 제공함을 보였다. 또한 낸드 플래시 메모리의 Idle 시간을 활용하여 쓰기 시간에 발생할 수 있는 가비지 컬렉션을 미리 수행함으로써 가비지 컬렉션으로 인한 오버헤드를 줄였다.

본 논문에서 제안한 AGCP정책은 지움 블록을 찾는 연산 횟수를 기존 알고리즘과 비교하여 50%이상 줄였으며, 블록당 지움 횟수의 표준 편차가 타 기법보다 우수함을 보였다. 또한 지움 횟수가 지움 횟수 상한에 근접할 때에는 블록당 지움 횟수의 표준편차가 0에 가까워짐을 보였다.

본 논문에서 제시한 가비지 컬렉션 정책에서 사용한 힙은 블록의 번호를 저장해야 하기 때문에, 블록의 수만큼 RAM 사용량이 늘어난다. 따라서 이를 보안하기 위해 전체 블록이 아닌 현재 사용중인 블록들만 대상으로 힙을 구성하는 연구가 필요하다.

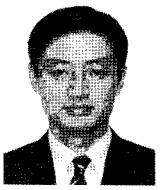
참 고 문 헌

[1] 민용기, 박승규, "이동컴퓨터를 위한 플래시메모리 클리닝 정책", 한국통신학회논문지, 제24권 5호, pp. 657-

666, 1999.
[2] 한준영, 김성조, "낸드 플래시 메모리 기반 멀티미디어 파일 시스템에서의 효율적인 페이지 할당 및 회수 기법", 2007 한국컴퓨터종합학술대회 논문집(B), Vol.34, No.1, pp. 680-682, JUNE 2007.
[3] 박상오, 김성조, "NAND 플래시 메모리 기반 파일 시스템을 위한 빠른 마운트 및 안정성 기법", 정보과학회논문지 : 시스템 및 이론, Vol.34, No.12, pp. 683-695, DECEMBER 2007.
[4] M. L. Chang, P. C. H. Lee, R. C. Chang, "Managing Flash Memory in Personal Communicati on Devices," Proceedings of IEEE Symp. On Consumer Electronics, pp. 177-182, 1997.
[5] Atsuo Kawaguchi, Shingo Nishioka, Hiroshi Motoda "A Flash-Memory based File System," Proceedings of USENIX Technical Conference, pp. 155-164, 1995.
[6] 김정기, 박승민, 김채규, "임베디드 플래시 파일 시스템을 위한 순위별 지움 정책", 정보처리학회논문지, 제9-A권, 제4호, pp. 399-404, 2002.
[7] 이태훈, 이상기, 정기동, "임베디드 플래시 파일 시스템을 위한 플레인 지움 정책", 한국정보과학회 2005 추계학술 발표 논문집, pp. 778-780, 2005.
[8] T. Blackwell, J. Harris and M. Seltzer, "Heuristic Cleaning Algorithms in Log-Structured File Systems," Proceedings of the 1995 Winter Usenix, January 1995.
[9] http://www.linux-mtd.infradead.org/faq/nand.html
[10] Samsung Electronics Co,. "NAND Flash," http://www.samsung.com/global/business/semi conductor/products/flash/Products_NANDFlash.html

한 규 태

2007년 중앙대학교 컴퓨터공학과(공학사). 2007년~현재 중앙대학교 컴퓨터공학과(석사과정). 관심분야는 NAND 플래시 메모리 파일 시스템, 임베디드 시스템, Linux Kernel



김 성 조

1975년 서울대학교 응용수학과(공학사) 1977년 한국과학기술원 전산과(이학석사). 1977년~1980년 ADD(연구원). 1980년~현재 중앙대학교 컴퓨터공학부 교수 1987년 Univ. of Texas at Austin(공학박사). 1887년~1988년 Univ. of Texas at Austin(Research Fellow). 1996년~1997년 Univ. California-Irvine(Visiting Professor). 관심분야는 이동컴퓨팅, 임베디드 소프트웨어, 유비쿼터스컴퓨팅