데이터 무결성을 보장하는 플래시 저장 장치에서 잦은 쓰기 참조 흡수가 플래시 변환 계층에 미치는 영향

(The Effect of Absorbing Hot Write References on FTLs for Flash Storage Supporting High Data Integrity)

심 명 섭 † 도 인 환 ††
(Myoung Sub Shim) (In Hwan Doh)

문 영 제 †† 이 효 정 ††
(Young Je Moon) (Hyo J. Lee)

최 종 무 ††† 이 동 희 ††††
(Jongmoo Choi) (Donghee Lee)

노 삼 혁 †††††
(Sam H. Noh)

· 이 논문은 2007년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. R0A-2007-000-20071-0)
· 이 논문은 제35회 추계학술대회에서 '데이터 무결성을 보장하는 파일 시스템의 행동이 플래시 변환 계층들에 미치는 영향'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : 홍익대학교 컴퓨터공학과
sms1222@dreamwiz.com
†† 학생회원 : 홍익대학교 컴퓨터공학과
ihdoh@necsst.ce.hongik.ac.kr
lemonkenya@naver.com
hjlee0215@gmail.com
††† 종신회원 : 단국대학교 컴퓨터과학과 교수
choijm@dku.edu
†††† 종신회원 : 서울시립대학교 컴퓨터과학과 교수
dhl_express@uos.ac.kr
††††† 종신회원 : 홍익대학교 컴퓨터공학과 교수
samhnoh@hongik.ac.kr
논문접수 : 2009년 9월 2일
심사완료 : 2009년 11월 10일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제3호(2010.3)

요 약 플래시 저장장치는 컴퓨팅 시스템에서 휴대용 저장매체로 각광 받고 있다. 플래시 저장장치의 착탈성을 고려해 보면, 데이터의 무결성이 중요한 이슈로 부각된다. 본 연구는 데이터 무결성을 보장하려는 파일시스템 동작이 플래시 변환 계층(FTL) 기법들의 성능에 미치는 영향에 주목한다. 본 연구에서는 파일시스템이 데이터 무결성을 보장하기 위해서 발생시킨 잦은 쓰기 참조가 플래시 저장장치에 미치는 영향에 대해서 살펴본다. 또한, 비휘발성 램을 이용한 잦은 쓰기 참조의 흡수가 플래시 저장장치 내의 FTL 성능에 미치는 영향을 살펴본다. 실제 시스템 환경에서 실시된 성능 평가 결과는 잦은 쓰기 참조가 포함된 워크로드들에서 FTL 성능이 기존 연구에서 제시된 결과와 상이할 수 있음을 보여준다. 이와 더불어, 비휘발성 램을 이용하여 잦은 쓰기 참조를 흡수함으로써 FTL 기법들이 플래시 저장장치의 성능에 미치는 영향이 완화됨을 실험 결과를 통해서 알 수 있다.
키워드 : 플래시 변환 계층, 비휘발성 램, 쓰기 캐시

Abstract Flash storages are prevalent as portable storage in computing systems. When we consider the detachability of Flash storage devices, data integrity becomes an important issue. To assure extreme data integrity, file systems synchronously write all file data to storage accompanying hot write references. In this study, we concentrate on the effect of hot write references on Flash storage, and we consider the effect of absorbing the hot write references via nonvolatile write cache on the performance of the FTL schemes in Flash storage. In so doing, we quantify the performance of typical FTL schemes for workloads that contain hot write references through a wide range of experiments on a real system environment. Through the results, we conclude that the impact of the underlying FTL schemes on the performance of Flash storage is dramatically reduced by absorbing the hot write references via nonvolatile write cache.
Key words : Flash memory, FTL, NVRAM, Write cache

1. 서 론

플래시 메모리 기술이 성장함에 따라 USB 플래시 드라이브와 SD(Secure Digital) 카드와 같은 플래시 저장장치는 다양한 컴퓨팅 환경에서 널리 사용된다. 플래시 저장장치는 일반적으로 플래시 메모리를 효율적으로 관리하기 위해 FTL을 채택한다. FTL의 효율성은 플래시 저장장치의 성능에 큰 영향을 미치기 때문에, FTL의 성능을 향상 시키려는 많은 연구들이 기존한다[1,2]. 입출력 성능 향상에 대한 고려뿐 아니라, 갑작스런 정전이나 플래시 메모리의 제거와 같은 시스템 오류에 대해서 플래시 메모리에 저장된 데이터의 무결성을 보장하는 것도 중요하게 여겨진다.

데이터 무결성을 보장하려는 파일시스템의 노력은 플래시 저장장치에서 사용하는 FTL 기법에 따라서 플래시 저장장치의 입출력 성능에 미치는 영향이 다를 수 있다. 데이터 무결성을 보장하기 위해서 발생되는 파일들에 대한 동기식 쓰기 요청은 파일시스템의 메타데이터에 대한 잦은 쓰기를 동반한다. 이 잦은 쓰기 참조(Hot write reference)가 특정 FTL 기법들에서는 문제가 되지 않을 수도 있지만, 다른 FTL 기법들에서는 극심한 성능저하를 초래할 수 있다. 이처럼 플래시 저장장치의 성능에 악영향을 끼칠 수 있는 잦은 쓰기 참조들은 비휘발성 쓰기 캐시를 통해서 제거될 수 있다. 다양한 방식으로 도입될 수 있는 비휘발성 쓰기 캐시는 데이터 무결성을 보장함과 동시에 입출력 성능저하를 방지할 수 있다.

본 연구에서는 데이터 무결성을 보장하기 위해서 파일시스템이 발생시키는 잦은 쓰기 참조가 서로 다른 FTL 기법들의 성능에 미치는 영향을 정량화 하고자 한다. 동시에, 잦은 쓰기 참조를 제거하기 위한 비휘발성 쓰기 캐시의 도입이 각 FTL 기법들의 성능에 어떠한 영향을 초래하는지를 실험적으로 살펴보고자 한다. 이를 위하여, 실제 시스템 환경에서 VFAT 파일시스템이 데이터 무결성을 보장하기 위해서 매번 동기식으로 쓰기 요청을 할 때, 서로 다른 FTL 기법들에 의해서 관리되는 플래시 저장매체의 입출력 성능을 측정한다. 동일한 실험 환경에서 파일시스템 메타데이터에 대한 잦은 쓰기 요청이 비휘발성 램에 의해 흡수될 때, 각 FTL 기법들의 입출력 성능 변화를 분석한다.

실제 환경에서의 구현과 성능 평가를 통해서 다음과 같은 두 가지 흥미로운 관찰 결과를 제시한다. 첫째, 잦은 쓰기 참조가 포함된 특정 워크로드에 대해서 실제 구현된 FTL 기법들의 성능이 기존 연구에서 제시된 결과와 다소 차이가 있음을 보여준다. 구체적으로, 파일 데이터의 동기식 쓰기로 인해 발생한 잦은 쓰기 참조들이 특정 상황에서는 로그블록 기법(Log-block Scheme)이 교체블록 기법(Replacement-block Scheme)보다 안 좋은 성능을 보이고, 어떤 상황에서는 FAST(Full Associative Sector Translation) 기법이 로그블록 기법보다 낮은 성능을 보이는 상황이 관찰된다. 둘째, 비휘발성 쓰기 캐시를 통하여 잦은 쓰기 참조들을 흡수함으로써 각 FTL 기법들의 성능이 급격히 향상될 뿐만 아니라, 동시에 여러 FTL 기법들 간의 성능 격차가 현저하게 줄어든다는 사실을 관찰한 바 있다.

본 논문의 구성은 다음과 같다. 2절에서는 본 연구에서 고려하는 FTL 기법들에 대해서 언급한다. 3절에서는 FTL 기법의 구현과 잦은 쓰기 참조들을 흡수하는 방법에 대해 설명한다. 4절은 하드웨어 설정 그리고 실험에 이용된 워크로드에 대해 설명한다. 실험 결과는 5절에서 논의하고, 마지막으로 6절에서 전체적인 요약과 앞으로의 연구 방향과 함께 결론을 맺는다.

2. 배경지식

기존에 플래시 변환 계층(FTL)의 성능을 향상시키기 위한 많은 연구들이 있었다. 이런 FTL 기법들은 블록 사상(Block-mapped) FTL, 페이지 사상(Page-mapped) FTL, 그리고 하이브리드 사상(Hybrid-mapped) FTL로 크게 세 가지 형태로 분류된다[1-4]. 다양한 FTL 기법 중에서 대표적인 FTL 기법들로 널리 알려진 교체블록 기법, 로그블록 기법, 그리고 FAST 기법만을 본 연구에서 고려한다(각 FTL 특징은 [1-3] 참조).

일반적으로 FTL 기법들의 성능은 파일시스템의 행동과 밀접한 관련이 있다. 그러나 대부분의 FTL 기법 연구들이 시뮬레이션 환경에서 이루어졌고, 파일시스템의 행동과 독립적으로 FTL 기법들의 성능에 집중하였다. 이런 기존의 연구들과 달리 본 연구에서는 실제 시스템 환경에서 FTL 기법들을 평가한다. 특히, 파일시스템이 데이터 무결성을 추구하는 상황에서 각 FTL 기법의 성능을 평가한다.

본 연구와 밀접한 관련이 있는 또 다른 연구 분야는 쓰기 캐시에 관한 연구이다. 최근에 플래시 메모리에 대한 쓰기 캐시로써 비휘발성 램을 고려하고, 쓰기 캐시 정책이 FTL의 성능에 미치는 영향을 다각도로 분석한 연구가 발표된 바 있다[5]. 본 연구는 쓰기 캐시의 정책에 관한 연구이기 보다는 오히려 실제 구현과 실험을 통해서 자주 참조되는 쓰기 요청들이 흡수될 때 FTL의 성능 변화를 관찰하고 흥미로운 관찰 결과를 제시하는 데 초점을 둔다. 이런 면에서 본 연구는 기존의 쓰기 캐시 정책에 관한 연구와는 차별된다.

3. 구 현

본 절에서는 연구 수행을 위해서 구현된 내용에 대해서 기술한다.

3.1 FTL 구현

본 실험에서 고려하는 FTL 기법들을 Linux 2.6.21에서 구현하였다. 교체블록 기법은 Linux 2.6.21에서 포함된 M-Systems에서 만든 플래시 저장장치를 위해 제공된 NFTL 모듈을 수정하였다. 로그블록 기법과 FAST 기법은 MTD(Memory Technology Device) 계층상에 해당 논문의 설명에 기초하여 각각 구현하였다[1,2]. 다른 점이 있다면, 본 연구에서는 로그블록 기법에 한 가지 최적화 기법을 도입하였다. 이 최적화 기법은 병합 연산 시, 로그블록의 데이터가 모두 유효하고 순차적인 경우에 로그블록의 빈 페이지들을 해당 데이터 블록에서 가

져와 스위치 연산을 유도하는 방법이다. 이 방법은 FAST에 도입되어 있다[2]. 이는 데이터의 복사 오버헤드를 줄이고, 삭제 연산도 1번으로 줄일 수 있다. 모든 실험에서 로그블록과 교체블록의 개수를 64개로 고정한다.

3.2 비휘발성 쓰기 캐시

본 연구는 잦은 쓰기 참조들과 비휘발성 쓰기 캐시를 통한 잦은 쓰기 참조들의 흡수가 FTL 기법들에 미치는 영향에 주목한다. 데이터 무결성을 보장할 때 발생하는 이 잦은 참조들은 다양한 방법의 비휘발성 쓰기 캐시를 통해 흡수될 수 있다. 특히, 파일시스템 계층[6], 블록 디바이스 드라이버 계층[7], 그리고 FTL 계층[8]에 캐시를 도입할 수 있고, 최근에 접근된 데이터, 자주 접근되는 데이터, 그리고 파일시스템의 메타데이터를 잦은 참조로 간주하는 등 다양한 방법을 통해 잦은 참조를 선택하여 흡수 할 수 있다[7,9,10].

본 연구에서는 많은 플래시 저장장치가 FAT 파일시스템으로 미리 포맷되어 있기 때문에 호스트 파일시스템으로 리눅스에서 제공하는 VFAT 파일시스템을 사용한다. 또한, 데이터 무결성을 보장하기 위해 응용프로그램은 항상 파일을 동기식으로 쓴다고 가정한다. 본 연구에서 고려하는 쓰기 캐시는 파일시스템 계층에 위치하고, 비휘발성 매체로는 비휘발성 램인 FeRAM(Ferro-electronic RAM)을 사용한다[11]. 본 논문에서는 파일시스템 메타데이터를 빈번하게 참조되는 데이터로 선택하였다. 따라서 VFAT 파일시스템을 수정하여 파일시스템의 메타데이터만을 비휘발성 램에 쓰기 캐시하도록 한다.

4. 실험 환경

이번 절에서는 본 실험에 사용된 하드웨어와 소프트웨어 실험 환경에 대해서 설명한다.

4.1 하드웨어 환경

성능 평가를 위해, 각 FTL 기법은 실제 임베디드 시스템 개발 보드에서 동작한다. 이 임베디드 개발 보드는 64MB SDRAM과 64MB 플래시 메모리를 가지고 있다[12]. 파일시스템은 항상 플래시 메모리 32MB 파티션을 마운트하여 사용한다. 이 플래시 메모리는 512B 데이터 페이지들과 16B의 예비 영역(Spare area)으로 구성되어 있다. 비휘발성 램 도터 보드는 최대 64MB의 FeRAM을 장착할 수 있다. 하지만 실험에서 실제로 사용된 FeRAM의 최대 용량은 128KB이다. 비휘발성 램은 물리 메모리 주소 공간 내에 표시되며 메모리 사상 주소 접근 방식을 통해 CPU가 직접 비휘발성 램에 접근할 수 있다.

4.2 파일시스템 워크로드

본 연구에서 성능평가을 위해 Camera-MS500 벤치마크와 PostMark 벤치마크를 사용한다.

Camera-MS500: 아직까지 임베디드 시스템의 특성을 고려한 검증된 파일시스템 워크로드가 존재하지 않기 때문에 본 연구팀에서 모바일 폰에 탑재된 카메라 모듈의 동작을 시뮬레이션 하였다. Sourceforge 프로젝트의 일환으로써 발표된 BItPim 툴을 이용하여 사진을 찍고 삭제한 후의 파일시스템의 변화를 관찰하였다[13]. 이 관측 결과를 이용하여 디지털 카메라에 대한 파일시스템 워크로드를 생성하는 Camera-MS500 벤치마크 프로그램을 구현하였다. 이 벤치마크는 초기에 파일을 가지고 있지 않은 상태에서 특정 개수만큼의 트랜잭션을 수행한다. 본 실험에서는 트랜잭션의 수를 10으로 설정하였다. 하나의 트랜잭션 동안에 이 벤치마크는 300KB에서 500KB 사이의 크기를 갖는 20 개의 사진 파일들을 생성하고 생성된 파일 중 절반을 삭제한다. 만약 파일의 총 개수가 60개를 넘어가면 여유 공간을 확보하기 위해 모든 파일들을 삭제한다. 본 실험의 설정에서 Camera-MS500 벤치마크가 생성한 전체 파일 용량(Footprint)은 79.67MB이다. 데이터의 무결성을 보장하기 위해 각 사진 파일과 사진 인덱스 파일은 512B 단위로 동기식으로 기록된다.

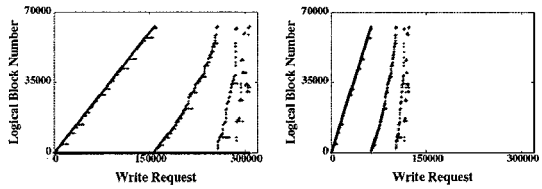
PostMark: 인터넷 서버의 워크로드를 생성하는 이 PostMark은 디스크 기반 파일시스템의 성능을 측정하는데 널리 사용된다[14]. 본 실험에서 사용한 PostMark 벤치마크 v1.5는 초기 512B에서 32KB 사이의 임의의 크기를 갖는 500개의 초기 파일과 10개의 디렉터리를 생성한다. 이 초기 파일을 생성한 이후에 PostMark 벤치마크는 특정 개수만큼의 트랜잭션을 수행한다. 본 실험에서는 10,000번의 트랜잭션을 수행한다. 본 실험에서 PostMark 벤치마크가 생성한 전체 파일 용량은 113.29 MB이다. 높은 수준의 데이터 무결성을 보장하기 위해 모든 파일과 메타데이터를 동기식으로 쓰도록 PostMark를 수정하였다.

5. 성능 평가

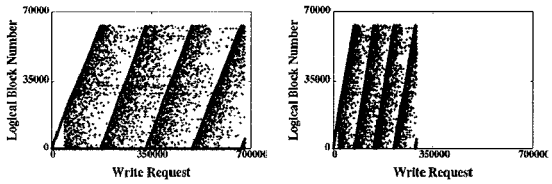
본 절에서는 실험에서 사용된 워크로드들의 쓰기 참조 패턴에 대해 논의하고 잦은 쓰기 참조들의 제거가 FTL 기법들에 미치는 영향에 대해서 살펴본다.

5.1 쓰기 참조 패턴

그림 1에서 Camera-MS500 벤치마크와 PostMark가 만들어낸 쓰기 참조 패턴을 볼 수 있다. 그림 1(a)는 Camera-MS500 벤치마크가 수행되는 동안 진행된 쓰기 요청(x축)과 참조된 블록들의 논리 블록 번호(LBN)(y축)를 나타낸다. 그림의 왼쪽 부분은 잦은 쓰기 참조들을 포함하고, 오른쪽 그림은 비휘발성 램을 이용하여 잦은 쓰기 참조들을 흡수한 워크로드에 대한 결과이다.

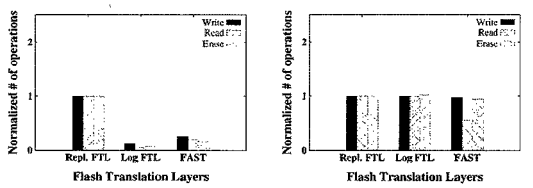


(a) Camera-MS500

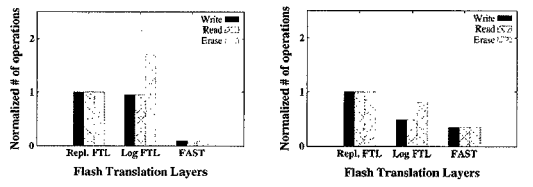


(b) PostMark

그림 1 쓰기참조 패턴: 잦은 쓰기 참조들을 흡수하기 전(왼쪽)과 후(오른쪽)의 모습

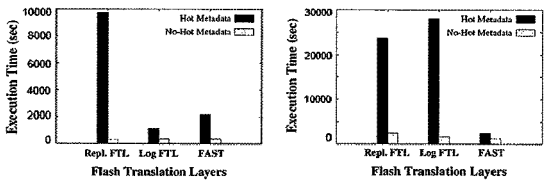


(a) Camera-MS500



(b) PostMark

그림 2 FTL 내부에서 수행된 Write /Read/Erase 연산 횟수(교체 블록 기법에 평준화): 잦은 쓰기 참조들을 흡수하기 전(왼쪽)과 후(오른쪽)의 연산 횟수



(a) Camera-MS500 (b) PostMark

그림 3 각 FTL에 대한 수행 시간: 잦은 쓰기 참조들을 흡수하기 전(Hot Metadata)과 후(No-Hot Metadata)

Camera-MS500 벤치마크는 낮은 섹터 번호에 해당되는 파일시스템 메타데이터에 대해 잦은 쓰기 참조와 함께 순차적인 쓰기 참조들을 만들어낸다. 이 형상이 그림 1(a)의 왼쪽 그림에 해당한다. 비휘발성 쓰기 캐시를 통해 잦은 쓰기 참조들을 흡수한 후, 쓰기 요청 횟수가 급격히 감소된다(그림 1(a) 오른쪽). 그림 1(a)의 오른쪽 그래프에서 x축의 밑 부분의 점들이 사라진 것을 볼 수 있다.

그림 1(b)는 PostMark에 대한 쓰기 참조 패턴을 보여준다. 이 그림을 보면 PostMark 벤치마크가 Camera-MS500 벤치마크보다 더 집중적으로 파일시스템 메타데이터와 파일 데이터를 쓴 것을 볼 수 있다. PostMark 벤치마크에서도 Camera-MS500 벤치마크가 보여준 것과 비슷한 모습이 관측된다. 게다가 비록 PostMark 벤치마크가 만들어낸 워크로드가 순차적 참조와 임의적 참조를 포함하고 있지만, 잦은 쓰기 참조들이 포함되어 있을 때 이 쓰기 참조 패턴은 임의 참조 패턴에 가깝게 된다. 이 잦은 쓰기 참조들을 흡수하는 것이 임의적 쓰기 패턴을 약화시키게 된다.

5.2 잦은 쓰기 참조 흡수에 의한 효과

그림 2는 Camera-MS500 벤치마크와 PostMark 벤치마크가 수행되는 동안에 FTL이 플래시 저장장치로 요청한 쓰기/읽기/소거 연산 횟수를 보여준다. 이 그래프들에서는 연산 횟수들 사이의 차이가 너무 커서 교체 블록 기법을 기준으로 그 비율적 차이를 보여준다. 이 워크로드들은 순차적 쓰기 패턴과 파일시스템 메타데이터의 잦은 업데이트를 포함하고 있고, 이 잦은 쓰기 참조들은 비휘발성 램 쓰기 캐시를 통해 흡수할 수 있다.

먼저 Camera-MS500 벤치마크에 대해 각 FTL 기법의 성능에 대해 논의한다. 로그블록 기법이 잦은 쓰기 참조들이 섞여 있는 워크로드에서 교체블록 기법과 FAST 기법에 대해 더 좋은 성능을 보인다. 예상한대로, 교체블록 기법은 잦은 쓰기 참조들에 취약하고, 로그블록 기법과 FAST 기법 둘 다 잦은 쓰기 참조들을 잘 극복한다. 그러나 기대와 달리 FAST 기법이 이 워크로드에서 로그블록 기법보다 안 좋은 성능을 보인다. 로그블록 기법은 하나의 데이터 블록에 대해 하나의 로그블록을 갖는 반면, FAST 기법은 플래시 메모리 내의 모든 데이터 블록에 대해 하나의 공유된 로그블록을 갖게 된다. FAST 기법은 로그블록에 잦은 쓰기 참조들과 다른 데이터 블록이 섞여있어, 로그블록과 데이터 블록의 스위치 병합 연산이 적게 발생하는 것으로 분석된다.

이처럼 각 워크로드에 대한 FTL의 읽기/쓰기/소거 연산 횟수의 차이는 각 FTL의 수행성능 차이로 실제 임베디드 시스템 기반의 실험 환경에서 나타난다. 그림 3은 Camera-MS500 벤치마크와 PostMark 벤치마크에 대해서 각 워크로드를 모두 수행하는데 소요된 시간을 FTL별로 보여준다. 그림 3(a)에서 나타난 바와 같이 Camera-MS500 워크로드에 대해서, 비휘발성 램을 통해 잦은 쓰기 참조들이 흡수 될 때 각 FTL 기법의 성

능이 적게는 3배 많게는 30배나 향상 된다. 구체적으로, 교체 블록 기법의 성능은 30배 향상되고, 로그 블록 기법과 FAST 기법은 각각 3.2배와 6.6배 좋아진다. 또한, Camera-MS500 워크로드에 포함되어 있는 잦은 쓰기 참조들이 비휘발성 램을 통해 흡수 될 때, 서로 다른 FTL 기법들의 성능 차이가 거의 무시할 정도로 비슷해진다. 엄밀히 말해, 교체블록 기법이 잦은 쓰기 참조들을 흡수한 후에 로그블록 기법보다 더 좋은 성능을 보였다. 이 이유는 잦은 쓰기 참조들을 흡수함으로써 로그블록이 갖는 장점이 줄어들고, 워크로드의 특성이 보다 순차적으로 바뀌었기 때문으로 분석된다. 게다가, 교체블록 기법이 단순하기 때문에 사상 테이블을 관리하는 오버헤드가 적어 실제 시스템 환경에서 보다 더 좋은 성능을 보인다. 그림 2(a)의 오른쪽 그림에서 보듯이 로그블록 기법은 병합 연산 시 대부분 2개의 블록을 소거하므로 약간의 연산 오버헤드가 있다.

PostMark 벤치마크에 대한 결과는 Camera-Ms500 벤치마크에서 보여준 결과와 비슷하게 잦은 쓰기 참조를 흡수한 후에 수행 시간이 급격히 향상되고, FTL 기법들 간의 성능 차이도 상당히 줄어든다. FAST 기법의 잦은 쓰기 참조들의 존재 유무에 상관없이 가장 좋은 성능을 보인다. 이러한 결과는 PostMark 벤치마크의 임의적 쓰기 참조 패턴을 고려한다면 당연한 결과이다. 그림 3(b)에서 주목할 만 한 점은 로그블록 기법이 PostMark 벤치마크에서 모든 파일의 데이터가 동기식으로 쓰일 때, 가장 안 좋은 성능을 보인다. 그 이유는 로그블록 기법이 완전한 임의 쓰기 참조들에 대해 로그블록들을 효율적으로 사용할 수 없기 때문으로 생각된다[2]. 특히, 잦은 쓰기 참조들이 로그블록의 이용률을 극도로 낮게 만들고, 그 결과 연쇄적인 병합 연산(Merge thrashing)이 발생하게 된다(그림 2(b) 왼쪽 그림 참조). 게다가, 교체블록 기법과 로그블록 기법 사이의 성능 차이는 로그블록 기법이 병합 연산과 같은 내부 연산을 수행하는데 더 많은 자원을 소비함으로써 발생한다.

6. 요약 및 향후 과제

본 논문에서는 파일시스템이 데이터 무결성을 보장할 때, 메타데이터에 대한 잦은 쓰기 참조들이 서로 다른 FTL 기법들에 미치는 영향에 대해 살펴보았다. 또한, 비휘발성 램을 이용한 잦은 쓰기 참조들의 흡수가 플래시 저장장치 내의 FTL 기법들의 성능에 미치는 영향에 대해서 논의하였다.

다양한 성능 평가를 통해, 잦은 쓰기 참조들이 존재하는 경우 각 FTL 기법의 성능이 기존 FTL 기법 연구 결과들과 다소 차이가 있을 수 있음을 살펴보았다. 또한, 비휘발성 램을 이용해 잦은 쓰기 참조들을 흡수함으로써 전형적인 FTL 기법들이 플래시 저장장치의 성능에 미치는 영향이 급격히 줄어드는 것을 확인하였다.

참 고 문 헌

[1] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A Space-Efficient Flash Translation Layer for CompactFlash Systems," *IEEE Transactions on Consumer Electronic*, vol.48, no.2, pp.366-375, 2002.

[2] S. W. Lee, D. J. Park, T. S. Chung, D. H. Lee, S. Park, and H. J. Song, "A Log Buffer-Based Flash Translation Layer Using Fully-Associative Sector Translation," *ACM Transactions on Embedded Computing Systems*, vol.6, no.3, pp.18, 2007.

[3] A. Ban. "Flash file system," United States Patent, No.5, 404,485, 1995.

[4] A. Ban, "Flash file system optimized for page-mode flash technologies," United States Patent, No.5,937,425, 1997.

[5] S. Kang, S. Park, H. Jung, H. Shim, and J. Cha, "Performance Trade-Offs in Using NVRAM Write Buffer for Flash Memory-Based Storage Devices," *IEEE Transactions on Computer*, vol.58, no.6, pp. 744-758, 2009.

[6] K. Salem and S. Akyurek, "Management of Partially Safe Buffers," *IEEE Transactions on Computers*, vol.44, no.3, pp.394-407, 1995.

[7] B. S. Gill and D. S. Modha, "WOW:Wise Ordering for Writes - Combining Spatial and Temporal Locality in Non-Volatile Caches," *In Proceedings of the 4th USENIX Conference on File and Storage Technologies*, pp.129-142, 2005.

[8] H. Kim and S. Ahn, "BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage," *In Proceedings of the 6th USENIX Conference on File and Storage Technologies*, pp.239-252, 2008.

[9] J. W. Hsieh, T. W. Kuo, and L. P. Chang, "Efficient Identification of Hot Data for Flash Memory Storage Systems," *ACM Transactions on Storage*, vol.2, No.1, 2006.

[10] I. H. Doh, J. Choi, D. Lee, and S. H. Noh, "Exploiting Non-Volatile RAM to Enhance Flash File System Performance," *In Proceedings of the 7th ACM & IEEE International Conference on Embedded Software*, pp.164-173, 2007.

[11] Ramtron, http://www.ramtron.com.

[12] Falinux EZ-X5, http://www.falinux.com.

[13] BitPim Project, http://www.bitpim.org.

[14] J. Katcher. PostMark: A New Filesystem Benchmark. Technical Report TR3022, Network Appliance, 1997.