임베디드 플래시 파일시스템을 위한 순위별 지움 정책

김 정 기[†]·박 승 민[†]·김 채 규[†]

요 약

최근 정보 산업과 이동 통신 기술이 발전함에 따라 PDA(personal digital assistant), HPC(hand-held PC), 세탑박스(set-top box), 정보가전 등의 임베디드 시스템(embedded system)이 개발되고 있으며, 이러한 시스템을 운영할 실시간 운영체제(RTOS)와 파일시스템의 요구는 필수적이다. 휴대의 용이성과 빠른 접근시간, 저전력을 요구하는 임베디드 시스템의 특성상 데이터를 저장하기 위한 저장 매체로 하드 디스크(hard disk)를 이용하는 것은 비효율적이며, 플래시 메모리(flash memory)가 주로 사용되고 있다. 플래시 메모리는 비휘발성과 빠른 접근 시간을 갖는 장점이 있지만, 상대적으로 느린 지움 시간과 지움 회수의 한계 등은 극복해야 할 문제점이다. 본 논문에서는 이러한 단점을 보완하기 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 향상시킨다. 제안된 방법은 기존의 Greedy 및 Cost-benefit 방법에 비해 저장 연산 수행에서 10%~50%의 성능 향상을 보인다.

A Ranking Cleaning Policy for Embedded Flash File Systems

Jeong-Ki Kim[†]·Sung-Min Park[†]·Chae-Kyu Kim[†]

ABSTRACT

Along the evolution of information and communication technologies, manufacturing embedded systems such as PDA (personal digital assistant), HPC (hand-held PC), settop box, and information appliance became realistic. And RTOS (real-time operating system) and filesystem have been played essential roles within the embedded systems as well. For the filesystem of embedded systems, flash memory has been used extensively instead of traditional hard disk drives because of embedded system's requirements like portability, fast access time, and low power consumption. Other than these requirements, nonvolatile storage characteristic of flash memory is another reason for wide adoption in industry. However, there are some technical challenges to cope with to use the flash memory as an indispensable component of the embedded systems. These would be relatively slow cleaning time and the limited number of times to write-and-clean. In this paper, a new cleaning policy is proposed to overcome the problems mentioned above and relevant performance comparison results will be provided. Ranking cleaning policy (RCP) decides when and where to clean within the flash memory considering the cost of cleaning and the number of times of cleaning. This method will maximize not only the lifetime of flash memory but also the performance of access time and manageability. As a result of performance comparison, RCP has showed about 10~50% of performance evolution compared to traditional policies, Greedy and Cost-benefit methods, by write throughputs.

키워드 : 임베디드 시스템(Embedded System), 플래시 메모리(Flash Memory), 파일 시스템(File System), 순위별 지움 정책(Ranking Cleaning Policy), RCP

1. 서 론

최근에 정보 산업과 이동 컴퓨팅 기술이 발전함에 따라 PDA(personal digital assistants), HPC(hand-held PC), 휴대폰, 전자책(e-book) 등을 이용하여 어디서나 통신과 컴퓨팅이 가능하게 되었다. 이러한 휴대 컴퓨팅 기기들은 운영체제가 내장되는 임베디드 시스템으로 발전되고 있으며, 작은 크기와 적은 전력 소모를 요구하고 있다. 이러한 임베디드 시스템을 관리할 실시간 운영체제는 필수적이며, 시스템 데이터와 사용자 데이터를 저장하고 관리해야 한다. 이를 위해 기존의 하드 디스크를 이용한 파일시스템을 구성하는 것은 매우 비효율적이다. 하드 디스크는 부피가 클 뿐만 아니라 무겁고 소비전력 또한 크기 때문이다. 이를 대체하기 위한 방안으로 플래시 메모리(flash memory)의 사용은 매우 유용하다[1, 6].

플래시 메모리는 비휘발성의 특성을 갖고 있으며, 하드 디스크에 비해 견고하다. 저 전력으로 동작이 가능하며 접근 시간이 RAM과 유사할 만큼 빠르다. 또한 크기가 작아 휴대 기기에 적합하다. 그러나, 단점으로는 하드 디스크에 비해 가격이 5~10배정도 비싸며, 이미 데이터가 있는 공간에 새로운 데이터를 쓰고자 할 때는 지움(cleaning) 과정을 수행한 다음에서야 다시 저장 할 수 있다. 또한 읽는 속도

† 정 회 원 : 한국전자통신연구원 정보가전연구부
논문접수 : 2002년 9월 28일, 심사완료 : 2002년 11월 4일

는 매우 빠르지만, 쓰기 속도와 지우는 속도가 상대적으로 느리고, 한번에 지울 수 있는 크기가 일정하며, 상온에서 지울 수 있는 회수가 약 10만 번으로 정해져 있다. 이렇게 한번에 지울 수 있는 플래시 메모리의 공간을 *지움 블록*(*erase block*) 또는 *세그먼트*(*segment*)라 한다.

이러한 세그먼트를 지울 수 있는 회수는 상온에서 약 10만 번으로 정해져 있기 때문에 일정 세그먼트를 계속해서 사용하게 되면 한계 수명에 도달하게 되고 더 이상 사용할 수 없게 된다. 이는 메모리가 줄어드는 결과를 가져오고 지움 과정이 더 빈번히 발생하여 시스템 사용에 영향을 준다. 그러므로 지움 과정이 일어나는 세그먼트를 고르게 안배해야 임베디드 시스템 성능을 향상시킬 수 있다. 이렇게 세그먼트에 대해 지움 과정을 고르게 안배하는 과정을 지움 평준화(wear-leveling)라 한다. 플래시 메모리의 일반적인 특징은 <표 1>과 같다[7].

〈표 1〉 플래시 메모리 특성 (Intel 28F640J3A)

특 징	값
읽기 접근 시간 (r)	100~150 *nsec*
버퍼 이용 쓰기 시간 (w) 지움 블록 쓰기 시간	218 μ*sec/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, NOR나 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 CPU가 수행하는 코드(code)를 저장하는 목적으로 주로 사용되며, NAND 플래시는 비트 당 접근이 불가능하고 상대적으로 느린 임의 접근 때문에 음악 파일이나 이미지 파일 등 상대적으로 큰 용량의 데이터를 한번에 저장하는 용도로 주로 이용된다.

본 논문에서는 임베디드 시스템에서 사용되는 플래시 메모리 관리의 단점을 보완하고 효율적인 관리를 위해 순위별 지움 정책(Ranking Cleaning Policy, RCP)을 제안하고 성능 평가를 실시한다. 논문의 구성은 다음과 같다. 2장에서는 기존의 플래시 파일 시스템과 지움 정책에 대해 설명하고, 3장에서는 새로운 순위별 지움 정책을 설계한다. 4장에서 성능을 평가하고, 5장에서 결론과 향후 연구를 제시한다.

2. 기존의 플래시 파일 시스템과 지움 정책 방법

앞에서 언급한 플래시 메모리의 한계 내에서 최대한 효율적인 이용을 위하여 여러 가지 플래시 파일시스템이 개발되었다. 특히 플래시 메모리의 가장 큰 제약인 지우는 속도가 상대적으로 느리고 세그먼트 당 지울 수 있는 회수가 한정된다는 문제점을 극복하기 위하여 지움 정책(cleaning policy)은 가장 큰 쟁점 중의 하나이다.

데이터가 존재하는 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.

다른 방식으로는 UNIX 시스템의 데이터 접근 패턴을 고려하여[9] 수정이 빈번하게 일어나는 데이터(hot data)와 수정이 거의 없는 데이터(cold data)를 분리하여 성격이 비슷한 데이터를 한쪽으로 집중함으로써 지움 회수를 줄이는 방향성 지움 정책[1]과 클리닝 지표(cleaning index)를 이용한 사이클 평준화 방법[3] 등이 제시되고 있다.

일반적으로 세그먼트를 지울 때 유효한(valid) 블록은 다른 세그먼트로 옮기고 지워야 하기 때문에 무효한(invalid) 블록이 가장 많은 것을 골라서 지우게 되는데, 방향성 지움 정책에서는 갱신이 빈번하게 일어나는 hot 블록을 일정한 공간에 모음으로써, hot 블록의 갱신으로 일정 공간에 무효화 블록이 많이 발생한다. 그러면, 무효화 블록이 대부분인 세그먼트를 지우는 비용이 적어지고 성능이 향상된다. 그러나 세그먼트 간 지움 평준화를 이루기 위해 hot 블록을 모으는 공간을 계속해서 이동하거나 또는 일정량의 작업을 수행한 다음 위치를 변경해야 하는데, 이런 변경에 대한 시점이 분명하지 않으며, 이런 변경과정을 수행할 때는 성능이 매우 떨어진다는 문제점이 있다.

사이클 평준화 방법도 hot 데이터와 cold 데이터를 분리하여 저장하는데, 클리닝 비용을 계산하고 사이클 평준화(지움 평준화)를 이루기 위해 클리닝 지표를 계산한다. 클리닝 지표에 의해 가장 작은 값을 갖는 세그먼트를 지움 대상으로 선정한다. 그리고 정규 사이클 평준화도가 어떤 설정값을 넘어서면 위치를 변경하는 방식이다. 이 방법은 방향성 지움 정책을 개선한 방법이긴 하지만, 마찬가지로 hot 데이터가 집중되는 위치를 계속해서 옮길 때 성능이 저하되는 문제점이 있다.

Kawaguchi 등은 플래시 메모리를 위한 장치 관리자(device driver)를 구현하기 위해 표준 UNIX 파일시스템을 지원하고 하드 디스크와 유사한 형태로 설계하고 구현하였다[10]. 플래시 메모리에서 물리적으로 한번에 지울 수 있는 공간이 세그먼트(128KB) 단위이므로 UNIX 파일시스템에 맞추기 위해 세그먼트를 512B나 1KB 크기의 데이터 블록으로 쪼개서 저장한다. 플래시 메모리는 갱신이 일어날 때, 디스크처럼 같은 곳에 저장되지 않으므로 UNIX 파일시스

템과 플래시 메모리의 물리적인 데이터 저장 위치를 연결하기 위해 매핑 테이블(mapping table)이 필요하다. 세그먼트의 앞 부분은 블록 관리에 대한 요약 정보가 약 2KB 정도 들어간다[4, 5].

이렇게 블록 디바이스 형태를 에뮬레이트(emulate)하여 구현한 플래시 파일시스템이 FTL(Flash Translation Layer)[11]과 TrueFFS[12]이다. FTL은 Intel사에 의해 개발되었으며, wear-leveling에 대한 개념이나 갑작스런 전원 오류시 복구 방법이 미약한 단점이 있다. TrueFFS는 WindRiver 사의 WxWorks RTOS에 맞게 구현한 시스템으로 블록 디바이스 인터페이스를 제공한다. wear-leveling 알고리즘을 구현하고 있으며, "저장 뒤 지움(erase after write)" 정책을 이용하여 오류 복구(fault recovery) 방법을 구현하고 있다.

또한 wear-leveling을 완전히 해결할 수 있는 플래시 파일시스템으로는 JFFS[13]가 있다. 이 방법은 완전한 LFS 형식을 따르고 있다. 대부분의 플래시 파일시스템이 플래시 메모리에 블록 디바이스를 에뮬레이트(emulate)하고 그 위에 파일 시스템을 올리는 형식이지만, JFFS는 발생되는 데이터의 크기에 관계없이 플래시 메모리의 공간에 순차적으로 저장한다. 마지막에 이르면 처음으로 돌아와서 다시 반복한다. 그러므로 가장 큰 문제점은 변경이 전혀 없는 세그먼트도 순차적인 접근 때문에 내부의 모든 데이터를 다른 곳으로 옮기고 지워야 하는 문제점이 발생한다. 데이터 갱신이 한곳으로 집중될 경우 매우 속도가 느려지는 단점이 있다.

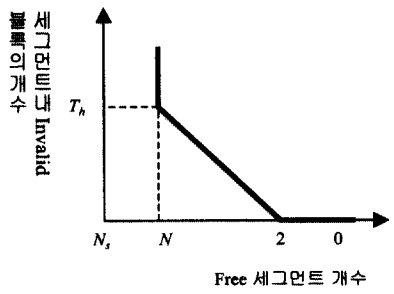
지움 정책의 역할은 어떤 세그먼트를 언제 지울 것인지를 결정한다. 일단 플래시 메모리가 지워지면, 세그먼트 내의 모든 데이터 블록은 *Free* 상태이다. 유용한 데이터가 저장되면 *Valid* 상태가 된다. *Valid* 상태의 데이터를 갱신하면 플래시 메모리의 특성상 기존에 저장된 위치에 바로 저장할 수 없으므로 다른 Free 상태의 공간에 저장하고 이전 데이터 위치는 *Invalid* 상태로 변경한다. 또한 지워진 데이터의 블록은 바로 *Invalid* 상태가 된다. Invalid 상태의 데이터가 실제로 사라진 것은 아니고 요약 정보 부분에 표시해 둘 뿐이다. 나중에 세그먼트를 지울 때 Invalid 블록은 그냥 지우고 Valid 블록은 유용한 데이터이므로 다른 세그먼트로 옮기고 Invalid 상태를 만들어서 지운다.

지움 정책의 세그먼트 선택을 위한 기존의 방법은 Greedy와 Cost-benefit 방법이 있다[10]. 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 u는 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 Invalid된 이후 현재까지의 시간이다.

다음은 언제 세그먼트를 지울 것인지를 결정해야 하는데 가능한 미루어지는 것이 좋다. 왜냐하면, 한 세그먼트에서 블록이 Invalid되는 가능성이 높아지므로 지우는 동안 이동해야 하는 데이터 양이 적어지기 때문이다. 그러나 너무 많은 지연은 지움 과정이 다른 프로세스(process)와 동시에 수행되게 하는 가능성을 줄인다. 그러므로 cost-benefit 방법에서는 (그림 1)과 같이 임계값(threshold)을 설정하여 지우는 세그먼트를 결정한다. 전체 플래시 메모리에서 최소 2개 이상의 Free 세그먼트를 유지해야 하며, N개 이상의 세그먼트가 Free이면 전혀 지울 필요가 없다. 2와 N 사이 일 때, 세그먼트 내에 T_h개 이상의 Invalid 블록을 가진 세그먼트는 모두 지운다.



(그림 1) 임계값 결정 그래프

3. 순위별 지움 정책 방법

기존의 Greedy 방법은 wear-leveling을 무시하고 있으며, Cost-benefit 방법에서는 *age*로 지움 평준화(wear-leveling)를 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 Cr은 선택된 세그먼트에서 유용한(Valid) 데이터를 읽어서 다른 세그먼트에 옮기는 비용과 물리적으로 지우는 비용을 포함한다. 그리고 Cw는 Free 공간으로 남아있

는 곳을 다시 지움으로써 낭비되는 비용이다. LFS 방식을 이용할 경우 일반적으로 Cw가 발생되지 않지만, JFFS[13] 같은 몇몇 시스템은 세그먼트에 Free 공간이 발생된다. 그러므로, V, I, F를 각각 Valid 공간의 크기, Invalid 공간의 크기, Free 공간의 크기라고 할 때 <표 1>에서 읽기 시간, 쓰기 시간, 지움 시간에 대한 기호를 비트 당 수행으로 설정할 경우 위의 식은 다음처럼 계산될 수 있다.

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

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

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

여기서, S와 w는 플래시 메모리 특성에 의해 정해지는 값이므로 I에 의해 지움 비용(Ce)을 결정할 수 있다. 즉, 지우는 비용이 가장 적은 세그먼트를 선택하는 것은 무효한(Invalid) 공간이 가장 많은 세그먼트를 선택하는 것과 같다. Greedy 방법에서는 지울 세그먼트를 선택할 때 유효한 블록이 가장 적은 것을 선택하는데, LFS처럼 플래시 메모리의 세그먼트에 데이터를 순차적으로 저장하는 방식은 세그먼트 내에 Free 공간이 생기지 않으므로 타당한 방법이다.

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

먼저, i번째 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 한계값($Emax$)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.

$$Pe(Si) = R = \alpha \frac{Ce}{Er} = \alpha \frac{Ce}{En - Emax},$$
$$\text{or} \quad R = \alpha \cdot Ce - \frac{1}{\alpha} Er$$

여기서 α는 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($Cmax$)은 모두 유용한(Valid) 데이터가 들어있는 세그먼트를 지우는 비용이다.

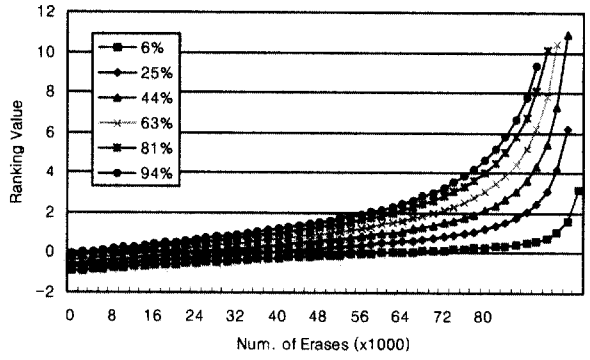
$$c = \frac{Ce}{Cmax}, \quad e = \frac{Er}{Emax}$$

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 α를 결정하면 세그먼트를 지울 순위를 결정할 수 있다. 기본적으로 지움 회수가 최대값에 가까워지면, 순위 값을 높임으로써 지움 가능성을 줄여야 한다. 즉 e가 0(zero)에 가까워지면, R 값이 증가해야 한다. 그러므로, α를 결정하는 한가지 방법은 e값과 반비례하는 것으로 보고 $1/e$로 결정할 수 있다. 즉, 순위 값을 결정하는 한가지 예는 위의 두 번째 식을 이용하여 다음과 같이 정의할 수 있다.

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



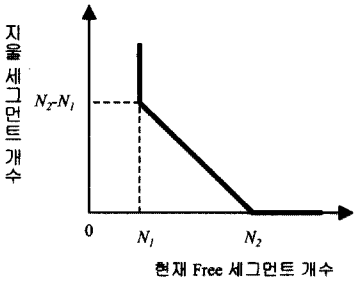
(그림 2) 유용 데이터 양과 지움 회수에 따른 순위 값의 분포도

(그림 2)는 이 식을 이용한 순위 값의 분포를 보이고 있다. 세그먼트에 free 공간이 없을 때 Valid 데이터의 양과 지움 회수에 따른 순위 값의 분포이다.

Cost-benefit 방법에서의 임계값 결정은 세그먼트 내에 Invalid 블록의 개수를 이용하여 지울 세그먼트를 결정하기 때문에 실제로 지워지는 세그먼트의 개수를 정확히 알 수 없다. 그러므로 필요이상 세그먼트가 지워질 가능성이 높다. 특히 갱신되는 데이터가 전체 플래시 메모리에 골고루 분포할 경우, 즉 Invalid 블록이 세그먼트에 골고루 분포될 경우, Invalid 블록의 개수가 임계값에 이르지 못하여 지워지는 세그먼트가 거의 없을 수도 있고, 반대로 한번에 너무 많은 세그먼트가 지워질 수도 있다. 즉, 지워질 세그먼트의 개수를 예측하기 힘들다는 문제점이 있다.

이러한 문제점을 해결하기 위해 본 논문에서는 (그림 3)과 같은 새로운 임계값 결정 그래프를 이용한다. N_2 이상의 Free 세그먼트를 확보하고 있으면, 더 이상 지움 과정이 발생하지 않는다. Free 세그먼트의 개수가 N_1에서 N_2 사이에 있을 때는 최대한 N_2 이상의 Free 세그먼트를 확보하기 위하여 그래프의 지울 세그먼트 개수만큼 지운다. Free 세그먼

트의 개수가 N_1 이하이면, 어떤 연산보다 우선하여 세그먼트를 확보한다. 지움 연산을 수행한 뒤로도 Free 세그먼트의 개수가 N_1 이하이면 더 이상의 저장공간이 없다는 뜻이다.



(그림 3) RCP의 임계값 결정 그래프

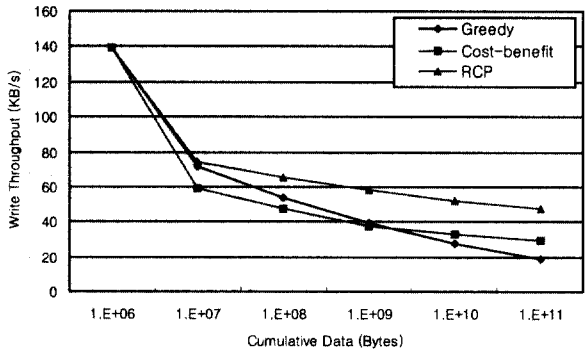
지움 과정은 읽기나 쓰기에 비해 많은 시간을 소모하므로 플래시 파일시스템에 대한 일정량의 접근이 수행된 다음에 읽기나 쓰기가 발생하지 않는 휴지 상태(idle time)에 있을 때 일반적으로 발생한다. 이러한 휴지 상태는 일정시간 동안 접근이 없을 때로 설정할 수 있다. 그리고, 대부분의 플래시 메모리는 지움 과정을 수행하다가 읽기 및 쓰기 연산의 요구가 들어오면, 지움 과정을 보류하고(suspend) 우선 순위가 높은 연산을 수행하도록 하는 기능을 제공한다.

4. 성능 평가

플래시 메모리의 읽기 속도는 RAM과 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.

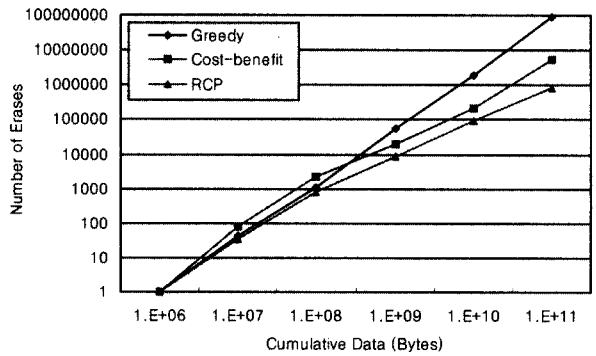
본 논문에서의 실험은 SMDK2400이라는 테스트 보드를 이용해서 수행되었다. S3C2400(ARM290) CPU와 32MB RAM, 2개의 Intel 28F640J3A 플래시 메모리(2×8MB)로 구성되어 있다. 초기에 플래시 메모리 공간에 일정량의 데이터를 저장시킨 다음, 새로운 데이터를 갱신하는 형태로 테스트를 수행하였다. Greedy와 Cost-benefit 방법에서 임계값 T_h는 한 세그먼트에 저장 가능한 504개 데이터 블록 중 455개의 Invalid 블록을 기준으로 했으며, N = 12로 설정했다[10]. RCP 방법에서도 다른 방법과 유사성을 위해 N_1 = 2, N_2 = 12로 설정했다. 데이터 갱신은 UNIX의 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 4)는 쓰기 처리능력(write throughput)에 따른 쓰기 성능평가를 보이고 있다. 초기 데이터는 60%을 이용하고 계속해서 데이터를 누적하여 저장함으로써 성능을 평가했다. 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



(그림 4) 쓰기 연산 성능평가

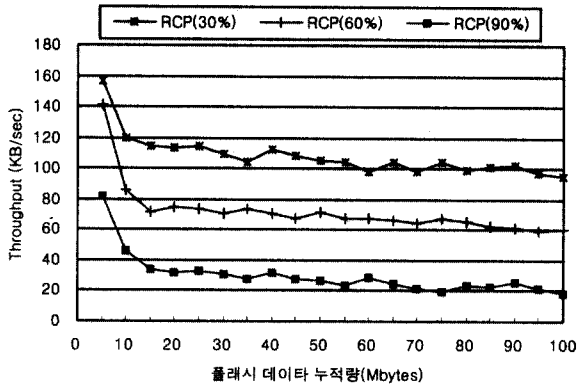
않기 때문이다. Greedy 방법은 몇 개의 세그먼트가 지움 회수 최대값에 도달하면서부터 성능이 현저히 감소된다. Cost-benefit 방법은 *age* 값을 이용하여 wear-leveling을 고려함으로써 오랫동안 플래시 메모리의 수명을 유지할 수 있다. RCP 방법은 전체적인 범위에서 좋은 성능을 보인다. 왜냐하면, 데이터가 적을 때 세그먼트의 지움 회수가 적으면 지움 평준화를 별로 고려하지 않지만, 지움 회수가 한계 값에 가까워지면, 순위 값이 커져 지움 평준화가 고려되기 때문이다.



(그림 5) 지움 회수에 대한 성능평가

(그림 5)는 60% 초기 데이터에 대한 누적 데이터에 따른 지움 회수의 성능평가를 보이고 있다. Greedy 방법의 지움 회수는 처음에 RCP 방법과 같이 완만히 증가하다가, 몇 개의 세그먼트가 지움 회수의 한계치에 이르러서부터 빠르게 증가한다. Cost-benefit 방법은 처음에 Greedy 방법보다 빠르게 증가하지만, 후반부에 완만히 증가한다. RCP 방법이 전체적으로 가장 좋은 성능을 보인다.

(그림 6)은 RCP방법에서 초기 데이터가 각각 30%, 60%, 90% 일 때 쓰기 연산에 대한 성능평가를 보이고 있다. 전체적으로 데이터 누적량이 적을 때 좋은 성능을 보이는 것은 Free 세그먼트가 아직 남아 있기 때문이다. 그리고, 누적량이 많아지면서 전체적으로 일정한 성능을 보이는 것은 쓰기와 지우기의 양이 일정해 지기 때문이다. 또한 초기 데이터의 양이 많을수록 성능이 나빠지는 이유는 Free 세그먼트를 만들기 위해 옮겨야 하는 데이터의 양이 많고 빈번히 지움 과정이 일어나기 때문이다.



(그림 6) RCP에서 초기 데이터 량에 따른 쓰기 성능평가

5. 결 론

본 논문에서는 임베디드 플래시 파일시스템에서 플래시 메모리의 관리를 위해 순위값을 이용한 새로운 플래시 메모리 지움 정책을 제안했다. 플래시 메모리는 비휘발성의 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 보인다. 또한, 플래시 공간에 대한 지움 평준화를 이룸으로써 사용 수명이 오래가도록 했다.

제안된 RCP 방법은 NOR 형식의 플래시 메모리를 기준으로 설계되고 평가되었지만, 최근에 많이 사용되고 있는 NAND 형식의 플래시 메모리도 같은 방식으로 적용할 수 있다. 또한 제안된 방식은 [1]과 [3]에서 제안하고 있는 hot 블록과 cold 블록을 구분하여 저장하는 방식과 결합하여 이용할 수 있다. RCP 방법은 플래시 메모리의 세그먼트에 대한 지움 비용과 지움 회수만 고려할 뿐이지 hot 블록과 cold 블록을 구분하지 않는다. 저장하는 공간을 구분하여 저장한다면 더 좋은 성능을 보일 것으로 기대한다. 이것은 향후 연구 내용으로 남겨둔다.

비휘발성 메모리의 연구는 최근에 활발히 연구되고 있으며, 특성 면에서 플래시 메모리보다 우수한 성능을 보이는 FRAM(Ferroelectric RAM)은 이미 상용화 단계에 있고, MRAM(Magnetic RAM) 등이 활발히 연구되고 있다. 이러한 새로운 메모리 소자의 개발로 인해 비휘발성의 저장매체에 대한 관리 방법도 새롭게 개발되어야 할 것이다.

참 고 문 헌

[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, 제7권 제3호, pp.56-65, 2000.
[3] 김한준, 이상구, "신뢰성 있는 플래시메모리 저장시스템 구축을 위한 플래시메모리 저장 공간 관리 기법", 정보과학회논문지 : 시스템 및 이론, 제27권 제6호, pp.567-582, 2000.
[4] 박상호, 안우현, 박대연, 김정기, 박승민, "플래시 메모리를 위한 파일시스템 구현", 정보과학회논문지 : 컴퓨팅의 실제, Vol. 7, No.5, pp.402-415, 2001.
[5] 김정기, 박승민, 김채규, "임베디드 플래시 파일시스템", 정보처리학회지, 제9권 제1호, 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 28F128 J3A, 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] Atsuo Kawaguchi, Shingo Nishioka, and Hiroshi Motoda, "A Flash-Memory Based File System," Proc. of USENIX Technical Conference, pp.155-164, 1995.
[11] Intel Corporation, "Understanding the Flash Translation Layer(FTL) Specification," Technical Paper, 1998.
[12] WindRiver, "TrueFFS for Tornado Programmer's Guide 1.0," 1999.
[13] David Woodhouse, "JFFS : The Journaling Flash File System," Proc. of Ottawa Linux Symposium and Technical Paper of RedHat Inc., 2001.

김 정 기
e-mail : jkk@etri.re.kr
1992년 전북대학교 컴퓨터공학과(공학사)
1994년 전북대학교 컴퓨터공학과(공학석사)
1999년 전북대학교 컴퓨터공학과(공학박사)
1996년~1998년 시스템공학연구소 연구원
1998년~현재 한국전자통신연구원 정보가전연구부 선임연구원
관심분야 : 임베디드 시스템, 파일시스템, 병렬 정보검색

박 승 민
e-mail : minpark@etri.re.kr
1981년 울산대학교 전자공학과(공학사)
1983년 홍익대학교 전자공학과(공학석사)
1998년~현재 충남대학교 전자공학과 박사과정
1983년~1984년 (주)LG전자
1984년~현재 한국전자통신연구원 정보가전연구부 실시간미들웨어연구팀장/책임연구원
관심분야 : 정보가전, RTOS, 이동 통신망

김 채 규
e-mail : kyu@etri.re.kr
1978년 고려대학교 수학과(이학사)
1993년 호주 시드니 공과대 전산과학(석사)
1994년 호주 Wollongong대 전산과학(박사)
1997년~현재 한국전통신연구원 정보가전연구부장/책임연구원
관심분야 : 정보가전, 실시간 운영체제, 멀티미디어, 데이터베이스, 전자상거래 등