쓰기 패턴 기반 플래시 메모리 지움 정책 설계

강성구, 노한영, 고광선, 엄영익
성균관대학교 정보통신공학부
{lived, kirseia, rilla91, yieom}@ece.skku.ac.kr

Design of Flash Memory Cleaning Policy based on Writing Patterns

Seong-Goo Kang, Han-Young Noh, Kwangsun Ko,
and Young Ik Eom
School of Information and Communication,
Sungkyunkwan University

요 약

본 논문에서는 최근에 각광받고 있는 저장장치인 플래시 메모리의 특징과 플래시 메모리의 단점인 지움 횟수 제한과 느린 지움 속도를 극복하는 지움 정책의 종류를 살펴보고 그 중에서 CAT 기법의 랜덤 쓰기에서의 지움 횟수 증가와 지움 평준화를 위한 비효율성이라는 단점을 개선한 지움 정책을 제시한다. 개선된 지움 정책은 CAT 기법에 쓰기 패턴을 기반한 가중치를 부여해 랜덤 쓰기에서의 성능 하락과 지움 평준화의 비효율성을 보완하는 것이다. 이 때 사용되는 사용자의 쓰기 패턴 파악은 유효한(valid) 데이터 블록과 유효하지 않은(invalid) 데이터 블록의 리스트를 만들어 그 시간 값의 평균을 이용한다.

1. 서론

최근 각광받고 있는 플래시 메모리는 저전력, 가벼운 무게를 필요로 하는 다양한 컴퓨터 장비에 사용이 되고 있는데 이는 다음과 같은 장점이 있기 때문이다. 첫째, 플래시 메모리는 기존의 디스크와 같은 회전식 자기 매체가 아니라 전자적으로 읽고, 쓰는 메모리이기 때문에 빠른 접근 속도를 가지므로 높은 성능을 제공한다. 둘째, 플래시 메모리는 외부 충격에 강하고 비휘발성이다. 셋째, 저전력 구동이 가능하고 크기가 작다[1][2].

하지만 이를 저장 장치로 활용하기 위한 파일 시스템의 구현에서 몇 가지 설계상 제약이 있다. 대표적으로 이미 데이터가 저장되어있는 곳에 새로운 데이터를 쓰기 위해서는 메모리의 지움(cleaning) 과정이 반드시 선행되어야 한다는 점과 지움 과정이 세그먼트 단위로 이루어져야 하는 점 때문에 원하지 않는 데이터도 지워야 하는 단점이 있다. 또한 지움 과정의 경우, 지움 횟수가 제한되어있으며 지우는 시간 또한 읽기나 쓰기에 비해 상당히 길다[1][2].

이러한 제약을 극복하고 플래시 전용 파일 시스템을 설계하기 위해서는 기존의 파일시스템에 대한 이해와 함께 플래시 메모리의 특징을 고려해야 한다.

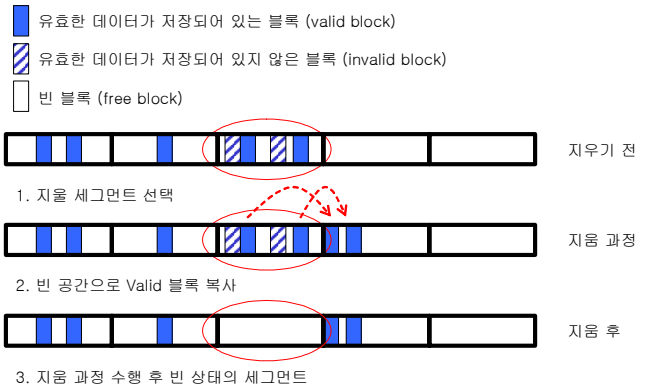
본 논문은 플래시 메모리에서 사용될 파일 시스템에서 사용되는 지움 정책들을 소개하고 그 특징들에 대해 고찰한다. 그리고 그 중에서 CAT(Cost Age Times) 기법에서 랜덤 쓰기의 단점을 쓰기 패턴을 반영하여 개선한 기법을 제안한다.

2. 기존 연구

플래시 메모리를 저장 매체로 사용하기 위해서는 세 가지 문제점을 고려해야 한다. 첫 번째로 데이터를 한 번 쓴 영역에는 지움 과정을 수행하기 전에는 다시 쓸 수 없으며, 두 번째로 지우는 속도가 읽는 속도 및 쓰기 속도에 비해 매우 길다는 점이다. 세 번째로 플래시 메모리를 지울 수 있는 횟수가 상온에서 약 10만회 정도로 정해져 있고, 지우는 방법도 지움 단위(erase unit), 지움 블록(erase block) 또는 세그먼트라고 하는 일정 크기로 이루어진다는 것이다. 이러한 플래시 메모리의 문제점을 극복하기 위해 여러 가지 방법이 제시 되고 있는데, 이 중에서 플래시 메모리의 가장 큰 제약인 지우는 속도가 느리고 지우는 횟수가 정해져 있다는 문제점을 극복하기 위한 지움 정책이 가장 큰 쟁점이다[2].

플래시 메모리는 세그먼트와 블록의 구조를 가지고 있다. 세그먼트는 지움 단위이고, 블록은 쓰기의

단위이다. 한 개의 세그먼트는 여러 개의 블록으로 구성된다. 블록 수정이 필요하면 플래시 메모리 특성상 지움 과정이 수행 된 후 새로 기록되어야 하므로 세그먼트에 대해 지움 과정이 수행된다. 그렇게 되면 세그먼트 내에 수정이 불필요한 블록들까지 지움 과정이 수행이 되기 때문에 이는 비효율적이다. 따라서 수정이 필요한 블록에는 invalid 플래그를 표시한 후 새로운 블록에 자료를 쓰는 방식으로 갱신이 이루어진다. 공간이 부족해지면 지움 과정을 거치게 되는데 그림 1처럼 3단계의 과정을 거쳐서 공간을 확보하게 된다. 이것이 지움 정책이다.



(그림 1) 세그먼트의 지움 과정

1단계는 지우기 전에 지우기 위한 세그먼트를 선택한다. 이 때 지울 세그먼트를 어떻게 선정하느냐에 따라 지움 정책이 정해진다. 2단계는 지울 세그먼트를 정한 후, 해당 세그먼트의 valid 블록을 다른 세그먼트로 복사한다. 마지막 3단계에서는 지우고 난 후에는 해당 세그먼트는 데이터의 쓰기가 가능한 상태가 된다.

2.1 Greedy 기법

Greedy 기법은 세그먼트 내에서 valid 블록이 가장 적은 세그먼트를 지울 대상으로 선정하는 방식이다. 이 기법은 valid 블록이 가장 적은 세그먼트를 선택하기 때문에 valid 블록을 다른 세그먼트로 옮기는 비용이 적게 들고, 별도의 연산 작업이 필요하지 않다는 장점이 있다. 하지만, 시간적, 공간적으로 특정 세그먼트에 수정이 집중되는 지역성 쓰기일 경우 비용이 커진다는 단점이 있으며, 또한 각 세그먼트의 지움 횟수를 균일하게 해주는 지움 평준화(wear-leveling)가 전혀 고려되지 않는다[2][3].

2.2 Cost-Benefit 기법

Cost-Benefit 기법은 비용(블록 삭제 비용 + valid 블록을 다른 곳으로 옮기는 비용)에 비해 이익이 많이 남는 세그먼트를 선택해서 삭제하는 지움 정책으로, 비용 대비 이익 계산 값은 식 1로 결정한다. 단, 비용은 삭제 및 이동에 필요한 시간이고 이익은 지움 과정 후 추가로 얻는 용량이다.

$$\frac{Benefit}{Cost} = \frac{Age \times (1-u)}{2u} \quad \cdots \text{ 식 (1)}$$

u는 해당 세그먼트 내 valid 블록의 비율이고, $(1-u)$는 invalid 블록의 비율이다. $2u$는 세그먼트의 valid 블록을 다른 세그먼트로 옮길 때 발생하는 읽고 쓰는 valid 블록의 비율이다. Age는 세그먼트에 블록이 기록된 후 경과 시간이다.

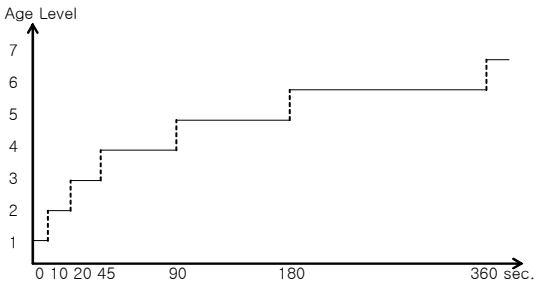
Cost-Benefit 기법에서는 Age이 있어서 지움 평준화를 추구하지만, 이는 세그먼트의 누적 지움 횟수가 아닌 세그먼트에 기록된 시간이므로 근본적으로 지움 횟수에 대한 완벽한 지움 평준화는 되지 않는다[2][3].

2.3 CAT 기법

CAT 기법은 식 2로 각 세그먼트 마다 값을 계산하여 그 값이 가장 작은 세그먼트를 지움 정책의 대상으로 삼는 기법으로 세그먼트 내 valid 블록의 비율과 Age, 그리고 세그먼트의 누적 지움 횟수를 이용한 기법이다.

$$\frac{u}{(1-u)} \times \frac{1}{Age} \times NumberOfCleaning \quad \cdots \text{ 식 (2)}$$

지움의 비용은 $\frac{u}{(1-u)}$로 표현이 되고, u와 $(1-u)$는 Cost-Benefit 기법과 같은 의미로 invalid 블록에 대한 valid 블록의 비율이다. Age는 세그먼트가 기록된 이후의 경과 시간을 말하며, Age은 그림 2의 사전에 정의된 Age 변환 함수에 의해 결정 된다. 마지막으로, NumberOfCleaning은 세그먼트가 지워진 누적 횟수를 말한다[4].



(그림 2) Age 변환 함수

CAT 기법의 기본 원리는 수정이 빈번하게 일어나는 데이터(hot 데이터)와 수정이 거의 없는 데이터(cold 데이터)의 분리이다. 만약, hot 데이터와 cold 데이터가 같은 세그먼트에 기록된다면 hot 데이터는 곧 invalid 블록이 되어버리고, 한 세그먼트 안에 invalid 블록과 valid 블록이 같이 존재하게 된다. 이런 상황에서 공간이 부족하다면 지움 정책이 수행이 되는데, Greedy 기법에서는 valid 블록을 다른 곳으로 옮기게 되고 이는 오랫동안 수정이 없을 cold 데이터를 옮기는 것으로 효율성이 떨어지기 때문에, CAT 기법에서는 Age이 큰 세그먼트에 가중치를 낮게 두어 지움 정책을 수행할 때 세그먼트 선

택 우선순위를 낮게 한다. 그리고 Age이 작은 세그먼트는 우선순위를 높게 두어 결과적으로 cold 데이터와 hot 데이터를 분리 한다[4]. 그 결과 수정이나 삭제가 빈번한 지역성 쓰기일 때, CAT 기법은 cold 데이터의 이동, 지움 횟수가 줄어들어 우수한 성능을 보여준다.

3. 개선된 CAT 기법

3.1 CAT 기법의 문제점

CAT 기법은 기존의 지움 정책들에 비해서 valid 블록의 비율, Age, 지움 횟수를 모두 고려해서 속도와 지움 평준화를 만족시켜준다. 표 1에서 보이는 바와 같이 사용자 쓰기 패턴이 순차 쓰기이거나 지역성이 있는 쓰기일 경우는 지움 횟수나 복사 횟수가 Greedy 기법과 Cost-Benefit 기법보다 적기 때문에 더 나은 성능을 보여주지만, 랜덤 쓰기일 경우에는 Greedy 기법 보다 지움 횟수와 복사 횟수가 더 많고 평균 처리량이 더 적어 효율성면에서 오히려 더 낮은 성능을 보여준다[4].

본 논문에서 정의하는 순차 쓰기는 새로운 데이터가 기록되는 것으로 빈 블록에 데이터가 기록되어 valid 블록이 되는 작업이다. 반대로 지역성 쓰기는 기존의 데이터를 수정하는 작업으로 valid 블록이 invalid 블록으로 바뀌고 빈 블록에 수정된 데이터가 기록되는 작업이다. 랜덤 쓰기는 순차 쓰기와 지역성 쓰기가 섞여서 나타나는 특성을 보인다.

<표 1> 쓰기 특성에 따른 지움 정책의 성능

		순차	지역성	랜덤
Greedy	지움횟수	1,567	8,827	7,103
	복사횟수	0	225,068	171,624
	평균 처리량	134,441	22,680	27,989
Cost-Benefit	지움횟수	1,568	5,733	7,265
	복사횟수	0	129,142	176,634
	평균 처리량	134,334	34,980	27,118
CAT	지움횟수	1,568	4,138	7,241
	복사횟수	0	79,705	175,891
	평균 처리량	133,950	44,263	25,055

(평균 처리량의 단위는 bytes/s)

또한, 식 2의 NumberOfCleaning으로 지움 평준화를 추구하는데, 세그먼트 내에 블록들 간의 지움 횟수의 차가 작을 때와 같이 지움 평준화가 중요하지 않을 때도 이 NumberOfCleaning의 중요성은 항상 같은 가중치를 갖기 때문에 효율성을 무시한 세그먼트 선택이 있을 수 있다.

3.2 쓰기 패턴을 반영한 CAT 지움 정책

앞서 살펴본 CAT 기법의 랜덤 쓰기 단점에 대한 해결책을 Greedy 기법에서 찾아보았다. 랜덤 쓰기 일 경우에는 Greedy 기법이 효율이 더 좋기 때문에, 이를 응용해서 랜덤 쓰기일 경우에는 Greedy 기법을 응용한 방식으로 지움 정책을 수행하고, 순차 쓰기나 지역성 쓰기일 때는 기존의 CAT 기법과 같은 지움 정책을 수행한다. 또한, 식 2의 NumberOfCleaning의 중요도가 항상 같지 않다는 점을 이용해 NumberOfCleaning에 가중치를 두었다. 이를 반영해 CAT 기법의 원래의 식 2를 수정하여 식 3으로 변형한다.

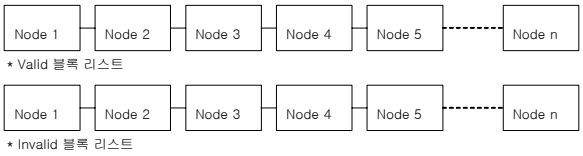
$$[\alpha \times \frac{u}{1-u}] \times [\beta \times \frac{1}{Age}] \times [\gamma \times NmberOfCleaning] \quad \cdots \text{ 식(3)}$$

식 3에서 α는 invalid 블록에 대한 valid 블록 비율 가중치, β는 Age에 대한 가중치, γ는 지움 횟수에 대한 가중치로 정의하며, 사용자의 쓰기 패턴에 따라 각 가중치를 변화시켜 최적의 지움 정책을 구현한다. 랜덤 쓰기의 경우에는 Greedy 기법과 같이 Age보다 세그먼트 내의 valid 블록의 비율이 중요하므로 α는 크게 잡고, β는 작게 잡아준다. 또한 순차 쓰기나 지역성을 보이는 쓰기 일 때에는 기존의 CAT 기법과 같이 Age이 중요하기 때문에 Age의 가중치인 β를 크게 잡고, α는 작게 잡는다.

γ는 전체 세그먼트의 지움 횟수의 차이로 정해지는 값이다. 전체적인 지움 평준화가 잘 되어있다면 NumberOfCleaning은 큰 의미를 갖지 않는다. 하지만, 평준화가 잘 안되어 있다면 이 값은 매우 중요한 값이기 때문에 큰 값을 갖도록 한다.

3.3 사용자의 쓰기 패턴 파악

남은 문제는 사용자의 쓰기 패턴을 어떻게 파악하여 순차 쓰기, 랜덤 쓰기, 혹은 지역성 쓰기 여부를 파악하는 것이다. 본 논문에서는 그림 3과 같은 두 개의 리스트를 만들어 관리하는 것으로 쓰기 패턴을 파악하고자 한다.



(그림 3) 최근 valid, invalid 블록 리스트

각 리스트는 블록이 쓰기 혹은 지우기 작업이 수행될 때마다 갱신되며 각각 n개의 노드로 구성되어 있다. 새로운 노드가 리스트에 들어올 때는 리스트의 맨 앞으로 들어오게 된다. 그리고 리스트에 노드가 n개일 때 새로운 노드가 들어오면 마지막 노드를 제거한다. 각 노드는 리스트에 삽입되었을 때의 시간을 가지고 있다.

valid 블록 리스트에는 빈 블록에 쓰기 작업을 수행한 순간의 시간이 차례로 기록되며, invalid 블록 리스트 역시 마찬가지로 invalid 블록이 된 순간의 시간이 차례로 기록된다.

예를 들어 빈 블록이 많이 있고 단순한 순차쓰기를 하는 경우에는 valid 블록 리스트에만 노드가 추가되고 invalid 블록 리스트에는 노드가 추가되지 않는다. 반대로 사용자가 어떤 데이터를 수정하는 작업을 하게 되면 기존의 valid 블록이 invalid 블록이 되고 빈 블록에 새로운 데이터를 쓰게 되므로 양 쪽에 노드가 하나씩 추가된다. 만약 빈 블록이 부족하

여 지움 과정이 수행되어 valid 블록의 이동이 생기더라도 사용자가 의도한 작업이 아니므로 리스트에 추가적으로 노드가 삽입되지 않는다.

valid 리스트와 invalid 리스트의 각각 노드들의 평균 삽입 시간을 Avg_{valid}와 $Avg_{invalid}$라고 한다. 두 값의 차이를 이용하여 사용자의 쓰기 패턴을 파악하면 표 2와 같이 나온다.

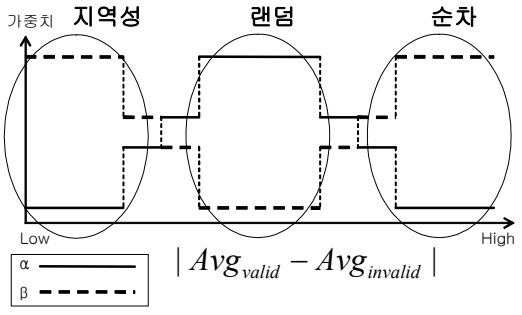
<표 2> 노드들의 평균 삽입 시간을 이용한 쓰기 패턴 파악

	지역성	랜덤	순차
$\lvert Avg_{valid} - Avg_{invalid} \rvert$	Low	Middle	High

Avg_{valid}가 $Avg_{invalid}$보다 크다면, valid 블록이 삽입된 시간은 invalid 블록들이 삽입된 시간에 비해 최근이라는 것이다. 즉, 빈 블록에 새로 기록된 데이터는 많은데 지워진 데이터 혹은 수정된 데이터는 적다는 뜻으로 이는 순차 쓰기를 의미한다고 할 수 있다. 또한 Avg_{valid}와 $Avg_{invalid}$의 차이가 작다면 두 개의 리스트에 비슷한 시기에 노드가 추가되었음을 의미하므로 데이터의 수정이 많이 일어났다고 볼 수 있다. 이 경우는 지역성이 있는 쓰기라고 할 수 있다.

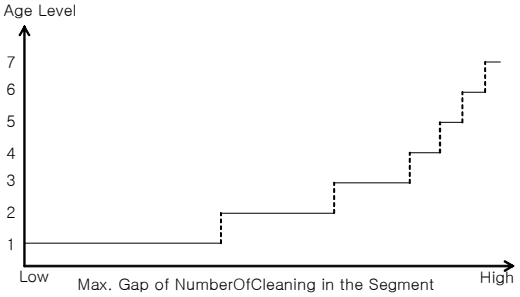
즉, Avg_{valid}와 $Avg_{invalid}$의 차이가 매우 크다면 이는 순차 쓰기, 값이 비슷하다면 지역성을 보이는 쓰기, 그 중간정도는 순차 쓰기와 지역성을 보이는 쓰기가 모두 일어났다는 것을 의미하고 이를 랜덤 쓰기로 가정한다.

가중치 α와 β에 대한 값은 시뮬레이션을 통해서 최적의 값을 정해야 하지만, 여기서는 앞에서 가정한 상황들을 바탕으로 가중치를 임의적으로 설정하여 그림 4로 표현하였다. X축은 Avg_{valid}와 $Avg_{invalid}$의 차이를 나타내고 있다. 차이가 커져가는 것에 맞춰 가중치가 적용되도록 하였다.



(그림 4) α와 β의 가중치 함수

NumberOfCleaning에 대한 가중치 γ 역시 계산이 필요하다. 가장 많은 지움 횟수를 가진 세그먼트와 가장 적은 지움 횟수를 가진 세그먼트의 NumberOfCleaning의 차가 커지면 지움 평준화가 잘 안되고 있다는 뜻이므로 γ를 크게 하여 지움 평준화에 가중치를 준다. 지움 횟수 차이가 작으면 지움 평준화가 잘 되고 있다는 뜻이므로 γ의 비중은 작아진다. 앞에서의 그림 2와 같이 표현하면 그림 5의 함수처럼 표현할 수 있다.



(그림 5) γ의 가중치 함수

Avg_{valid}와 $Avg_{invalid}$를 이용하여 그림 4와 그림 5의 가중치 α, β, γ에 대한 가중치 함수를 도출하여 개선된 CAT 기법의 식 3에 적용할 수 있도록 하였다. 계산된 가중치로 인해 지역성이나 순차 쓰기에서 뿐만 아니라 랜덤 쓰기에서도 높은 성능을 낼 수 있게 된다.

4. 결론

본 논문에서는 플래시 메모리의 현황과 특징에 대해 알아보고, 플래시 메모리의 특징에 따른 단점을 극복하기 위한 지움 정책을 살펴보았다. 그리고 지움 정책 중에서 CAT 기법에서의 랜덤 쓰기에서의 단점을 사용자의 쓰기 패턴을 이용해 원래의 식에 가중치를 적용한 새로운 식을 제안하였다. 사용자의 쓰기 패턴은 valid와 invalid 블록의 리스트를 구성하여 가중치 함수를 도출하여 파악하였다. 향후 가중치 α, β, γ의 변화에 따른 성능 그래프를 시뮬레이션을 통해 구하는 것과 세그먼트 선택 알고리즘과 이와 관련된 지움 정책의 시작 타이밍, 그리고 얼마만큼의 세그먼트에 대해 지움 정책을 수행할지에 대해서 연구할 계획이다.

참고문헌

[1] 이경남, 빅인규, “Flash Memory 파일관리에 관한 연구,” 대한전자공학회 99 추계종합학술대회 논문집, 1999권, pp. 993-996

[2] 김정기, 박승민, 김채규, “임베디드 플래시 파일 시스템을 위한 순위별 지움 정책,” 정보처리학회 논문집, 제 9-A권 4호, pp.399-404, 2002. 12.

[3] 민용기, 박승규, “플래시 메모리를 사용하는 이동컴퓨터에서 클리닝 정책,” 학술대회 논문집, Vol. 21, No. 2, 1998, pp. 495-498.

[4] Mei-Ling Chiang, Paul C. H. Lee, Ruei-Chuan. Chang, “Cleaning policies in mobile computers using flash memory,” The Journal of Systems and Software, Vol. 48, Issue 3, Nov. 1999, pp. 213-231.