SSD에서 오프라인 중복 데이터 제거를 위한 플래시 메모리 블록 구분 기법

Block Separation Technique for Offline Deduplication on Solid State Drives

저자
(Authors)

강윤지, 안정철, 신동군
Yunji Kang, Jeongchoel An, Dongkun Shin

출처
(Source)

한국정보과학회 학술발표논문집 39(1A), 2012.6, 379-381 (3 pages)

발행처
(Publisher)

한국정보과학회
KOREA INFORMATION SCIENCE SOCIETY

URL

http://www.dbpia.co.kr/Article/NODE01901543

APA Style

강윤지, 안정철, 신동군 (2012). SSD에서 오프라인 중복 데이터 제거를 위한 플래시 메모리 블록 구분 기법. 한국정보과학회 학술발표논문집, 39(1A), 379-381.

SSD에서 오프라인 중복 데이터 제거를 위한 플래시 메모리 블록 구분 기법

강윤지, 안정철, 신동군
성균관대학교 정보통신공학부
oso41@skku.edu, luckyjc7@skku.edu, dongkun@skku.edu

Block Separation Technique for Offline Deduplication on Solid State Drives

Yunji Kang, Jeongchoel An, Dongkun Shin
Sungkyunkwan University

요 약

중복 제거(deduplication)기법은 저장장치의 공간을 효율적으로 사용할 수 있도록 해 주기 때문에 기존의 스토리지 시스템에서 많이 사용된 기법이다. 최근에는 플래시 메모리 기반의 SSD를 위한 중복 제거 기법도 많이 제안되었지만, 플래시 메모리의 특성을 고려하지 못하고 있다. 본 논문에서는 오프라인 중복 제거 기법을 대상으로 SSD의 특성을 고려하여 가비지 컬렉션의 비용을 절감할 수 있도록 중복 가능성이 있는 데이터와 중복 가능성이 없는 데이터를 온라인에 구분하여 플래시 메모리의 다른 영역에 기록하여 오프라인 중복 제거 후에 가비지 컬렉션 성능을 향상 시키는 기법을 제안하였다. 실험결과, 제시된 기법은 가비지 컬렉션 비용인 페이지 이동 횟수를 약 80%이상 감소시켰다.

1. 서 론

파일 시스템에서 데이터 중복은 흔히 일어나는 일이다. 예를 들어, 다수의 사용자들이 사용하는 서버 컴퓨터에서는 같은 파일이 중복하여 생성되는 경우가 많으며, 개인 사용자도 동일한 부분이 많은 다수의 파일을 스토리지에 기록하는 경우가 많다. 이렇게 중복되는 데이터를 찾아 내어 한 개의 데이터를 여러 사용자나 파일이 공유하도록 하는 기법인 중복 제거(Deduplication) 기법이 기존의 시스템에서 많이 사용 되어져 왔다. 중복 제거는 스토리지의 공간을 늘릴 수 있다는 장점이 있으며, 특히 쓰기 횟수가 제한되어 있는 플래시 메모리 기반의 SSD(Solid State Drive)에서는 SSD의 수도 연장시켜 주는 유용한 기법이다.

중복제거 기법은 중복제거 과정이 언제 수행되는가에 따라서 온라인 중복제거와 오프라인 중복제거로 나눌 수 있다. 온라인 중복제거는 런타임에 사본 여부를 판단하여 사본 처리를 하는 기법이고, 반면에 오프라인 중복제거는 데이터를 일단 스토리지에 기록한 후에 유휴시간에 중복제거를 하는 기법이다. 온라인 기법은 런타임에 사본 처리를 하므로 쓰기 연산을 줄여서 쓰기 성능을 향상 시킬 수 있지만, 중복 데이터가 많지 않거나 다수의 쓰기 요청이 동시에 올 경우에는 사본 탐색 시간이 병목이 되어 오히려 응답시간을 증가시킬 수 있다. 반면에 오프라인기법은 모든 데이터를 일단 스토리지에 기록하므로 중복 제거를 통한 쓰기 성능 향상은 없지만, 쓰기 요청 처리를 위한 오버헤드가 없어 중복 데이터가 적은 경우에 유리하며, 가비지 컬렉션(Garbage Collection) 비용을 감소시켜 전체적으로 쓰기 성능을 개선할 수 있다. 본 논문은 SSD를 대상으로 하는 데, 최근에 SSD가 사용되는 개인용 컴퓨터 시스템에서는 서버 컴퓨터나 백업 장비에 비해 데이터 중복률이 낮기 때문에 본 논문에서는 SSD를 위한 오프라인 중복제거에 대해 다룬다.

기존에 다양한 온라인/오프라인 중복제거 기법들이 제안되었지만, 대부분이 HDD를 대상으로 하고 있어 SSD의 특성을 고려하지 않고 있다. 본 논문에서는 데이터를 SSD에 기록할 때 중복제거가 될 가능성을 조사하여 가능성 여부에 따라서 서로 다른 플래시 메모리 블록에 기록하여 SSD에서 가비지 컬렉션 비용을 감소시키는 기법을 제안한다.

본 연구는 지식경제부 및 정보통신산업진흥원의 지원사업의 연구결과로 수행되었음

2. 관련 연구

2.1 중복제거

중복제거는 스토리지 시스템에서 동일한 데이터는 제거하고, 공유하는 기법이다. 그림 1은 스토리지 내에 데이터 A가 서로 다른 위치에 중복 저장되어 있고, 각각 서버A와 서버B가 사용하고 있는 경우 어떻게 중복제거를 하는지 보여준다. 그림1과 같이 한 쪽의 데이터는 제거하고, 다른 한 쪽의 데이터로 공유하게 하여, 데이터의 손실 없이 스토리지의 빈 공간을 확보한다.

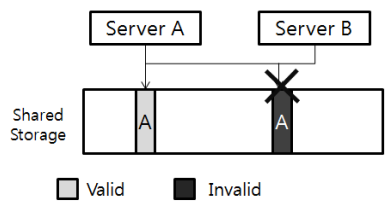

그림 1 중복제거

SSD를 위한 중복제거 기법으로는 CAFTL[1]이 제안되었다. 온라인 중복제거와 오프라인 중복제거 둘 다 지원하고 있으나 주로 온라인 중복제거에 초점을 맞춘 연구이다.

2.2 온라인 중복제거

온라인 중복제거는 런타임에 중복제거를 하는 기법이다. 먼저 쓰기 요청이 오면 데이터는 쓰기 버퍼에 저장이 된다. 그 다음 데이터를 청크(chunk)라는 단위로 분리하고, 해시 엔진을 통해 청크의 핑거프린트를 계산한다. 핑거프린팅(Fingerprinting)은 SHA-1이나 MD5와 같이 충돌이 거의 없는 암호화 알고리즘을 이용해 청크에 대한 키를 만드는 과정이다[2]. 데이터를 나누는 기준에 따라, 일정한 크기의 블록들로 나누는 정적 청킹과 데이터의 값에 따라 서로 다른 크기의 블록들로 나눌 수 있는 가변 청킹이 있다. 핑거프린트가 계산이 되면 핑거프린트 테이블에서 동일한 값이 존재 하는지 검사한다. 동일한 핑거프린트가 존재한다면 사본이 존재한다는 뜻이므로 데이터를 쓰지 않고 매핑 테이블만 조작하여 사본을 가리키도록 한다. 만약, 핑거프린트를 찾지 못한다면 그대로 스토리지에 데이터를 쓴다.

많은 수의 쓰기 요청이 동시에 발생하면 핑거프린팅 및 검색을 위한 지연시간이 발생하여 스토리지 시스템의 성능이 저하 될 수 있다.

2.3 오프라인 중복제거

중복제거 시스템에서 CAFTL의 오프라인 중복제거는 단순하다. 데이터를 핑거프린팅과 검색을 거치지 않고 스토리지에 먼저 쓴 후, 읽기/쓰기 요청이 없는 유휴시간에 데이터들에 대한 핑거프린팅 및 검색을 수행하여 사본을 무효화 시킨다. SSD에서 오프라인 중복 제거는 무효화된 페이지를 많이 생성시켜 가비지 컬렉션 비용을 감소시켜 준다.

3. 본 론

3.1 오프라인 중복 제거를 위한 블록 구분 기법

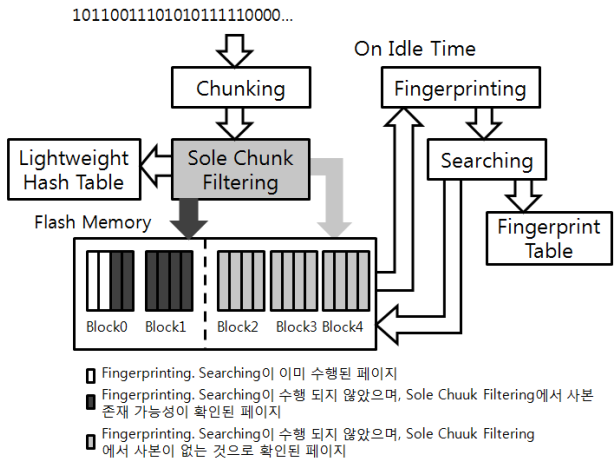

그림 2 오프라인 중복 제거를 위한 블록 구분 기법

제안하는 기법에서는 그림 2과 같이 데이터가 들어오면 먼저 청킹 과정을 거치게 된다. 청킹 과정에서는 데이터 접근 용이성을 위하여 페이지 단위의 정적 청킹을 수행한다.

청킹 후 데이터는 유일 청킹 필터(Sole Chunk Fliter)를 통해 사본 존재 여부의 가능성만을 판단하는 데, SHA-1, MD5등의 암호화 알고리즘에 비해 충돌 가능성은 높지만 연산속도가 매우 빠른 CRC32와 같은 비교적 간단한 해시 알고리즘을 이용한다. CRC32는 충돌 비율이 높으므로 중복여부를 정확히 판단해 주지는 않지만, 해당 데이터가 스토리지에서 유일한 데이터인 경우에는 그 사실을 확인할 수 있는 경우가 생긴다. 즉, 어떤 데이터의 CRC32 키 값이 기존 데이터들의 CRC32 테이블에서 발견되지 않으면 해당 데이터는 중복되지 않은 데이터가 명확하며, 같은 키 값이 테이블에서 발견된다면 중복 데이터가 있을 수도 있고 없을 수도 있는 불명확한 상태이며 이것은 오프라인 중복 제거에서 충돌 비율이 낮은 해시 기법으로 재조사가 필요하다.

쓰고자 하는 데이터 페이지가 유일 청킹 필터링 단계에서 유일 데이터 여부가 불명확한 것으로 판단되면, 그림 2의 0,1번 블록처럼 사본이 있을 가능성이 있는 데이터 페이지를 저장하는 영역의 빈 페이지에 기록하며, 유일 데이터 여부가 확실하면 그림2의 2,3,4번 블록처럼 사본이 없는 데이터 페이지를 저장하는 영역의 빈 페이지에 기록한다. 따라서 블록들은 사본이 있을 가능성이 있는 페이지만을 포함하거나, 사본이 없는 페이지를 포함하는 블록들로 나뉘게 된다.

읽기/쓰기 요청이 없는 유휴 시간에는 전체 SSD 영역이 아닌 중복 가능성이 확인되지 않은 블록들에 대해서만 검색을 수행하여 사본을 제거하면 되기 때문에 스토리지 스캔 시간을 크게 줄일 수 있다. 중복 데이터로 확인된 페이지는 매핑 테이블에 공유 정보 설정 후 무효 상태로 변경 된다.

사본이 있을 가능성이 큰 데이터와 사본이 없는 데이터를 다른 블록에 구분 저장하였기 때문에 오프라인 중복 제거 후에는 사본이 있을 가능성이 큰 데이터를 기록했던 블록들이 주로 많은 무효 페이지를 가지게 된다.

3.2 제안 기법의 효과

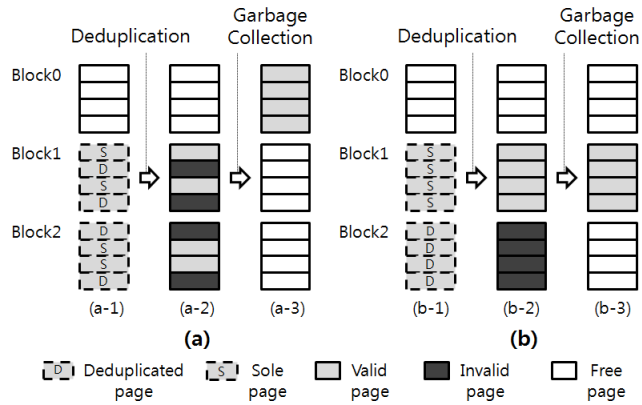

그림 3 (a) 기존의 오프라인 중복제거 기법
(b) 제안한 오프라인 중복제거 기법

그림3의 (a)는 기존의 오프라인 중복제거 기법에서 가비지 컬렉션을 수행할 때 블록들의 상태를 나타낸 것이다. 기존 기법에서는 쓰고자 하는 데이터들을 구분 없이 일괄적으로 쓴다. 따라서 쓰기 완료 후 플래시 메모리 블록들은 사본이 존재하는 페이지와 사본이 존재하지 않는 페이지가 섞여 있는 상태가 된다. 이후 유휴시간에 오프라인 중복 제거에 의해서 사본이 존재하는 페이지들은 공유 정보 설정 완료 후 무효

페이지가 된다. 따라서 가비지 컬렉션을 수행 할 때 4개의 유효 페이지를 복사하는 비용이 발생한다.

그림 3의 (b)는 제안한 오프라인 중복제거 기법을 사용했을 경우이다. 데이터 쓰기 직전 사본이 존재할 가능성에 따라서 다른 블록에 저장하기 때문에 오프라인 중복 제거 후에 사본이 존재할 가능성이 큰 페이지들은 대부분 무효 페이지가 되며, 각 블록들은 유효 페이지와 무효 페이지가 섞이지 않은 상태가 된다. 따라서 가비지 컬렉션 시간에 유효 페이지를 복사하는 비용을 크게 줄일 수 있다.

4. 실 험

4.1 실험 환경

본 논문의 실험에서는 SSDSim[3]을 수정하여 중복제거를 시뮬레이션 하도록 하였다. 실험에 사용한 트레이스는 WebVM[4]과 어플리케이션 인스톨이 있다. WebVM은 웹 메일 프록시와 온라인 코스 관리에 대한 트레이스이고 어플리케이션 인스톨은 여러 어플리케이션을 버전 별로 설치하면서 수집한 트레이스이다. CRC32과 SHA-1의 소프트웨어 해싱 시간은 Simple Scalar ARM버전을 이용하여 측정한 값인 0.013ms, 0.152ms으로 설정하였고, 핑거프린트 저장을 위한 RAM 크기는 2MB로 설정하였다.

4.2 응답 시간

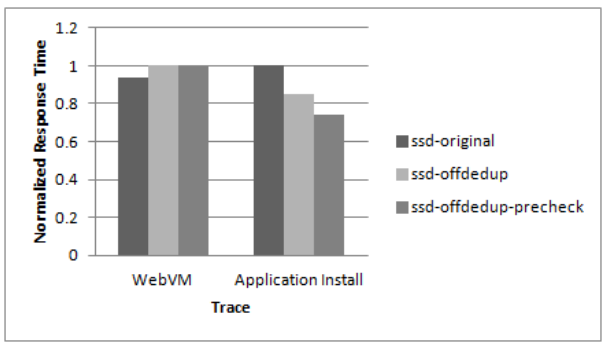

그림 4 응답 시간

그림 4는 중복제거를 하지 않은 경우(ssd-original), 기존 오프라인 중복제거(ssd-offdedup)기법과 제안한 오프라인 중복제거(ssd-offedup-precheck)기법의 이렇게 세 가지 설정에서 각 IO의 평균 응답시간을 측정한 것이다.

이 실험에서 응답시간에 영향을 미치는 세 가지 요소가 있다. 첫 번째는 유휴시간에 중복처리를 하는 도중에 읽기/쓰기가 발생했을 때 중복처리 완료를 기다리는 시간으로 이것은 응답시간을 증가시킨다. 두 번째로는 중복제거로 인해 줄어든 가비지 컬렉션의 횟수로서 응답시간을 감소시킨다. 마지막으로 제안된 오프라인 기법에서 줄어든 가비지 컬렉션 비용으로서 응답시간을 감소시킨다.

WebVM은 초당 약 10개의 요청을 받는다. 한꺼번에 들어오는 요청의 수가 적으므로 쓰기를 할 때, 동시에 가비지 컬렉션을 하지 않는다. 그러므로 두 번째 경우의 효과는 받지 못하며 첫 번째 경우로 인하여 오리지날 보다 오프라인 중복제거가 약간의 응답시간이 크다. 기존 오프라인 중복제거보다 제안된 오프라인 중복제거가 응답시간이 빠른 것은 세 번째 경우의 영향을 받은 것이다.

반면에 어플리케이션 인스톨은 약 300개의 요청을 받는다. 한번에 들어오는 요청의 수가 많으므로 쓰기와 가비지 컬렉션을 함께 하게 된다. 그림 4에서 어플리케이션 인스톨은 두 번째, 세 번째 영향을 많이 받고 있음을 알 수 있다.

4.3 가비지 컬렉션과 페이지 이동 횟수

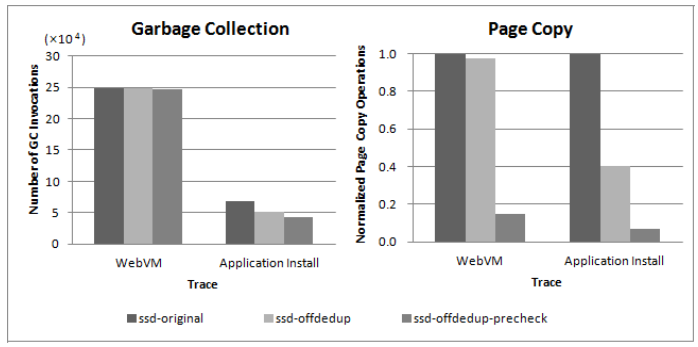

그림 5 총 가비지 컬렉션 횟수와 페이지 이동 횟수

그림 5는 가비지 컬렉션 횟수와 페이지 이동 횟수를 비교한 그래프이다. WebVM에서는 가비지 컬렉션의 횟수에서 약간의 감소가 보이며 가비지 컬렉션 비용인 페이지 이동횟수에서는 오리지날과 기존 중복제거 대비 약 85%가 감소하였다는 것을 볼 수 있다. 어플리케이션 인스톨에서도 가비지 컬렉션 횟수가 오리지날 대비 약 38%, 기존 오프라인 중복제거 대비 약 18% 감소하였다. 페이지 이동은 93%와 82% 감소하였다.

5. 결 론

본 논문에서는 기존의 오프라인 중복제거 기법을 개선한 블록 구분 기법을 제안하였다. 데이터를 쓰기 전 비교적 낮은 비용의 해시 알고리즘을 이용하여 사본이 존재할 가능성이 큰 데이터를 구분해서 플래시의 다른 영역에 모아 저장함으로써 페이지의 이동 횟수를 줄이는 기법이다. 실험 결과, 페이지 이동횟수에서 기존 오프라인 중복제거보다 80% 이상 페이지 복사 횟수를 감소시켜 가비지 컬렉션 비용을 감소시킴을 확인하였다.

참고문헌

[1] F. Chen et al, CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives, Proc. of the USENIX Conference, 2011.

[2] M. Rabin, Fingerprinting by random polynomials, Center for Research in Computing Technology Harvard University Report TR-15-81, 1981.

[3] N. Agrawal et al, Design Tradeoffs SSD Performance, Microsoft Research, 2008.

[4] R. Koller et al, I/O Deduplication: Utilizing Content Similarity to Improve I/O Performance, Proc. of USENIX File and Storage Technologies, 2010.