ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 42, No. 1, pp. 7-14, 2015. 1
http://dx.doi.org/10.5626/JOK.2015.42.1.7

스토리지 클래스 메모리를 위한 롤백-복구 방식의 데이터 일관성 유지 기법

(Data Consistency-Control Scheme Using a Rollback-Recovery Mechanism for Storage Class Memory)

이 현 구 † (Hyun Ku Lee)　김 정 훈 † (Junghoon Kim)　강 동 현 † (Dong Hyun Kang)　엄 영 익 †† (Young Ik Eom)

요 약 스토리지 클래스 메모리(SCM)는 메모리와 스토리지의 장점을 동시에 가지고 있기 때문에 기존의 스토리지를 대체할 차세대 스토리지로 주목 받고 있다. 하지만 현재까지 제안된 SCM 전용 파일시스템은 데이터 일관성을 충분히 보장하지 않거나 혹은 보장될 경우, 과도한 일관성 유지 비용을 발생시키는 문제점을 지니고 있다. 본 논문에서는 보편적으로 사용하는 WAL(Write Ahead Logging) 방식의 일관성 유지 기법 대신 롤백-복구 방식을 이용하여, 블록내의 변경되는 데이터의 비율에 따라 로그 데이터 기록 방식을 변경하는 데이터 일관성 유지 기법을 제안한다. 본 기법은 데이터 일관성 손실 없이 로그 데이터의 크기를 줄여 데이터 쓰기 및 동기화 비용을 최소화시킬 수 있다. 제안한 기법을 평가하기 위해 리눅스 3.10.2 상에 구현하여 성능을 측정한 결과, 다른 일관성 유지기법에 비해 평균적으로 9배 정도의 데이터 쓰기 성능이 향상됨을 볼 수 있었다.

키워드: 스토리지 클래스 메모리, 파일 시스템, 일관성 유지 기법, 롤백-복구 방식, 데이터 복구

Abstract Storage Class Memory(SCM) has been considered as a next-generation storage device because it has positive advantages to be used both as a memory and storage. However, there are significant problems of data consistency in recently proposed file systems for SCM such as insufficient data consistency or excessive data consistency-control overhead. This paper proposes a novel data consistency-control scheme, which changes the write mode for log data depending on the modified data ratio in a block, using a rollback-recovery scheme instead of the Write Ahead Logging (WAL) scheme. The proposed scheme reduces the log data size and the synchronization cost for data consistency. In order to evaluate the proposed scheme, we implemented our scheme on a Linux 3.10.2-based system and measured its performance. The experimental results show that our scheme enhances the write throughput by 9 times on average when compared to the legacy data consistency control scheme.

Keywords: storage class memory, file system, data consistency control, rollback-recovery, data recovery

· 본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT 연구센터 육성지원 사업의 연구결과로 수행되었음(NIPA-2014(H0301-14-1020))
· 이 논문은 2014 한국컴퓨터종합학술대회에서 '스토리지 클래스 메모리를 위한 롤백 방식의 데이터 일관성 유지기법'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 성균관대학교 정보통신대학
hyungooo@skku.edu
myhuni20@skku.edu
kkangsu@skku.edu
†† 종신회원 : 성균관대학교 정보통신대학 교수(Sungkyunkwan Univ.)
yieom@skku.edu
(Corresponding author임)

논문접수 : 2014년 7월 22일
(Received 22 July 2014)
논문수정 : 2014년 9월 16일
(Revised 16 September 2014)
심사완료 : 2014년 9월 28일
(Accepted 28 September 2014)

정보과학회논문지 제42권 제1호(2015. 1)

1. 서 론

데이터의 읽기, 쓰기 연산에 직접적인 영향을 미치는 스토리지는 시스템의 성능을 결정하는 중요한 요인 중 하나이다. 이중 하드 디스크 드라이브는 가장 대중적으로 사용되는 스토리지 이지만 디스크의 물리적인 검색 시간, 느린 전송속도 등의 제한 때문에 높은 성능을 요구하는 시스템에 사용하기에는 한계가 존재한다. 이러한 한계를 해결하기 위해 현재 SSD, eMMC 등 NAND 플래시 메모리를 이용한 스토리지가 개발 및 사용되고 있으나, 데이터의 제자리 덮어쓰기가 안되고 블록 단위의 데이터 삭제 명령으로 인해 늘어나는 데이터 관리비용 등의 플래시 메모리의 물리적인 한계를 극복하지 못하고 있다. 본 연구에서 다루고자 하는 스토리지 클래스 메모리(SCM)는 기존의 스토리지가 지닌 물리적인 한계를 극복하기 위해 새롭게 연구 및 개발되고 있는 메모리로 메모리와 스토리지의 장점을 동시에 지닌 이상적인 메모리를 총칭하는 용어이다[1]. SCM의 주요한 특징은 DRAM 처럼 바이트 단위 접근이 가능하고 성능이 좋으며 하드디스크처럼 데이터의 제자리 덮어 쓰기가 가능해 플래시 메모리의 FTL(Flash Translation Layer) 같은 데이터 관리가 필요하지 않다. 현재 PCM, STT-MRAM, FeRAM, memristor 등이 SCM으로 불리며 개발 및 상용화가 진행되고 있다[2]. 몇 년 안에 상용화의 문제가 되었던 SCM의 밀접도가 향상되고 단위당 단가 또한 낮아질 예정으로[3], 대용량 SCM이 상용화되면 기존의 스토리지를 대체할 것으로 현재 많은 이들의 주목을 받고 있다. 하지만 현재 시스템의 소프트웨어 스택에서는 SCM과 같은 특성을 지닌 스토리지가 도입된다 해도 이러한 장점들을 제대로 활용 하기 어렵다. 기존의 스토리지 관리 방식은 DRAM과 같은 메모리보다 훨씬 느린 스토리지를 고려하여 개발이 되어왔기 때문이다. 그러므로 현재의 시스템에 몇가지 변경이 필요하다.

가장 처음으로 고려할 점은 바이트 단위 접근이 가능하고 탐색 시간이 0 또는 0에 가까운 특징을 고려한 전용 파일시스템을 사용해야 한다는 것이다. 하드 디스크 드라이브나 SSD등의 블록 단위 접근을 위해 개발된 기존의 파일 시스템은 SCM의 빠르고 바이트 단위로도 접근이 가능한 장점을 이용하지 못해 불필요한 데이터 접근 또는 쓰기가 발생 할 수 있기 때문이다. 둘째로는 운영체제에서 사용되는 페이지 캐시의 존재 여부를 재고 해야 한다. 메모리와 스토리지의 속도 차이를 보조하기 위해 추가된 페이지 캐시는 같은 데이터를 스토리지, 페이지 캐시 두 곳에 중복 관리를 해야 하기 때문에 메모리와 비슷한 성능을 지닌 스토리지에선 오히려 데이터 쓰기량을 증가시키는 요인이 될 수 있기 때문이다. 마지막으로 고려 할 점은 바로 동기화 문제이다. 바이트 단위 접근을 지원하기 위해 시스템 메모리와 함께 메모리 버스를 공유하는 SCM은 일관성을 위해 페이지 캐시가 아닌 CPU 캐시와의 동기화가 필요하다. 하지만 CPU 캐시와 메모리의 동기화를 위한 cache flush 명령은 캐시내의 데이터를 메모리에 업데이트 한 후 캐시에서 삭제하므로 해당 데이터가 필요해질 경우 CPU 캐시에 다시 불러 오는 등의 추가비용이 요구된다. 전통적인 파일 시스템에서 데이터 일관성 유지는 로그 데이터가 스토리지에 미리 저장됨을 보장함으로써 이루어지며 SCM 스토리지 환경에서는 이러한 데이터 일관성 유지를 위해 cache flush 명령을 사용해야 한다. 그러므로 로그 데이터가 증가하면 데이터 쓰기량의 증가뿐만 아니라 해당 로그 데이터의 캐싱 비용의 증가 또한 초래한다.

본 연구에서는 이러한 SCM의 특성을 고려하여 새로운 방식의 일관성 유지 기법을 제안한다. 우선 범용적으로 사용되는 저널링, Shadow Paging 기법[4] 등의 Write-ahead Logging(WAL)이 아닌 롤백-복구 방식을 사용하여 문제가 되고 있는 로그 데이터의 사이즈를 줄이도록 하였다. 또한 롤백-복구 방식의 이전 데이터만 로그 데이터로 저장하는 특성과 SCM의 제로 탐색 시간의 특성을 이용하여 로그 데이터 크기가 지정한 범위보다 클 경우 해당 블록 자체를 로그 데이터로 지정하고 해당 블록의 포인터만 따로 기록하도록 하여 일관성 유지 비용을 최소화하였다. 또한 롤백-복구 방식의 로그 데이터는 시스템 오류로 인해 데이터 복구를 하기 이전에는 사용되지 않으므로 동기화로 인해 CPU 캐시에서 지워져도 필요에 의해 다시 불려지는 비용을 줄일 수 있다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 현재까지 개발 및 연구된 SCM전용 파일시스템의 일관성 유지기법을 살펴보고 문제점을 파악한다. 3장에서 제안한 일관성 유지기법을 자세히 설명하고, 4장에서 다양한 벤치마크 툴을 이용해 제안한 일관성 유지기법의 성능을 다른 일관성 유지 기법들과 비교 및 분석한다. 마지막으로 5장에서 본 연구의 결론 및 향후 연구방향을 도출 하도록 한다.

2. 일관성 유지기법 관련 연구

SCM의 특성을 지원하는 SCM 전용 파일 시스템은 현재 많은 연구가 진행되고 있다. 하지만 각각의 전용 파일시스템의 데이터 일관성 유지 기법은 아직 여러 문제점들이 존재한다.

SCMFS(Storage Class Memory File System)[5]는 메타데이터를 배열로 관리하며 가상 메모리에 데이터를 저장하는 SCM 전용 파일 시스템이다. 데이터 일관성

유지를 위해 MFENCE와 CLFLUSH를 이용하여 CPU 캐시의 연산순서를 보장 하고 메타데이터가 변경될 때마다 CPU 캐시와의 동기화를 한다. 하지만 데이터 변경에 관한 일관성 유지 기법이 없기 때문에 스토리지의 데이터 업데이트 중 시스템 에러가 발생되었을 경우 해당 파일의 데이터의 일관성을 보장할 수 없다.

BPFS(Byte-addressable Persistent File System)[4]는 SCSP(Short-Circuit Shadow Paging)을 사용해 기존의 Shadow Paging 기법 보다 데이터 쓰기 비용을 줄인 파일 시스템이다. 논문에서 제안한 특별한 하드웨어적 기능인 *epoch barrier* 이용하여 CPU 캐시와 스토리지와의 일관성을 유지한다. 하지만 SCSP를 이용한 데이터 쓰기 시 64bit 원자적 업데이트가 가능할 때까지 데이터 복사가 전파되므로 다른 일관성 유지 기법보다 데이터 쓰기 량이 많아진다. 그러므로 *epoch barrier*를 지원하지 않는 시스템일 경우 BPFS는 일반적인 파일 시스템보다 많은 cache flush 명령이 필요하게 된다.

Shortcut-JFS[6]는 SCM의 특성을 이용하도록 바이트 단위 접근으로 변경한 저널링 기법을 사용한다. 추가로 저널링 영역에 저장한 한 블록의 데이터 크기가 블록크기의 반 보다 크면 데이터를 해당 블록에 덮어쓰지 않고 간접 포인터 변경을 통하여 저널링 영역의 데이터를 원본 데이터로 여겨 데이터 복사 비용을 줄였다. 이런 방식을 통하면 일관성 유지 기법의 데이터 쓰기 량을 줄일 수 있는 장점이 있지만 페이지 캐시를 사용하지 않고 스토리지가 CPU 캐시와의 동기화가 필요한 상황에서는 일반적인 파일시스템과 마찬가지로 cache flush 명령이 과도하게 발생하게 된다.

현재까지 연구된 SCM 전용 파일 시스템은 일관성 유지 기법이 시스템 성능을 감소시키거나 완벽한 데이터 일관성을 유지하지 못하는 등의 단점이 존재한다. 그러므로 SCM을 스토리지로 사용할 경우 데이터 일관성 유지와 함께 시스템 성능 저하를 최소한으로 유지하기 위해선 본 연구에서 제안하는 새로운 방식의 데이터 일관성 기법이 필요하다.

3. 제안 일관성 유지기법

데이터 업데이트 도중 시스템 오류 발생시 데이터 일관성을 보장하기 위한 일관성 유지 기법은 로그 데이터의 특성에 따라 크게 WAL(Write Ahead Logging) 방식과 롤백-복구 방식으로 나뉜다. 저널링과 Shadow Paging은 대표적인 WAL방식의 일관성 유지 기법으로 새로운 데이터를 업데이트 하기 전에 해당 내용을 스토리지의 다른 영역에 먼저 기록하여 데이터 일관성을 유지한다. 만약 데이터 업데이트 도중 시스템 오류가 발생되면 기록해놓은 데이터를 이용하여 실패한 업데이트를

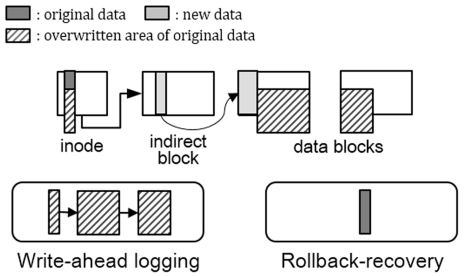


그림 1 WAL과 롤백-복구 방식의 로그데이터 비교
Fig. 1 Comparison of WAL and Rollbak-Recovery Schemes

다시 시도한다. 하지만 앞에서 설명하였듯이 SCM을 스토리지로 사용하는 환경에서는 로그 데이터가 커질수록 데이터 쓰기량의 증가뿐만 아니라 cache flush 명령의 수행 및 CPU caching 비용 또한 높아지므로 로그 데이터를 최소화하는 것이 매우 중요하다. 로그 데이터를 최소화하기 위해 제안하는 일관성 유지 기법 에서는 데이터베이스 관리 시스템에서 주로 사용하는 롤백-복구 방식을 사용하기로 하였다. 그림 1에서와 같이 롤백-복구 방식은 새로운 데이터가 아닌 변경되는 원본 데이터만 따로 기록하여 데이터 일관성을 유지하므로 WAL 방식에 비해 로그 데이터 사이즈가 현저히 적다. 또한 SCM의 특성과 롤백-복구 방식을 이용하여 로그 데이터 사이즈를 더욱 최소화시켰다. 롤백-복구 방식의 일관성 유지 기법은 파일 시스템의 최소 단위인 블록 단위로 이루어 지게 된다. 쓰기 연산 시 업데이트 하는 데이터의 블록마다 변경되는 데이터를 따로 저장하게 되는데, 만약 해당 블록에서 변경되는 데이터의 비율이 높을 경우 그 블록 자체를 로그 데이터로 지정하고 새 블록을 할당하여 새로운 데이터를 쓰는 것이 더 효율적일 것이다. 이 경우 일반적인 스토리지라면 새로운 블록이 기존의 데이터보다 물리적으로 멀리 떨어진 영역에 쓰여질 가능성이 높은데, SCM은 하드 디스크 드라이브나 SSD와는 달리 데이터 탐색 비용이 적으므로 이러한 방식을 사용하여도 성능의 저하가 생기지 않는다. 또 하나 고려할 점은 블록 자체를 로그 데이터로 여길 경우 블록 내의 변경되지 않은 데이터를 로그 데이터에서 새로운 블록으로 복사해야 하므로 이러한 부가 비용을 계산하여 해당 블록을 로그 데이터로 여길지, 아니면 로그 데이터가 저장되는 블록을 따로 생성할지 결정하여야 한다. 제안하는 기법에선 변경되는 블록의 데이터 비율이 50% 이상일 때 해당 블록을 로그 데이터로 여기고 데이터를 위한 새로운 블록을 할당한다. 또한 로그 데이터로 지정된 기존 데이터의 위치를 저장하기 위해 해당 블록의 포인터를 저장한다. 이러한 기법을 제안하는 일

관성 유지기법에선 포인터 로깅 기법이라 한다. 만약 50% 미만일 경우 변경되는 데이터를 다른 곳에 저장하며 이를 데이터 로깅 기법이라 한다.

앞에서 설명하였듯이 제안하는 일관성 유지 기법을 사용하면 로그 데이터의 쓰기 비용 최소화는 물론 CPU 캐시와 SCM의 일관성 유지를 위한 cache flush 명령의 비용 또한 줄어 든다. 일정 영역의 데이터에 cache flush 명령을 사용하면 CPU 캐시에 존재하였던 데이터를 메모리에 업데이트한 후 해당 데이터를 CPU 캐시에서 flush 시킨다. 만약 flush된 데이터가 다시 필요하다면 스토리지 시스템은 해당 데이터를 CPU 캐시에 다시 캐싱해야 한다. 제안하는 일관성 유지 기법은 롤백 방식의 로깅 기법을 사용하므로 로그 데이터로 지금 사용할 수 있는 새로운 데이터가 아닌 변경되는 예전 데이터를 사용한다. 그러므로 시스템 오류가 발생해 파일의 복구가 필요한 경우가 아니라면 로그 데이터를 다시 읽을 필요가 없어 cache flush 명령 이후 해당 데이터를 다시 캐싱하는 비용이 현저히 줄어들게 된다.

3.1 데이터 쓰기 과정

제안하는 일관성 유지 기법은 데이터 쓰기 요청이 발생되면 각각의 쓰기 요청을 대표하는 로깅 헤더를 생성한다. 로깅 헤더에는 inode 번호, 상태 flag, 로그 데이터의 목록 등 해당하는 데이터 쓰기의 정보가 담겨있다. 로깅 헤더의 생성이 완료되면 그림 2의 (1) *minimal logging* 을 수행한다. 해당 단계에서는 데이터를 업데이트 하기 전 업데이트 되는 원본 데이터를 로그 데이터로 스토리지에 저장하는 작업을 하는데 로깅하려는 블록마다 변경되는 데이터 비율을 측정하여 비율에 따라 데이터 로깅 또는 포인터 로깅을 수행한다. 로그 데이터의 쓰기가 완료되면 쓰기 완료된 상태를 SCM과 동기화를 하기 위해 (2) *logged data flushing* 이 수행된다. 이때 CPU 캐시에 있는 데이터를 SCM에 업데이트 하기 위해 cache flush 명령을 사용한다. 로그 데이터 모두 스토리지에 완벽히 쓰여진 상태, 즉 두 번째 작업이 끝난 상태를 COMMIT 상태라 하며 COMMIT 상태 이후 요청된 새로운 데이터로 원본 데이터를 업데이트 하는 (3) *new data writing* 을 수행한다. 원본 데이터 변경 시에는 업데이트 하는 블록마다 수행된 로그 기법에 따라 업데이트 하는 방식이 달라지게 된다. 만약 해당 블록이 데이터 로깅 방식으로 저장 된 경우 새로운 데이터를 해당 블록에 직접 업데이트 하지만, 포인터 로깅 방식인 경우 해당 블록을 변경하지 않고 새로운 블록을 할당하여 그곳에 요청된 새로운 데이터와 기존의 원본 데이터를 복사하여 데이터 쓰기 과정을 마무리 한다.

3.2 Checkout 과정

요청된 새로운 데이터를 업데이트하는 데이터 쓰기 과정이 완료되어도 로그 데이터를 제외한 새로운 데이터는 CPU 캐시와 SCM의 동기화가 되어있지 않으므로 모든 데이터가 완벽히 쓰여있다고 확신할 수 없다. 그러므로 업데이트한 데이터를 CPU 캐시와 완벽히 동기화 시키고 해당 영역을 위한 로그 데이터를 삭제하는 일련의 과정이 필요하다. Checkout 과정에서 이러한 작업을 수행하며 새로 쓰여진 모든 영역을 cache flush 명령을 이용하여 CPU 캐시와 동기화 시키고 동기화가 완료되면 해당 블록의 로그 데이터를 모두 삭제한다. Checkout process는 일반적인 파일시스템의 일관성 유지 기법에서와 마찬가지로 로깅 영역이 75% 이하로 남아 있을 경우 또는 5초마다 수행된다.

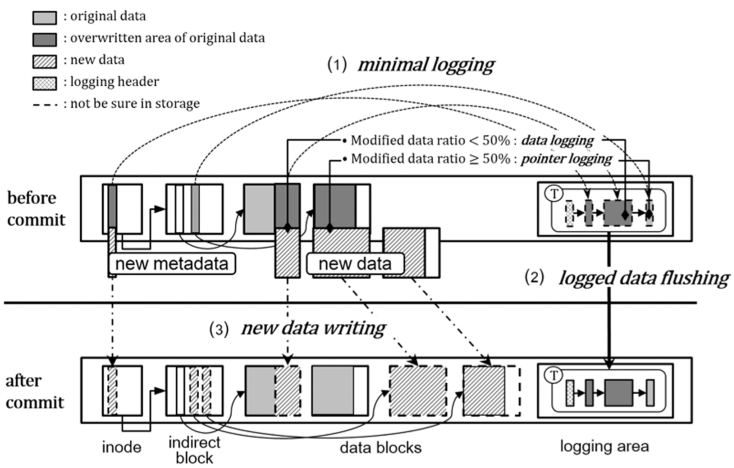


그림 2 제안하는 일관성 유지 기법의 데이터 쓰기 과정

Fig. 2 Data Write Process of Proposed Data Consistency Control Scheme

3.3 데이터 복구 과정

데이터 업데이트 도중 시스템 오류가 발생되었을 경우 제안하는 일관성 유지 기법은 저장해 놓은 로그 데이터를 이용하여 데이터를 변경 전으로 복구해 데이터 일관성을 보장한다. 그림 3에서와 같이 COMMIT 상태 이후 데이터를 업데이트 하는 도중에 시스템 오류가 발생하면 복구하려는 블록의 로그 데이터에 따라 복구 방식을 다르게 하여 데이터 복구를 진행한다. 만약 복구하려는 블록이 데이터 로깅 과정으로 저장 되어 있었다면 해당 로그 데이터를 이용하여 오류가 발생한 파일에 덮어쓰기를 통해 기존의 데이터로 복구 하고, 만약 포인터 로깅 과정으로 저장이 되어 있었다면 로그 데이터를 이용하여 원본 블록의 포인터를 복구하고 새로 할당한 블록을 삭제하여 업데이트 하기 전의 데이터로 복구하도록 한다. 만약 COMMIT 상태 이전일 때 시스템 오류가 발생되었다면 스토리지 내의 실질적인 데이터는 변경이 되지 않은 상태이므로 데이터 복구 과정이 요구되어지지 않는다.

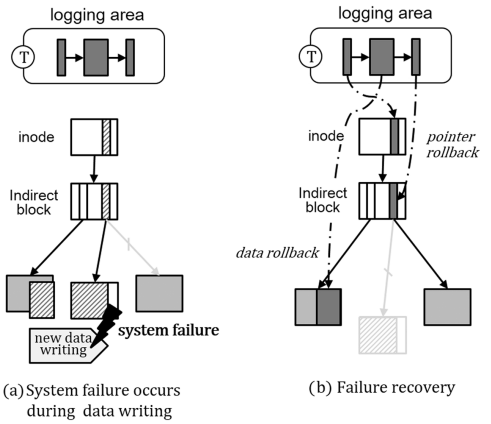


그림 3 제안하는 일관성 유지 기법의 데이터 복구 과정
Fig. 3 Data Recovery Process of Proposed Data Consistency Control Scheme

4. 성능 평가 및 분석

제안한 일관성 유지기법의 성능을 평가하기 위해 Intel Core i7 3.4GHz, 8G DRAM의 머신을 사용하였으며 Linux kernel 3.13.0을 기준으로 기법을 구현하였다. 현재 대용량의 SCM이 상용화 되어있지 않아 DRAM을 SCM으로 간주하여 실험하였다. SCM의 성능은 PCM과 STT-MRAM을 기준으로 DRAM과 성능이 비슷하거나 쓰기 성능의 차이가 조금 있을 것으로 예상되어[7] 실제 SCM을 사용하여도 본 실험에서의 결과와 크게 다르지 않을 것으로 예측된다. 스토리지의 성능을 평가하기 위해 범용적으로 쓰이는 벤치마크 중 TPC-C 벤치마크를 이용하는 DBT2 tool[8]과 IOZone 벤치마크[9]를 사용하였다. DRAM을 SCM대신 스토리지로 사용하기 위해 페이지 캐시를 스토리지로 사용하고 일관성 유지기법이 없는 ramfs를 사용하여 실험을 진행 하였다. 제안한 기법의 성능 평가를 위해 ramfs에 본 논문에서 제안하는 일관성 유지기법을 추가하였고(Min) 다른 NVRAM 전용 파일 시스템의 일관성 유지 기법과의 비교를 위해 EXT3 저널링 기법을 SCM에 맞춰 개량한 shortcut-JFS의 differential logging 기법을 ramfs내에 구현하였다(Opt-J). 해당 기법들의 구현을 위해 스토리지 내의 데이터를 기존의 데이터 검색 방법이 아닌 페이지 캐시 검색을 이용하여 확인하였고 요청되는 트랜잭션 마다 하나의 로그 데이터 구조체를 생성해 데이터 쓰기 과정에서 생성되는 모든 로그 데이터의 내용을 관리하도록 하였다. 마지막으로 현재 이용하는 기본 블록 계층을 사용하여 블록 단위로 데이터에 접근하는 기존 파일 시스템과의 비교를 위해 EXT3 파일시스템을 ram-disk에 마운트 하여 실험을 진행 하였다(Blk-J). 우선 전체적인 성능을 평가하기 위해 전자상거래 서버의 워크로드를 지닌 TPC-C 벤치마크를 이용한 DBT2 tool을 사용하였다. TPC-C는 작은 단위의 파일 생성 및 삭제, 크기 변경 등의 입출력 연산을 수행하여 파일 시스템의 파일 데이터 연산 뿐 아니라 메타 데이터 연산의 성능도 크게 좌우한다. 본 논문에서는 warehouse 갯수를 10개로 설정하고 분당 처리 건수인 tpmC를 계산하여 각각의 일관성 유지 기법을 비교하였다[10]. 그림 4에 보여지는 것 같이 일관성 유지기법이 없는 ramfs를 기준으로 제안하는 기법을 추가한 결과 약 3%의 일관성 유지 비용이, 저널링 기법을 SCM에 맞춰 개량한 Opt-J는 약 11%의 성능 하락이 있었다. 기존의 블록기반 저널링 기법인 Blk-J는 제안한 기법보다 약 21%의 성능의 하락을 보여 일관성 유지 기법이 없을 때의 성능의 약 78%의 성능이 나타남을 확인할 수 있었다.

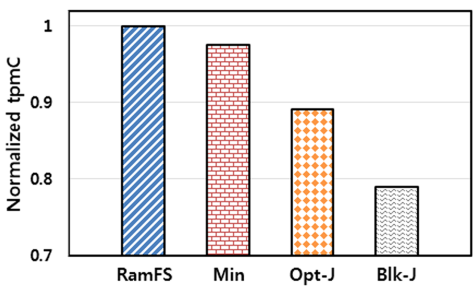
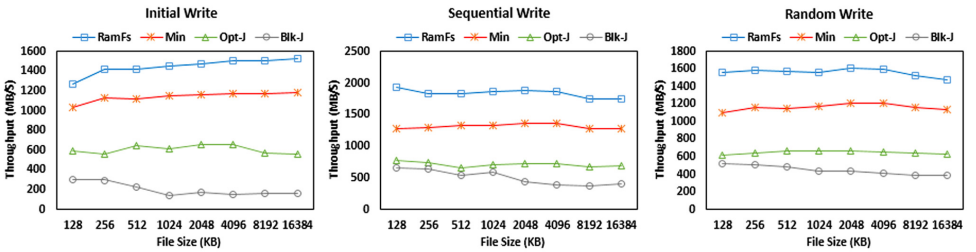
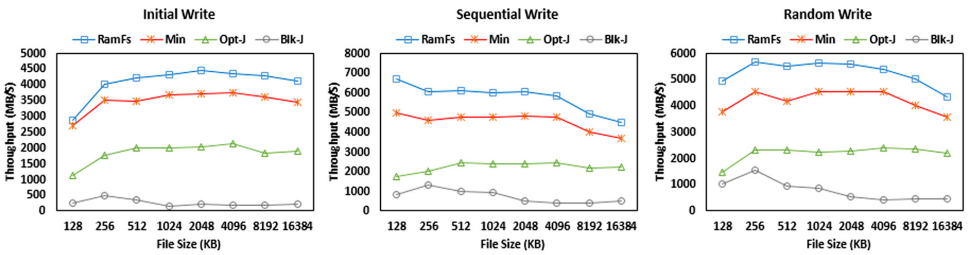


그림 4 DBT2 Tool(TPC-C) 실험 결과
Fig. 4 Result of DBT2 Tool(TPC-C)

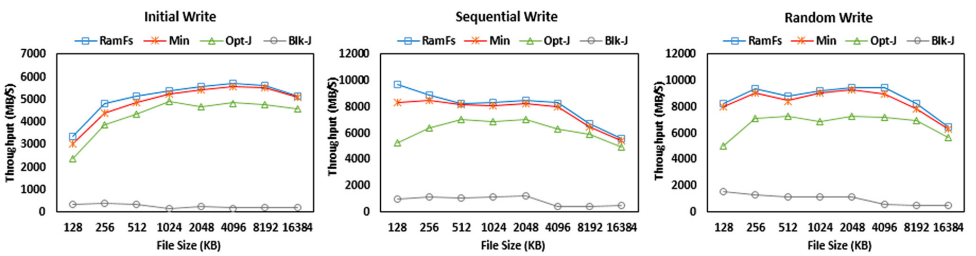
그림 5는 일련 한 데이터의 읽기 및 쓰기 연산을 수행하는 IOZone 벤치마크의 결과를 나타낸 그래프이다. Initial write, sequential write, random write 시나리오를 기준으로 1KB~100KB의 레코드 크기와 128KB~16MB의 데이터 크기로 나누어 실험 및 평가를 하였다. 결론적으로 본 연구에서 제안한 일관성 유지기법은 IOZone 벤치마크를 이용하였을 때 Opt-J 기법보다 145%~216%의 성능 향상과, Blk-J 기법보다 2배에서 21배의 성능 향상을 볼 수 있었다. 이는 제안한 일관성 유지기법이 저널링 기법을 사용하는 Opt-J 기법보다 스토리지에 기록해야 하는 로그 데이터 크기가 작고 이에 따라 cache flush 하는 비용이 줄어들기 때문으로 볼 수 있다. 또한 제안하는 기법은 블록 단위로 데이터를 다루는 기존의 일관성 유지 기법과 비교하여 최고 21배의 성능 차이를 보이는데 이러한 성능 차이는 블록 기반으로 수행되는 데이터 관리가 바이트 단위 접근이 허용되는 SCM 환경의 장점을 이용하지 못하기 때문인 것으로 보여진다. 또한 그림 5의 (a), (b), (c)에서 보여지듯, 데이터 쓰기 요청 시 쓰여지는 레코드 크기가 커질수록 각각의 일관성 유지기법의 쓰기 속도 차이는 더욱 크게 나타난다. 결과를 정확히 분석하기 위하여 IOzone 벤치마크의 레코드 크기가 1KB일때와 100KB일때의 제안한 일관성 유지 기법과 Opt-J의 로그 데이터의 크기를 측정하여 그림 6에 나타내었다. 그 결과 레코드 크기가 커질수록 로그 데이터의 크기 차이가 기존의 기법과 비교하여 많이 발생하는 것을 확인할 수 있었다. 이는 한번에 쓰여지는 레코드 크기가 클수록 변경되는 블록의 비율이 높아져 제안한 일관성 유지 기법에서 포인터 로깅이 많이 발생하기 때문이다. 그림 5와 그림 6을 비교 하였을 때 로그 데이터의 크기가 작을수록 쓰기 성능이 향상되는 것을 확인 할 수 있는데 이는 기록하는 데이터의 크기가 줄어들 뿐 아니라 cache flush를 해야 하는 범위 또한 줄어들기 때문에 일관성 유지 비용이 감소되기 때문이다. 이에 반해 저널링 기법을 사



(a) Record size of Iozone (data wiring unit) : 1KB



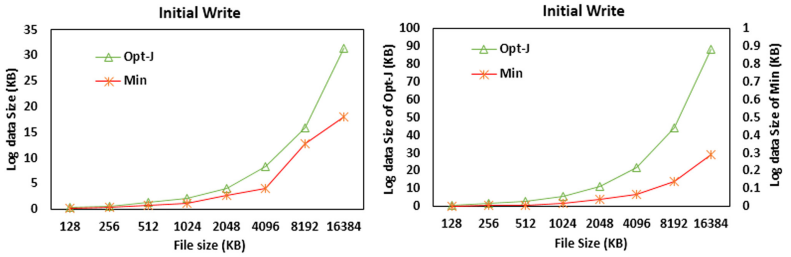
(b) Record size of Iozone (data wiring unit) : 4KB



(c) Record size of Iozone (data wiring unit) : 100KB

그림 5 IOZone 벤치마크 실험 결과

Fig. 5 Result of IOZone Benchmark



(a) Record size of IOzone : 1KB (b) Record size of IOzone : 100KB

그림 6 IOZone의 로그데이터 비교

Fig. 6 Comparison of IOZone Log Data

용하는 일관성 유지기법은 새로운 데이터 그대로 저장하므로 레코드 사이즈 변화와 상관없이 로그 데이터의 크기가 일정하게 커지게 된다. 그러므로 본 연구에서 제안하는 기법은 레코드 사이즈가 클수록 다른 일관성 유지 기법에 비해 일관성 유지비용이 최소화됨을 알 수 있다.

5. 결론 및 향후 연구

본 연구는 SCM의 장점을 효과적으로 활용하기 위해 기존의 블록 디바이스 기반의 스토리지 시스템에서 변경되어야 하는 부분과 SCM의 특성을 활용한 새로운 일관성 유지 기법을 제안하였다. 또한 프로토 타입의 구현 및 실험을 통해 새로운 일관성 유지 기법을 사용하면 데이터 일관성의 감소 없이 SCM 기반 스토리지의 성능을 최대한으로 이끌어 낼 수 있음을 확인할 수 있었다. 향후에는 현재 제안한 일관성 유지기법의 정책을 바탕으로 효율적인 포인터 로깅 시점을 다양한 실험을 바탕으로 확정하고 BPFS의 SCSP 기법과의 비교, 보다 다양한 벤치마크에서의 실험 등을 통해 제안하는 기법의 장점을 구체적으로 비교 및 분석해 볼 예정이다.

References

[1] R. F. Freitas and W. W. Wilcke, "Storage-Class Memory: The Next Storage System Technology," *Journal of IBM Journal of Research and Development*, Vol. 52, No. 4.5, pp. 439-447, Jul. 2008.

[2] Everspin's CEO Explains The Company's Technology And Business [Online]. Available: http://www.mram-info.com/everspins-ceo-explains-companys-technology-and-business.

[3] Everspin Officially Announces The World's First ST-MRAM Chip [Online]. Available: http://www.mram-info.com/everspin-officially-announces-worlds-first-st-mram-chip-will-be-available-2013.

[4] J. Condit, E. B. Nightingale, C. Frost, E. Ipek, B. Lee, D. Burger, and D. Coetzee, "Better I/O through Byte-Addressable, Persistent Memory," *Proc. of the ACM SIGOPS 22nd Symposium on Operating systems Principles*, pp. 133-146, 2009.

[5] X. Wu and N. Reddy, "Scmfs: A File System for Storage Class Memory," *Proc. of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, No. 39, pp. 1-11, 2011.

[6] E. Lee, S. Yoo, J.-E. Jang, and H. Bahn, "Shortcut-JFS: A Write Efficient Journaling File System for Phase Change Memory," *Proc. of Mass Storage Systems and Technologies 2012*, pp. 1-6, Apr. 2012.

[7] Spin-Torque MRAM Technical Brief [Online]. Available: http://www.everspin.com/PDF/ST-MRAM_Technical_Brief.pdf

[8] OSDL Database Test 2 (DBT-2) [Online]. Available: http://osdldbt. sourceforge.net

[9] D. Norcott, Iozone filesystem benchmark [Online]. Avaliable: http://www.iozone.org/docs/IOzone_msword_98.pdf

[10] S. T. Leutenegger and D. Dias, "A Modeling Study of the TPC-C Benchmark," *Proc. of ACM SIGMOD 1993*, pp. 22-31, 1993.

이 현 구

2013년 성균관대학교 전자전기공학부 학사. 2013년~현재 성균관대학교 전자전기컴퓨터공학과 석사과정. 관심분야는 운영체제, 파일 시스템, 가상화 기술

김 정 훈

2010년 성균관대학교 컴퓨터공학과 학사 2012년 성균관대학교 휴대폰학과 석사 2012년~현재 성균관대학교 IT융합학과 박사과정. 관심분야는 시스템 소프트웨어, 운영체제, 가상화 기술, 파일 시스템

강 동 현

2007년 한국산업기술대학교 컴퓨터공학과 학사. 2010년 성균관대학교 전자전기컴퓨터공학과 석사. 2013년~현재 성균관대학교 전자전기컴퓨터공학과 박사과정 관심분야는 스토리지 시스템, 운영체제

엄 영 익

1983년 서울대학교 계산통계학과 학사
1985년 서울대학교 전산과학과 석사
1991년 서울대학교 전산과학과 박사
1993년~현재 성균관대학교 정보통신대학 교수. 관심분야는 시스템 소프트웨어, 운영체제, 가상화, 파일 시스템