K 분할 기반 플래시 메모리 균등소거 방법론

박 제 호†

요 약

플래시 메모리의 활용성이 높은 특성으로 인해 모바일 기기와 유비쿼터스 관련 기기에 대한 적용이 확장되고 있다. 하지만, 이러한 경향은 플래시 메모리의 물리적 특성으로 인해 제한 받을 수 있다. 이 논문에서는 플래시 메모리 공간의 재활용을 위한 방법론을 제안하다. 이 방법론은 메모리 재활용에 필요한 비용과 재활용 성능을 동시에 최적화하는 것을 목표로 한다. 제안하는 방법론은 특정시간에 재사용되는 메모리 세그먼트를 선택할 때 대상이 되는 메모리 공간을 다수의 하부 공간으로 분할하여 탐색 비용을 최적화한다. 아울러, 자유 세그먼트의 선택이라는 측면에서 전체 메모리 공간의 균등한 소거를 위한 방법론 또한 논의한다. 제안된 방법론들은 기존의 방법론과 함께 실험을 통해 검증하였으며, 방법론의 수행을 위한 최적화된 시스템 구성을 실험을 통하여 밝혔다.

키워드 : 플래시 메모리 파일 시스템, 세그먼트 클리닝, 균등소거

K Partition-Based Even Wear-Leveling Policy for Flash Memory

Je-ho Park†

ABSTRACT

Advantageous features of flash memory are stimulating its exploitation in mobile and ubiquitous related devices. The hardware characteristics of flash memory however place restrictions upon this current trend. In this paper, a cleaning policy for flash memory is proposed in order to decrease the necessary penalty for recycling of memory minimizing the degradation of performance at the same time. The proposed cleaning algorithm is based on partitioning of candidate memory regions, to be reclaimed as free, into a number of groups. In addition, in order to improve the balanced utilization of the entire flash memory space in terms of "wearing-out", a free segment selection algorithm is discussed. The impact of the proposed algorithms is evaluated through a number of experiments. Moreover, the composition of the optimal configuration featuring the proposed methods is tested through experiments.

Key Words : Flash Memory File System, Segment Cleaning, Even Wear-leveling

1. 서 론

최근의 경향을 볼 때, 플래시 메모리 관련 기술로 빠른 발전에 힘입어 용량과 입출력 성능 면에서 모바일 기기 보조 저장매체 뿐만 아니라 일반 컴퓨터 환경에 필요한 요건을 만족시키고 있다[2-4, 9]. 아울러 유비쿼터스 환경에 대한 연구와 개발이 활발해지면서 플래시 메모리의 활용에 대한 관심이 점진적으로 가속화되고 있다[1, 5]. 이러한 경향과 함께 대용량 플래시 메모리의 실용적인 활용이 증가되면서 차세대 저장매체로서의 역활에 대한 기대가 크다. 그러나, 전형적인 RAM과 비교할 때, 플래시 메모리의 늦은 반응속도와 제한된 갱신 횟수라는 물리적 특성을 이해하지 않고서는 실제적 응용을 고려할 수 없다[4, 7]. 플래시 메모리의 일반적인 특성은 <표 1>에서 보는 바와 같다.

플래시 메모리는 생산할 때 정의되는 세그먼트의 집합으로 구성되며, 한 세그먼트는 읽기/쓰기가 가능한 블록들로 구성된다. 플래시 메모리에서 자료 갱신은 특성 상 세그먼트에 있는 기존의 자료를 먼저 소거하고 새로운 자료를 저장한다. 여기서 주목할 점은 플래시 메모리의 소거 단위는 블록이 아닌 세그먼트라는 것이다. In-place 갱신은 소거 대상 세그먼트에 있는 유효 블록들을 시스템 버퍼에 임시로 저장 한 뒤, 해당 세그먼트를 소거한다. 그리고, 시스템 버퍼에 저장된 블록들은 제위치에 재저장되고, 새로이 갱신되는 블록도 함께 저장된다.

플래시 메모리를 대용량 저장매체로 사용할 때 고려해야 하는 단점은 세그먼트 갱신 횟수에 대한 제한이다. 플래시 메모리

〈표 1〉 플래시 메모리의 특성

특 성	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

※ 이 연구는 2004학년도 단국대학교 대학연구비의 지원으로 연구되었음.
† 종신회원 : 단국대학교 컴퓨터과학과 교수
논문접수 : 2006년 3월 24일, 심사완료 : 2006년 5월 22일

는 물리적 특성으로 인해 메모리 셀의 갱신 횟수가 제한되며, 결과적으로 다수의 셀로 구성되는 세그먼트 또한 갱신에 제한을 받게 된다. 만일 한 셀의 갱신 횟수가 제한된 범위에 근접하거나 한계를 넘게 되면 해당 셀은 손상된다. 따라서, 해당 영역에 쓰기를 시도하면 장애가 발생되고, 이로 인해 적절한 반응 속도를 보장할 수가 없게 된다. 최악의 경우에는 플래시 메모리 컴포넌트 전체의 장애를 가져올 수도 있다. 따라서, 특정 세그먼트들에 집중된 갱신으로 인한 장애를 방지하기 위해서는 전체 세그먼트에 대한 갱신 횟수의 균등한 분포가 절대적으로 필요하다. 이러한 점에 착안하여 non-in place방법론은 갱신 과정에 포함된 읽기와 쓰기 과정을 분리하여 수행한다[7, 8, 10].

플래시 메모리에서 재활용을 위한 세그먼트를 확보하기 위해서는, 부분적으로 무효화된 자료(블록)를 포함한 세그먼트에서 유효한 자료를 다른 세그먼트로 옮긴 뒤, 해당 세그먼트를 소거한다. 이 때 소거 대상 세그먼트는 클리닝(cleaning) 정책에 의해 결정된다. <표 1>에서와 같이 세그먼트 소거는 시간이 많이 소요되고, 에너지 소모량이 다른 동작에 비해 상대적으로 많기 때문에, 클리닝 정책은 소거 횟수를 최소화하여 전력 소비와 반응 속도 측면에서 시스템 성능을 개선해야 한다. 반면에 클리닝 정책은 일부 세그먼트에 소거가 집중 되는 것을 방지하여 wear leveling 문제를 동시에 완화시켜야 한다.

가장 기본적인 소거 대상 세그먼트 선택은 무작위 선택이다. 탐욕적 방법은 대상 세그먼트들 중에서 무효 블록와 사용되지 않는 블록을 최대로 포함하는 세그먼트를 소거 대상으로 선택한다[2]. 비용/편익 기반 알고리즘은 획득할 수 있는 블록의 갯수와 함께 마지막 갱신 이후의 시간을 고려하여 선택을 결정한다[2, 7]. 무작위 선택을 제외하고, 세그먼트 균등소거와 관련된 기존 연구들은 소거 대상 결정에 사용되는 속성값에 초점을 두고, 유효한 컨텐츠를 가진 세그먼트 전체를 대상으로 하는 전역 탐색을 기반으로 한다. 따라서 플래시 메모리의 대용량화로 인해 성능 측면에서의 필요 비용과 탐색 영역의 크기 간의 관계는 주의 깊은 관찰을 필요로 한다.

본 논문에서 제안하는 방법론은 대상 세그먼트 집합을 다수의 부분집합들로 분할하여, 탐색영역을 축소시켜 소거 비용의 최소화와 개선된 균등소거를 통해 플래시 메모리의 성능과 생명주기의 연장을 동시에 만족시키게 한다. 이를 위해 소거 대상 세그먼트 탐색 비용을 최소화할 수 있는 세그먼트 분할을 위한 방법론을 제안하고, 실험적 방법을 통해 기존의 방법론들과의 비교를 통해, 제안된 방법론이 비용과 성능 측면에서 우수함을 검증한다. 또한 제안하는 방법론으로 시스템 구성 시 필수적인 최적화된 분할 구성을 실험을 통하여 검증하고, 새롭게 컨텐츠를 저장하기 위해 사용되는 자유 세그먼트의 선택과 균등소거와의 관계에 대해서 논의한다.

2. K-분할 플래시 메모리 관리

2.1 탐색 비용의 최소화

무작위 선택 방법론을 제외하고, 이제까지 연구된 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[1-3, 8]. 한 개의 플래시 메모리 모듈에 포함된 세그먼트의 개수를 N이라고 하고, 적어도 한 개 이상 유효 데이터 블록을 포함하는 세그먼트들의 전체 메모리에 대한 비율을 u 라고 정의하자. 전역 탐색을 적용할 때, 최적값 탐색에 필요한 세그먼트의 수 S는 다음과 정의된다.

$$S = \lceil N * u \rceil \quad (1)$$

플래시 메모리의 크기가 작거나 유효 데이터율이 미미한 경우 S 의 크기가 작아, 탐색 비용 또한 간과해도 될 정도로 작아진다. 하지만, 대용량 플래시 메모리에 대한 탐색 비용은 저장매체의 성능에 심각한 장애가 될 가능성이 높다. 이를 방지하기 위해K-분할 플래시 메모리 관리는 탐색 비용을 최소화하는 동시에 소요 비용 대비 결과를 최대화하려는 것이다.

K-분할 플래시 메모리 관리는 탐색 영역 S 를 다수의 하부 탐색 영역으로 분할하는 데 그 출발점을 두고 있다. 이 논문에서는 최적의 하부 탐색 영역 카디널리티(cardinality)를 K라 명명하고, K값은 실험적인 방법론을 통하여 결정을 한다.

각 하부 탐색 영역의 이상적인 크기 T는 다음과 같이 정의된다.

$$T = \lceil (N * u) / K \rceil \quad (2)$$

K-분할 플래시 메모리 관리에서 소거 대상 세그먼트를 선택할 때, 탐색 영역의 크기는 식 (2)에서 구한 크기 T에 의해 한정된다. 각 하부 영역은 탐색에 대한 우선도를 표현하는 대표 속성값이 주어지게 된다. 특정 하부 검색 영역에 속하는 세그먼트들의 속성값은 동일하지는 않으나, 전체 소거 대상 세그먼트 집합을 분할할 때 사용되는 속성이 전역 탐색 시의 속성과 같다고 가정할 때, 최상위 대표 속성값를 가지는 하부 영역만을 탐색한 결과는 적용된 알고리즘에 따라 차이를 보일 것이라고 기대된다. 하부 영역에서 소거할 세그먼트의 선택은 크게 두 가지 방법에 의해 수행될 수 있다: 무작위 선택과 전역 탐색 기반 선택. 무작위 선택은 해당 하부 영역에 속하는 세그먼트들 중에서 속성값의 우열에 상관 없이 무작위로 선택하는 방법이고, 전역 탐색 기반 선택은 해당 하부 영역에서 최적 속성값을 가지는 세그먼트를 소거 대상으로 한다.

전체 소거 대상 세그먼트 집합에서 각 세그먼트의 특정 속성값 f는 중복되지 않는다고 가정하고, 속성값 특성에 따른 최적값을 가지는 세그먼트를 S_{max}라고 정의하자. 분할에 속하는 세그먼트가 하나 이상이고 최상위 대표 속성값을 가진 하부 탐색 영역 R_K에 속하는 세그먼트들을 다음과 같이 표현하자:

$$S_i \in R_K \quad (3)$$

여기서 $1 \le i \le T$이고, i 에 에 대해 $f(S_i) \le f(S_{i+1})$가 성립한다고 하면, R_K에 속하는 세그먼트들 중에서 최적 속성값을 가지는 세그먼트S_T는 S_{max}와 동일하게 된다 - $f(S_T) = f(S_{max})$.

우수한 속성값을 가진 세그먼트의 소거는 레벨링 성능을 개선한다는 관련 연구 결과에 착안할 때, 전체 영역에 무작위 선택을 적용한 경우보다 하부 탐색 영역 R_K에 무작위 선택을 적용한

경우가 레벨링 효과를 개선할 수 있다. 이러한 기대는 최대 대표 속성값을 가지는 하부 탐색 영역 R_K 는 분할을 통하여 속성값의 최적성이 높은 세그먼트들로 구성되기 때문이다. 전체 S개의 소거 대상 세그먼트들 중에서 S_{max}가 선택될 확률은 1/S이고, 분할을 통하여 특정 하부 탐색 영역 R_K 에서 S_{max} 가 선택될 확률은 1/T이므로, $1/S < 1/T$가 성립되어 분할 기반 정책이 우수한 결과를 보일 것으로 기대한다. 일반적 경우를 고려할 때, 분할 기반 정책에서 R_K 는 소속 세그먼트들의 최적성을 고려하여 구성하기 때문에, R_K에서 무작위 방법에 의해 선택되는 세그먼트들이 전체 집합에서 최적성의 고려 없이 무작위로 선택한 세그먼트들의 레벨링 에 대한 기여도 면에서 우수할 것이다. 무작위 선택의 특성 상 탐색 비용은 하부 영역 탐색이나 전역 탐색 두 경우 모두 최소화된다. 분할 기반 소거 방법론과 무작위 기반 정책의 성능에 대한 비교는 실험을 통해 수행될 것이다.

전역 탐색은 소거 세그먼트 선택이 항상 최적성에 기반하여 수행되기 때문에, 동일한 속성값 함수 f를 적용할 때, 전역 탐색 정책과 분할 기반 정책은 성능면에서 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(S) 라는 비용을 들여야 한다. 이와는 달리 수행시간을 최소화하기 위해 검색 트리와 같은 자료 구조를 사용할 경우 탐색 수행 시간 $O(S \cdot \log S)$ 을 필요로 한다. 아울러, C 를 각 세그먼트 정보 유지에 필요한 메모리 비용이라고 할 때, 전체 $O(S \cdot C)$ 의 메모리 비용이 소요된다.

무작위 선택을 이용한 분할 기반 정책은 탐색 비용 면에서 무작위 선택의 특성에 의해 탐색비용이 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 탐색을 위해 해당 영역에 대한 전역 탐색을 할 경우, 레벨링 성능은 전역 탐색 기반 클리닝 기법의 성능과 유사할 것이라는 기대에는 무리가 없을 것으로 사려된다.

2.2 세그먼트 분할을 위한 사례 연구

전체 탐색 영역에 속하는 구성 세그먼트들은 실제로 플래시 메모리의 수명 동안 계속적으로 변화한다. 따라서, 하부 탐색 영역으로 대상 세그먼트들을 분할하기 위해서는 고정된 하부 영역 속성값을 설정하여야 한다. 이를 위해 본 논문에서는 탐욕적 방법론에 사용되는 세그먼트 속성값 공간을 K 개의 하부 영역으로 분할하는 방법을 제안한다.

탐욕적 방법론에서 사용하는 속성값 G 는 한 세그먼트에 존재하는 무효 데이터 블록의 수와 데이터 블록으로 사용되지 않은 블록의 수의 합으로 한다. 따라서, B를 한 세그먼트에 속하는 블록수로 정의할 때, B는 다음의 영역을 갖는다.

$$1 \le G \le (B-1) \tag{4}$$

이 영역을 K 개로 분할하기 위해 다음과 같이 구간 간격 W를 정의한다.

$$W = \lceil (B - 2) / K \rceil \tag{5}$$

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 lower 와 상한값 upper를 가지도록 정의한다.

$$lower = 1 + (P - 1) * W \tag{6}$$

$$upper = lower + (W - 1) \tag{7}$$

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.

클리닝 정책과는 별도로, 플래시 메모리 관리자가 자료를 저장할 새로운 세그먼트를 필요로 하는 경우 다수의 자유 세그먼트들 중에서 하나를 선택한다. 자유 세그먼트는 포함된 모든 블록의 자료가 무효화된 세그먼트를 의미한다. 일반적으로 플래시 메모리 크기가 증가 하면서 자유 세그먼트 집합의 크기도 비례하여 증가할 가능성이 크다.

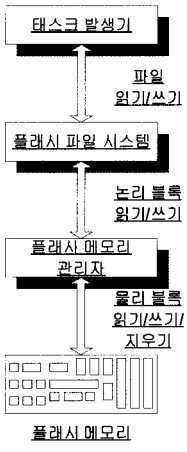
기존의 연구에서는 자유 세그먼트 선택이 균등소거에 미치는 영향을 고려하지 않았다. K-분할 플래시 메모리 관리는 자유 세그먼트 집합에서 새롭게 컨텐츠를 저장할 세그먼트를 선택할 때 최소 소거 횟수를 가지는 세그먼트를 우선적으로 선택하는 방법을 적용하여 자유 세그먼트 선택이 레벨링에 미치는 영향을 고찰하고자 한다. 이 방법은 비용/편익 기반 클리닝 정책이 가지는 특성, 즉 모든 세그먼트에 대한 소거 횟수의 균등한 분포를 유지하기 위해 대상 세그먼트 선택 시 최소 소거 횟수를 고려한다는 점에서 착안한 것이다. 본 논문에서 제안하는K-분할 플래시 메모리 관리의 유효성과 자유 세그먼트 선택에 따른 영향성은 시뮬레이션 기반 실험을 통해 검증하였다.

3. 구현 사례 및 결과 분석

3.1 구현 사례

시뮬레이션 기법에 기반한 실험의 목적은 첫째로K-분할 플래시 메모리 관리와 균등 레벨링을 위한 경험적 자유 세그먼트 선택 방법의 유효성을 실험 결과치를 통해 검증하고, 둘째로 K-분할 플래시 메모리 관리에서 최적K값과 관련하여 실험적 방법을 통해 밝혀보고자 하는 데 있다. 유효성 검사는 동일한 플래시 메모리 모델을 사용하여 기존의 클리닝 정책들과의 비교 분석 방법을 선택하였다. 시뮬레이션에 사용된 플래시 메모리 모델의 구성은 (그림 1)에서 볼 수 있다. 사용된 플래시 메모리 모델은 플래시 메모리의 전형적인 모델을 따라 구성하였다[1, 2].

태스크 발생기는 접근할 논리 블록 번호를 주어진 환경값에 따라 임의적으로 생성한다. 접근 논리 번호는 플래시 파일 시스템에 의해 플래시 블록 번호로 변환되어, 해당 블록의 읽기/쓰기



(그림 1) 플래시 메모리 관리 시뮬레이션 구성

〈표 2〉 세그먼트 헤더의 구조와 정보

정보 이름	정보 내용
No of Erasures	세그먼트 소거 횟수
Timestamp	마지막 오퍼레이션 작동 시간
In Used Flag	세그먼트의 자유성 여부
Starting Block	세그먼트의 첫번째 블록 번호
Ending Block	세그먼트의 마지막 블록 번호
Valid Blocks	세그먼트의 유효 블록의 갯수
First Free Block	세그먼트의 첫번째 유효 블록 번호

〈표 3〉 플래시 블록 정보

정보 이름	정보 내용
LBlock No	상응 논리 블록 번호
Timestamp	마지막 오퍼레이션 작동 시간
Updates	쓰기를 위해 사용된 횟수
In Used Flag	사용 여부
Invalid Flag	자료의 무효화 여부

를 한다.

플래시 메모리의 구조는 관장하는 메모리 공간을 정해진 수의 플래시 메모리 블록을 포함하는 세그먼트로 나누어 구성한다. 각 세그먼트 헤더를 통해 관리되는 정보의 내용은 <표 2>에 상세하게 나타나 있다. 전체 세그먼트들에 대한 정보인 총 세그먼트 수와 세그먼트 당 블록 갯수는 세그먼트 정리 헤더(Segment Summary Header)에 기록한다. K-분할 플래시 메모리 관리의 경우 분할에 대한 정보를 세그먼트의 헤더에 포함시키는 방법과 별도의 자료구조를 이용한 방법을 사용할 수 있다. 본 실험에서는 별도의 자료구조를 이용하여 분할 정보를 관리한다. 각 플래시 블록은 <표 3>에 정리되어 있는 정보를 유지한다.

3.2 성능 비교를 위한 실험

플래시 메모리 사이즈가 큰 경우, 본 논문에서 비교를 위해 사용되는 세그먼트 선택 알고리즘들은 그 성능 측면에서 별 다른 차이를 보이지 않는 것으로 알려져 있다[7, 9]. 이 연구 결과에 따라, 본 논문에서는 플래시 메모리의 사이즈를 2 GByte로 설정하여 실제로 응용할 수 있는 플래시 메모리 시스템을 검증하려고 노력하였다. 일반적인 구성에 따라서, 세그먼트 크기는 64KByte로 설정하고, 메모리 블록은 2KByte로 설정하였다. 플래시 메모리 활용도 역시 플래시 메모리 시스템에 크게 영향을 미치는 것으로 알려져 있다[2, 7, 9]. 성능면에서 관찰할 때 병목현상은 주로 플래시 메모리의 80%의 활용도에서 나타나기 때문에 본 논문도 활용도 80%를 설정하였다. 실험은 접근 집약성(locality)를 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.

〈표 4〉 세그먼트 선택에 따른 실험 결과 범례

범례	소거 세그먼트 선택	K-분할 사용 시 세그먼트 선택
Random-X	무작위	해당없음
Greedy-X	탐욕적	해당없음
CostBenefit-X	비용/편익	해당없음
MultisetR-X	K-분할	무작위
MultisetY-X	K-분할	최소 소거

소거 대상 세그먼트 선택 알고리즘은 무작위 정책, 탐욕적 정책, 비용/편익 정책, 그리고 K-분할 기반 정책을 구현 대상으로 삼았다. 구현된 알고리즘을 간략히 정리하면 다음과 같다.

- 무작위 선택 알고리즘 : 대상 세그먼트의 순서에 대한 무작위 번호를 발생하여 소거 세그먼트를 선택한다[3].
- 탐욕적 선택 알고리즘: 각 세그먼트에 대해(무효 블록 수 + 사용되지 않은 블록 수) 최대값을 가지는 세그먼트를 소거 대상으로 선택한다[2].
- 비용/편익 선택 알고리즘 : 해당 세그먼트에서 마지막으로 블록에 대한 무효화된 시간과 계산 수행까지의 시간을 age라 하고, 해당 세그먼트의 유효 데이터의 비율을 u라고 정의한다. 각 세그먼트에 대한 (age • (1-u))/2 • u를 계산하고, 최대값을 가지는 세그먼트를 소거 대상으로 선택한다[7].
- K-분할 선택 알고리즘 : 2.2절에서 설명한 방법에 기반하여, 탐욕적 선택 알고리즘에서 사용하는 속성값을 세그먼트 속성값으로 채택하여 분할을 구성한다. 각 분할은 정수값으로 표현된 대표 속성값을 부여한다. 선택은 비어 있지 않은 최대 대표 속성값을 가지는 분할에 속하는 세그먼트 중에서 순서에 의한 무작위 선택과 최대 속성값에 따른 선택을 각기 알고리즘으로 구현하였다.

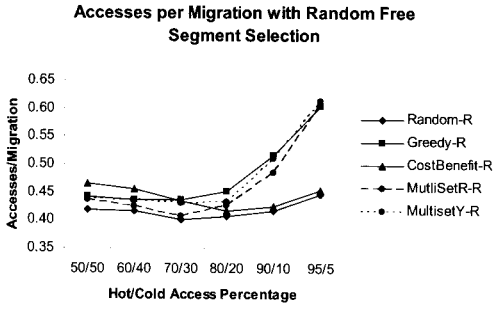
자유 세그먼트 선택 방법론은 유효한 컨텐츠를 포함하지 않은 세그먼트 중에서 순서에 대한 무작위 선택과 최소 세그먼트 소거 기반 선택을 구현하였다. <표 4>는 4개의 소거 대상 세그먼트 선택 알고리즘의 실험 결과를 보여 주기 위한 범례를 정리한 것으로, 각 범례의 마지막 영문자 X는 R과 Y로 그림에서 분류된다. R은 자유 세그먼트 선택 시 무작위 기반으로 수행한 것이고, Y는 최소 세그먼트 소거 횟수에 기반하여 선택을 한 것을 표현한다.

3.3 성능 비교 실험 결과 분석

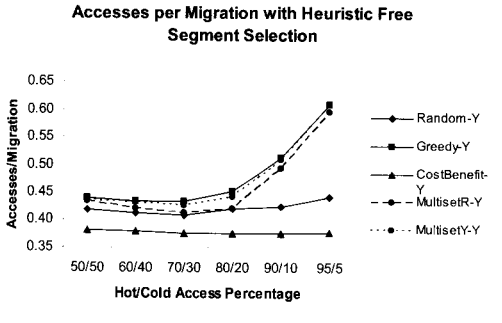
플래시 메모리 시스템의 성능을 분석하기 위한 측정치로는 한 블록 갱신 당 지원 접근수를 수집하였으며, 균등소거 성능을

측정하기 위해서는 세그먼트 소거 횟수에 대한 표준편차를 수집하였다. (그림 2)는 자유 세그먼트 선택을 무작위 기반으로 수행했을 때 한 개의 블록 갱신 당 지원 접근수를 보여 주고 있으며, (그림 3)은 최소 세그먼트 소거 횟수에 기반으로 수행했을 때 한 개의 블록 갱신 당 지원 접근수를 보여 주고 있다. 예시되어 있는 것과 같이 두 경우 모두 탐욕적 방법과 K-분할 기반 정책이 다른 방법들보다 우수한 성능을 보이고 있다.

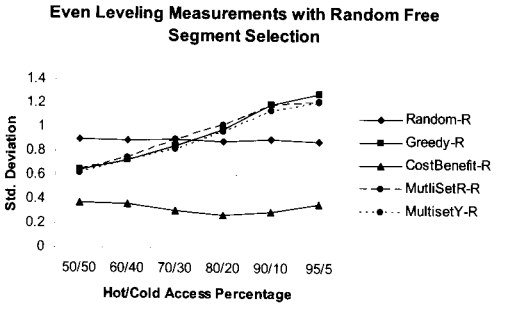
(그림 4)는 레벨링 효과를 보여주고 있는데, 사용된 방법 중에서 비용/편익 기반 세그먼트 선택 알고리즘은 무작위 자유 세그먼트 선택을 할 경우에도 우수한 성능을 보여주고 있다. 이는 비용/편익 기반 세그먼트 선택 알고리즘이 세그먼트를 선택할 때 균등한 소거를 고려하기 때문이다. 같은 이유로 자유 세그먼트 선택에서도 유사한 방법을 사용할 때 현저하게 레벨링 효과가 개선되는 것으로 나타났다.



(그림 2) 블록 갱신 당 접근수(무작위 자유 세그먼트 선택)



(그림 3) 블록 갱신 당 접근수(최소 세그먼트 소거 기반 자유 세그먼트 선택)



(그림 4) 레벨링(무작위 자유 세그먼트 선택)

3.4 최적 K값 실험

K-분할 플래시 메모리 관리는 K값에 의해 그 성능에 영향을 미칠 수 있다. 따라서, 최적 K값을 결정하기 위하여 앞의 실험에서 사용한 시스템을 이용하여 K값을 변화시키면서 블록 갱신 당 접근수와 레벨링을 측정하였다. 접근 집약성은 95%로 설정하여 대부분의 플래시 관리 알고리즘이 높은 성능을 보이는 환경을 채택하였다. 아울러, 소거 대상 세그먼트의 선택과 자유 세그먼트의 선택 방법에 대하여, 무작위 선택과 최소 소거 세그먼트 선택에 대한 모든 조합을 실험하였다. 사용된 조합은 <표 5>에서 종합해서 정리를 하였다.

K값의 집약성과 메모리 크기에 대한 민감도를 검증하기 위하여, 여러 집약성과 메모리 크기에 대해서도 실험을 수행하였다.

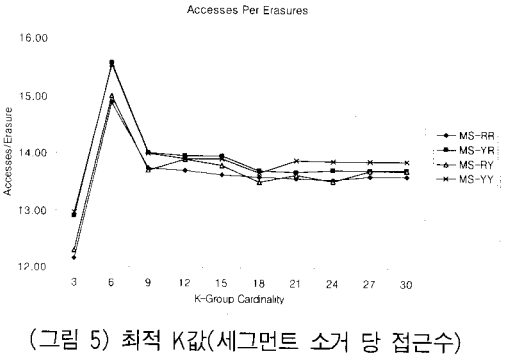
3.5 최적 K 값 실험 결과 분석

K-분할 플래시 메모리 관리에서 성능에 영향을 미치는 K값을 3에서 30까지의 범위에서 증가치를 3으로 사용하였다. 실험에서 한 세그먼트의 블록수는 32개로 설정하였다.

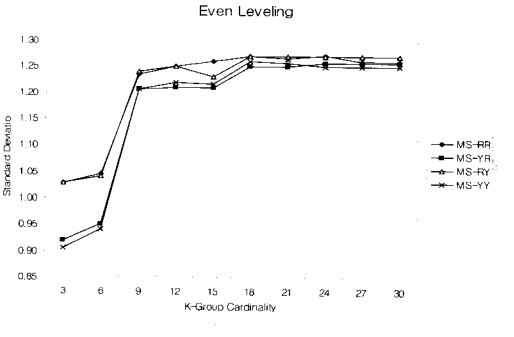
세그먼트 소거 당 접근수는 (그림 5)에서 예시되었다. (그림 5)에서 보여지고 있는 것처럼 6개의 분할을 사용할 때 K-분할 플래시 메모리 관리가 최적의 성능을 보이고 있음을 알 수 있다. (그림 6)은 동일한 실험 환경에서 구한 소거 횟수에 대한

〈표 5〉 최적K 값 실험 결과 범례

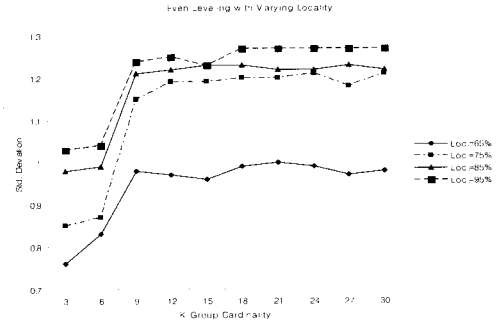
범례	소거 세그먼트 선택	자유 세그먼트 선택
MS-RR	무작위	무작위
MS-YR	최소 소거	무작위
MS-RY	무작위	최소 소거
MS-YY	최소 소거	최소 소거



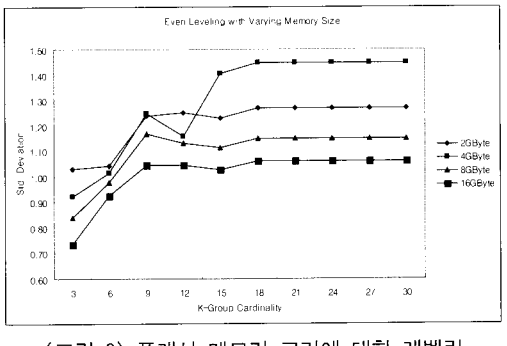
(그림 5) 최적 K값(세그먼트 소거 당 접근수)



(그림 6) K값 변화에 따른 레벨링



(그림 7) 다수의 집약성에 대한 레벨링



(그림 8) 플래시 메모리 크기에 대한 레벨링

표준편차를 통해 레벨링 효과를 나타내고 있다. (그림 6)은 6개의 분할을 사용하였을 때 최적의 균등 소거가 이루어짐을 보이고 있다.

(그림 7)은 집약성에 따른 민감도를 알아보기 위하여 집약성을 65%, 75%, 85% 그리고 95%로 각기 설정하여, K값을 변화시키면서 수집한 결과를 보여주고 있다. 여기서 알 수 있는 것은 같은 메모리 크기와 같은 속성값을 사용한다고 가정할 때, K-분할 플래시 메모리 관리에 유사한 최적값을 사용할 수 있다는 점이다. 또한, 일반적으로 시스템의 성능에 병목현상을 일으키는 시점을 75% 이상의 집약성을 나타내는 환경이라는 것을 고려할 때, 레벨링은 시스템 환경에서 유사하게 악화되는 것을 관찰할 수 있다.

(그림 8)은 서로 다른 메모리 크기에 대한 최적값을 보여주고 있다. 메모리의 크기가 다른 경우에도 K그룹의 최적값은 비슷한 영역에서 나타나고 있으며, 결과적으로 최적값은 메모리 크기에 별로 영향을 받지 않은 것으로 보인다. 동일한 세그먼트 구성과 속성값을 채택하여 K-분할 정책을 적용할 경우 동일한 대표 속성값을 가지는 분할에 속하는 세그먼트가 많아지게 되고, 그 결과로 레벨링의 성능이 대체로 향상되는 경향을 (그림 8)에서 볼 수 있다.

4. 결 론

이 논문은 K 개의 집합을 이용하여 비용과 효율면에서 우수한 플래시 메모리 관리 알고리즘을 소거 대상 세그먼트 선택과 자유 세그먼트 선택이라는 측면에서 고찰하였다. 제안된 K-분할 플래시 메모리 관리는 소거 대상 세그먼트 선택에 있어서 그 비용이 낮은 반면에 성능은 기존의 방법과 유사하거나 보다 우수하다는 것을 실험을 통해 검증했다. 또한 자유 세그먼트를 선택함에 있어, 균등 레벨링을 고려한 방법은 소거 대상 세그먼트 선택을 할 때 균등 레벨링과 함께 사용되었을 때 현저히 성능을 개선 시킬 수 있음을 실험 결과 알 수 있었다. K-분할 플래시 메모리 관리는 시스템 매개 변수로서 사용되는 집합의 수를 변화시킬 수 있다. 따라서, 최적 집합의 개수를 결정하는 것은 전체 플래시 메모리 관리에 중요한 과제이다. 이 최적값을 실험을 통해 추정한 결과 한 세그먼트 당 블록수의 15%에서 19% 사이의 값을 K 값으로 설정하는 것이 최적의 시스템을 구성하는 것으로 나타났다.

참 고 문 헌

[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, pp.837-863, 2004.

[2] Mei-Ling Chiang and Ruei-Chuan Chang. "Cleaning Policies in Mobile Computers Using Flash Memory." Journal of Systems and Software, Vol.48, No.3, pp.213-231, 1999.

[3] 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, pp.267-290, 1999.

[4] Fred Douglis et al. "Storage Alternatives for Mobile Computers," In OSDI, November 14-17, Monterey, California, USA.

[5] Joshua B. Fryman et al. "Energy-efficient Network Memory for Ubiquitous Devices.", IEEE Micro, Vol.23, No.5, pp.60-70, 2003.

[6] 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.

[7] Atsuo Kawaguchi , Shingo Nishioka and Hiroshi Motoda. "A Flash-Memory Based File System." In USENIX Winter, pp.155-164, 1995.

[8] Han-joon Kim and Sang-goo Lee. "An Effective Flash Memory Manager for Reliable Flash Memory Space Management." IEICE Transaction on Information and Systems, Vol.E85-D, No.6, pp.950-964, June, 2002.

[9] Brian Marsh, Fred Douglis and P. Krishman. "Flash Memory File Caching for Mobile Computers." In HICSS(1), pp.451-461, 1994.

[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, pp.26-52, 1992.

박 제 호

e-mail : dk_jhpark@dankook.ac.kr
1985년 서강대학교 전자계산학과(학사)
1993년 Polytechnic Univ.(공학석사)
2001년 Polytechnic Univ.(공학박사)
2003년~현재 단국대학교 컴퓨터과학과 교수
관심분야 : 데이터베이스 구조, 분산 데이터베이스, 데이터 저장 시스템, 웹 데이터 정보검색 시스템