Cold 블록 영역과 hot 블록 영역의 주기적 교환을 통한 wear-leveling 향상 기법

장시웅*

*동의대학교

A wear-leveling improving method by periodic exchanging of cold block areas and hot block areas

Si-woong Jang*

*Dongeui University

E-mail : swjang@deu.ac.kr

요 약

플래시 메모리에서 읽기 작업은 속도도 빠르고 제약이 없으나 데이터 변경 시에는 덮어쓰기(overwrite)가 되지 않아 해당 데이터를 새로운 영역에 쓰고 이전에 존재하던 데이터는 무효시켜야한다. 무효화시킨 데이터는 가비지컬렉션을 통해 지움 연산을 수행해야 한다. 지역 접근성을 가지는 데이터에 대해 가비지컬렉션을 통해 클리어 시킬 대상 목록을 선정할 때 cost-benefit 방법을 사용하면 성능은 좋으나 wear-leveling이 나빠지는 문제점이 있다.

본 연구에서는 wear-leveling을 개선하기 위해 플래시 메모리를 hot 데이터 그룹들과 cold 데이터 그룹들의 다수의 그룹으로 분할한 후 데이터를 배치하고 주기적으로 hot 데이터 영역과 cold 데이터 영역을 교체함으로써 wear-leveling과 성능을 개선하였다.

ABSTRACT

While read operation on flash memory is fast and doesn't have any constraints, flash memory can not be overwritten on updating data, new data are updated in new area. If data are frequently updated, garbage collection, which is achieved by erasing blocks, should be performed to reclaim new area. Hence, because the number of erase operations is limited due to characteristics of flash memory, every block should be evenly written and erased. However, if data with access locality are processed by cost benefit algorithm with separation of hot block and cold block, though the performance of processing is high, wear-leveling is not even.

In this paper, we propose CB-MG (Cost Benefit between Multi Group) algorithm in which hot data are allocated in one group and cold data in another group, and in which role of hot group and cold group is exchanged every period. Experimental results show that performance and wear-leveling of CB-MG provide better results than those of CB-S.

키워드

Flash File System, Wear-leveling, Garbage Collector, Database, Flash Memory

Ⅰ. 서 론

플래시 메모리는 임베디드 시스템의 구성에 중요한 역할을 하고 있으며, 플래시 메모리가 대형화됨에 따라 플래시 파일시스템에 관한 많은 연구가 진행되고 있다[1-5]. 플래시 메모리는 디스크와는 달리 덮어쓰기가 불가능하여 데이터가 변경되면 변경된 데이터에 해당하는 이전 영역은 쓸모없게 되고, 새로운 데이터는 새로운 영역에 갱신된다. 데이터가 빈번히 변경되면 새로운 영역을 확보하기 위해 가비지 컬렉션을 통하여 새로운 영역을 확보하게 된다. 데이터를 읽고 쓰는 것은 페이지 단위로 수행하나 데이터를 지우는 연산은 블록단위로 수행된다[6]. 데이터에 대한 가비지

컬렉션을 수행할 때 유효한 페이지는 새로운 블록으로 이동한 후 해당 데이터 블록을 지우는데, 이때 지움(erase) 연산을 수행할 수 있는 회수는 플래시 메모리의 특성에 의해 한정된 회수(약 100, 000회)로 제한을 받게 되므로 플래시 메모리의 모든 블록은 고르게 쓰여 지고 지워져서 평등한 wear-leveling이 제공되어야 한다.

데이터의 갱신, 추가, 삭제가 반복되는 동안 자유공간이 부족해지면 새로운 데이터를 추가하기 위해 자유공간을 확보하여야 하는데 이때 자유공간을 확보하기 위해 지움 연산을 수행할 블록을 선정해야한다. 지움 연산을 수행하기 위해 블록을 선정하는 전통적인 방법으로 Greedy 방법과 Cost benefit 방법이 존재하는데, Greedy 방법은 uniform한 작업 부하에 좋은 성능을 보이는 반면 접근 지역성을 가지는 작업부하에 대해서는 좋지 않은 성능을 보인다. 따라서 접근 지역성을 가지는 작업부하를 위하여 Cost benefit 방법이 사용되며, 가비지 컬렉션 수행시 cold 데이터와 hot 데이터를 분리하여 저장하면 Greedy 정책에 비해 월등한 성능을 보이는 것으로 나타났다[6].

그러나 Cost benefit 방법에서 cold 데이터와 hot 데이터를 분리하여 저장하면 wear-leveling의 편차가 커져 플래시 메모리의 수명에 큰 영향을 미치게 된다. 플래시 파일 시스템의 wear-leveling을 개선하기 위한 연구로는 [7]이 있으며, 특정한 cold 데이터가 한 블록에 오래 머무는 것을 방지하여 블록 간의 지움 연산의 회수의 차이를 최소화하는 방법을 제시하였으나 접근 지역성(access locality)을 가지는 작업부하에는 효과적이지 않다. [8]에서는 wear-leveling을 개선하기 위해 멀티 뱅크에서 하나의 뱅크는 cold 데이터를 다른 뱅크는 hot 데이터를 할당하고, 시간이 흐름에 따라 일정주기로 cold 뱅크와 hot 뱅크를 교환하는 방법을 제안하였다. [8]의 방법은 hot 데이터의 블록 수와 cold 데이터의 블록수가 유사할 경우에는 효과적으로 이용될 수 있지만, 그렇지 않은 경우에는 플래시 메모리 공간의 효율성이 좋지 않은 단점이 있다.

본 연구에서는 플래시 메모리 공간을 다수의 그룹으로 나누고 hot 데이터와 cold 데이터의 비율에 따라 일정 수의 그룹은 hot 데이터 블록들을 저장하고, 나머지 그룹은 cold 데이터 블록들을 저장한 후, 주기적으로 hot 데이터 영역과 cold 데이터 영역을 교체함으로써 wear-leveling을 개선하였다.

II. 관련 연구

플래시 메모리는 여러 개의 블록으로 구성되며 하나의 블록은 일정수의 페이지로 나뉘어 관리된다. 한 페이지의 크기는 하드 디스크의 최소 단위인 섹터의 배수로 구성된다. 플래시 메모리는 읽기와 쓰기 연산을 페이지 단위로 수행한다. 한편 쓰기 연산은 지워진 영역에만 가능하므로 데이터를 쓰기 위해서는 해당 영역을 미리 지워야하는데 지움 연산은 블록 단위로 수행된다. 플래시 메모리는 덮어쓰기(overwrite)가 되지 않으므로 데이터를 갱신하기 위해서는 갱신할 데이터에 해당하는 페이지를 무효화시킨 후 갱신된 데이터를 새로운 블록의 페이지에 써야 한다. 많은 데이터에 대해 갱신 연산이 발생하면 무효화된 데이터가 많아져서 유효한 블록이 소진되게 된다. 따라서 자유블록의 수가 임계치 이하로 떨어지면 가비지 컬렉션 정책에 의해 특정 블록을 선정하여 유효 페이지를 새로운 페이지를 할당 받아 이동한 후 지움 연산을 수행한다.

가비지 컬렉션 정책에는 무효 페이지가 가장 많은 블록을 선정하는 Greedy 알고리즘과 블록의 age와 이용율을 고려한 Cost benefit 알고리즘이 있다. Cost benefit 알고리즘은 블록 이용율(u)과 age를 고려하여 $\frac{age(1-u)}{2u}$를 최대로 하는 블록을 선정[8]하여 지움 연산을 수행한다. 이 방법을 이용하면 cold 블록들은 hot 블록보다 낮은 이용율을 가짐에도 불구하고 age가 클 경우에는 지움 연산을 위한 블록으로 선정된다.

III. 다중 그룹을 이용한 Wear-leveling 향상 기법

종래의 Cost benefit 방법은 균등한 wear-leveling을 제공하지만 접근 지역성을 가지는 작업부하에서는 Greedy 방법처럼 좋지 못한 성능을 제공한다.

3.1 CB-S (Cost Benefit with Separation)

CB-S는 hot 데이터와 cold 데이터를 분리하여 관리하는 Cost benefit 방법으로 하나의 논리적 블록에는 hot 데이터의 페이지와 cold 데이터의 페이지가 공존하는 형태로 존재한다. 그러나 논리적 페이지 데이터들을 물리적 뱅크에 할당할 때는 hot 페이지 데이터들은 hot 페이지만 포함하는 물리적 블록에 저장하고, cold 페이지 데이터들은 cold 페이지만 포함하는 물리적 블록에 저장한다.

3.2 CB-MB(Cost Benefit between Multi Bank)

CB-MB는 hot 데이터와 cold 데이터를 분리하여 비용효율적인 방법으로 관리하되 hot 데이터의 페이지와 cold 데이터의 페이지를 서로 다른 물리적 뱅크에 나누어 저장하고 관리하는 방법으로 hot 데이터의 페이지와 cold 데이터의 페이지가 공존하는 형태로 존재한다. 그러나 논리적 페이지 데이터들을 물리적 뱅크에 할당할 때는 hot 페이지 데이터들은 hot 블록만 포함하는 물리적

뱅크에 저장하고, cold 페이지 데이터들은 cold 블록만 포함하는 물리적 뱅크에 저장한 후, 일정한 주기로 hot 뱅크와 cold 뱅크의 역할을 바꾸도록 하여 뱅크 간에 균등한 부하를 받도록 하였다.

3.3 CB-MG(Cost Benefit between Multi Group)

CB-MG에서는 플래시 메모리를 다수의 그룹으로 나누어 일정 수의 그룹은 hot 데이터를 저장하고, 나머지 그룹은 cold 데이터를 저장한다. CB-MG는 hot 데이터와 cold 데이터를 분리하여 비용효율적인 방법으로 관리하되 hot 데이터의 페이지와 cold 데이터의 페이지를 서로 다른 그룹에 배치하여 서로 혼합되지 않도록 한다. 이런 경우, hot 블록만 포함하는 그룹과 cold 블록만 포함하는 그룹 모두 그룹내에서는 균등한 wear-leveling을 제공할 수 있지만, hot 블록만 포함하는 그룹이 빨리 소진되는 현상이 나타날 것이다. 따라서 본 논문에서는 일정한 주기로 hot 그룹과 cold 그룹의 역할을 바꾸도록 하여 전체 플래시 메모리의 wear-leveling을 향상시켰다.

Ⅳ. 실험

본 논문에서 제안한 CB-MG의 성능 및 wear-leveling을 CB-S 기법과 비교하기 위해 시뮬레이션을 수행하였다.

4.1 실험 방법

실험은 1개 뱅크의 크기에 해당하는 논리적 페이지에 대한 페이지 번호를 발생시키되 발생된 홀수 페이지 번호는 번호가 하나 작은 짝수 페이지로 바꾸어 논리적인 홀수 페이지는 없도록 하여 1개의 물리적 뱅크에 할당되는 물리적인 페이지의 수는 물리적 뱅크의 절반만 차지하도록 하였다. 이때 새로운 물리적 블록이 일정개수(1/100) 이하가 되면 가비지 컬렉션이 수행되도록 하여 dirty 페이지가 많은 블록들이 클리어 되도록 하였다. 이와 같은 방법으로 작업부하를 1000회 연속 발생시켜 실험을 수행하였다. CB-MG의 성능을 평가하기 위해 플래시 메모리를 10개의 그룹으로 나눈 후, 발생된 페이지 번호에 따라 그룹으로 나뉘어 관리되도록 하였다.

실험 수행시 사용된 파라메터는 표1과 같다. 표1의 실험 파라메터는 블록의 크기가 크고 대용량인 플래시 메모리의 사양으로 사용되는 파라메터이다.

표 1. 실험 파라메터

특성	값	특성	값
읽기(512B)	0.025ms	페이지 크기	2 KBytes
쓰기(512B)	0.200ms	블록 크기	64 KBytes
지우기(16KB)	1.5ms	뱅크 크기	8GBytes

본 논문에서 제안하는 CB-MG의 성능을 상대적으로 평가하기 위해 기존에 제안된 CB-S 방법의 성능을 함께 평가하여 비교한다. 성능비교시 CB-S 및 CB-MG의 CPU 수행시간을 고려하였다. 본 실험은 플래시 메모리를 사용하는 환경을 PDA와 같은 임베디드 시스템을 고려하므로 CPU의 수행시간을 PC 수행시간의 5배로 환산하여 계산하였다.

작업부하는 [6]에 있는 것과 유사한 방법으로 지역성을 가지는 작업부하를 2개의 등급으로 나누어 실험한다. 참조의 지역성을 표시하기 위해 x/y로 표현하는데, 이것은 y%의 데이터에 x%의 접근이 이루어지는 것을 의미한다. 따라서, 90/10의 작업부하는 10%의 데이터에 90%의 참조가 이루어지고, 90%의 데이터에는 10%의 참조가 발생하는 것을 의미한다.

4.2 성능 평가 결과

4.1절의 실험 방법으로 CB-S 및 CB-MG에 대해 실험한 전체 수행시간과 wear-leveling을 비교하여 분석하면 각각 그림 1 및 표 2와 같다.

그림 1에서 보는 것처럼 CPU 수행시간을 포함한 CB-MG의 성능은 CB-S에 비해 좋은 성능을 나타내는 것으로 보인다. 이는 CB-S는 가비지 컬렉션시 전체 플래시 메모리에서 대상 블록을 찾는 반면 CB-MG는 그룹내부에서 대상 블록을 찾기 때문인 것으로 분석된다. 접근 지역성이 높은 경우(90/10)인 경우에는 hot 데이터를 포함한 그룹과 cold 데이터를 포함한 그룹의 교환은 교환주기가 10회에서 100회까지 교환 주기가 증가할수록 전체 시스템의 성능이 좋아지는 것으로 분석되었다. 접근 지역성이 비교적 낮은(80/20) 경우에는 교환주기가 평균쓰기 50회 이상이 되면 전체 시스템의 성능은 좋지 않은 것으로 분석되었다.

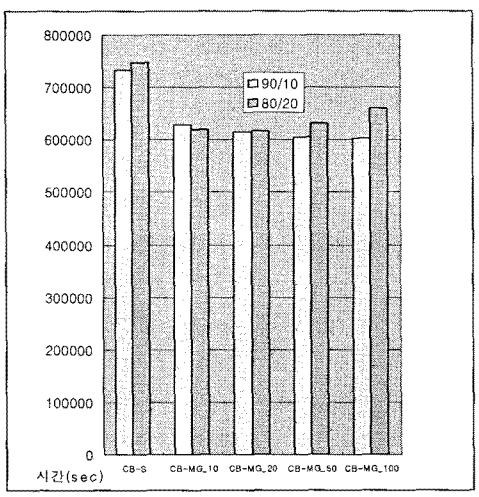

그림 1. 그룹교환 주기에 따른 수행시간의 비교

이는 접근지역성이 낮은(80/20) 경우에는 hot 그룹의 수가 접근 지역성이 높은 경우(90/10) 경우에 비해 2배이므로 교환 주기를 2배 정도 빠르게 해 주는 것이 필요한 것으로 분석된다.

접근 지역성이 높은 경우(90/10)에는 교환주기가 증가할수록 표준편차가 감소하여 균등한 wear-leveling을 제공한다. 그러나 CB-MG_100인 경우에는 표준편차는 적지만 전체 블록의 평균 지움연산 회수가 상대적으로 증가하는 문제점이 발생한다. 접근지역성이 낮은(80/20) 경우에는 그룹 교환의 주기가 50까지는 표준편차와 평균 지움연산 회수 모두 감소하지만 100이상인 경우에는 표준편차와 평균 지움연산 회수가 모두 증가하는 현상이 발생한다.

표 2. 그룹교환 주기에 따른 wear-leveling의 비교

	10/90	20/80
CB-S	22.36 (1152~1287)	14.22 (1196~1283)
CB-MG_10	9.11 (1277~1335)	5.44 (1269~1303)
CB-MG_20	6.41 (1259~1299)	4.54 (1271~1299)
CB-MG_50	4.53 (1248~1276)	3.88 (1272~1296)
CB-MG_100	3.41 (1269~1291)	4.13 (1396~1421)

본 논문에서 제안한 CB-MG 방법은 접근지역성이 증가함에 따라 CB-S에 비해 성능은 더 좋아지고 wear-leveling 또한 양호한 결과를 보여준다. 전체적으로 CB-MG는 교환 주기가 평균쓰기 50회인 경우에 성능과 wear-leveling 측면에서 좋은 결과를 보여준다.

V. 결 론

본 논문은 hot 데이터와 cold 데이터를 분리하여 저장하여 관리하는 Cost benefit 방법(CB-S)의 단점인 wear-leveling을 개선하기 위한 방법으로 hot 데이터와 cold 데이터를 서로 다른 그룹에에 저장하여 관리하고 일정주기로 그룹의 역할을 교환하는 방법을 제안하고 성능을 평가하였다. hot 데이터와 cold 데이터를 블록단위로 분리하여 저장하여 관리하는 Cost benefit 방법은 좋은 성능을 보이지만 균등하지 않은 wear-leveling을 제공하는 것이 단점으로 지적된 반면, 본 연구에서 제안한 CB-MG 방법은 접근지역성을 보이는 작업부하에 대해 CB-S 방법에 비해 좋은 성능을 보임과 동시에 균등한 wear-leveling을 보이는 것으로 나타났다. CB-MG는 그룹의 교환 주기가 평균 쓰기 50회로 하면 가장 좋은 성능과 좋은 wear-leveling을 보이는 것으로 분석되었다.

참고문헌

[1] Jen-Wei Hsieh, Li-Pin Chang and Tei-Wei Kuo. "Efficient On-line Identification of Hot Data for Flash-Memory Management", In SAC, pp.838-842, 2005.

[2] Y. Ryu and K. Lee. Improvement of space utilization in nand flash memory storages. Lecture Notes in Computer Science, 3820:766-775, 2005.

[3] Chang, L, Kuo, T., and Lo, S., "Real-Time Garbage Collection for Flash-Memory Storage Systems of Real-Time Embedded Systems," In ACM Trans. on Embedded Computing Systems, November 2004, pp. 837-863

[4] Li-Pin Chang, Tei-Wei Kuo and Shi-Wu Lo. "A Real-Time Garbago Collection for Flash-Memory Storage Systems of Real-Time Embedded Systems." ACM Trans. in Embedded Computing Systems, Vol.3, No.4, pp.837-863, 2004.

[5] ERAN GAL AND, SIVAN TOLEDO, "Algorithms and Data Structures for Flash Memories", ACM Computing Surveys, Vol.37, Issue.2, pp.138-163, June, 2005.

[6] L.Z Han, Y.S Ryu, T.S Chung, M.H Lee, S.W Hong, "An Intelligent Garbage Collection Algorithm for Flash Memory Storages," Lecture Notes in Computer Science, 3980:1019-1027, 2006.

[7] Atsuo Kawaguchi, Shingo Nishioka, and Hiroshi Motoda, "Flash Memory Based File System," Proceedings of USENIX Technical Conference, New Orleans, LA, pp.155-164, 1995.

[8] 장시웅, "플래시 파일시스템에서 wear-leveling 개선을 위한 블록할당 정책," 한국해양정보통신학회, 2007년도 추계종합학술대회, Vol.11, No.2 pp.574-577, 2007.