

트랜잭션 단위 쓰기를 보장하는 스토리지 클래스 메모리 쓰기 버퍼캐시의 설계 및 구현

(Design and Implementation of
Transactional Write Buffer Cache
with Storage Class Memory)

김 영 진 ^{*}

(Young Jin Kim)

도 인 환 ^{*}

(In Hwan Doh)

김 은 삼 ^{**}

(Eunsam Kim)

최 종 무 ^{***}

(Jongmoo Choi)

이 동 희 ^{****}

(Donghee Lee)

노 삼 혁 ^{*****}

(Sam H. Noh)

요약 최근 등장한 비휘발성 속성과 램의 속성을 동시에 제공하는 스토리지 클래스 메모리(SCM)는 스토리지 시스템의 입출력 성능과 안정성 향상을 위한 시스템 소프트웨어 연구에 있어 새로운 가능성을 열어준다. 이에 본 연구에서는 트랜잭션 단위 쓰기를 보장하는 SCM 버퍼 캐시를

* 이 논문은 2007년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. R0A-2007-000-20071-0)

** 이 논문은 2009 한국컴퓨터종합학술대회에서 '트랜잭션 단위 쓰기를 보장하는 스토리지 클래스 메모리 쓰기 버퍼캐시의 설계 및 구현'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 홍익대학교 컴퓨터공학

yjkim@necsst.ce.hongik.ac.kr
ihdoh@necsst.ce.hongik.ac.kr

** 정 회 원 : 홍익대학교 컴퓨터공학 교수

eskim@hongik.ac.kr

*** 종신회원 : 단국대학교 컴퓨터공학 교수

choijm@dankook.ac.kr

**** 정 회 원 : 서울시립대학교 컴퓨터과학 교수

dhl_express@uos.ac.kr

***** 종신회원 : 홍익대학교 컴퓨터공학 교수

samhnoh@hongik.ac.kr

논문접수 : 2009년 8월 14일

심사완료 : 2009년 11월 10일

Copyright©2010 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.

정보과학회논문지: 컴퓨팅의 실제 및 테크 제16권 제2호(2010.2)

통하여 스토리지 시스템의 안정성과 성능을 동시에 향상시키는 한편 시스템 붕괴 시 즉각적인 복구를 가능하게 한다. 본 연구에서 제안하는 트랜잭션 단위 쓰기를 보장하는 SCM 버퍼 캐시 기법은 리눅스 저널링 블록 디바이스(JBD)의 트랜잭션 메커니즘을 기반으로 하므로 JBD 만큼의 안정성을 제공한다. 동시에 실제 시스템에서의 성능 평과 결과에서 트랜잭션 단위 쓰기를 보장하는 SCM 버퍼 캐시를 적용한 EXT3 파일 시스템은 높은 수준의 안정성을 보장하는 동시에 최소한의 안정성만을 제공하는 파일시스템보다 더 좋은 수행성을 보여주었으며, 시스템 붕괴 시 시스템을 즉각적(약 0.2초)으로 복구함을 보여주었다.

키워드 : SCM(스토리지 클래스 메모리), 파일시스템, 버퍼 캐시, 트랜잭션

Abstract Using SCM in storage systems introduce new potentials for improving I/O performance and reliability. In this paper, we study the use of SCM as a buffer cache that guarantees transactional unit writes. Our proposed method can improve storage system reliability and performance at the same time and can recover the storage system immediately upon a system crash. The proposed method is based on the LINUX JBD(Journaling Block Device), thus reliability is equivalent to JBD. In our experiments, the file system that adopts our method shows better I/O performance even while guaranteeing high reliability and shows fast file system recovery time (about 0.2 seconds).

Key words : SCM(Storage Class Memory), File System, Buffer Cache, Transaction

1. 서 론

스토리지 시스템의 안정성과 성능은 지난 수십 년간 시스템 소프트웨어 분야에서 꾸준히 연구되어 오고 있는 주제이다. 전통적인 스토리지 시스템 구조에서의 안정성 보장을 위한 노력은 다각도로 진행되어 왔고 그 결과로 효과적인 기법들이 이미 현실화 되었다. 그 예로써, 저널링 파일시스템과 소프트업데이트 기법은 모두 파일시스템의 일관성을 효과적으로 보장하며 현재 널리 사용되고 있다[1]. 그럼에도 불구하고 기존의 기법들은 시스템의 입출력 성능을 회생하거나 최근에 개선된 데이터들의 무결성을 회생한다는 측면에서 여전히 개선의 여지가 있다.

스토리지 클래스 메모리(Storage Class Memory; SCM)는 메모리 셀 자체적으로 비휘발성 속성을 제공하며 동시에 고속의 바이트 단위 랜덤 접근을 지원하는 메모리 기술을 총칭한다. 현재 대표적인 SCM으로는 PCM(또는 PRAM), FeRAM, 그리고 MRAM이 주목받고 있다[2].

시스템 입출력 계층구조에서 SCM의 효과적인 활용

은 전통적으로 트레이드오프 관계로 여겨지는 입출력 성능 향상과 안정성 향상이라는 두 가지 요구사항을 동시에 만족시키는 것을 가능하게 한다. 이에 본 연구에서는 SCM을 버퍼캐시 용도로 도입함으로써 동기식 쓰기를 제거하여 입출력 수행 성능을 향상하고자 한다. 동시에 SCM 버퍼캐시에 트랜잭션 단위의 쓰기를 지원함으로써 이를 적용한 파일시스템에 대해 기존의 저널링 파일시스템 수준의 높은 안정성을 보장하고자 한다.

SCM 하드웨어 모듈과 리눅스 운영체계에 동적으로 적재 가능한 소프트웨어 모듈을 활용한 실제 시스템 환경에서의 성능평가 결과에서, 본 논문에서 제안하는 기법의 SCM 쓰기 버퍼캐시를 적용한 EXT3 파일시스템은 시스템 붕괴 후 파일시스템의 일관성을 매번 즉각적(대략 0.2초)으로 복구하며, EXT2 파일시스템은 물론 메타데이터만 저널링 하는 EXT3 파일시스템보다 뛰어난 수행성능을 나타낸다.

이후 논문의 구성은 다음과 같다. 본 연구와 밀접한 관련이 있는 기존 연구들을 다음 절에서 언급하고, 3절에서는 본 논문에서 제안하는 SCM을 활용하는 쓰기 버퍼캐시 기법에 대해서 구체적으로 설명한다. 이어서 4절과 5절에서 실험 환경과 성능 평가 결과에 대해서 각각 논의하고, 6절에서 결론을 맺는다.

2. 관련 연구

본 절에서는 기존의 대표적인 파일시스템 안정성 향상 기법에 대해서 언급한 후, SCM을 포함하여 비휘발성 램을 입출력 계층구조에서 활용하고자 하는 선행연구들을 정리한다.

기존에 파일시스템의 일관성을 보장하는 대표적인 방법으로는 저널링 파일시스템을 들 수 있다. 저널링 기법은 WAL(write-ahead logging)을 이용하여 파일시스템 일관성과 데이터에 대한 높은 안정성을 보장하는 기법이다. 이 같은 방식은 추가적인 기록과정으로 인한 파일시스템의 성능 저하를 동반하며, 저널공간이 늘어남에 따라 시스템 붕괴 이후의 복구시간이 길어진다는 한계를 지닌다. 본 연구는 SCM을 도입하여 파일 시스템의 성능 저하 없이 높은 안정성을 보장하는 동시에 즉각적인 복구를 가능하게 하는 점에서 새롭다고 할 수 있다.

비휘발성 램을 입출력 계층구조에 도입하려는 시도는 다양한 방식으로 진행되어오고 있다. Chen 등은 무정전 전력공급 장치(UPS)의 지원으로 전체 시스템 메모리가 비휘발성 속성을 가질 경우에 파일캐시의 안전한 관리 및 복구 방법을 제안한 바 있다[3]. Doh 등은 메타데이터를 SCM에 유지하고 파일데이터만 플래시 메모리에 저장하는 플래시 파일시스템을 제안한 바 있다[4]. 본 연구는 SCM을 비휘발성 램으로 고려하며, 휘발성 램

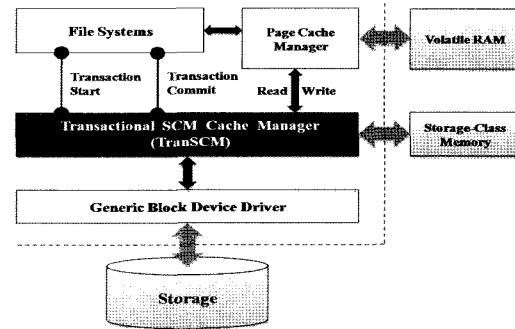


그림 1 TranSCM을 포함하는 시스템 입출력 계층구조

버퍼캐시와 SCM버퍼캐시가 공존하는 환경을 고려하면서 트랜잭션 쓰기를 지원한다는 점에서 기존의 연구들과 차별된다.

3. TranSCM: 트랜잭션 단위의 쓰기를 지원하는 SCM 쓰기 버퍼캐시 관리 모듈

TranSCM의 설계 목표는 메타데이터와 데이터의 일관성 보장을 통한 파일시스템의 안정성 향상과 극단적인 지연 쓰기를 통한 파일시스템의 성능 향상이다.

3.1 전체적인 구조

SCM 쓰기 버퍼캐시는 파일시스템 독립적인 추가적인 소프트웨어 모듈인 TranSCM에 의해서 관리되며, TranSCM은 기존의 파일시스템들이 트랜잭션 단위 쓰기 캐시 기능을 사용할 수 있도록 인터페이스를 제공한다. 그림 1은 TranSCM을 포함하는 입출력 계층구조를 간략하게 보여준다.

TranSCM이 제공하는 인터페이스를 통해서 기존의 파일시스템들은 트랜잭션 단위로 파일데이터와 메타데이터를 일관성 있게 생성하는 것이 가능하다. 트랜잭션 시작 요청 이후에 생성된 파일시스템의 모든 정보들은 SCM 상에서 하나의 트랜잭션으로 구성되어 트랜잭션 종료 요청이 발생하면 SCM 쓰기 버퍼캐시로 즉각 반영된다. 이를 통하여 파일시스템의 일관성이 보장되며, 종료되지 않은 마지막 트랜잭션의 생성된 정보를 제외한 모든 데이터는 안정적으로 유지된다.

3.2 내부 설계

TranSCM의 역할은 트랜잭션 단위 데이터 캐시와 시스템 붕괴 이후 복구로 나눌 수 있다. 본 절에서는 트랜잭션 단위 데이터 캐시를 위한 TranSCM의 트랜잭션 구성 메커니즘과 SCM 버퍼 캐시 생성 과정 그리고 시스템 붕괴 이후의 복구 과정에 대하여 자세히 설명한다.

TranSCM의 트랜잭션 메커니즘은 리눅스 저널링 블록 디바이스(JBD)의 트랜잭션 메커니즘을 그대로 차용한다.[5] 이를 통하여 TranSCM은 개별적인 파일시스템

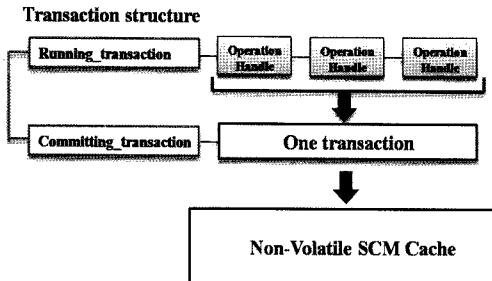


그림 2 TranSCM 트랜잭션 메커니즘

연산을 최소 단위로 트랜잭션을 구성하는 메커니즘을 지원하는 동시에 JBD에서 지원하는 세 가지 저널링 모드(Writeback, Ordered, Full Journal)에 해당하는 트랜잭션 구성 방법을 모두 지원할 수 있다.

그림 2는 TranSCM의 트랜잭션 구조 및 메커니즘을 보여준다. TranSCM은 트랜잭션 구조 및 관리를 위하여 메모리상에 두 개의 리스트 구조를 유지한다. Running_transaction은 개별적인 파일 시스템 연산들마다 할당되는 오퍼레이션 핸들을 포함한다. 각각의 오퍼레이션 핸들은 해당 파일 시스템 연산에 의하여 수정된 메타데이터 및 데이터의 페이지를 포함한다. 시스템의 트랜잭션 종료(커밋) 요청 발생 시 Running_transaction은 현재 유지하고 있는 오퍼레이션 핸들을 하나의 트랜잭션으로 묶어 Committing_transaction으로 이동시킨 후 SCM 버퍼캐시로 즉각 반영시킨다. 이로써 파일 시스템은 최근의 트랜잭션 종료 요청 이전까지 수행된 파일 시스템 연산에 대한 안정성을 보장받게 된다. 구성된 트랜잭션을 SCM 버퍼캐시로 반영하여 생성하는 과정은 이후 자세히 설명한다.

SCM 버퍼 캐시를 위한 TranSCM은 SCM 공간을 트랜잭션 처리를 위한 영역(Transaction Manager; TM 영역)과 순수 버퍼캐시 영역(Buffer Pool; BP영역)으로 구분하여 관리한다. 그림 3은 TranSCM이 트랜잭션 메커니즘을 통하여 SCM 버퍼 캐시를 생성하는 일련의 과정을 보여준다.

그림 3(a)는 파일시스템 연산에 의하여 일부 페이지가 수정되었음을 보여준다. 수정된 페이지들은 TranSCM의 트랜잭션 메커니즘에 의하여 파일 시스템 연산 단위로 관리되며 트랜잭션 종료 요청 발생 시 하나의 트랜잭션으로 구성되어 SCM 버퍼캐시의 TM 영역으로 복사된다. 그림 3(b)는 트랜잭션이 복사되는 과정을 보여준다. SCM 버퍼캐시 TM 영역으로의 트랜잭션 복사는 SCM 버퍼 캐시 부분 생성 문제의 해결을 위하여 반드시 시작(Start)과 끝(End)을 명시하여 진행된다. 하나의 트랜잭션이 완전하게 SCM 버퍼캐시의 TM 영역으로 복사

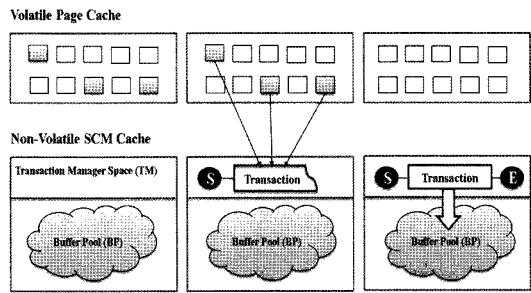


그림 3 트랜잭션 단위 쓰기 보장을 위한 내부 동작

된 후 TranSCM은 SCM 버퍼캐시의 생성(BP 영역 생성)을 시작한다. 그림 3(c)는 완전하게 복사된 트랜잭션으로 SCM 버퍼 캐시를 생성하는 과정을 보여준다.

비휘발성 SCM 버퍼캐시를 복구하는 절차, 다시 말하면 TranSCM이 적용된 파일시스템의 복구 절차는 간단하게 이루어진다. SCM 버퍼캐시의 생성은 그림 3(c)에서 볼 수 있듯이 하나의 새로운 트랜잭션이 시작과 끝을 명시하여 SCM 공간으로 완전히 복사되었을 때만 수행된다. 시스템 붕괴가 발생했을 때 만약 트랜잭션이 완전한 상태에 있다면 BP영역 생성 도중 시스템 붕괴가 발생했을 가능성이 있으므로 해당 트랜잭션에 속한 모든 버퍼를 BP영역으로 반영하는 것으로 복구가 완료되며, 트랜잭션이 불완전한 상태라면 단순히 트랜잭션을 폐기하는 것으로 복구가 완료된다.

4. 실험 환경

본 절에서는 TranSCM이 적용된 파일시스템의 수행 성능과 안정성 평가를 위하여 본 연구에서 구현한 소프트웨어들과 실험이 수행된 하드웨어 시스템 환경에 대해서 살펴본다.

4.1 시스템 소프트웨어 환경

기본적으로 리눅스 2.6.21 커널에서 TranSCM을 커널 모듈 형태로 구현하고, 대표적인 저널링 파일시스템인 EXT3 파일시스템이 TranSCM를 운용하도록 수정한다. 이후 논의의 편의를 위해서 EXT3-TranSCM이라 칭한다. 성능 평가를 위하여 TranSCM과 동일한 안정성을 지니는 동시에 SCM을 전혀 사용하지 않는 EXT3 파일 시스템과 SCM을 저널링 공간으로 사용하도록 수정한 JBD를 운용하는 EXT3 파일시스템(이후, EXT3-JbdSCM이라 명명)을 채택한다. EXT3, EXT3-JbdSCM, EXT3-TranSCM의 비교는 동일한 정도의 안정성을 바탕으로 SCM의 사용여부 및 사용구조에 따른 성능 차이를 보여준다.

성능 평가에 사용된 벤치마크는 PostMark 버전 1.51

이며, 생성된 워크로드에서 쓰기 요청된 총 용량은 3022MB이다[6].

4.2 하드웨어 환경

TranSCM이 적용된 파일시스템의 평가를 위해서 임베디드 시스템 환경과 PC 환경을 함께 사용한다. 실험에 사용된 임베디드 시스템은 실제 SCM의 한 형태인 FeRAM을 탑재할 수 있지만 임베디드 시스템이 가지는 하드웨어 수준을 고려할 때 파일시스템의 수행성능을 충분히 평가하는데 한계가 있다. 이에 추가적인 실험 환경으로 메인메모리의 일부를 SCM 영역으로 에뮬레이션한 PC 환경을 사용한다.

임베디드 시스템은 PXA255 ARM RISC 400MHz MCU, 64MB SDRAM을 갖추고 최대 64MB의 FeRAM이 가용하다. 실험 PC 환경은 Intel Core2 Duo E4400 2.00GHz CPU, 2GB RAM, Seagate Barracuda 7200.10 400GB 디스크를 갖추고 있다. PC환경에서 진행된 모든 실험에서 2GB의 시스템 메모리 중 768MB가 SCM 영역으로 에뮬레이션 되었다.

5. 성능 평가 결과

본 절에서는 TranSCM이 적용된 파일시스템의 수행성능과 복구시간에 초점을 맞춘다. TranSCM은 리눅스 JBD와 동일한 트랜잭션 메커니즘을 사용하므로 안정성에 대한 논의는 생략한다.

5.1 파일시스템 복구

실험을 위해 임베디드 시스템 환경에서 벤치마크 수행도중 임의의 시점에서 전원을 차단하여 시스템을 봉괴시킨 후 파일시스템 복구과정을 수행한다. EXT3 파일시스템의 디스크 저널 크기, EXT3-JbdSCM의 SCM 저널 크기, EXT3-TranSCM에 사용된 SCM 버퍼캐시의 크기는 모두 64MB로 동일하다.

표 1은 시스템 봉괴 이후 각 파일시스템의 복구시간을 보여준다. 각 파일 시스템은 SCM을 활용하는 구조적 차이에서 시스템 봉괴 이후 메타 데이터 일관성 복구를 위하여 수행하는 작업의 차이가 발생하므로 상이한 복구 성능을 보여 준다.

비정상적인 종료 이후 파일시스템 마운트 수행 시 EXT3는 디스크의 저널 공간전체를 탐색하고 필요한 트랜잭션을 메인 파일시스템에 기록하여야 하므로 50초 내외로 가장 느리게 복구된다. EXT3-JbdSCM의 경우

표 1 시스템 봉괴 이후 복구시간

File Systems	Recovery time(sec)
EXT3	49.897
EXT3-JbdSCM	0.248 ~ 38.697
EXT3-TranSCM	0.217

SCM 저널 공간을 탐색하고 필요한 트랜잭션을 메인 파일시스템으로 기록하는데, 파일시스템으로 기록해야 하는 트랜잭션의 양에 따라서 0.2초에서 38초 내외로 복구 시간에 많은 차이를 보여준다. EXT3-TranSCM의 경우 SCM 버퍼 캐시상의 TM영역에 존재하는 마지막 트랜잭션을 분석하고 SCM 버퍼캐시로 반영 또는 미반영 하는 것으로 복구 과정이 완료 되므로 매번 0.2초로 즉각적인 복구가 가능하다.

5.2 수행 성능

그림 4는 PC 실험 환경에서 디스크 저널 크기, SCM 저널 크기 및 SCM 버퍼캐시의 크기 변화에 따른 각 파일시스템의 벤치마크 수행성능을 보여준다. 실험에서 EXT3는 JBD가 지원하는 저널링 모드 중 메타데이터의 일관성만 보장하며 수행속도가 가장 빠른 Writeback 모드로 동작한다. EXT3-JbdSCM과 EXT3-TranSCM은 메타데이터의 일관성과 데이터의 안정성까지 모두 보장하는 Full Journal 모드에서 동작한다. 수정된 모든 메타데이터와 데이터는 EXT3-JbdSCM에서는 SCM 저널 공간에 기록되며, EXT3-TranSCM에서는 SCM 버퍼캐시에 반영하여 유지된다.

그림 4에서 EXT3-JbdSCM과 EXT3-TranSCM은 Writeback 모드로 동작하는 EXT3보다 더 높은 안정성을 제공하는 동시에 모든 설정에서 EXT3보다 좋은 수행성능을 보여준다. EXT3-JbdSCM은 EXT3와 동일한 파일 시스템 구조를 취한다. 따라서 EXT3-JbdSCM이 EXT3보다 많은 양의 데이터를 저널 공간에 기록함에도 불구하고 더 높은 성능을 보이는 이유에 저널 공간으로 사용되는 디스크와 SCM간의 매체의 접근 속도 차이가 기인한다고 볼 수 있다. EXT3-TranSCM과 EXT3-JbdSCM의 성능 차이는 시스템 구조적 측면의 차이에서 기인된다. EXT3-TranSCM은 SCM을 버퍼 캐시로 활용함으로써 휘발성 램 페이지 캐시와는 독립적으로 추가적인 캐시 효과를 제공할 수 있다.

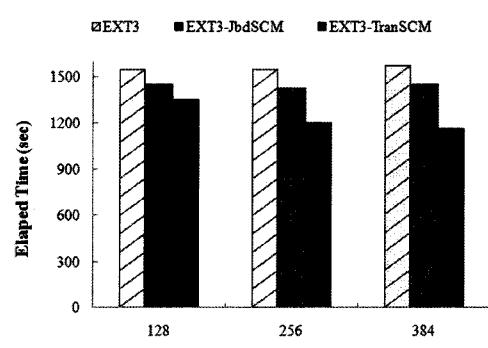


그림 4 파일시스템 성능 비교

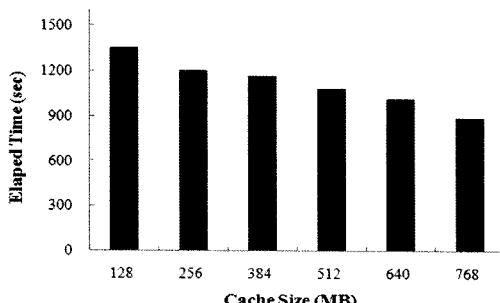


그림 5 캐시 크기에 대한 EXT3-TranSCM의 성능

TranSCM 구조는 휘발성 램 페이지 캐시상의 데이터를 기록하기 위한 주기적인 디스크 접근이 존재하지 않으므로 충분한 SCM 크기를 가정한다면 이는 파일 시스템 수행성능에 충분히 긍정적인 영향을 미칠 수 있는 요소이다. 이에 따라 그림 4에서 사용되는 SCM의 크기에 비례하여 EXT3-TranSCM의 성능이 증가하는 것을 확인할 수 있다. 그림 5는 위의 실험 설정에서 SCM 버퍼캐시 크기 증가에 따른 EXT3-TranSCM의 수행 성능 변화 추이를 보여준다. 그림 4의 결과와 마찬가지로 SCM 캐시 크기에 비례하여 성능이 좋아지고 있는 것을 확인할 수 있으며, 이는 단순히 빠른 저널링 매체를 사용함으로써 얻을 수 있는 성능 이득의 한계를 보여줌과 동시에 본 연구에서 제안하는 TranSCM 구조의 성능상의 효과를 잘 나타내고 있다.

5.3 짧은 트랜잭션 반영 주기를 통한 안정성 향상 시 수행 성능 평가

TranSCM 구조에서 시스템 붕괴 발생 시 가장 마지막 트랜잭션에 속한 데이터들은 손실의 가능성을 지닌다. 트랜잭션 반영 주기를 짧게 하는 것은 시스템 붕괴 후 복구 시 손실의 가능성을 지니는 데이터양을 감소시켜 안정성 향상을 가능하게 한다.

그림 6은 EXT3-TranSCM의 트랜잭션 반영 주기에

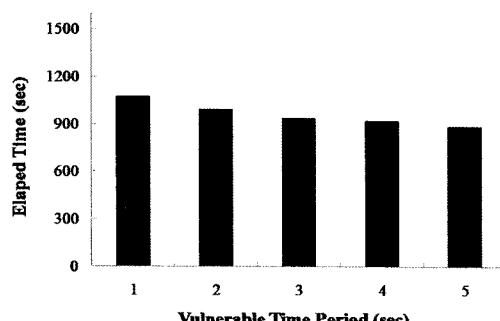


그림 6 트랜잭션 반영 주기에 따른 EXT3-TranSCM의 성능

따른 수행 성능을 보여준다. 실험에 사용된 SCM 캐시 크기는 768MB이다. 실험 결과에서 짧은 트랜잭션 반영 주기가 수행성능의 저하를 가져옴을 확인할 수 있으며, 이는 짧은 트랜잭션 구성 및 SCM 버퍼 캐시 갱신에 따른 오버헤드이다. 하지만 실제 5초의 트랜잭션 반영 주기를 1초로 줄였을 때 증가한 수행 시간은 20% 정도로, 이 결과는 트랜잭션 반영 주기 조정을 통해서 데이터 안정성과 입출력 성능 사이의 수용 가능한 트레이드 오프 관계를 형성할 수 있음을 보여준다.

6. 결 론

본 연구에서는 입출력 계층구조에 SCM을 파일시스템의 버퍼캐시로 도입하고 해당 버퍼캐시에 트랜잭션 단위의 쓰기를 지원하는 기법을 제안하였다. 실제 시스템 환경에서의 성능 평가 결과는 제안된 SCM 쓰기 버퍼캐시의 활용을 통해 파일시스템 안정성을 희생하지 않으면서 입출력 성능을 향상하는 것이 가능함을 보여준다.

참 고 문 헌

- [1] M. I. Seltzer, G. R. Ganger, M. K. McKusick, K. A. Smith, C. A. N. Soules, and C. A. Stein, "Journaling versus Soft Updates: Asynchronous Meta-data Protection in File Systems," In *Proceedings of the 2000 USENIX Annual Technical Conference*, 2000.
- [2] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," *IBM Journal of Research and Development*, vol.52, no.4/5, 2008.
- [3] P. M. Chen, W. T. Ng, S. Chandra, C. Aycock, G. Rajamani, and D. Lowell, "The Rio File Cache: Surviving Operating System Crashes," In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS-VII)*, 1996.
- [4] 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 (EMSOFT'07)*, 2007.
- [5] K. Sovani, "Linux: The Journaling Block Device," Kernel Trap (<http://kerneltrap.org/node/6741>), 2006.
- [6] J. Katcher, "PostMark: a New Filesystem Benchmark," Network Appliance, Tech. Rep. TR3022, 1997.