로그 구조 파일 시스템의 성능 향상을 위한 논리적 복사 기법

강윤지 ○, 한규화, 신동군
성균관대학교 정보통신공학부
oso41@skku.edu, hgh6877@skku.edu, dongkun@skku.edu

Improving the Performance of Log-Structured File Systems Using Logical Copy Technique

Yunji Kang○, Kyuhwa Han, Dongkun Shin
School of Information and Communication Engineering, Sungkyunkwan University

요 약

로그 구조 파일 시스템(LFS)은 모든 쓰기를 순차 쓰기로 기록하기 때문에 플래시 메모리에 적합한 파일 시스템이다. 그러나, LFS의 세그먼트 클리닝은 많은 복사 비용을 발생시켜 성능 병목의 원인이 된다. 본 논문에서는 거의 무 비용의 세그먼트 클리닝을 수행 할 수 있는 호스트와 저장장치간의 새로운 인터페이스인 논리적 복사를 제안한다. 제안한 기법은 파일 시스템 블록을 복사하기 위하여 데이터 쓰기 대신에 SSD 펌웨어가 관리하는 주소 매핑만을 수정한다. 우리는 실제 SSD 플랫폼과 F2FS 파일 시스템에 논리적 복사 기법을 구현하였다. 실험 결과, 논리적 복사 기법을 적용한 LFS는 기존의 복사 기법을 사용한 LFS와 비교하여 쓰기 성능이 최대 3배 향상을 보였으며 쓰기 양도 약 80% 감소하였다.

1. 서 론

최근에 플래시 메모리 Solid State Disks (SSDs)는 높은 성능과 낮은 전력 소비로 하드 디스크를 대체하고 있다. SSD는 여러 NAND 플래시 메모리 칩, 컨트롤러, 호스트 인터페이스, DRAM 쓰기 버퍼 등으로 구성되어 있으며 여러 플래시 칩에 병렬 처리를 함으로써 높은 I/O 성능을 제공한다. 그러나 플래시 메모리는 기존의 하드 디스크와 달리 지우기 연산 전에 덮어 쓰기가 불가능하기 때문에 SSD는 클린 페이지에 새로운 데이터를 기록하고 이전 데이터가 기록된 페이지를 무효화 시킨다. 플래시 변환 계층(Flash Translation Layer, FTL)은 상위 계층의 논리 주소를 실제 플래시 메모리의 물리 주소로 변환하여 접근할 수 있도록 주소 매핑 테이블을 관리한다. 또한 FTL은 무효화된 페이지를 회수하는 가비지 컬렉션 (Garbage Collection, GC)을 수행한다. SSD의 GC는 여러 유효 페이지를 복사하고 블록 지우기 연산을 수행하여 SSD 성능을 하락시킨다.

SSD에서 작은 크기의 임의 쓰기 성능은 다양한 이유로 큰 크기의 순차 쓰기보다 성능이 낮다. 먼저, 임의 쓰기는 여러 산재된 블록에 무효 페이지를 생성하기 때문에 GC 비용이 증가한다. 두 번째는 작은 크기의 요청이 들어왔을 때, SSD의 병렬 구조를 완전히 활용할 수 없다. 큰 크기의 요청은 여러 플래시 페이지들을 한 번에 기록할 수 있지만, 작은 크기의 요청은 많은 플래시 칩들을 유휴 상태로 만들기 때문이다. 세 번째는 작은 크기의 요청은 상대적으로 큰 처리 비용이 든다. 마지막으로, SSD의 제한된 메모리 크기 때문에 메모리에 주소 매핑 테이블의 일부만 캐시 할 수 있다면, 임의 쓰기 요청은 순차 쓰기 요청과 비교하여 맵 페이지 적재를 자주 요청하게 된다.

SSD의 높은 I/O 성능을 활용하기 위하여, 로그 구조 파일 시스템(Log-structured File System, LFS)[2]은 모든 쓰기를 큰 순차 쓰기로 기록하기 때문에 SSD에 적합한 파일 시스템이다. LFS는 여러 순차적 블록으로 구성된 세그먼트(Segment)의 단위로 저장장치에 기록한다. 전통적인 파일 시스템과 달리 LFS는 모든 쓰기를 저장장치에 순차적으로 기록하기 때문에 임의 쓰기의 성능이 매우 뛰어나다. 그러나, 현재 LFS는 SSD에서 기대만큼의 좋은 성능을 보여주지 못한다. 가장 큰 문제점은 LFS의 높은 세그먼트 클리닝(호스트 GC) 비용이다. 파일 시스템에 여유 세그먼트가 부족하면, LFS는 파일 시스템 블록을 회수하기 위하여 세그먼트 클리닝을 수행한다. 이러한 호스트 GC는 희생(Victim) 세그먼트의 유효 블록들을 다른 세그먼트로 복사하기 위하여 호스트와 저장장치 사이에서 많은 I/O를 요청한다. 또한, GC는 복사 연산의 많은 데이터 쓰기로 인하여 플래시 칩의 제한된 수명에 영향을 준다.

본 논문에서는 호스트와 SSD 사이에 데이터 전송 없이 파일 시스템 블록의 복사가 가능한 논리적 복사(LC) 기법을 제안한다. 호스트 GC의 블록 복사 연산은 데이터 값 변경 없이 데이터의 논리 주소만 변경 되므로 FTL이 관리하는 주소 매핑 테이블의 수정만으로 복사 연산이 가능하다. 제안된 LC 기법은 호스트 GC 부하를 제거함으로써 LFS의 성능과 SSD의 수명을 크게 향상 시켰다.

2. 관련 연구

2.1 LFS의 세그먼트 클리닝

LFS의 세그먼트 클리닝은 세 가지 단계로 진행된다. 첫

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 서울어코드활성화지원사업의 연구결과로 수행되었음 (IITP-2015-R0613-15-1062)
본 연구는 미래창조과학부 및 정보통신기술진흥센터의 SW중심대학-ICT/SW창의연구과정 지원사업의 연구결과로 수행되었음 (IITP-2015-R2215-15-1005)

번째로 세그먼트 클리닝을 진행할 희생 세그먼트를 선정한다. 두 번째로 희생 세그먼트의 유효 블록들을 여유 세그먼트로 복사한다. 이 과정에서 파일 시스템은 많은 읽기와 쓰기를 요청한다. 그 다음, 이전의 체크포인트가 희생 세그먼트의 블록을 참조하기 때문에 LFS가 희생 세그먼트를 재활용하기 위해서는 체크포인트 과정을 수행해야 한다. 체크포인트는 파일 시스템의 모든 더티(Dirty) 메타 데이터를 저장장치에 기록함으로써 파일 시스템의 상태를 저장한다. 체크포인트를 수행한 뒤, LFS는 무효화된 블록에 trim 명령을 보낸다.

GC의 복사 연산을 피하기 위해서 LFS는 일반적인 로깅 대신 TL (Threaded Logging)[2, 3] 기법을 사용할 수 있다. TL은 여유 세그먼트가 아닌 무효 블록이 존재하는 더티 세그먼트를 할당 받아 더 이상 쓰지 않는 무효 블록에 덮어 쓰기를 수행한다. 그러나 이러한 방식은 데이터를 분산하여 기록하는 임의 쓰기가 발생하기 때문에 일반적인 로깅과 비교하여 쓰기 성능이 떨어질 수 있다.

2.2 디바이스 레벨 주소 매핑 변경 기법

본 논문에서는 디바이스 내부의 맵 주소 변경을 통하여 복사 연산을 구현하였다. 이와 유사하게 FTL 맵 수정을 통하여 쓰기 양을 감소 시키는 연구들이 존재한다. JFTL[7]은 저널링 파일 시스템의 중복 쓰기 문제를 다루었고, XFTL[8]은 WAL 기법의 중복 쓰기 문제를 해결하였다. 이 두 기법은 LC와 달리 일반적인 복사 연산에 대해서 사용할 수 없다. ANVIL[6]은 LC와 비슷하게 주소 재배치를 통한 복사, 이동, 삭제의 인터페이스를 제안하였다. 그러나 실제 디바이스에 이러한 기법을 적용하였을 때 발생하는 맵 관리 문제와 복구 문제를 다루지 않았다.

3. 설계

3.1 논리적 복사 기법

논리적 복사 기법(Logical, copy, LC)기법은 SSD의 매핑 테이블을 이용하여, LFS에서 세그먼트 클리닝의 복사 연산 부하를 감소시키는 것이다. 이 기법은 복사 연산을 수행할 때 읽기와 쓰기 대신 새로운 명령인 LC를 통하여 SSD 내부에서 복사 연산을 수행하도록 한다. 이 때, 디바이스 내부에서는 데이터를 복사하는 것이 아니라 주소 매핑 테이블을 수정하여 두 LPN(Logical Page Number)들이 하나의 PPN(Physical Page Number)를 가리키게 한다.

LC 기법을 현재 SSD 디바이스에 구현하는 것은 단순한 일이 아니다. 일반 SSD에서 P2L(Physical-to-Logical) 매핑 테이블은 페이지 쓰기 이후 수정되지 않지만 LC 기법에서는 LC로 인하여 P2L 매핑 테이블이 수정된다. 또 다른 문제는 LC를 수행한 후 trim 명령이 요청되기 전 까지 두 LPN이 하나의 PPN을 공유해야 한다. 그러므로 두 LPN을 관리하기 위한 P2LL(Physical-to-Logical&Logical) 매핑 테이블이 필요하다. 그러나 SSD가 모든 PPN에 대하여 P2LL 매핑 테이블을 관리하게 되면, 두 개의 LPN들이 참조하는 PPN는 전체 주소 공간의 일부이기 때문에 메모리 공간이 낭비된다.

이러한 문제를 해결하기 위하여, 두 개의 LPN을 매핑하고 있는 일부 PPN을 위하여 버추얼 맵(Virtual Map)을 도입하였다. 예를 들어, 그림 1에서 파일 시스템의 GC 프로세스는 희생 세그먼트의 두 블록 A, B를 복사하기 위하여 LC 명령을 요청하였다. LC 명령은 SSD에게 복사할 데이터가 있는 주소, 복사 위치 주소, 복사 크기를 알려준다. SSD에 LC_1 명령이 도착하면, FTL은 L2P(Logical-to-Physical) 테이블을 참조하여 LPN 10의 데이터가 PPN 300에 저장되었음을 알아낸다. 그 다음, 버추얼 맵에서 VPN(Virtual Page Number)을 할당하여 PPN 300과 복사할 데이터가 있는 주소(LPN 10)와 복사 위치 주소(LPN 30)를 기록한다. 마지막으로 L2P와 P2L 테이블에 할당한 VPN인 V1을 기록한다.

3.2 복구

제안된 LC 기법은 갑작스런 전원 종료가 발생 시, 안전하게 복구할 수 있다. 그림 2는 저장장치 펌웨어와 파일 시스템의 복구 예시를 보여준다. 파일 시스템은 고장이 발생하면 CP_{10}, CP_{11}과 같은 가장 최근의 체크포인트로 복구 되며, SSD는 맵이 저장장치에 반영되었을 때의 상태로 복구된다. C_1과 C_2 시점에서 고장이 발생하면 파일 시스템은 CP_{10}의 상태로 돌아간다. SSD 복구 모듈 또한 LC로 수정된 매핑 테이블을 플래시에 반영하지 않았기 때문에 CP_{10}의 상태로 복구 된다. C_3에서 고장이 발생하면 파일 시스템은 CP_{10}의 상태로 복구되지만 SSD에서는 활성 블록이 변경되어 맵이 저장장치에 기록되었으므로 W_3 연산 까지 반영 된 상태로 복구된다. 하지만 버추얼 맵이 존재하기 때문에 LC 수행 전의 블록도 접근 가능하여 파일 시스템의 데이터 손실이 없다. 마지막으로 C_4에서 고장이 발생하였을 때, 체크포인트 CP_{11} 이후이기 때문에 SSD 복구 모듈과 파일 시스템 복구 모듈이 호스트 GC 수행이 반영되어 있는 CP_{11} 상태로 복구 된다.

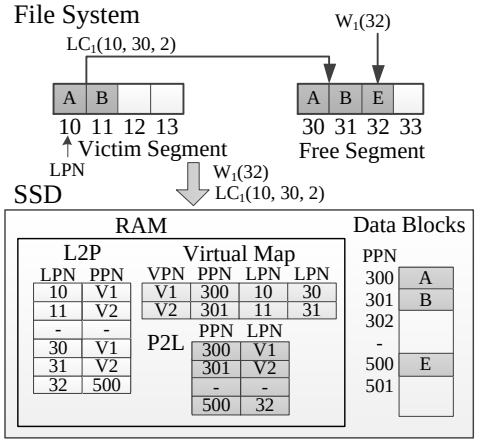

그림 1 LFS 가비지 컬렉션 수행 시 논리적 복사 연산

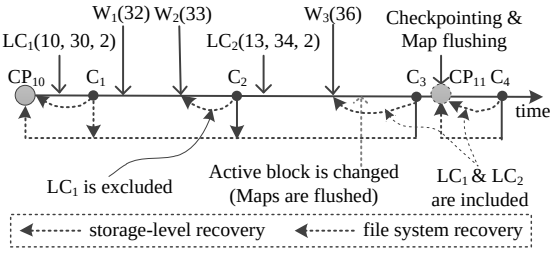

그림 2 복구 예시

4. 실험

4.1 실험 환경

제안한 논리적 복사 기법의 유효성을 측정하기 위하여, 우리는 Jasmine OpenSSD 플랫폼[5]에 논리적 복사를

구현하였다. 호스트 시스템에서는 리눅스 3.12.7 커널 버전의 수정된 F2FS[3]를 사용하였다. 사용한 OpenSSD의 용량은 4GB이며 플래시 페이지의 크기는 8KB, 블록의 크기는 1MB이다. OpenSSD의 FTL로는 파일 시스템의 블록 단위와 같은 4KB 단위의 페이지 매핑을 사용하였다.

실험은 EXT4와 세 가지 버전의 F2FS의 성능과 쓰기 양을 비교하였다. 먼저, 호스트 GC 사용을 피하고 최대한 TL을 수행하는 기존 F2FS(ORG), NL(Normal Logging)만 수행하는 F2FS (NL-RW) 그리고 GC를 논리적 복사로 수행하는 F2FS (NL-LC)이다. 각 실험을 진행하기 전에, 호스트 GC가 발생하는 환경을 만들기 위하여 미리 저장장치를 초기화 하였다. 초기화는 1000개 파일 생성과 16KB 크기의 임의 갱신으로 진행된다. 파일 크기는 파일 시스템 활용률(Utilization)에 따라 결정되며 호스트 GC가 발생하기 직전까지 여러 파일들을 갱신하였다.

사용한 벤치마크는 네 가지 이다. Tiobench는 4개의 쓰레드가 100MB 파일에 25MB를 임의로 갱신한다. Copy&tar는 150MB의 압축된 파일을 복사한 후 압축을 푼다. Varmail은 16개 쓰레드가 2000개의 파일을 생성하고 fsync 시스템 콜을 요청한다. Dbench는 8개의 쓰레드가 여러 파일 연산을 수행하고 fsync 시스템 콜을 요청한다.

4.2 실험 결과

그림 3은 파일 시스템에 따른 I/O 성능을 보여 준다. EXT4와 ORG는 작은 I/O 크기로 인하여 SSD의 병렬 구조를 전부 활용하지 못하였다. 반면에 NL-RW는 큰 I/O로 SSD의 병렬 구조를 전부 활용하였지만 호스트 GC의 복사 연산으로 인하여 성능이 하락하였다. 그리고 파일 시스템 활용률이 높아질수록 더 큰 성능 감소를 볼 수 있다. 제안한 기법이 적용 된 NL-LC는 NL-RW의 단점인 호스트 GC 비용을 감소시켜 NL-RW와 비교하여 Tiobench 실험에서 최대 3배, Varmail에서 1.3배, Dbench에서는 1.5배의 성능 향상을 보였다. 예외로 Copy&Tar의 경우, I/O 패턴이 순차 쓰기 이므로 EXT4가 수행 시간이 가장 짧다. 제안된 NL-LC는 NL-RW와 ORG과 비교하여 성능이 각각 최대 2.8배, 1.2배 향상하였다.

그림 4는 SSD 펌웨어에서 측정한 쓰기 양을 나타낸 것이다. Tiobench, Copy&tar 실험에서 NL-RW은 호스트 GC으로 인하여 많은 쓰기를 수행하는 것을 확인할 수 있으며 파일 시스템 활용률이 높을 수록 호스트 GC의 양이 많아 진다. 제안된 NL-LC는 쓰기와 읽기 대신 LC를 사용하여 기존 NL-RW와 비교하였을 때 쓰기를 최대 80% 감소시켰다. 그러나, NL-LC는 FTL 메타데이터(P2L) 갱신으로 인하여 EXT4와 비교해서 10-30% 정도 쓰기 양이 증가하였지만 순차 쓰기로 인하여 더 좋은 성능을 보여준다. Varmail, Dbench는 특정 공간에만 집중적으로 갱신하는 특성이 있어 비교적 호스트 GC 복사의 양이 적다. 또한 잦은 fsync 연산으로 flush 명령이 자주 요청되어 많은 FTL 메타데이터가 기록된다. 특히 EXT4와 ORG은 임의 쓰기를 수행하기 때문에 더 많은 L2P 맵을 기록하여 유저 쓰기 대비 FTL 메타데이터 쓰기의 양이 크다.

5. 결론

본 논문에서는 논리적 복사 인터페이스를 제공하여 로그 구조 파일 시스템의 거의 비용이 들지 않는 세그먼트 클리닝을 수행할 수 있도록 하였다. 논리적 복사는 데이터 접근 없이 디바이스 내부의 매핑 테이블만을 수정하여 복사와 동일한 효과를 낼 수 있다. 실험 결과, 기존 시스템과 비교하여 최대 3배의 성능 향상을 보였으며 약 80%의 쓰기를 감소 시켰다.

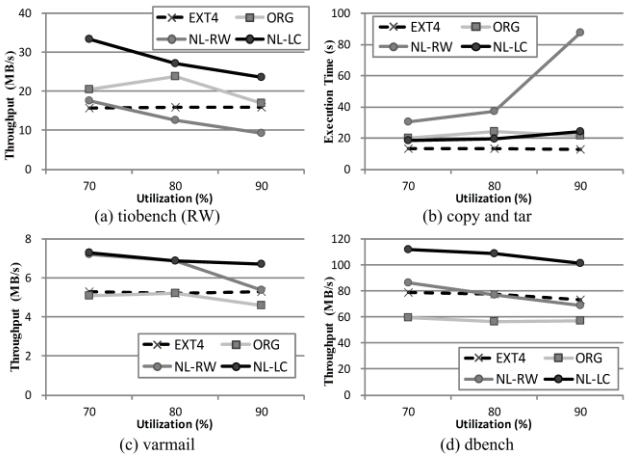

그림 3 I/O 성능 비교

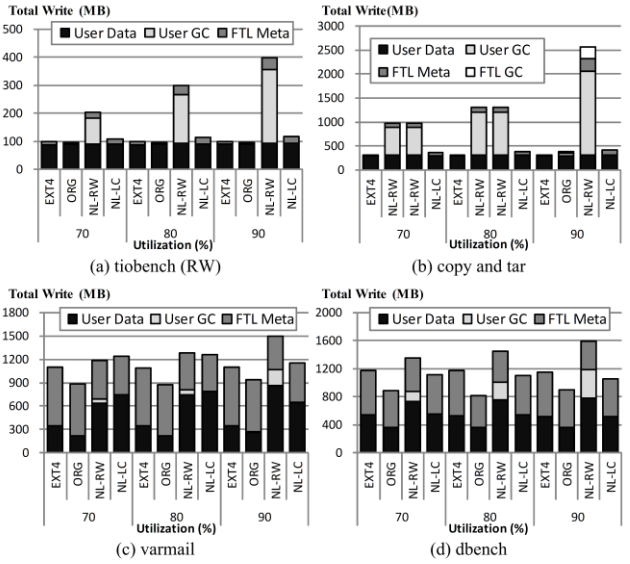

그림 4 쓰기 양 비교

참고문헌

[1] The OpenSSD Project. http://www.openssd-project.org/.
[2] Rosenblum, et al. "The design and implementation of a log-structured file system." ACM Transactions on Computer Systems (TOCS) 10.1, pages 26-52, 1992.
[3] Lee, Changman, et al. "F2FS: A new file system for flash storage." In Proceedings of the USENIX Conference on File and Storage Technologies (FAST), pages 273-286, 2015.
[4] Choi, Hyun Jin, et al. "JFTL: A flash translation layer based on a journal remapping for flash memory." ACM Transactions on Storage (TOS), 4.4, pages 1-22, 2009.
[5] Kang, Woon-Hak, et al. “X-FTL: Transactional FTL for SQLite Databases.” In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD), pages 97–108, 2013.
[6] Weiss, Zev, et al. "ANViL: advanced virtualization for modern non-volatile memory devices." In Proceedings of the USENIX Conference on File and Storage Technologies (FAST), pages 111-118, 2015.