윈도우즈(Windows) 파일 시스템에서 플래시 메모리의 FTL(Flash Translation Layer) 알고리즘 성능 분석*

박원주[O] 유현석 박성환 김도윤 박상원
한국외국어대학교 정보통신공학과
{wjpark[O], hsyoo, shpark, dykim, swpark}@islab.hufs.ac.kr

Performance Analysis of FTL Algorithms in Flash Memory for Windows File Systems

Won-Joo Park[O] Hyun-Seok Yoo Sung-Hwan Park Do-Yun Kim Sangwon Park
Information and Communication Engineering, Hankuk University of Foreign Studies

요 약

이동 기기의 저장장치로 널리 사용되고 있는 플래시 메모리는 하드웨어적 특성으로 인하여 쓰기 전 소거(erase before write) 기법이 사용되고 있다. 이러한 특성으로 인하여 플래시 메모리에서는 성능을 증진시키기 위한 기법이 필요하게 되었으며, 이러한 소프트웨어 모듈을 FTL이라 한다. 플래시 메모리의 용량이 크게 늘어나면서 디스크를 대체할 제품이 등장하고 있으며, 이러한 디스크가 일반 컴퓨터에서의 저장장치로 채택되는 경우가 많아지고 있다. 본 연구에서는 플래시 메모리 기반의 디스크를 이용한 윈도우 파일 시스템에서의 여러 FTL 알고리즘의 성능을 분석, 비교하고, FTL 알고리즘의 올바른 개선 방향을 제시한다.

1. 서론

최근 디지털 카메라, MP3 플레이어, 핸드폰, 개인 휴대 정보 단말기(PDA)등이 많이 사용되고 있다. 플래시 메모리는 저전력, 비휘발성, 고성능, 물리적 안정성, 휴대성 등의 특징으로 인해 이러한 휴대장치의 저장장치로 많이 사용되고 있다. 특히 플래시 메모리의 용량이 크게 늘어나면서 디스크를 플래시 메모리로 대체하는 경향이 많아지고 있다.

플래시 메모리는 물리적인 특성으로 인하여 쓰기 전 소거(erase-before-write)연산을 실시한다. 이것은 섹터(sector)[1]에 쓰기 연산을 할 경우 그 섹터가 속한 블록(block)[2]을 소거 연산을 통해 지운 후 쓰기 연산을 수행하는 것이다. 그러므로 섹터에 덮어쓰기 연산을 수행할 수 있는 하드 디스크에 비하여 동일한 I/O에 대해 더 많은 소요 시간이 걸릴 수 있다. 또한 플래시 메모리의 블록은 10만 번 정도의 소거 연산을 수행하면 더 이상 소거 연산을 수행할 수 없다. 따라서 소거 연산은 플래시 메모리의 수명을 단축시키는 연산이다.

이러한 특성으로 인해 플래시 메모리의 특정 섹터에 대한 쓰기 연산을 수행하면 플래시에서는 비어있거나 소거 연산의 횟수가 적은 블록을 할당하여 쓰기 연산을 수행한다. 이러한 소프트웨어를 FTL(flash translation layer)라 부르며 FTL은 그림 1에서 보는바와 같이 파일 시스템의 섹터 번호[3]를 플래시의 섹터 번호[4]로 맵핑하는 기능을 수행한다. FTL 알고리즘은 이러한 소거 연산을 가능하면 적게 해야 할 것이고, 가능하면 한 개의 블록에 집중 되지 않도록 해야 한다.

기존의 FTL 알고리즘들은 소형 기기에서 동작하는 것을 목표로 하여 만들어졌다. 본 논문에서는 플래시가 디스크를 대체하여 컴퓨터의 보조 기억 장치로 사용될 경우 각 FTL 알고리즘의 성능을 분석하였다. 그림 1에서 보는 바와 같이 윈도우즈 파일 시스템에서 디스크에 대한 I/O를 수행할 경우 FTL은 맵핑 테이블을 이용하여 LSN을 PSN으로 변환하게 되며 필요한 여러 연산을 수행하게 된다. 파일 시스템에서 디스크에 요청하는 I/O 연산을 가로채어 기존의 여러 FTL 알고리즘에서 어떻게 동작하는지 분석하였다.

* 본 논문은 2005년도 한국외국어대학교 학술연구비 지원에 의해서 연구되었음

1) 보통 512 바이트
2) 여러 개 섹터의 집합
3) 논리적 섹터 번호 (LSN, logical sector number)
4) 물리적 섹터 번호(PSN, physical sector number)

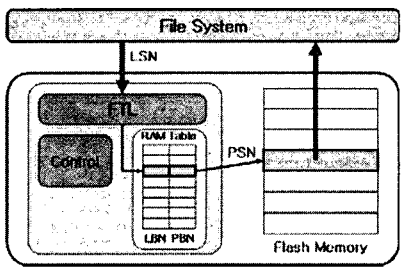

그림 1 플래시 메모리의 구조

본 논문에서는 소거 연산의 횟수와 읽기/쓰기/소거 연산의 수행 시간을 비교하였다.

2. 관련 연구

FTL알고리즘에서 사용되는 맵핑은 섹터 맵핑, 블록 맵핑, 하이브리드(hybrid) 맵핑으로 나누어지는데 섹터 맵핑은 섹터 단위로 맵핑 테이블을 만들어 물리적인 섹터와 논리적인 섹터를 맵핑하는 것이고, 블록 맵핑은 물리적인 블록과 논리적인 블록을 맵핑하는 것이다. 하이브리드 맵핑은 두가지 맵핑을 동시에 사용하는 것으로 두 가지 맵핑의 장점을 취할 수 있다. 이러한 맵핑 테이블은 플래시 컨트롤러의 RAM에 저장된다.

한 블록에 저장된 섹터들이 정해진 특정 위치에 고정되는 것을 고정 섹터 방식(in-place)이라 부르며, 섹터들이 블록 내의 임의의 위치에 존재하는 것을 변동 섹터 방식(out-of-place)라 부른다.

또한 각 알고리즘은 내부적으로 기본적인 읽기/쓰기/소거 연산 외에 필요에 따라 교환(switch) 연산이나 합병(merge) 연산, 할당(allocate) 연산 등이 있다. 교환 연산은 블록의 내용을 변화시키지 않은 채 원래 블록을 소거 하고, 블록 맵핑 테이블의 값만 바꾸어주는 연산이다. 합병 연산은 두 개의 블록에서 최신 정보만 복사하여 새로운 블록에 쓰고, 맵핑 테이블을 새로운 블록으로 바꾸어 준 다음 합병에 사용된 두 개의 블록을 소거하는 연산이다. 할당 연산은 빈 블록이 필요할 경우 빈 블록을 탐색, 반환하는 연산이다. 보통은 라운드 로빈(round robin)방식을 사용한다.

본 논문에서는 여러 FTL 알고리즘 중 대표적인 다섯 가지

에 대하여 그 특성을 분석하였다. 선정된 알고리즘은 Misubish사의 알고리즘(MSBS)[1]과 M-System의 FMAX[2], Lexar Media사의 알고리즘(LM)[3], 로그블록을 사용한 알고리즘(로그 블록 기법[4]과 FAST[5])으로 각각을 분석하였다. 표 1은 각 알고리즘의 특징을 나타낸 것이다.

표 1 알고리즘의 특성 비교

	MSBS	FMAX	LM	로그블록기법	FAST
데이터	in place	in place	in place	inplace	in place
맵핑	Block	Block	Block	Hybrid	Hybrid
덮어쓰기	out of place	in place → out of place	in place	out of place	in place & out of place
특징	블록 내의 Spare area를 사용	각 블록당 1:1로 복사 블록을 사용	논리적인 블록중 하나를 이동블록으로 사용	한 블록에 로그 블록 하나를 할당	로그블록을 순차쓰기용과 임의쓰기용으로 구분

3. 실험

3.1 실험 환경

현재 휴대용 제품의 운영체제로 윈도우즈 계열이 많이 사용되고 있다. 따라서 본 연구에서는 윈도우즈 운영체제에서 많이 사용하는 파일 시스템인 FAT과 NTFS에서 디스크로 보내지는 I/O를 캡처하여 실험하였다. I/O 캡처를 위해 윈도우즈 디바이스 드라이버 중 디스크 드라이버의 하위 필터 드라이버를 작성하여 설치하였다. 디바이스 드라이버는 커널영역에서 동작하는 코드이므로 안전성을 위해 가상 머신[5]을 사용하고 캡처 드라이버는 I/O에 대해 읽기와 쓰기 연산을 구분한 구분자와 시작 섹터, 길이를 공유영역에 저장하였다. 실험의 편의를 위해 캡처한 I/O 데이터를 파일에 저장하여 실험하였다.[6]

3.2 실험 도구

Log-File Creater(LFC)를 작성하여 I/O 캡처 필터 드라이버가 설치된 디스크에서 로그 파일을 만든 후 각 알고리즘에 적용하여 실험하였다.

표 2 LFC

디스크		100M 바이트
파일	MP3	150M 바이트, 평균 4.7M 바이트, 1500회
	JPG	150M 바이트, 평균 240K 바이트, 1500×19회
방법		● 무작위로 파일을 선택하여 복사 및 삭제 ● 복사 및 삭제 연산 비율을 1:1로 함 ● 단일 및 다중 프로세스 환경에서 추출

FTL 시뮬레이터는 LFC에서 캡처한 I/O 파일을 실시간으로 읽어서 원하는 FTL 알고리즘으로 실험하여 각 I/O당 플래시 메모리에서 발생하는 읽기/쓰기/소거의 횟수를 측정하였다.

3.3 실험 방법

VMware에서 가상으로 만든 디스크에 I/O 캡처 필터 드라이버를 설치한 뒤 LFC를 이용하여 로그 파일을 만들고 시뮬레이터로 로그 파일을 이용하여 FTL 알고리즘을 실험하였다. 실험의 편의를 위해 각 로그파일을 미리 생성하여 저장해두고, 시뮬레이터로 여러 개의 알고리즘을 실험하였다.

4. 실험 결과

소요시간은 NAND 플래시 메모리의 읽기를 기준으로 1:7:63의 상대적인 시간 비율로 각 읽기/쓰기/소거 연산 횟수를 곱하여 계산하였다. 그리고 소요시간에 가장 큰 영향을 미치는 소거 횟수를 비교하였다.

4.1 파일 시스템에 따른 분석

NTFS는 FAT보다 보안성, 안정성 면을 더 향상하기 위해 파일 시스템에서 더 많은 메타 데이터를 사용하고 있다. 그래서 메타 데이터에 대한 덮어쓰기가 더 많이 발생한다. 그림 5와 그림 6에서 I/O의 양은 NTFS가 더 많지만 모든 I/O가 종료될 때까지의 소요시간은 비슷하다. 따라서 모든 알고리즘은 메타 데이터의 I/O 때문에 발생하는 소거와 소거에 따른 소요시간에서 차이를 보인다. 특히 LM과 로그 블록 기법은 메타 데이터에 대한 I/O가 적을수록 같은 데이터 블록에 대한 덮어쓰기가 줄어들기 때문에 이동블록과 로그블록의 소거횟수가 크게 작아진다.

4.2 파일 크기에 따른 분석

그림 3과 그림4에서 JPG파일을 복사/삭제한 로그 파일이 MP3 파일을 복사/삭제한 로그 파일보다 더 많은 I/O가 발생한 것을 알 수 있다. JPG 파일은 크기가 작기 때문에 더 많은 횟수로 파일을 복사/삭제 실시했고, 이 때문에 각 파일의 메타 데이터에 대한 쓰기 혹은 덮어쓰기가 더 많이 발생하였다.

4.2.1 MSBS, FMAX

각각의 블록은 모두 복사 블록 혹은 여유 공간(spare area)을 가지기 때문에 순차적인 쓰기나 임의적인 쓰기에 거의 관계없이 일정한 성능을 보인다. 따라서 소거횟수와 소요시간이 파일 크기에 따라 큰 차이를 보이지 않는다.

4.2.2 LM, FAST

그림 3과 4에서 이 두 가지 알고리즘은 순차적인 덮어쓰기 대하여 고정 섹터 방식으로 대응하기 때문에 소거 횟수를 많이 줄 일 수 있다. 하지만 LM에서는 항상 고정 섹터 방식을 사용하기 때문에 JPG의 경우에 소거횟수가 많이 증가 하였다.

4.2.3 로그 블록 기법

로그블록을 사용하기 때문에 FAST와 거의 유사한 소거 횟수를 보인다. 임의적인 덮어쓰기가 많이 발생하는 JPG의 경우 메타 데이터에 대한 소거횟수가 증가하여 MP3의 경우보다 소요시간에서 차이를 보인다.

표 3 전체 평균 및 표준편차

단위 I/O (100개)	평균				표준편차			
	읽기	쓰기	소거	소요시간	읽기	쓰기	소거	소요시간
MSBS	147214	20549	629	330688	28633	3226	161	60288
FMAX	64077	21801	647	257466	10271	3454	162	44215
LM	35083	33520	739	316291	5492	5238	158	51723
로그블록기법	22887	23089	384	208719	2900	2965	64	27306
FAST	23144	22365	376	203358	3121	3013	70	28403

4.3 안정성 분석

안정성은 I/O에 대해 소요시간이 거의 일정해야 함을 말하는데 표 3의 표준편차에서처럼 로그블록을 사용하는 것이 다른 알고리즘보다 편차가 매우 적다. 또, 하나의 로그블록을 소거할 경우 여러 개의 데이터 블록이 소거가 되는 FAST보다 하나의 로그블록과 하나의 데이터 블록만 소거하면 되는 로그블록 기법이 소요시간 편차가 더 작다.

5) 본 실험에서는 VMware를 사용하여 실험을 하였다.

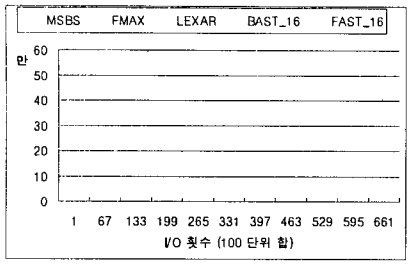

그림 2 FAT JPG 소거횟수 누적

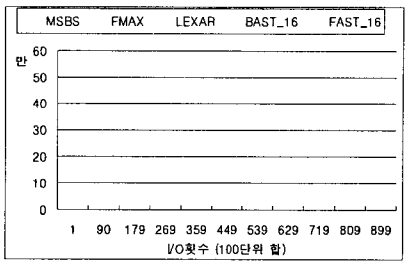

그림 3 NTFS JPG 소거횟수 누적

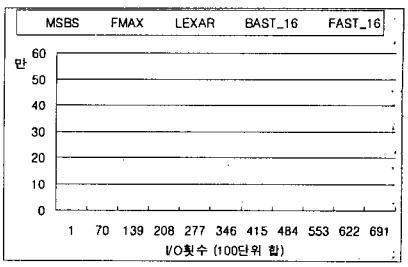

그림 4 NTFS MP3 소거횟수 누적

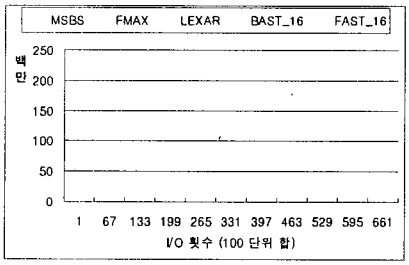

그림 5 FAT JPG 소요시간 누적

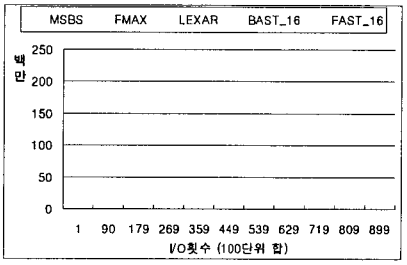

그림 6 NTFS JPG 소요시간누적

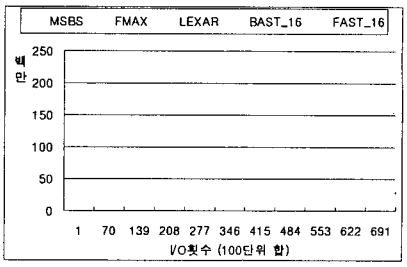

그림 7 NTFS MP3 소요시간누적

4.4 알고리즘 특성에 따른 분석

4.4.1 LM 대 MSBS

그림 7의 소요시간에서 LM이 MSBS보다 더 작은 이유는 표 3의 평균 소거 횟수와 평균 쓰기 횟수에서 LM이 더 많지만, 평균 읽기 횟수에서 큰 차이로 적기 때문이다. 이는 덮어쓰기에 대해 변동 섹터 방식을 사용 하면 I/O에 대하여 더 많은 읽기 연산을 유발함을 말한다.

대 로그 블록 기법

그림 4와 그

4.4.2 FAST 림 5에서 보듯이 로그블록을 사용하는 두 알고리즘은 소거횟수에서는 거의 차이를 보이지 않는다. 하지만 데이터 블록에 로그 블록의 정보를 기록할 필요가 없는 FAST가 각 블록에 로그 블록의 정보를 기록해야 하는 로그 블록 기법보다 쓰기 측면에서 더 좋은 성능을 보임을 표 3에서 확인 할 수 있다. 또한, FAST가 여러 개의 블록에 대한 덮어쓰기를 하나의 로그블록에 기록하여 좀 더 로그블록을 효율적으로 사용하기 때문에 평균 소거횟수와 평균 소요시간이 더 작다.

4.5 종합분석

전체적인 성능은 전체 소거횟수, 전체 소요시간 과 평균 소거 횟수, 평균 소요시간, 각각의 표준편차 등 모든 면에서 FAST가 절대적으로 좋다. 이는 하이브리드 맵핑을 통해서 순차적인 덮어쓰기와 임의적인 덮어쓰기에 대해 가장 효과적으로 대응하기 때문이다.

파일 시스템은 각 알고리즘의 성능에 영향을 주지 않지만 NTFS를 쓰면 전체 I/O양이 늘어나므로 가능한 적은 양의 I/O를 위해 FAT을 쓰는 것이 좋다. 같은 양의 파일을 복사/삭제 할 경우 파일 크기가 크면 순차적인 쓰기/덮어쓰기가 발생하므로 더 적은 소거횟수를 가진다.

모든 알고리즘들은 디스크 대체용으로 플래시 메모리가 사용되었을 때에 발생할 수 있는 대용량화, 고속화에 대한 해결 방안을 모색해야 한다.

5. 향후 발전 방향

우선 종합 분석에서 가장 좋은 성능을 보이는 FAST는 제한적인 크기의 로그블록을 가진다는 점과 임의쓰기용 로그블록을 원형 큐 방식으로 동작하는 점을 보완 한다면 좀 더 나은 알고리즘으로 발전 할 수 있다. 로그블록 기법은 로그 블록을 좀 더 효율적으로 사용할 수 있는 방안을 찾아야 한다. 다른 알고리즘 들은 FAST처럼 하이브리드 맵핑 방식을 통해서 여러 가지 I/O 패턴에 대해 효과적으로 대처 할 수 있도록 해야 한다. FMAX와 MSBS는 논리적으로 제공되는 플래시 메모리의 양보다 더 많은 양의 물리적인 플래시 메모리를 제공해야 하는 점을 보완해야 한다.

실험에 사용한 알고리즘들은 하나의 섹터에 대한 I/O를 수행하고 있다. 디스크 I/O의 인자는 시작 섹터 번호와 섹터의 개수이기 때문에 디스크 대체용 플래시 메모리는 디스크처럼 섹터 개수를 고려하여 I/O를 수행하는 방법이 추가되어야 하고 대용량화에 따른 동작도 고려해야 한다.

마지막으로 플래시 메모리의 수명과 관계되는 소거연산 평준화(wear leveling)를 각 알고리즘 별로 잘 구현하여 각 블록의 평균 수명을 일정하게 유지하고 대역폭을 높인다면 디스크를 대체하였을 경우 좋은 성능을 보여줄 것이다.

참고문헌

[1]Takayuki Shinohara. Flash memory card with block memory address arrangement, 1999. United States Patent, no. 5,905,993.

[2]Amir Ban. Flash file system optimized for page-mode flash technologies, 1999. United States Patent, no. 5,937,425.

[3]Petro Estakhri and Berhanu Iman. Moving sequential sectors within a block of information in a flash memory mass storage architecture, 1999. United States Patent, no. 5,930,815.

[4]Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min, and Yookun Cho. A space-efficient flash translation layer for compact flash systems. IEEE Transactions on Consumer Electronics, 48(2), 2002.

[5]Sang-Won Lee and Dong-Joo Park. FAST:An efficient flash translation layer for flash memory, Submitted for publication, 2005.

[6]Sung-Hwan Park, Hyun-Seok Yoo, Do-Yun Kim, Won-Joo Park, Kisun Oh, Sang-Won Lee and Sangwon Park. Development of performance analysis tool for flash disk on windows platform, http://islab.hufs.ac.kr/flash, 2005.