플래시메모리를 사용하는 이동컴퓨터에서 클리닝 정책

민용기, 박승규
아주대학교 정보 및 컴퓨터공학부

A Cleaning Policy for Mobile Computers using Flash Memory

Yong-Ki Min, Seung-Kyu Park
Division of Information and Computer Engineering, Ajou University
Tel : 0331-219-2532, E-mail : sparky@madang.ajou.ac.kr

Abstract

Mobile computers have restrictions for size, weight, and power consumption that are different from traditional workstations. Storage device must be smaller, lighter. Low power consumed storage devices are needed. At the present time, flash memory device is a reasonable candidate for such device. But flash memory has drawbacks such as bulk erase operation and slow program time. This causes of worse average write performances. This paper suggests a storage method which improves write performance.

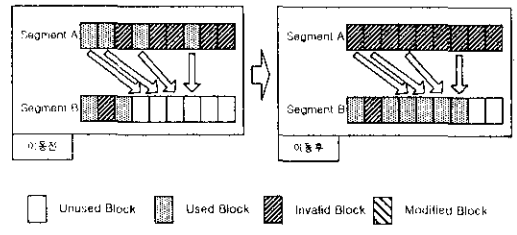
1. 플래시메모리의 특성

플래시메모리는 그 구조가 간단하여서 집적도가 높고 가격이 싸다. 그리고 DRAM과 비슷한 성능을 나타내는데, 읽기(read)가 85ns정도이고, 쓰기(write)가 4-10 micro second정도이다. 전력소모의 경우 DRAM 경우 약 1A인데 반해 플래시메모리는 동작상태에서 30-50mA 이다. 이런 장점으로 앞으로 이동컴퓨터 분야에 적용될 가능성이 크다[1]. 그러나 플래시메모리는 데이터의 덮어쓰기가 불가능하다. 즉 같은 주소에 수정된 내용을 바로 다시 쓰는 것이 불가능하다. 만약 같은 자리에 다시 쓰고자 한다면 그 블록이 포함된 세그먼트 전체를 소거한 후(erase operation) 다시 쓸 수 있다. 여기서 세그먼트는 여러 개의 페이지들이 모여 하나의 세그먼트를 이루게 되는데 플래시메모리의 종류에 따라 8KB, 16KB, 64KB, 128KB등으로 다양하다. 소거작업은 세그먼트 당 약 800ms정도의 시간이 소요된다. 그리고 각 세그먼트 당 소거작업의 수의 한계가 있어서 무한정 소거작업을 할 수 없도록 되어 있다. 제조 업체에서는 한 세그먼트 당 소거작업의 횟수를 보증한다[2][5]. 따라서 그 회수 이상 소거작업을 하게되면 오동작을 일으킬 수 있게 된다. 디스크 같은 경우 수명이 다하면 읽기와 쓰기 모두 불가능해 지지만 플래시메모리의 경우는 쓰기는 불가능하여도 읽기는 가능하다.

* 본 연구는 한국과학재단 특정 기초연구비 (961-0100-001-2)지원으로 수행되었으며 지원에 감사드립니다.

2. 세그먼트 클리닝

앞서 언급한데로 플래시메모리는 같은 주소에 덮어쓰기가 불가능하다. 따라서 어떤 블록을 수정하려고 한다면 먼저 있던 블록을 무효화시키고 쓰기가 가능한 세그먼트를 찾아 수정된 내용을 다시 기록하여야 한다. 쓰기와 수정이 계속된다면 플래시메모리에는 더 이상 데이터를 기록할 공간이 없어지게 되는데 이는 일반적인 디스크나 DRAM이 다 차는 경우와는 조금 틀리다. 디스크나 DRAM같은 경우 일반적으로 다 찼다는 것은 유효한 데이터들이 다 차는 경우인데 반해 플래시메모리의 경우는 무효한 데이터와 유효한 데이터들로 가득 차 더 이상 기록할 공간이 없는 경우를 말한다. 이러한 상태가 오기 전에 세그먼트 클리닝작업을 통해서 쓸 수 있는 플래시메모리의 공간을 항상 확보하고 있어야 한다.



<그림 1> 클리닝 작업

세그먼트 클리닝 작업은 <그림 1>과 같이 유효한 블록들을 쓰기가 가능한 다른 블록으로 옮겨서 세그먼트 전체를 무효한 블록으로 만드는 것이다. 이렇게 세그먼트 클리닝이 된 세그먼트들은 소거작업(erase operation)을 통해서 다시 쓰기 가능한 블록을 가지는 세그먼트가 된다.

3. 세그먼트 클리닝 비용

일반적인 디스크에서는 이러한 세그먼트 클리닝 작업이 필요하지 않지만 플래시메모리 구조에서는 세그먼트

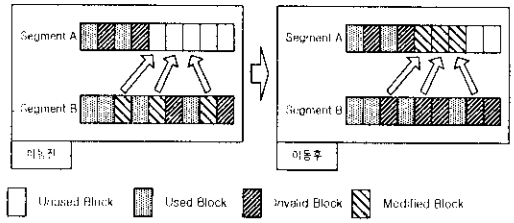
클리닝 작업이 필요하게 된다. 세그먼트 클리닝작업을 하는데 소모되는 작업을 측정하기 위해서는 세그먼트 클리닝 비용을 계산해야 한다. 클리닝 비용은 세그먼트 안에 옮겨야 할 블록의 수로 결정된다. 옮겨야할 블록의 수가 많으면 이동하는데 드는 비용이 늘어나게 되므로 클리닝 비용은 증가하게 되고 반대로 옮겨야할 블록의 수가 작으면 클리닝 비용은 감소한다. 그리고 전체 메모리 중에 유효한 데이터들이 많아 그 비율이 높아지게 되면 잦은 세그먼트 클리닝을 요구하게 되므로 또한 클리닝 비용을 증가시키는 요인이 된다.

4. 방향성 클리닝 정책

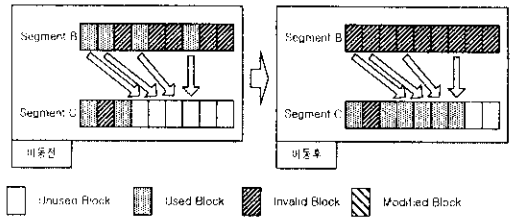
세그먼트 클리닝 비용은 시스템의 성능에 매우 큰 영향을 미치므로 세그먼트 클리닝 정책을 적용할 때 중요한 참고가 된다. 일반적으로 disk에서 읽기 보다는 쓰기가 더 많이 발생하여 더욱 그 중요성이 크다[3]. 비용을 낮추기 위해서 취할 수 있는 일반적인 방법은 greedy 방법이라고 하는 것이 있는데, 먼저 전체 세그먼트 중에서 무효한 블록이 많은 것을 찾아 선택하여 세그먼트 클리닝 작업을 하고 이 세그먼트는 다음 쓰기(write)작업 시에 가장 먼저 선택되어 채워진다. 이 방법은 먼저 쓰여진 것이 먼저 세그먼트 클리닝 될 확률이 높아 FIFO(first-in-first-out)의 성격을 가지고 있다. 이 방법은 구현이 간단하지만 클리닝 비용에서는 그다지 이득을 보지 못한다.

일반적으로 데이터들은 일정한 수정 빈도를 가지고 있지 않는다. 어떤 데이터 블록들은 한번 쓰여지면 읽기만 계속해서 수행하고 반대로 어떤 데이터 블록들은 수정작업이 빈번하게 일어난다. 수정이 빈번하게 일어나는 데이터 블록을 hot 블록, 수정작업이 거의 없는 블록을 cold 블록이라고 할 때, greedy 방법에서 hot 블록과 cold 블록이 결과적으로 균일하게 섞이게 되므로 블록의 이동이 일어나자 않아야 좋은 cold 블록들까지 이동이 일어나 결국 세그먼트 클리닝 비용이 많이 들게 된다. 뿐만 아니라 필요이상으로 블록의 이동이 일어나 결국 세그먼트 클리닝의 횟수가 늘어나 수명에도 좋지 않은 영향을 준다. 이 같은 점을 해결하기 위해서 hot 블록과 cold 블록을 분리해내는 작업이 필요하다. 실제로 유닉스의 예를 들면 읽기(read)작업보다 쓰기(write)작업이 조금 더 많고 데이터의 성격에 따라 어떤 데이터는 자주 쓰여지고 어떤 데이터는 덜 쓰여진다. 여기서 쓰기 작업이 읽기 작업보다 많은 이유는 실제적인 데이터의 쓰기 작업에 메타데이터를 포함되기 때문이다. 쓰기 작업이 빈번할수록 hot 블록과 cold 블록을 구분하여 세그먼트 클리닝 비용을 줄이는 것이 중요하다.

방향성 클리닝 정책에서는 hot 블록과 cold 블록을 구분해내고 또 그 작업을 간단히 하는 구조로 컴퓨터 연산 처리 능력이 약한 이동 컴퓨터에 적합하도록 한다. 실제로 hot 블록과 cold 블록을 구분하는 것은 앞으로 쓰기 작업을 예측하여 구분하는 것이지만 현실적으로 불가능하기 때문에 여기서는 수정작업이 일어나는 횟수를 이용해 구분하도록 한다. 즉 이전에 수정이 빈번하게 일어났던 블록은 hot 블록으로 가정하고 그렇지 않은 것은 cold 블록으로 가정한다. 수정 횟수를 기록하여 나중에 참고하는 것이 일반적인 방법이지만 이런 방법은 블록마다 저장공간을 따로 필요로 한다. 그리고 예전에 많이 수정되었던 블록이더라도 최근 얼마간은 수정되지 않은 경우 앞으로 수정이 빈번하지 않으리라 예상됨에도 불구하고 hot 블록으로 간주될 수 있다. 따라서 이 방법은 불합리하다. 방향성 클리닝 정책에서는 직접적으로 횟수를 기록하기보다는 세그먼트 번호로 hot 블록과 cold 블록을 구분하도록 한다.



<그림 3> 수정으로 인한 블록의 이동

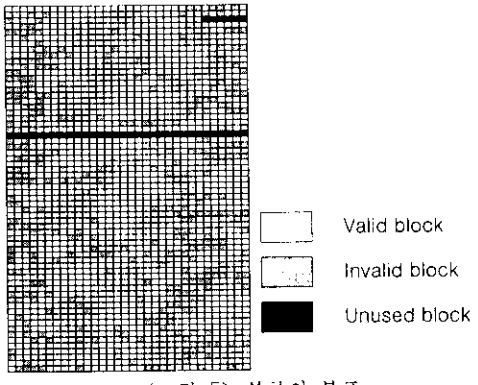


<그림 4> 클리닝 작업으로 인한 블록의 이동

<그림 3>과 <그림 4>는 세그먼트 B에 있는 hot 블록과 cold 블록이 분리되는 과정을 보여 주고 있다. 세그먼트 번호는 낮은 순서대로 A, B, C이다. 블록의 수정이 필요하여 데이터가 이동한다면 이것은 hot 데이터로 간주하고 쓰기 가능한 세그먼트 중에서 자신이 속한 세그먼트 번호 보다 낮은 세그먼트로 이동하는데 이때 자신이 속한 세그먼트와 가장 가까운 세그먼트를 선택하도록 한다. 세그먼트 클리닝 때에는 수정작업 때와는 반대로 쓰기 가능한 세그먼트 중에서 자신이 속한 세그먼트 번호 보다 높은 세그먼트로 이동하는데 이때 자신이 속한 세그먼트와 가장 가까운 세그먼트를 선택한다. 자신이 속한 세그먼트 번호가 최소이거나 최대일 경우에는 자신과 가장 가까운 쓰기 가능한 세그먼트를 선택한다. 그리고 수정의 경우 자신의 세그먼트 번호보다 낮은 세그먼트 중에서 쓰기 가능한 블록이 없는 경우와 반대로 클리닝 작업 때에는 클리닝되는 세그먼트 번호보다 높은 세그먼트 중에서 쓰기 가능한 블록이 없는 경우 역시 자신과 가장 가까운 쓰기 가능한 세그먼트를 선택하도록 한다.

블록의 수정과 세그먼트 클리닝 작업이 계속해서 이루어지게 되면 hot 블록들은 세그먼트 번호가 낮은 곳에 집중되게 되고 반대로 cold 블록들은 세그먼트 번호가 높은 곳에 집중되게 된다. 방향성 클리닝 정책에서는 hot 블록과 cold 블록을 구분해내기 위해서 별도의 연

산이 필요 없어 작업에 부하가 적게 걸리고 hot 블록이었다가 cold 블록으로 또는 cold 블록이었다가 hot 블록으로 변하는 경우에도 유동적으로 변화에 대처할 수 있다. 예를 들면 처음에는 hot 블록이어서 세그먼트 번호가 낮은 곳에 있던 블록이 수정 작업을 한동안 거치지 않게 되면 세그먼트 클리닝 작업을 계속해서 거치게 되어 cold 블록이 모이는 곳으로 이동하게 된다.



<그림 5> 블록의 분포

<그림 5>는 쓰기, 지우기, 세그먼트 클리닝작업이 어느 정도 이루어졌을 때 블록의 분포를 나타내고 있다. 세그먼트 번호가 낮을수록 데이터 블록의 수정이 빈번하게 일어나 무효한 블록의 수가 많이 분포하고 있고 세그먼트의 번호가 높을수록 유효한 블록의 수가 많아지고 있다. 이것은 hot 블록이 모여있는 세그먼트는 클리닝 비용을 낮추는 작용을 하게되고 반대로 cold 블록에서는 불필요한 이동을 억제하도록 하고 있다.

방향성 클리닝 정책을 사용하기 위해서는 세그먼트를 관리해주어야 하는데 <그림 6>은 세그먼트를 관리하는데 필요한 table의 구성을 나타낸다. 소거작업의 수는 현재 세그먼트가 몇 번 소거작업을 수행했는지를 나타낸다. 이것은 세그먼트 평준화를 위해서 사용되어진다. 세그먼트에 새로운 블록이나 수정된 블록을 추가하기 위해서는 세그먼트에 쓰기 가능한 블록이 얼마나 있는지, 또 그 위치가 얼마나 있는지를 알아야 한다. 이는 유효한 블록도 쓰기 가능한 블록의 경우와 마찬가지로 그 정보를 저장하도록 한다. 무효한 블록의 수는 쓰기 가능한 것과 유효한 블록의 조합으로 대신하도록 한다. 이 table은 수정이 빈번하게 일어나므로 쓰기 작업의 시간이 읽기에 비해 오래 걸리는 플래시메모리에 저장하기보다는 DRAM이나 SRAM에 저장하도록 한다. 또한 이동 컴퓨터의 전원 공급이 중단되었을 경우를 대비해 대기 전원을 별도로 공급하도록한다. 이 table은 전체 플래시메모리에 비해 상대적으로 작기 때문에 작은 크기의 DRAM이나 SRAM이라도 충분히 저장 가능하다.

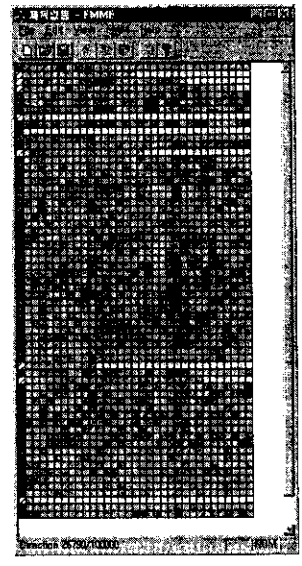
소거작업의 수	쓰기 가능한 블록의 수, 위치	유효한 블록의 수, 위치

<그림 6> 세그먼트 관리 table 구조

클리닝 정책에서 고려해야할 사항으로 세그먼트 평준화(wear leveling, cycle leveling)가 있다[4]. 플래시메모리는 세그먼트마다 수명이 있기 때문에 만약 어느 세그먼트만 집중적으로 소거작업이 일어나게 되면 전체 플래시메모리가 고르게 한계 수명까지 쓰는 것이 아니라 크기가 줄어드는 결과가 나오고 결과적으로 사용률이 증가하게되어 세그먼트 클리닝 작업이 빈번하게 발생하게 되고 이는 또 세그먼트의 수명에 영향을 주어 메모리 크기가 줄어드는 악순환을 거듭하게 된다. 방향성 클리닝 정책에서도 hot 블록이 집중되는 세그먼트가 다른 세그먼트에 비해 여러번 소거작업이 일어나는 구조를 가지고 있다. 그래서 이러한 문제점을 해결하기 위해 방향성 클리닝 정책에서는 일정 수의 작업이 일어난 후에 블록의 이동방향을 바꾸어 이 문제를 해결하도록 하였다. 즉 블록의 수정으로 인한 이동의 겨우 세그먼트 번호가 낮은 방향으로 이동하던 것을 주기적으로 바꾸어주어 세그먼트의 소거작업이 고르게 일어나도록 한다. 물론 세그먼트 클리닝 작업의 경우도 같이 바꾸어 주도록 하였다.

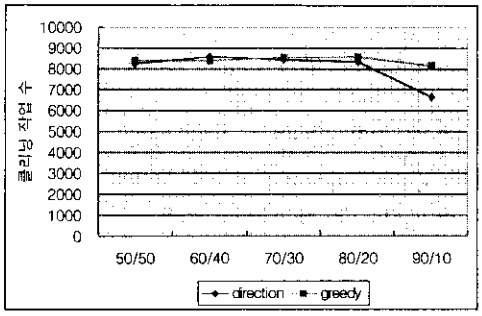
5. 시뮬레이션

시뮬레이션결과 쓰기 작업의 집중하는 정도에 따라 greedy 방법과 방향성 클리닝 정책을 비교하였다. 시뮬레이션은 <그림 7>은 시뮬레이션 작업화면이다. 전체 10만회중 2천여회를 넘었을 때의 메모리의 분포이다. 시뮬레이션 조건은 버퍼가 없는 상태에서 읽기작업을 제외하고 쓰기작업만을 10만회 수행하였다. 10만회중 무작위로 쓰기작업을 하여 수정작업과 쓰기가 이루어졌으며 메모리가 모두 차는 것을 방지하기 위하여 쓰기작업의 일부 작업을 지우기 작업으로 대치하였다. 전체 3회를 수행하여 그 값을 평균내었다. 전체 세그먼트의 수는 64개로 하였고 한 세그먼트 안의 블록의 수는 32개로 하였다.



<그림 7> 작업화면

<그림 8>은 시뮬레이션의 결과를 그래프로 보여주었다. 세로축은 세그먼트 마다 클리닝 작업이 수행된 수를 합한 수를 나타내 주고 가로축은 데이터의 집중도를 보여준다. 예를 들면 70/30은 70%의 쓰기작업이 30%의 데이터에 집중된다는 것을 나타내고 있다. <그림 7>에서 보여지듯이 클리닝작업의 수가 데이터의 집중도가 높아지는 80/20 정도에서부터 차이가 나기 시작해 90/10에서는 1484번의 클리닝을 덜하게 되었다.



<그림 8> 자료의 집중도에 따른 클리닝 작업의 수

<표 1>은 시뮬레이션한 결과를 표로 나타낸 것이다. 쓰기작업의 집중도가 높은 80/20에서부터 클리닝 작업이 수행될 때 이동하는 블록의 수가 greedy방법에 비해 방향성 클리닝 정책이 적어졌다. 따라서 90%의 쓰기작업이 10%의 데이터에 집중될 때 한번 클리닝을 하는데 이동해야하는 블록의 수가 greedy 정책에 비해 평균 2.12 블록 덜 이동하게되었다. 그리고 전체 클리닝 작업의 수도 감소하여 greedy작업에 비해 14.84%의 향상을 보게 되었다.

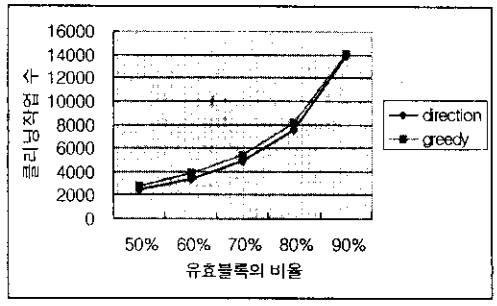
<표 1> 쓰기작업의 집중에 따른 클리닝 비용

	클리닝 회수		이동된 블록의 수		평균 클리닝 비용	
	direction	greedy	direction	greedy	direction	greedy
50/50	8230	8409	185456	190966	22.53	22.71
60/40	8620	8378	197619	189974	22.93	22.67
70/30	8443	8573	192064	196076	22.75	22.87
80/20	8331	8601	188411	197001	22.62	22.90
90/10	6664	8147	135173	182594	20.28	22.40

<표 2>는 플래시메모리 내에 유효한 블록의 비율에 따라 클리닝비용이 변화하는 것을 보여준다. 쓰기작업이 90%의 쓰기가 10%에 집중되는 경우에 한해서 시뮬레이션을 실행하였다. 같은 90/10에 비해서 차이가 얼마 나지 않는 이유는 빈 세그먼트의 수를 제한한 상황에서 실행하였기 때문이다. 그 결과를 그래프로 나타낸 것이 <그림 9>이다. 유효한 데이터의 비율이 70% - 80%인 경우에 약 5%정도의 클리닝 회수를 이득보았다.

<표 2> 유효블록의 비율에 따른 클리닝 비용

	클리닝 회수		이동된 블록의 수		평균 클리닝 비용	
	direction	greedy	direction	greedy	direction	greedy
50%	2444	2747	23690	33605	9.69	12.23
60%	3408	3866	44530	58785	13.07	15.21
70%	4876	5410	81281	98993	16.67	18.30
80%	7568	8112	158241	122563	20.91	15.15
90%	13842	14053	350372	357410	25.31	25.43



6. 결론

플래시메모리는 이동컴퓨터에 적합하여 앞으로 널리 사용될 것이다. 그러나 플래시메모리의 단점 때문에 쓰기작업에서 성능의 저하가 있게 된다. 본 논문에서는 기존 플래시메모리 저장기법을 이동컴퓨터에 적합하도록 하고 클리닝 회수를 줄임으로써 쓰기성능을 개선하도록 했다. 시뮬레이션결과 일반적인 디스크사용유형인 쓰기작업이 집중되는 환경에서 클리닝회수의 줄임을 보이고 전체 메모리 중 70%-80% 의 데이터가 유효한 블록인 경우에도 클리닝회수의 줄어듦을 보였다.

참고문헌

[1] Fred Douglis, Ramon Caceres, Brian Marsh, Frans Kaashoek, Kai Li, and Joshua Tauber, Storage Alternatives for Mobile Computers, Proceedings of the First Symposium on Operating Systems Design and Implementation, USENIX Association, pp. 25-37, November 1994.

[2] Michael Wu and Willy Zwaenepoel, eNVy : A Non-Volatile, Main Memory Storage System, Proceedings 6th International Conference on Architectural Support for Programming languages and Operating System, 1994.

[3] C. Reummer and J Wilkes, UNIX disk access patterns, In Proceedings of the 1993 Winter USENIX Conference, pp. 405-420.

[4] Deborah See and Clark Thurlo, Managing Data in an Embedded System Utilizing Flash Memory, Intel, 1995.

[5] Flash Memory Data Book, samsung, 1997.