반도체및디스플레이장비학회지 제6권 제2호(2007년 6월)
Journal of the Semiconductor & Display Equipment Technology, Vol. 6, No. 2. June 2007.

동적 임계값을 이용한 메모리 소거*

박 제 호†
단국대학교 컴퓨터과학과

Dynamic Threshold based Even-wear Leveling Policies

Je Ho Park†
Dankook University, Computer Science

ABSTRACT

According to the advantageous features of flash memory, its exploitation and application in mobile and ubiquitous related devices as well as voluminous storage devices is being increased rapidly. The inherent properties that are determined by configuration of flash memory unit might restrict the promising expansion in its utilization. In this paper, we study policies based on threshold values, instead of using global search, in order to satisfy our objective that is to decrease the necessary processing cost or penalty for recycling of flash memory space at the same time minimizing the potential degradation of performance. The proposed cleaning methods create partitions of candidate memory regions, to be reclaimed as free, by utilizing global or dynamic threshold values. The impact of the proposed policies is evaluated through a number of experiments, the composition of the optimal configuration featuring the methods is tested through experiments as well.

Key Words : Flash Memory File System, Segment Cleaning, Even Wear-Leveling, Dynamic Threshold

1. 서 론

플래시 메모리의 실용화 기술과 대용량화가 가능해짐에 따라 경제적 생산성이 빠르게 발전하여 모바일 기기에 사용되는 보조 저장매체뿐만 아니라 일반 컴퓨터 구성의 디스크 시스템을 대체하는 저장 매체, 유비쿼터스 환경에서의 다양한 응용 등 플래시 메모리를 저장 매체로 사용하는 분야가 다양해지고 있다[1, 2, 3, 4, 8, 9]. 일반적으로 메인 메모리에 사용되는 메모리 소재와 비교할 때, Table 1에서 보는 바와 같이, 플래시 메모리의 특성 즉 상대적으로 늦은 반응속도와 제한된 갱신 횟수라는 물리적 특성은 여러 가지 상황에서 제한적 상황을 야기시킬 수 있다[6].

플래시 메모리의 구성은 생산자에 의해 정의되는 읽기/쓰기가 가능한 특정 수의 블록들로 정의되는 세그먼트의 집합들을 기본으로 한다. 하지만, 플래시 메모리의 소거 단위는 블록이 아닌 세그먼트이기 때문에 기존의 메모리와는 다른 갱신 과정을 필요로 한다. 이를 보완하기 위해in-place 갱신은 소거 대상 세그먼트에 있는 유효 블록들을 시스템 버퍼에 임시로 저장 한 뒤, 해당 세그먼트를 소거하고, 버퍼에 저장된 블록들과 새로이 갱신되는 블록을 함께 기존의 세그먼트에 저장을 한다.

갱신에 따르는 플래시 메모리의 특성과 함께 고려해야 하는 또 다른 특성은 세그먼트 갱신 횟수에 대한 제한이다. 플래시 메모리의 메모리 셀의 갱신 횟수가 물리적 특성으로 인해 제한되어, 다수의 셀로 구성되는 세그먼트 또한 갱신에 제한을 받게 된다. 만일 한 셀의 갱신 횟수가 제한된 범위에 도달하면 해당 셀은 손상되고, 해당 영역에 대한 쓰기 시도는 장애가 발생시켜 적절한 반응 속도를 보장할 수가 없게 된다. 특정 세그먼트들에 집중된 갱신으로 인한 메모리 컴포넌트 전체의 장애를 방지하기 위해서는 전체 세그먼트에 대한 갱신 횟수의 균등한 분포가 절대적으로 필요하다. 이러

†E-mail : dk_jhpark@dankook.ac.kr

한 점에 착안하여 non-in place방법론은 갱신 과정에 포함된 읽기와 쓰기 과정을 분리하여 수행한다[10].

Table 1. Characteristics of Flash Memory

특성	특성값
블록 읽기 시간	120 ~ 250 nanosec
블록 쓰기 시간	6 ~ 9 microsec/byte
세그먼트 쓰기 시간	0.4 ~ 0.6 sec
세그먼트 소거 시간	0.6 ~ 0.8 sec
세그먼트 크기	64/128 Kbytes
세그먼트 소거 한도	100 K ~ 1,000 K

재활용을 위한 플래시 메모리 세그먼트를 확보하기 위해서는, 부분적으로 무효화된 자료(블록)가 포함된 세그먼트에서 유효한 자료를 다른 세그먼트로 옮긴 뒤, 해당 세그먼트를 소거한다. 이러한 소거 대상 세그먼트의 선택은 클리닝(cleaning) 정책에 의해 결정된다. Table 1에서 보여지는 특성값은 세그먼트 크기와 다른 구성 요소에 따라 복합적인 영향을 받는 것으로 알려져 있다. 공통적으로 세그먼트 소거는 소요 시간이 클 뿐 아니라 에너지 소모량이 다른 동작에 비해 많기 때문에, 클리닝 정책의 목표는 소거 횟수를 최소화하여 전력 소비와 반응 속도 측면에서 시스템 성능을 개선하는 동시에 일부 세그먼트에 소거가 집중 되는 것을 방지하여wear leveling 문제를 완화시켜야 하는 양면적 기능을 수행하여야 한다.

본 논문에서는 탐색 대상 세그먼트 집합을 축소하여 소거 비용의 최소화와 개선된 균등소거를 통해 저비용으로 플래시 메모리의 성능과 생명주기의 연장을 동시에 만족시키는 방법론에 대해 논의를 한다. 이를 위해 소거 대상 세그먼트 탐색 비용을 최소화할 수 있는 탐색 세그먼트 영역 축소를 위한 방법론을 제안하고, 실험적 방법을 통해 기존의 방법론들과의 비교하여 제안된 방법론이 비용과 성능 측면에서 우수함을 검증한다. 또한 제안하는 방법론으로 시스템 구성에 필요한 최적화된 분할 구성을 실험을 통하여 검증한다.

2. 플래시 메모리 재생 공간 분할

2.1. 기존의 소거 대상 선택 방법론

가장 기본적인 소거 대상 세그먼트 선택을 위한 방법론은 무작위 선택이다. 무작위 방법은 대상 세그먼트의 속성을 고려하지 않은 상태에서 무작위로 소거 대상을 결정한다. 무작위 선택 방법론을 제외한 기존의 소거 대상 세그먼트 선택 알고리즘들은 최적(최소/최대) 속성값을 검색하기 위하여 모든 대상 세그먼트의 속성을 각각 계산하고 대상 세그먼트 전체에 대한 전역 탐색을 실시 한다[1, 2, 3, 7]. 탐욕적 선택 방법론은 대상 세그먼트들 중에서 (무효 블록 수+미사용 블록 수) 값의 최대값을 가지는 세그먼트를 소거 대상으로 선택한다[3]. 비용/편익 기반 알고리즘은 특정 세그먼트에 대해 마지막 블록 무효화 시간과 계산 수행까지의 시간을 age라 하고, 그 세그먼트의 유효 데이터의 비율을 u라고 정의하고 (age* (1-u))/(2*u)의 최대값을 가지는 세그먼트를 소거 대상으로 선택한다[3, 6].

전역 검색을 수용하는 방법론은 플래시 메모리의 대용량화 현상에 따라 탐색 영역 크기의 확장이 불가피해지게 되고, 이에 따른 시스템 운영에 따르는 소요 비용의 급증은 플래시 메모리에 대한 기대 성능에 부응하지 못하는 결과를 가져 올 것이다. 본 논문에서 제안하는 방법론은 속성값 정의에 관한 것이 아니라 소거 대상 검색 영역을 축소하여 기존의 방법론이 가지는 검색 비용을 최소화할 뿐 아니라 최적값 기반 방법론에 비교될 수 있는 비최적값 기반의 방법론을 제안한다. 최적값을 사용하지 않을 경우 고려해야 하는 대상 세그먼트의 집합은 일정 수준의 질을 보장할 수 있는 방법론을 필요로 하게 되며 이를 위해 제안하는 방법론은 임계값을 사용한다.

2.2. 정적 분할 기반 플래시 메모리 관리

2.2.1. 탐색 비용

한 개의 플래시 메모리 모듈에 포함된 세그먼트의 개수를 SegNo이라고 하자. 또한 적어도 한 개 이상 유효 데이터 블록을 포함하는 세그먼트들의 전체 메모리에 대한 비율을 Util 이라고 정의하자. 전역 탐색을 적용할 때, 최적값 탐색에 필요한 세그먼트의 수 Search-Space 는 다음과 정의된다.

$$SearchSpac\ e = \lceil SegNo * Util \rceil \quad (1)$$

플래시 메모리의 크기가 작거나 유효 데이터율이 미미한 경우 탐색 영역이 작아, 탐색 비용 또한 간과해도 될 정도로 작아질 수 있지만, 대용량 플래시 메모리에 대한 탐색 비용은 저장매체의 성능에 심각한 장애가 될 수 있다. 이를 방지하기 위해 임계값 기반 플래시 메모리 관리는 탐색 비용을 최소화하는 동시에 소요 비용 대비 성능을 최대화하려는 것이다. 여기서 우리는 임계값을 위해 두 가지 방법을 고려하고자 한다: 고정 임계값과 유동 임계값. 고정 임계값은 탐색 시 고려 대상이 되는 속성값의 범위를 미리 알 수 있는 경우 사용할 수 있으며, 유동 임계값은 속성값의 한계가 시

간에 따라 변하는 경우에 사용할 수 있다.

2.2.2. 고정 임계값 기반 분할

고정 임계값 기반 플래시 메모리 관리(K-STFM)는 탐색 영역을 다수의 동일한 영역 크기를 가지는 하부 탐색 영역으로의 분할을 기초로 한다. 이 논문에서는 최적의 하부 탐색 영역 카디널리티(cardinality)를 K 라 명명하고, K 값은 실험적인 방법론을 통하여 결정을 한다.

각 하부 탐색 영역의 이상적인 크기 SubSize는 다음과 같이 정의된다.

$$SubSize = \left\lceil \frac{SearchSpac\,e}{K} \right\rceil \quad (2)$$

K-STFM에서 소거 대상 세그먼트를 선택할 때, 탐색 영역의 크기는 식 (2)에서 구한 크기 SubSize 에 의해 한정되고, 각 하부 영역은 탐색 우선 순위를 나타내는 대표 속성값을 가지게 된다. 특정 대표 속성값을 가지는 분할(하부영역)에 속하는 세그먼트들은 주어진 영역 내에서 서로 다른 속성값을 가질 수 있다. 따라서, 전역 검색 시 사용되는 속성값과 분할 시 사용되는 속성값이 같다고 가정할 때, 최상위 대표 속성값에 상응하는 하부 영역을 탐색한 결과는 적용된 알고리즘에 따라 차이를 보일 수 있으며, 크게 두 가지 방법에 의해 수행될 수 있다: 무작위 선택과 전역 탐색 기반 선택.

전체 소거 대상 세그먼트 집합에서 각 세그먼트의 특정 속성값 AttrVal 이 중복되지 않는다고 가정하고, 사용되는 알고리즘에 따라 최적 속성값을 가지는 세그먼트를 S_{opt} 라고 정의하자. 분할에 속하는 세그먼트가 하나 이상이고 최상위 대표 속성값을 가진 하부 탐색 영역 Region(K)에 속하는 세그먼트들 Seg_i에 대해, $1 \le i \le SubSize$이고, i에 대해 $AttrVal(Seg_j) \le AttrVal(Seg_{i+1})$ 가 성립한다고 하면, Region(K)에 속하는 세그먼트들 중에서 최적 속성값을 가지는 세그먼트$Seg_{SubSize}$ 는 S_{opt}와 동일하게 된다.

적절한 속성값을 가진 세그먼트의 소거는 균등성을 향상시켜 결과적으로 레벨링을 개선한다는 관련 연구 결과에 착안할 때, 전체 영역에 무작위 선택을 적용하는 것보다 하부 탐색 영역 Region(K)에 무작위 선택을 적용한 경우가 보다 개선된 레벨링 효과를 가져올 수 있다. 일반적 경우를 고려할 때, Region(K)에 속하는 세그먼트들은 높은 최적성에 따라 분할되었으므로, Region(K)에 무작위 방법을 적용하는 것이 전체 세그먼트 집합에서 최적성의 고려 없이 무작위 선택을 적용한 것보다 레벨링 기여도 면에서 우수할 것이다.

전역 탐색은 소거 세그먼트 선택 시 항상 최적성에 기반하여 수행되기 때문에, 동일한 속성값 함수 AttrVal 을 적용할 때, 전역 탐색 정책과 분할 기반 정책은 성능 측면에서 차이를 보일 것으로 예측된다. 하지만, 전역 탐색의 경우 비교에 의한 최적값 선택이 필수적이므로 탐색 비용은 분할 기반 정책보다 높다. 탐색 비용은 탐색 수행 시간과 메모리 사용량 측면에서 고려할 수 있으며, 이미 플래시 메모리에 있는 자료 구조만을 이용할 경우 수행시간은 O(SearchSpace) 라는 비용을 들여야 한다. 이와는 달리 수행시간을 최소화하기 위해 검색 트리와 같은 자료 구조를 사용할 경우 탐색 수행 시간 O(SearchSapce*log SearchSpace) 를 필요로 한다. 아울러, Cost 를 각 세그먼트 정보 유지에 필요한 메모리 비용이라고 할 때, 전체 O(SearchSpace*Cost)의 메모리 비용이 소요된다.

무작위 선택을 이용한 분할 기반 정책은 탐색 비용이 최소화되며, 세그먼트의 자료구조에 포인터 형태의 집합 표현을 사용할 경우 메모리 비용도 최소화된다. 따라서, 비용 대비 효과라는 측면에서 전역 탐색 방법론과 비교할 때, 분할 기반 정책은 성능의 손실은 최소화하면서, 탐색 비용의 최소화라는 목표를 만족시킬 수 있을 것으로 기대한다. 또한 특정 하부 영역에서 최적 속성값 탐색 시 해당 영역에 대한 전역 탐색을 할 경우, 레벨링 효과는 전역 탐색 기반 클리닝 기법의 성능과 유사할 것이라는 기대에는 무리가 없을 것이다.

2.2.3. 정적 분할 방법

전체 탐색 영역에 속하는 세그먼트들은 실제로 플래시 메모리의 수명 동안 계속적으로 변화한다. 따라서, 하부 탐색 영역으로 대상 세그먼트들을 분할하기 위해서는 고정된 하부 영역 속성값의 설정이 필요하다. 이를 위해 본 논문에서는 탐욕적 방법론에 사용되는 세그먼트 속성값 공간을 고정 임계값을 이용하여 K개의 하부 영역으로 분할하는 방법을 제안한다. 탐욕적 방법론에서 사용하는 속성값 G 는 한 세그먼트에 존재하는 무효 데이터 블록 수와 데이터 블록으로 사용되지 않은 블록 수의 합으로 한다. 따라서, B를 한 세그먼트에 속하는 블록수로 정의할 때, G는 다음의 영역을 갖는다.

$$1 \le G \le (B-1) \quad (3)$$

이 영역을 K개로 분할하기 위해 다음과 같이 구간 간격 W를 정의한다.

$$W = \left\lceil \frac{B-2}{K} \right\rceil \quad (4)$$

여기서 각 하부 영역은 대표 속성값 P를 가진다고 가정하면, 대표 속성값은 1부터 K까지의 값을 가진다. 따라서, 대표 속성값이 P인 하부 영역은 세그먼트 속성값에 대해 다음과 같은 하한값 L와 상한값 U를 가지도록 정의한다.

$$L = 1 + (P-1)*W \quad (5)$$

$$U = L + (W-1) \quad (6)$$

하부 영역 대표 속성값이 K인 경우는 상한값을 (B-1)로 정의한다. 한 세그먼트의 속성값에 변화가 생길 경우, 위에 주어진 방법에 따라 대표 속성값이 계산되고, 해당 세그먼트는 대표 속성값에 따라 소속 하부 영역을 변경한다.

2.2.4. 동적 분할 기반 플래시 메모리 관리

고정 임계값을 이용하여 소거 대상 세그먼트들을 분할하는 방법은 세그먼트 속성값의 최대값이 정해지지 않는 경우에는 적용하기가 힘들다. 이런 경우에는 동적으로 속성값 구간을 결정할 수 있는 임계값의 결정이 플래시 메모리가 실제로 사용되는 동안에 결정되어야 한다. 이 논문에서는 유동 임계값의 유효성을 검증하기 위하여 주어진 시간창 내에서 구해진 속성값의 기본 통계를 이용하여 K개의 구간을 위해 임계값을 설정하는 방법을 소개한다. 주어진 시간창 동안 관찰된 대상 속성값의 평균을 $Attr_{Ave}$라 정의하고, 최대값을 $Attr_{Max}$이라고 정의할 때, K개의 분할은 다음 식에 의하여 결정된다.

$$W_D = \left\lceil \frac{Attr_{Ave} - Attr_{Max}}{K} \right\rceil \quad (7)$$

이 식을 이용하여 대표 속성값 P를 가지는 하부영역의 정의를 위해 하한값 L_D와 상한값 U_D를 다음과 같이 정의한다.

$$L_D = 1 + (P-1)*W_D \quad (8)$$

$$U_D = L_D + (W_D - 1) \quad (9)$$

위의 식을 응용하여 구간을 나눌 때, 대표 속성값 1을 가지는 구간의 하한값은 $Attr_{Ave}$로 설정하고, 대표 속성값 K를 가지는 구간은 $Attr_{Max}$를 상한값으로 설정한다. 본 논문에서 제안하는 플래시 메모리 관리의 유효성과 자유 세그먼트 선택에 따른 영향성은 시뮬레이션 기반 실험을 통해 검증하였다

3. 구현 사례 및 결과 분석

3.1. 구현사례

시뮬레이션 기법에 기반한 실험의 목적은 첫째로 제안하는 플래시 메모리 관리의 유효성을 실험 결과치를 통해 검증하고, 둘째로 제안하는 플래시 메모리 관리와 분할된 하부영역과의 관련성을 실험적 방법을 통해 밝혀보고자 하는 데 있다. 사용된 플래시 메모리 모델은 플래시 메모리의 전형적인 모델을 따라 구성하였다[1, 3]. 시뮬레이션은 태스크 발생기를 이용하여 사용할 논리 블록 번호를 임의적으로 생성한다. 생성된 접근 논리 번호는 플래시 파일 시스템에 의해 플래시 블록 번호로 변환되어, 해당 블록의 읽기/쓰기를 한다.

3.2. 성능 분석을 위한 실험

본 논문에서는 플래시 메모리의 사이즈를 4 GByte로, 세그먼트 크기는 64 KByte로 설정하고, 메모리 블록은 2 KByte로 설정하였다. 플래시 메모리 활용도 역시 플래시 메모리 시스템에 크게 영향을 미치는 것으로 알려져 있다[3, 7, 9]. 성능 면에서 관찰할 때 병목현상은 주로 플래시 메모리의 80%의 활용도에서 나타나기 때문에 본 논문도 활용도 80%를 설정하였다. 실험은 접근 집약성(locality)을 증가시키면서 수행되었다. 이는 접근 집약성에 따른 플래시 메모리의 성능 변화가 플래시 메모리 시스템 구현의 중요한 사항으로 인식되고 있기 때문이다[7]. 접근 집약성을 실험에 적용시키기 위하여 태스크 발생기에 사용 빈도수가 높은 영역에 대한 접근(Hot Access)과 사용 빈도수가 낮은 영역에 대한 접근(Cold Access)을 설정하고, 접근 발생 시 종류를 조절하여 접근 집약성을 구현하였다.

성능 분석을 위해 구현된 알고리즘들은 기존의 무작위 선택 (Random), 탐욕적 선택(Greedy), 비용-/편익 선택 (CostBenefit) 알고리즘들과 제안하는 고정 임계값 기반(MultisetS)과 유동 임계값 기반(MultisetD) 방법론들이다.

3.3. 실험 결과 분석

성능 분석을 위한 측정치로는 한 세그먼트 소거 및 블록 갱신 당 접근 수를 수집하였다. 균등소거 성능을 측정하기 위해서는 세그먼트 소거 횟수에 대한 표준편차를 수집하였다. Fig. 1은 한 개의 세그먼트 소거 당 지원 접근 수를 보여 주고 있다. 무작위 선택 알고리즘은 메모리의 속성을 고려하지 않는 관계로 대체로 최소한의 성능만을 보이고 있으며, 접근 집약성이 커질수록 임계값 기반 선택 알고리즘들이 우수한 성능을 보

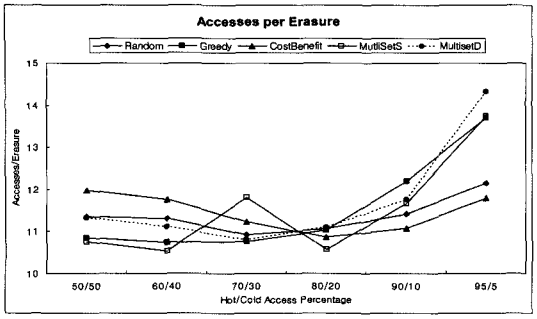

Fig. 1. Block Accesses per Segment Erasure.

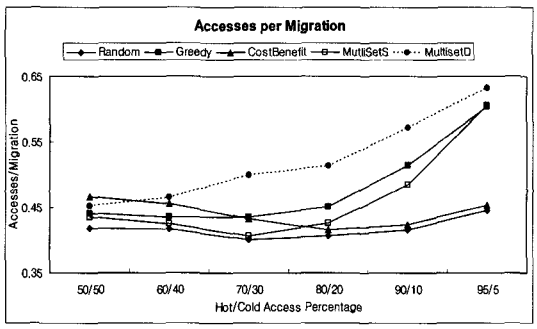

Fig. 2. Block Accesses per Block Migration.

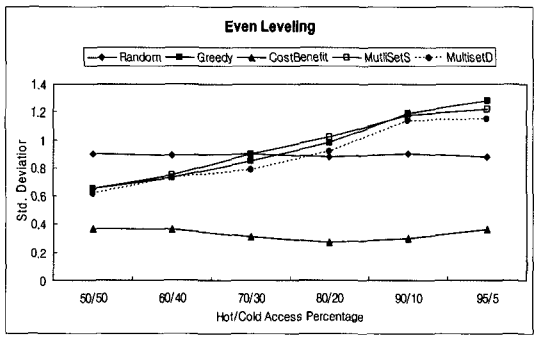

Fig. 3. Even Leveling Measurements.

였다.

Fig. 2는 한 개의 블록 이동 당 지원 접근 수를 보여준다. 이 결과값에 의하면 임계값 기반 알고리즘들은 비교적 블록의 이동에 따른 효율이 일반적 접근 집약성 수준에서 우수한 것으로 나타나고, 무작위 선택과 비용/편익 알고리즘은 접근 집약성이 커질수록 블록 이동에 따른 효율이 떨어지는 것으로 나타났다.

Fig. 3에 예시되어 있는 레벨링 효과 측면에서는 탐욕적 선택 알고리즘과 임계값 기반 알고리즘들이 다른 방법론들보다 우수한 성능을 보이고 있다.

임계값 기반 플래시 메모리 관리에서 구간의 개수 K

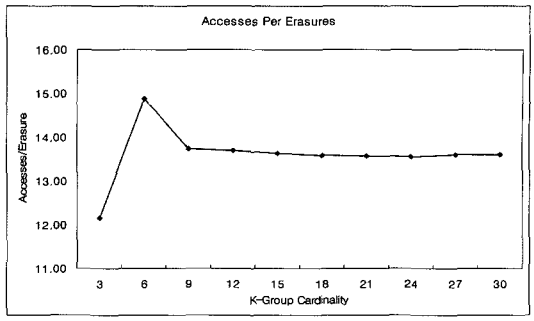

Fig. 4. Optimal Cardinality.

값은 시스템의 효율성에 많은 영향을 미칠 수 있으므로 앞의 실험에서 사용한 시스템을 이용하여 K 값을 변화시키면서 블록 갱신 당 접근수와 레벨링을 측정하였다. 이 실험을 위해서는 고정 임계값 방법론을 사용하였으며, 대부분의 관련 알고리즘이 탁월한 성능을 보이는 환경을 위해 접근 집약성은 95%로 설정하여 채택하였다. Fig. 4에서 6개의 분할을 사용할 때 최적의 성능을 보이고 있음을 알 수 있다.

4. 결 론

이 논문에서 우리는 플래시 메모리 공간에서 재활용을 목적으로 최적화된 메모리 내부에 데이터를 재배치하는 방법에 대해 논의를 하였다. 제안된 임계값 기반 분할 방법은 저비용으로 기존의 방법과 성능 면에서 뒤떨어지지 않고 있음을 실험을 통해 검증했다. 제안하는 방법은 다수의 하부영역을 사용하므로, 하부영역의 개수는 성능에 영향을 미친다. 최적값의 존재 여부와 근사치를 얻기 위해 실험을 통해 추정한 결과 한 세그먼트 당 블록수의 15%에서 19% 사이의 값을 구간 수로 설정하는 것이 시스템 성능에 긍정적인 영향을 미치는 것으로 나타났다. 본 본문에서 논의한 저비용 고효율 플래시 메모리 관리 방법론을 발전시키기 위해서는 다양한 방법의 분할 기법을 적용하여 보다 실제 상화에서 발생할 수 있는 워크로드에 적응할 수 있는 분할 기법에 대해 연구를 진행해야 한다. 이를 위해서는 한 기법이 아니라 다양한 기법을 상황에 맞게 선택하는 선택적 알고리즘의 개발을 필요로 하며, 이에 대한 연구는 현재 진행 중이다.

감사의 글

이 연구는 2005 학년도 단국대학교 대학연구비의 지원으로 연구되었음.

참고문헌

1. Li-Pin Chang, Tei-Wei Kuo and Shi-Wu Lo. "A Real-Time Garbage Collection for Flash-Memory Storage Systems of Real-Time Embedded Systems." ACM Trans. in Embedded Computing Systems, Vol.3, No.4, 837-863 (2004).
2. Mei-Ling Chiang, Paul C.H. Lee and Ruei-Chuan Chang. "Using Data Clustering to Improve Cleaning Performance for Flash Memory." Software-Practice and Experience, Vol.29, No.3, 267-290 (1999).
3. Mei-Ling Chiang and Ruei-Chuan Chang. "Cleaning Policies in Mobile Computers Using Flash Memory." Journal of Systems and Software, Vol.48, No.3, 213-231 (1999).
4. Joshua B. Fryman et al. "Energy-efficient Network Memory for Ubiquitous Devices", IEEE Micro, Vol.23, No.5, 60-70 (2003).
5. Jen-Wei Hsieh, Li-Pin Chang and Tei-Wei Kuo. "Efficient On-line Identification of Hot Data for Flash-Memory Management", In SAC, p.838-842 (2005).
6. Atsuo Kawaguchi, Shingo Nishioka and Hiroshi Motoda. "A Flash-Memory Based File System." In USENIX Winter, p.155-164 (1995).
7. Han-joon Kim and Sang-goo Lee. "An Effective Flash Memory Manager for Reliable Flash Memory Space Management." IEICE Transaction on Information and Systems, E85-D(6): 950-964, June (2002).
8. Brian Marsh, Fred Douglis and P. Krishman. "Flash Memory File Caching for Mobile Computers." In HICSS(1), p.451-461 (1994).
9. Sang Lyul Min and Eyee Hyun Nam. "Current Trends in Flash Memory Technology" In Proceedings of ASP-DAC 2006, Yokohama, Japan, January 24-27, (2006).
10. M. Rosenblum and J. K. Ousterhout. "The Design and Implementation of a Log-Structured File System." ACM Trans. Computer Systems, Vol.10, No.1, 26-52 (1992).