제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

플래시 메모리 관리를 위한 순위별 지움 정책

김정기, 박승민, 김채규
한국전자통신연구원 정보가전연구부
e-mail : jkk@etri.re.kr

A Ranking Cleaning Policy for Flash Memory Management

Jeong-Ki Kim, Sung-Min Park, Chae-Kyu Kim
Dept. of Information Appliance, ETRI

요 약

본 논문에서는 이동 컴퓨터 등에서 사용되는 플래시 메모리 관리를 위해 새로운 순위별 지움 정책을 제안하고 성능평가를 실시한다. 지움 정책의 일반적인 역할은 플래시 메모리 공간의 어디를 언제 지울 것인가를 결정한다. 제안된 순위별 지움 정책은 지우는 비용과 한정된 지움 회수를 고려하여 지우는 공간과 시간을 결정함으로써 플래시 메모리의 수명을 최대한 연장하고 플래시 메모리 접근 및 관리의 성능을 향상 시킨다. 제안된 방법은 기존의 방법에 비해 저장 연산 수행에서 10%~50%의 성능 향상을 보인다.

1. 서론

최근에 이동 컴퓨팅 기술이 발전함에 따라 PDA (personal digital assistants), HPC(hand-held PC), 휴태폰, 전자책(e-book) 등을 이용하여 어디서나 통신과 컴퓨팅이 가능하게 되었고, 이러한 휴대 기기들은 작은 크기와 적은 전력 소모를 요구하고 있다. 이러한 특성 때문에 이동 컴퓨터에 데이터를 저장하는 저장 매체로 하드 디스크를 사용하는 것은 비 효율적이다. 하드 디스크는 부피가 클 뿐만 아니라 무겁고 소비전력 또한 크다. 이를 대체하기 위한 방안으로 플래시 메모리 (flash memory)의 사용은 매우 유용하다[1,6].

플래시 메모리는 비 휘발성의 특성을 갖고 있으며, 하드 디스크에 비해 견고하다. 저 전력으로 동작이 가능하며 접근 시간이 RAM 과 유사할 만큼 빠르다. 또한 크기가 작아 휴대 기기에 적합하다. 그러나, 단점으로는 하드 디스크에 비해 가격이 5-10 배 정도 비싸며, 이미 데이터가 있는 공간에 새로운 데이터를 쓰고자 할 때는 지움(cleaning) 과정을 수행한 다음에서야 다시 저장 할 수 있다. 또한 읽는 속도는 매우 빠르지만, 쓰기 속도와 지우는 속도가 상대적으로 느리고, 한번에 지울 수 있는 크기가 일정하며, 상온에서 지울 수 있는 회수가 약 10 만 번으로 정해져 있다. 이렇게 한번에 지울 수 있는 플래시 메모리의 공간을 *지움 블록(erase block)* 또는 *세그먼트(segment)*라 한다. 플래시 메모리의 일반적인 특징은 <표 1>과 같다[7].

<표 1> 플래시 메모리 특성 (Intel 28F640J3A)

특징	값
읽기 접근 시간 (*r*)	100~150 *nsec*
버퍼 이용 쓰기 시간 (*w*) 지움 블록 쓰기 시간	218 *μsec/32bytes* 0.8 *sec/block*
블록 지움 시간 (*e*)	1.0 *sec/block*
최대 지움 회수 (*Emax*)	100,000 times
지움 블록의 크기 (*S*)	128 *Kbyte*
전력 소모량	대기전력 - 50~120 *μA* 동작전력 - 15~70 *mA*

플래시 메모리는 셀(cell)을 구성하는 구조에 따라, NOR, NAND, AND 플래시 등으로 구분할 수 있으나, NOR 나 NAND 플래시 이외에는 거의 사용되지 않는다[2]. NOR 플래시는 임의 접근(random access)의 읽기 속도가 빠르고 비트 당 접근이 용이함으로 메모리 주소 공간에 직접 연결되어 CPU 가 수행하는 코드(code)를 저장하는 목적으로 주로 사용되며, NAND 플래시는 느린 임의 접근 때문에 음악 파일이나 이미지 파

일 등 대용량의 데이터를 한번에 저장하는 용도로 주로 이용된다.

본 논문에서는 이동 컴퓨터 등에서 사용되는 플래시 메모리의 단점을 보완하고 효율적인 관리를 위해 순위별 지움 정책(Ranking Cleaning Policy, RCP)을 제안하고 성능평가를 실시한다. 논문의 구성은 다음과 같다. 2 장에서는 기존의 지움 정책에 대해 설명하고, 3 장에서는 새로운 순위별 지움 정책을 설계한다. 4 장에서 성능을 평가하고, 5 장에서 결론과 향후 연구를 제시한다.

2. 기존의 지움 정책 방법

데이터가 있는 플래시 메모리의 공간은 지움 과정을 수행하기 전까지 갱신할 수 없다는 특징과 지울 수 있는 회수가 한정된다는 특징 때문에 플래시 메모리에서 파일시스템을 구성하는 방법은 LFS(Log-structured File System) 방식을 주로 이용한다[8]. 즉, 데이터 저장과 갱신은 저장공간에 대해 순차적으로 일어나며, 저장매체를 끝까지 다 사용한 다음에 처음으로 돌아와서 공간을 확보하고 다시 저장하는 과정으로 수행된다. 이런 방식은 지움 회수의 한계 때문에 지우는 부분이 편중되지 않고 고르게 안배되도록 하는 과정이다. 이런 과정을 *Wear-leveling* 이라 한다.

LFS 형식 이외의 방식으로는 UNIX 에서 데이터의 접근 패턴을 고려하여[9] 수정이 빈번하게 일어나는 데이터와 수정이 거의 없는 데이터를 분리하여 서로 다른 쪽으로 집중함으로써 지움 회수를 줄이는 방향성 지움 정책[1]과 클리닝 지표(cleaning index)를 이용한 사이클 평준화 방법[3] 등이 제시되고 있다.

일반적으로 플래시 메모리를 위한 장치 관리자(device driver)는 표준 UNIX 파일시스템을 지원하고 하드 디스크와 유사한 형태로 설계되고 구현된다[10,11]. 플래시 메모리에서 물리적으로 한번에 지울 수 있는 공간이 세그먼트(128KB) 단위이므로 UNIX 파일시스템에 맞추기 위해 세그먼트를 512B 나 1KB 크기의 데이터 블록으로 쪼개서 저장한다. 플래시 메모리는 갱신이 일어날 때, 디스크처럼 같은 곳에 저장되지 않으므로 UNIX 파일시스템과 플래시 메모리의 물리적인 데이터 저장 위치를 연결해줄 매핑 테이블(mapping table)이 필요하다. 세그먼트의 앞 부분은 블록 관리에 대한 요약 정보가 약 2KB 정도 들어간다[4,5].

일단 플래시 메모리가 지워지면, 세그먼트 내의 모든 데이터 블록은 *Free* 상태이다. 유용한 데이터가 저장되면, *Valid* 상태가 된다. 데이터가 갱신되어 다른 블록에 저장되거나 지워진 데이터의 블록은 *Invalid* 상태가 된다. Invalid 상태의 데이터가 실제로 사라진 것은 아니고 요약 정보 부분에 표시해 둘 뿐이다. 나중에 저장공간이 필요하면 세그먼트 단위로 지워야 하는데 Valid 블록은 다른 세그먼트로 옮기고 Invalid 블록은 그냥 지운다.

지움 정책의 역할은 어떤 세그먼트를 언제 지울 것인지를 결정한다. 세그먼트 선택을 위한 기존의 방법은 Greedy 와 Cost-benefit 방법이 있다[12]. Greedy 방법은 세그먼트 중 가장 적은 Valid 블록을 가진 세그먼트를 선택하고, Cost-benefit 방법은 다음 식에 의해 세그먼트를 선택한다.

$$\frac{benefit}{cost} = \frac{age \times (1-u)}{2u}$$

즉, 비용이 가장 적고 이익이 가장 많은 세그먼트를 선택한다는 뜻이다. 여기서 u 는 세그먼트의 이용률이다. 즉 전체 세그먼트 크기에 대한 Valid 블록의 크기이다. $2u$ 는 세그먼트를 지우기 위해 Valid 블록을 읽어서 다른 세그먼트에 저장해야 하는 비용을 의미한다. $(1-u)$는 새롭게 생성되는 Free 공간의 양이다. *age* 는 블록이 Invalid 된 이후 현재까지의 시간이다.

다음은 언제 세그먼트를 지울 것인지를 결정해야 하는데 가능한 미루어지는 것이 좋다. 왜냐하면, 한 세그먼트에서 Invalid 되는 가능성이 높아지고 지우는 동안 이동해야 하는 데이터 양이 적어지기 때문이다. 그러나 너무 많은 지연은 지움 과정이 다른 프로세스(process)와 동시에 수행되게 하는 가능성을 줄인다. 그러므로 임계값(threshold)을 설정하여 지우는 때를 결정한다.

3. 순위별 지움 정책 방법

기존의 Greedy 방법은 wear-leveling 을 무시하고 있으며, Cost-benefit 방법에서는 *age* 로 wear-leveling 을 고려하고 있지만 이것이 지움 회수가 적다는 것을 의미하지는 않는다. 이러한 문제점을 해결하기 위해 본 논문에서는 순위값에 따라 세그먼트의 지움 순서를 결정하는 순위별 지움 정책(Ranking Cleaning Policy, RCP)을 제안한다. 순위값은 세그먼트를 지우는 우선순위이며 작은 값을 갖는 세그먼트가 먼저 선택되어 지워진다.

플래시 메모리 관리자의 성능은 쓰기 연산에 의해 결정된다. 왜냐하면, 기본적인 쓰기 시간이 읽기 시간에 비해 매우 느릴 뿐 아니라 쓸 공간이 없어 지움 과정을 수행하는 것도 외부에서는 쓰기 연산으로 보이기 때문이다. 그러므로 성능을 높이기 위해 세그먼트의 지움 회수를 절대적으로 줄이는 것이 중요하다. 이를 위해 세그먼트 지움 과정은 지우는 비용이 가장 적은 것을 선택한다. 세그먼트의 지움 비용(erase cost, *Ce*)은 다음 식에 의해 계산될 수 있다.

$$Ce = Cr + Cw$$

여기서 Cr 은 선택된 세그먼트에서 유용한(Valid) 데이터를 읽어서 다른 세그먼트에 옮기는 비용과 물리

적으로 지우는 비용을 포함한다. 그리고 Cw 는 Free 공간으로 남아있는 곳을 다시 지움으로써 낭비되는 비용이다. LFS 방식을 이용할 경우 일반적으로 Cw 가 발생되지 않지만, JFFS2[13] 같은 몇몇 시스템은 세그먼트에 Free 공간이 발생된다. 그러므로, V, I, F 를 각각 Valid 공간의 크기, Invalid 공간의 크기, Free 공간의 크기라고 할 때 <표 1>의 기호를 이용하여 위 식은 다음처럼 계산될 수 있다.

$$Cr = V(r+w) + S \cdot e, \qquad Cw = F \cdot e$$
$$Ce = V(r+w) + S \cdot e + F \cdot e$$

여기서, 읽기 속도(r)는 쓰기 속도(w)에 비해 매우 작으며, 쓰기 속도와 지움 속도가 유사하기 때문에 간단히 다음처럼 표시할 수 있다.

$$Ce = (V + S + F) \cdot w = (2S - I) \cdot w$$

여기서, S 와 w 는 플래시 메모리 특성에 의해 정해지는 값이므로 지우는 비용이 가장 적은 세그먼트를 선택하는 것은 무효한(Invalid) 공간이 가장 많은 세그먼트를 선택하는 것이다. Greedy 방법에서는 지울 세그먼트를 선택할 때 유효한 블록이 가장 적은 것을 선택하는데, LFS 처럼 플래시 메모리의 세그먼트에 데이터를 순차적으로 저장하는 방식은 세그먼트 내에 Free 공간이 생기지 않으므로 타당한 방법이다.

그러나, 지움 비용 만을 가지고 지울 세그먼트를 선택할 수는 없다. 여기서 한가지 더 고려해야 할 것이 플래시 메모리의 수명을 결정하는 wear-leveling 이다. RCP 방법에서는 지움 비용뿐 아니라 지움 회수를 고려한 세그먼트 선정 방법을 제안한다.

먼저, i 번째 세그먼트에 대해 지움의 가능성 $Pe(Si)$ 은 지움 비용(Ce)과 지움 회수(En)에 반비례한다. 그러나, 지움 회수가 한계 값($Emax$)에 가까워지면, 지움 가능성은 현저하게 줄어들어야 타당하다. 그러므로 지움 회수의 한계 값($Emax$)을 이용하기 위해 세그먼트의 남은 지움 회수(Er)를 이용하여 식을 만드는 것이 타당하다. 즉, $Er = Emax - En$. 이제 어떤 세그먼트의 지움 가능성은 지움 비용(Ce)에 반비례하고 남은 지움 회수(Er)에 비례한다.

이런 사실을 바탕으로 순위 값(ranking value, R)을 다음처럼 정의할 수 있다. 값이 적은 순서로 지울 대상 세그먼트로 선정한다

$$R = \alpha \frac{Ce}{Er} = \alpha \frac{Ce}{En - Emax}, \quad or \quad R = \alpha \cdot Ce - \frac{1}{\alpha} Er$$

여기서 α 는 가중치이다. 지움 비용에 중점을 둘 것인지 지움 회수에 비중을 둘 것인지를 결정할 수 있다. 위 식에서 지움 비용과 지움 회수의 단위가 다르므로, 같은 비율로 본다면 각각의 값을 최대값에 대한 비율로 계산할 수 있다. 지움 비용에 대한 최대값($Cmax$)는 모두 유용한(Valid) 데이터가 들어있는 세그먼트를 지우는 비용이다.

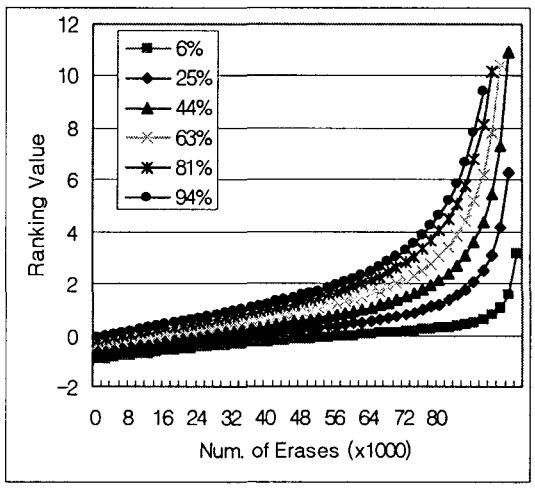
$$c = \frac{Ce}{Cmax}, \quad e = \frac{Er}{Emax}$$

위 식을 이용하여 순위 값을 다시 쓰면 다음과 같다.

$$R = \alpha \frac{c}{e}, \quad or \quad R = \alpha \cdot c - \frac{1}{\alpha} e$$

이제 가중치 α 를 결정하면 세그먼트를 지울 순위를 결정할 수 있다. 기본적으로 지움 회수가 최대값에 가까워 지면, 순위 값을 높임으로써 지움 가능성을 줄여야 한다. 즉 e 가 0(zero)에 가까워지면, R 값이 증가해야 한다. 그러므로, α 를 결정하는 한가지 방법은 e 값과 반비례한다는 것이다. 즉, 순위 값을 결정하는 한가지 예는 위의 두 번째 식을 이용하여 다음과 같이 정의할 수 있다.

$$R = \frac{c}{e} - e^2$$



(그림 1) 유용 데이터 양과 지움 회수에 따른 순위 값의 분포도

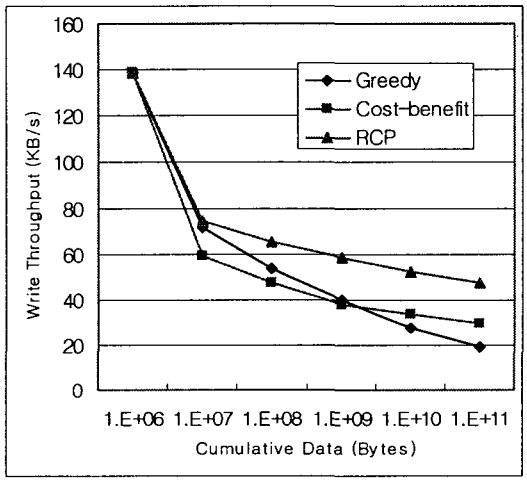
(그림 1)은 이 식을 이용한 순위 값의 분포를 보이고 있다. 세그먼트에 free 공간이 없을 때 Valid 데이터의 양과 지움 회수에 따른 순위 값의 분포이다.

4. 성능 평가

플래시 메모리의 읽기 속도는 RAM 과 유사하기 때문에 순차적 읽기 연산이나 무작위 읽기 연산에 대한 성능은 거의 같다. 그러므로 여러 가지 플래시 메모리 관리자(driver)나 파일시스템에서 읽기 성능 평가는 큰

의미가 없다. 그러나, 쓰기 연산에 대한 성능은 지움 정책의 영향을 많이 받는다.

본 논문에서의 실험은 SMDK2400 이라는 테스트 보드를 이용해서 수행되었다. S3C2400(ARM290) CPU 와 32MB RAM, 2 개의 Intel 28F640J3A 플래시 메모리 (2x8MB)로 구성되어 있다. 초기에 플래시 공간에 60%의 데이터를 저장시킨 다음, 새로운 데이터를 갱신하는 형태로 테스트를 수행했다. 데이터 갱신은 UNIX 의 일반적인 접근 패턴[9]을 고려하여, 90:10 법칙을 따른다고 가정한다. 쓰기 연산의 90%는 초기 데이터의 10%에 집중되고, 10% 만이 나머지 90%에 분포한다는 의미이다.



(그림 2) 쓰기 연산 성능평가

(그림 2)는 쓰기 처리능력(write throughput)에 따른 쓰기 성능평가를 보이고 있다. Greedy 방법은 데이터 량이 적을 때 Cost-benefit 방법보다 좋은 성능을 보이고 RCP 방법과 유사한 성능을 보인다. 왜냐하면, 데이터 량이 적을 때는 지움 회수에 대한 영향을 받지 않기 때문이다. Greedy 방법은 몇 개의 세그먼트가 지움 회수 최대값에 도달하면서부터 성능이 현저히 감소된다. Cost-benefit 방법은 *age* 값을 이용하여 wear-leveling 을 고려함으로써 오랫동안 플래시 메모리의 수명을 유지할 수 있다. RCP 방법은 전체적인 범위에서 좋은 성능을 보인다. 왜냐하면, 데이터가 적을 때 세그먼트의 지움 회수가 적으면 wear-leveling 을 별로 고려하지 않지만, 지움 회수가 한계 값에 가까워지면, 순위 값이 커져 wear-leveling 이 고려 되기 때문이다.

5. 결론

본 논문에서는 순위 값을 이용한 새로운 플래시 메모리 지움 정책을 제안했다. 플래시 메모리는 비 휘발성의 특성을 갖고 있지만, 관리하는 방법에 따라 접근 성능이 달라지고 오랜 수명을 유지할 수 있다. 본 논문에서 제안된 방법은 성능평가를 통해 기존의 방법에 비해 10%~50%까지 성능 향상을 보이고 플래시 공간에 대한 지움 평준화를 이룸으로써 사용 수명이 오래가도록 했다.

제안된 방법은 NOR 형식의 플래시 메모리를 기준으로 설계되고 평가 되었지만, 최근에 많이 사용되고 있는 NAND 형식의 플래시 메모리도 같은 방식으로 적용할 수 있다. 비 휘발성 메모리의 연구는 최근에 활발히 연구되고 있으며, 특성 면에서 플래시 보다 우수한 성능을 보이는 FRAM(Ferroelectric RAM)은 이미 상용화 단계에 있고, MRAM(Magnetic RAM) 등이 활발히 연구되고 있다. 이러한 소자의 개발로 인해 비 휘발성의 저장매체에 대한 관리 방법도 새롭게 개발되어야 할 것이다.

참고문헌

[1] 민용기, 박승규, "이동컴퓨터를 위한 플래시메모리 클리닝 정책," 한국통신학회논문지, Vol. 24, No. 5A, pp. 657-666, 1999.

[2] 서강덕, "본격적인 시장 도입기에 접어든 NAND Flash Memory", 전자공학회지, 7 권 3 호, pp. 56-65, 2000.

[3] 김한준, 이상구, "신뢰성 있는 플래시메모리 저장 시스템 구축을 위한 플래시메모리 저장 공간 관리 기법," 정보과학회논문지: 시스템 및 이론, 27 권 6 호, pp. 567-582, 2000.

[4] 박상호, 안우현, 박대연, 김정기, 박승민, "플래시 메모리를 위한 파일시스템 구현", 정보과학회 논문지: 컴퓨팅의 실제, Vol.7, No.5, pp.402-415, 2001.

[5] 김정기, 박승민, 김채규, "임베디드 플래시 파일시스템," 정보처리학회지, 9 권 1 호, pp. 43-49, 2002.

[6] M. L. Chang, P. C. H. Lee, R. C. Chang, "Managing Flash Memory in Personal Communication Devices," Proc. of IEEE Symp. on Consumer Electronics, pp. 177-182, 1997.

[7] Intel Corporation, "3 volt Intel StrataFlash Memory 28F128J3A, 28F640J3A, and 28F320J3A", 2001.

[8] M. Rosenblum and J. K. Ousterhout, "The Design and Implementation of a Log-Structured File System," ACM Transactions on Computer Systems, Vol. 10, pp 26-52, 1992.

[9] C. Ruemmler and J. Wilkes, "UNIX disk access patterns," Proc. Winter USENIX, 1993.

[10] Interl Corporation, "Understanding the Flash Translation Layer(FTL) Specification", Technical Paper, 1998.

[11] WindRiver, "TrueFFS for Tornado Programmer's Guide 1.0", 1999.

[12] Atsuo Kawaguchi, Shingo Nishioka, and Hiroshi Motoda, "A Flash-Memory Based File System," Proc. of USENIX Technical Conference, pp. 155-164, 1995.

[13] David Woodhouse, "JFFS: The Journaling Flash File System", Technical Paper of RedHat Inc. 2001.