논문 2012-49-12-17

SSD 기반 스토리지 시스템에서 중복률과 입출력 성능 향상을 위한 데이터 중복제거 및 재활용 기법

(Data De-duplication and Recycling Technique in SSD-based Storage System for Increasing De-duplication Rate and I/O Performance)

김 주 경*, 이 승 규*, 김 덕 환**

(Ju-Kyeong Kim, Seung-Kyu Lee, and Deok-Hwan Kim)

요 약

SSD(Solid State Disk)는 다수의 NAND 플래시 메모리로 구성되었으며 내부에 고성능 컨트롤러와 캐시 버퍼를 포함한 스토리지 장치이다. NAND 플래시 메모리는 제자리 덮어쓰기가 안되기 때문에 파일시스템에서 유효페이지가 갱신 및 삭제시 무효페이지로 전환되어 완전히 삭제하기 위해서는 가비지 컬렉션 과정을 거쳐야한다. 하지만 가비지 컬렉션은 지연시간이 긴 Erase 연산을 포함하기 때문에 SSD의 I/O 성능을 감소시키고 마모도를 증가시키는 문제가 된다. 본 논문에서는 입력데이터에 대하여 유효데이터와 무효데이터에서 중복검사를 실행하는 기법을 제안한다. 먼저 유효데이터에 대한 중복제거 과정을 거치고 그 다음에 무효데이터 재활용 과정을 거침으로써 중복률을 향상시켰다. 이를 통하여 SSD의 쓰기 횟수와 가비지 컬렉션 횟수를 감소시켜 마모도와 I/O 성능이 개선되었다. 실험결과 제안한 기법은 유효데이터 중복제거와 무효데이터 재활용을 둘다 하지 않는 일반적인 경우에 비해서 가비지 컬렉션 횟수가 최대 20% 감소하고 I/O 지연시간이 9% 감소하였다.

Abstract

SSD is a storage device of having high-performance controller and cache buffer and consists of many NAND flash memories. Because NAND flash memory does not support in-place update, valid pages are invalidated when update and erase operations are issued in file system and then invalid pages are completely deleted via garbage collection. However, garbage collection performs many erase operations of long latency and then it reduces I/O performance and increases wear leveling in SSD. In this paper, we propose a new method of de-duplicating valid data and recycling invalid data. The method de-duplicates valid data and then recycles invalid data so that it improves de-duplication ratio. Due to reducing number of writes and garbage collection, the method could increase I/O performance and decrease wear leveling in SSD. Experimental result shows that it can reduce maximum 20% number of garbage collections and 9% I/O latency than those of general case.

Keywords : SSD, Data De-duplication, Invalid Data Recycling, FTL, Garbage Collection

Ⅰ. 서 론

SSD(Solid State Disk)는 여러개의 NAND 플래시 메모리가 병렬로 결합하여 구성된 스토리지 장치이다. SSD는 기계적으로 동작하는 HDD(Hard Disk Drive)와 다르게 반도체 기반으로 동작하기 때문에 데이터 전송 속도와 랜덤한 입출력 측면에서 성능이 향상되었고 소

* 학생회원, ** 정회원, 인하대학교 전자공학과
(Department of Electronic Engineering, Inha University)
※ 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2012-0001773)
※ 본 과제(연구)는 지식경제부와 한국산업기술진흥원의 전략기술 인력양성 사업으로 수행된 결과임
접수일자: 2012년10월25일, 수정완료일: 2012년11월26일

비전력을 낮춤으로써 최근에 서버, 데스크탑, 노트북의 스토리지 장치로 활용률이 점점 높아지고 있다[1].

SSD를 구성하고 있는 NAND 플래시 메모리의 크리티컬한 특성으로 각각의 셀마다 쓰기/삭제 횟수에 제한이 있고 제자리 덮어쓰기가 되지 않는다[2]. SLC(Single Level Cell)는 약 10만번의 횟수 제한, MLC(Multi Level Cell)는 약 1만번의 횟수 제한, TLC(Triple Level Cell)은 약 1천번의 횟수 제한이 있다. 만약 제한된 횟수를 초과하면 해당 셀은 정상적인 동작이 이루어지지 않는다[16]. 또한, NAND 플래시 메모리는 HDD와 다르게 제자리 덮어쓰기가 안되기 때문에 빈 페이지에만 쓰기 연산이 가능하다[17].

본 논문에서 다루게 될 유효데이터와 무효데이터는 SSD의 쓰기/삭제 연산에서 생성된다. 유효데이터의 발생은 SSD에 입력데이터가 저장되면 쓰기 연산을 통해 생성되고 무효데이터는 파일시스템의 유효데이터 갱신 및 삭제로 발생한다. 무효데이터는 가비지 컬렉션이 발생하여 완전히 삭제되기 전까지 유지되며 가비지 컬렉션 과정 후에 빈공간이 된다. 가비지 컬렉션은 NAND 플래시 메모리에서 블록의 무효페이지들을 Erase연산하여 빈 블록으로 만드는 과정을 의미한다.

SSD에서 데이터 입출력을 하기 위해서는 제한된 쓰기/삭제 횟수 안에서 사용해야한다. 하지만 빈번하게 대용량의 데이터가 입출력되는 서버에서 마모도가 급격히 증가한다면 SSD는 신뢰성이 줄어든다[3]. 유효데이터 중복제거는 이러한 SSD의 마모도 증가율을 줄여서 수명의 향상을 가져오는 기법이다[4]. 입력데이터와 SSD에 저장되어 있는 유효데이터를 중복검사하여 중복될 때 참조함으로써 쓰기 연산의 발생을 줄일 수 있다. 중복검사 과정에서 I/O 오버헤드가 발생하지만 SSD의 수명을 향상시킬 수 있다. 이는 대규모의 데이터를 다루는 서버급 스토리지에서 신뢰성을 제공하기 위해서 사용되는 기법이다.

가비지 컬렉션 연산은 SSD의 제자리 덮어쓰기가 않되는 특성을 보완하지만 속도가 느린 Erase 연산 때문에 SSD의 I/O 속도를 느리게하는 원인이다[5]. 따라서 가비지 컬렉션이 발생하는 횟수를 줄일 수 있다면 I/O 속도의 향상을 기대할 수 있을 것이다. 다수의 무효데이터를 삭제하여 빈 공간을 확보하는 것이 가비지 컬렉션이기 때문에 무효데이터를 줄일 수 있다면 가비지 컬렉션의 발생 빈도도 감소할 것이다. 이러한 이유로 무효데이터와 입력데이터를 중복검사해서 중복될 경우 무효데이터를 재활용한다면 성능을 향상 시킬 수 있다[6].

본 논문에서 제안하는 기법은 유효데이터 중복제거와 무효데이터 재활용을 동시에 수행함으로써 더 높은 중복제거율을 얻을 수 있는 기법이다. 기존의 중복제거 기법과의 차이점은 중복검사의 범위가 유효데이터에서 무효데이터까지 확장되었다는 점이다. 제안한 기법으로 쓰기 횟수를 감소시키고 가비지 컬렉션 횟수를 줄여서 SSD의 마모도와 I/O 성능을 향상 시킬 수 있다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 관련연구로 기존의 중복제거 기법에 대하여 살펴보고 Ⅲ장에서는 제안하는 기법에 대해서 설명한다. Ⅳ장에서는 실험 및 제안한 기법의 성능평가를 하고 Ⅴ장에서 결론 및 향후 연구방향을 제시한다.

Ⅱ. 관 련 연 구

데이터 중복제거 기법은 데이터의 중복된 저장을 피하여 스토리지 공간을 효율적으로 사용하는 기법으로, 특히 많은 양의 데이터를 다루는 네트워크 서버 및 백업 스토리지에서 저장 공간이 크게 절감되기 때문에 많이 사용된다. SSD 및 NAND 플래시 메모리를 활용한 중복제거 기법 연구는 F. Chen의 CAFTL[7], B. Debnath의 ChunkStash[8], D. Meister의 Dedupv1[9]등이 있다.

CAFTL은 SSD 스토리지 상에서 중복제거를 위한 특징을 제시하였다[7]. Pre-hashing은 SHA-1 해시함수를 사용하기 전에 CRC-32 해시함수로 먼저 중복검사를 하는 방법으로 CRC-32는 SHA-1 보다 10배 빠르기 때문에 신속하게 중복여부를 판단할 수 있다. 하지만 해시충돌 확률이 높은 단점이 있어서 추가로 SHA-1으로 확인한다. 또한 CAFTL은 즉시 중복제거를 하는 동안에 IOPS가 증가하여 병목현상이 발생할 수 있기 때문에 쓰기 버퍼의 공간을 고려한다. 버퍼의 사용량이 상한 경계값을 초과할 경우 즉시 중복제거를 중지하고 사용량이 하한 경계값 이하일 경우 즉시 중복제거를 실행하면서 병목현상과 중복제거의 성능을 조절한다. 위의 방법을 통해서 CAFTL은 효과적으로 SSD에서 중복제거를 실행하였다.

ChunkStash는 NAND 플래시 메모리를 사용한 SHA-1 해싱 기반 중복제거 기법이다[8]. 청킹과 SHA-1

해시 값을 만드는 과정으로 메타데이터를 생성한다. ChunkStash는 각 청크의 메타데이터를 NAND 플래시 메모리에서 관리하여 응답시간을 줄였다.

Dedupv1은 기존의 HDD 기반의 중복제거 시스템에 SSD를 사용하여 입출력 속도를 증가시켰다[9]. Dedupv1은 청킹과 SHA-1 해시함수를 이용한 핑거프린팅으로 고유한 값을 생성한 후에 생성된 핑거프린트를 기반으로 Filter Chain라는 중복검사 과정을 거친다. Filter를 4번 거치면서 중복검사를 실행하여 중복될 확률에 따라서 확실할 경우 Existing, 대체적으로 중복 가능성이 클 경우에 Strong-Maybe, 중복 가능성이 낮을 경우 Weak-Maybe, 검사결과 중복이 안 될 경우 Non-Existing으로 나눈다. Filter Chain 과정 후에 결과에 따라서 쓰기 연산을 하거나 중복제거를 한다.

중복제거에서 입력 파일을 청크 단위로 분할하는 청킹의 종류는 고정된 청크 단위로 일관적으로 분할하는 고정청킹과 가변적으로 청크 단위를 결정하는 가변청킹이 있다[10]. 고정 혹은 가변청킹 방법으로 청크를 생성한 후에는 각각의 청크에 대해서 중복검사를 위한 고유값인 핑거프린트를 해시함수로 생성한다. SHA-1 해시함수는 임의의 크기를 가진 데이터에 대하여 160bit의 일정한 해시값을 생성한다. 정교한 중복제거를 위해서 SHA-1 해시함수가 쓰인다[11]. 이렇게 청크에 대하여 생성된 해시값은 해당하는 청크만의 고유값이 되어 다른 청크와 일치하는지를 비교하는데 사용된다. 중복검사는 입력데이터의 핑거프린트가 해시테이블에서 일치하는 값이 있는지 검사하여 동작한다.

중복제거는 실행되는 시점에 따라 즉시 중복제거와 예약 중복제거 방식이 있다. 즉시 중복제거 방식은 입력데이터를 저장하기 전에 중복검사를 하여 중복제거를 하고 스토리지에 저장하는 방식이다. 중복제거를 먼저 한 후에 저장장치에 쓰기를 하기 때문에 저장공간의 쓰기 횟수를 줄일 수 있는 장점이 있다. 하지만 입력데이터가 많이 발생할수록 중복제거 과정에서의 오버헤드로 인하여 병목현상이 발생하는 단점이 있다[12]. 예약 중복제거 방식은 입력데이터를 우선 스토리지에 저장한다. 그 후에 호스트 시스템의 유휴시간(Idle Time)에 저장된 데이터를 중복검사하여 중복제거를 한다. 예약 중복제거는 즉시 중복제거를 하지 않기 때문에 쓰기 연산 중에 병목현상은 없지만 입력데이터를 우선 스토리지에 저장한 후에 중복검사를 하기 때문에 즉시 중복제거에 비해서 추가적인 저장공간이 필요하다는 단점이 있다[12].

Ⅲ. 데이터 중복제거 및 재활용 기법

본 논문에서 제안한 기법은 유효데이터와 무효데이터에 대해서 중복제거를 고려하는 기법으로 무효데이터는 SSD의 무효페이지(Invalid Page)에 저장되어 있는 데이터를 의미하고 유효데이터는 유효페이지(Valid Page)에 저장된 데이터를 의미한다[6].

1. 제안한 기법의 구조

제안한 기법은 유효데이터 중복제거와 무효데이터 재활용 과정으로 분류된다. 유효데이터의 중복제거를 위해서 파일시스템에 중복제거 컨트롤 계층인 Dedup

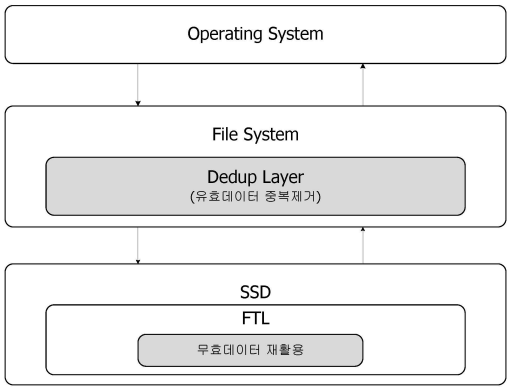


그림 1. 제안한 기법의 구조
Fig. 1. Structure of Suggested Method.

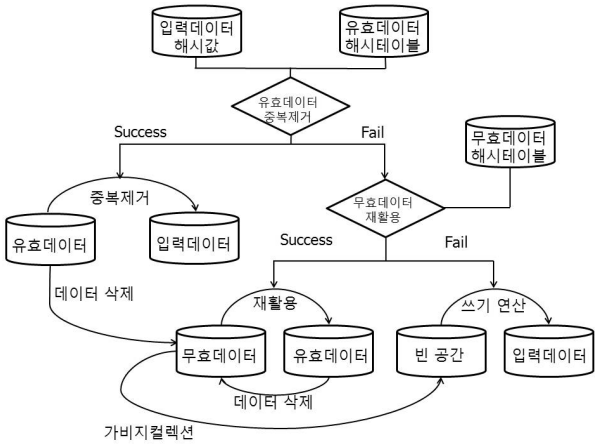


그림 2. 제안한 기법의 입력데이터 처리 과정
Fig. 2. Process of Input Data Handling in Suggested Method.

계층을 추가했다. 무효데이터 재활용은 무효페이지가 유효페이지로 전환되는 과정을 거쳐야하기 때문에 유효데이터의 중복제거와 다르게 SSD의 FTL에서 실행한다. 그림 1에 제안한 기법의 구조를 나타내었다.

그림 1에서와 같이 파일시스템으로 입력데이터의 쓰기 연산이 요청되면 우선 Dedup 계층에서 유효데이터와 중복되는지 중복검사를 실행한다. 중복되는 데이터가 없다면 FTL에서 무효데이터 재활용 과정을 거치게 된다. 무효데이터와도 중복이 되지 않는다면 SSD에 중복되는 데이터가 없으므로 빈 공간에 쓰기 연산을 실행한다. 그림 2는 제안한 기법에서 입력데이터의 처리 과정을 보여준다.

2. 유효데이터 중복제거

유효데이터의 중복제거는 입력데이터를 Dedup 계층에서 즉시중복제거 방법을 사용하여 실행한다. 입력데이터를 청킹할때 청크 단위는 SSD의 페이지 크기로 고정청킹한다. SSD의 페이지 크기로 고정청킹하는 이유는 무효데이터 재활용을 고려하기 때문이다. 무효데이터 재활용 과정에서 무효페이지가 유효페이지로 전환되기 때문에 청크가 페이지 크기와 일치해야한다.

그림3에서와 같이 입력데이터를 청크 단위로 분리한 후에는 각각의 청크에 SHA-1 해시함수를 적용하여 160bit의 핑거프린트를 생성한다. 그 다음 과정으로 입력데이터 청크의 핑거프린트 값과 유효데이터 해시테이블에서 동일한 값이 있는지 중복검사한다. 중복되었다면 입력데이터를 저장하지 않고 중복된 유효데이터를 참조한다. 중복된 데이터가 없을 경우에는 무효데이터 재활용 검사단계로 넘어간다.

유효데이터의 중복제거에서 해시테이블 공간이 필요하다. 페이지 크기가 4KB라고 한다면 청크마다 160bit의 SHA-1 해시값을 생성하기 때문에 청크 크기와 해시값 크기의 비율을 계산할 수 있다. 즉, 필요한 해시테이블의 저장공간은 4KB 크기의 입력데이터와 비교해서 약 0.5%인 20Byte를 차지한다. 따라서 유효데이터의 중복률이 0.5% 이상이면 중복제거를 통해 해시테이블의 저장공간이 확보된다. 여기서 중복률이란 스토리지에 저장된 데이터에 대한 중복 데이터의 비율을 의미한다.

유효데이터 중복제거에서 중복이 한 곳에서 다수 발생하여 하나의 유효데이터를 여러번 참조하게 된다면 유효데이터를 삭제하기 전에 고려해야 할 점이 있다.

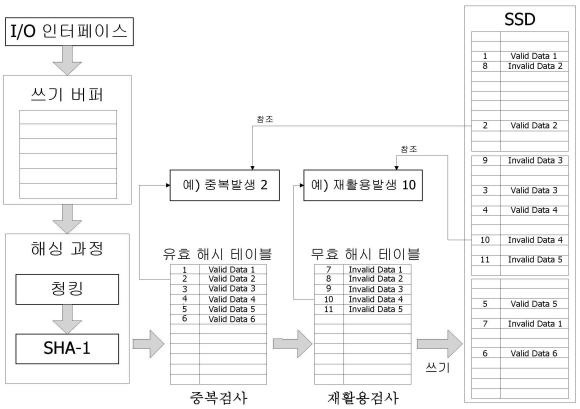


그림 3. 유효데이터 중복제거 및 무효데이터 재활용 기법

Fig. 3. Valid Data De-duplication and Invalid Data Recycling Method.

다수 중복이 발생한 유효데이터를 한 곳의 삭제 요청으로 삭제한다면 참조하고 있던 다른 곳에서는 실제 데이터가 삭제되어 오류가 발생한다[13]. 이런 오류를 방지하기 위해서 레퍼런스 카운트를 유효데이터의 메타정보로 추가하였다. 레퍼런스 카운트는 초기값으로 1이 주어지고 중복이 발생할 때마다 1씩 증가시킨다. 중복된 유효데이터를 참조하고 있는 논리 주소가 삭제되면 레퍼런스 카운트를 1씩 감소시킨다. 레퍼런스 카운트가 1일 때 삭제가 발생하면 유효데이터를 삭제하여 무효데이터로 전환한다.

3. 무효데이터 재활용

입력데이터가 유효데이터와 중복되지 않으면 무효데이터 재활용 과정을 거친다[6]. 무효데이터 재활용은 무효페이지를 유효페이지로 전환하기 때문에 페이지의 유효화/무효화를 관리하는 FTL에서 이루어진다. FTL은 SSD의 미들웨어로 주소 맵핑, 마모도 관리, 가비지 컬렉션의 역할을 한다[14]. FTL이 SSD 내부의 모든 블록과 페이지를 관리하기 때문에 무효데이터 재활용을 FTL에 추가했다.

NAND 플래시 메모리는 파일 엔트리에 블록에 대한 파일 정보가 유지된다. 각 파일 정보에는 현재 파일이 유효한지 또는 무효한지를 나타내는 플래그가 있다. 무효데이터 재활용시에 이 플래그를 수정하여 유효데이터로 전환한다.

입력데이터의 청킹과 핑거프린트 생성 과정은 이미 Dedup 계층에서 수행했기 때문에 바로 그림 3과 같이

무효데이터의 해시테이블과 중복검사를 한다.

유효페이지가 파일시스템의 갱신 및 삭제 요청으로 무효페이지로 전환된다면 해당 페이지의 해시정보는 무효데이터 해시테이블로 이동해야한다. 반대로 재활용 과정에서 무효페이지가 유효페이지로 전환된다면 유효데이터 해시테이블로 해시정보를 이동하여야 한다.

무효데이터 재활용 과정에서 FTL의 가비지 컬렉션 정책은 중복률에 영향을 준다. 기존 가비지 컬렉션 정책은 희생블록의 마모도와 유효페이지 복사에 필요한 비용만을 고려했다[15]. 하지만 무효데이터 재활용 관점에서는 가비지 컬렉션의 발생 빈도도 고려해야한다. 매우 빈번히 발생한다면 다수의 무효페이지가 삭제되어 재활용을 위한 무효페이지 수가 줄어들어 중복률이 감소한다. 반면에 가비지 컬렉션이 느슨하게 발생한다면 SSD에서 무효데이터가 차지하는 비중이 높아져서 저장공간의 활용률이 떨어지게 된다. 따라서 무효데이터 재활용을 위한 효율적인 가비지 컬렉션의 경계값을 구해야한다. 빈 블록의 비중이 경계값 이하로 감소하면 가비지 컬렉션을 실행하여 빈 블록을 확보하고 경계값 이상으로 빈 블록의 비중이 증가하면 가비지 컬렉션을 중단한다.

Ⅳ. 실험 및 성능평가

본 논문의 실험을 위한 환경은 다음과 같다. 실험 PC의 성능으로 CPU는 Intel Core2 Duo 2.40GHz, RAM은 DDR2 3GB이고 Linux Ubuntu 10.04에서 SSD 전용 시뮬레이터인 Microsoft Research SSD extension for DiskSim 4.0 기반으로 하였다.

1개의 블록은 64개의 페이지를 포함하였으며 페이지와 블록에 대한 기본 설정과 읽기, 쓰기, 삭제의 지연시간 설정은 다음의 표 1과 같다.

SSD의 총 용량은 64GB로 설정하였고 워크로드는 3

표 1. 실험 파라미터 설정
Table 1. Set-up Experiment Parameter.

단위	크기
페이지 크기	4 KB
블록 크기	256 KB
읽기 지연시간	25 μs
쓰기 지연시간	200 μs
Erase 지연시간	1500 μs

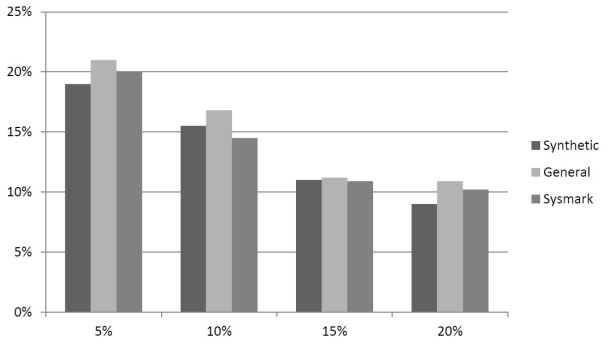


그림 4. 가비지 컬렉션 경계값에 따른 중복률
Fig. 4. De-duplication Ratio by Garbage Collection Boundary Value.

가지를 사용하였다. Disksim 4.0에서 기본적으로 제공하는 Synthetic 워크로드[1]와 office, download, web등의 수행과정에서 추출한 General 워크로드, E-learning, video, 3D modeling 작업의 수행과정에서 추출한 Sysmark 워크로드이다.

성능을 측정하기 위하여 제안한 기법의 경우, 무효데이터만 재활용 할 경우[6], 유효데이터 중복제거 기법인 CAFTL의 경우[7], 중복제거 및 재활용 하지 않는 일반적인 경우[1]를 비교하였다. 일반적인 방법은 DiskSim을 제안한 논문을 이용하여 실험하였다.

실험을 위하여 우선적으로 고려하여야 할 사항은 가비지 컬렉션의 최적의 경계값을 구하는 것이다. Synthetic, General, Sysmark의 세 가지 워크로드에서 무효데이터 재활용 기법만을 적용했을 때 가비지 컬렉션 경계값에 따른 무효데이터의 중복률을 측정했다. 그림 4를 보면 가로축의 경계값이 작을수록 세로축의 중복률이 증가한다. 그 이유는 무효데이터가 삭제되지 않고 많이 존재하여 중복이 많이 발생하기 때문이다. 제안한 기법을 위하여 가장 중복률이 좋은 5%를 경계값으로 사용하였다.

가비지 컬렉션의 경계값인 5%를 기준으로 제안한 기법, 무효데이터 재활용, 유효데이터 중복제거, 일반적인 방법의 네가지 경우를 비교하였다. 중복제거 및 재활용이 적용되지 않았을 때의 쓰기 횟수를 기준으로 정규화한 결과를 그림 5에 나타내었다. 제안한 기법이 General 워크로드에서 약 30%로 가장 쓰기횟수가 많이 감소하였고 그 다음으로 유효데이터 중복제거 기법, 무효데이터 재활용 기법 순으로 쓰기 횟수가 감소하였다. 워크로드 별로는 General 워크로드가 가장 많이 쓰기횟수를

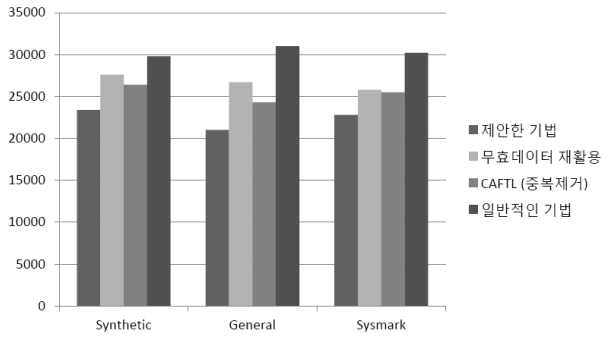


그림 5. 쓰기 횟수 비교
Fig. 5. Comparison of Number of Writes.

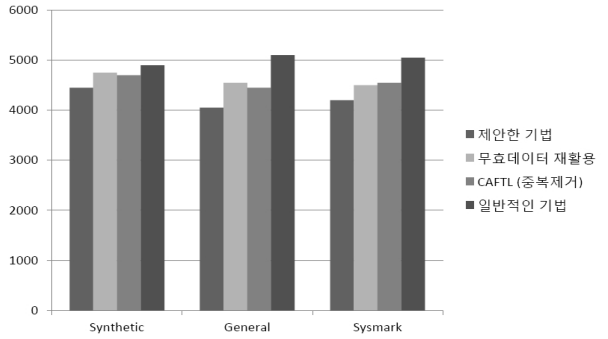


그림 6. 가비지 컬렉션 횟수 비교
Fig. 6. Comparison of Number of Garbage Collections.

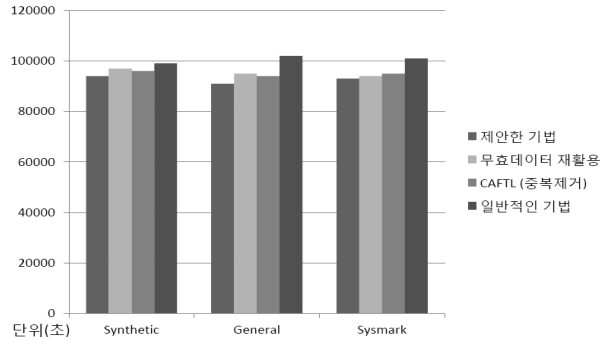


그림 7. 입출력 지연시간 비교
Fig. 7. Comparison of I/O Latency.

감소시켰다.

제안한 기법에서 쓰기 횟수 감소로 인하여 가비지 컬렉션 횟수도 감소하게 된다. 각각의 워크로드 별로 가비지 컬렉션의 횟수를 그림6에 나타내었다. 쓰기 횟수 비교에서와 마찬가지로 제안한 기법이 General 워크로드에서 약 20%로 가장 가비지 컬렉션 횟수가 적었다.

마지막으로 각 기법들의 I/O 지연시간을 비교하였다. 핑거프린트를 생성하고 비교하는 과정에서 지연시간이 생기지만 I/O 지연시간이 긴 가비지 컬렉션 횟수가 감소하여 제안한 기법에서 I/O 성능은 약간 상승하였다. 그림 7에서 보면 General 워크로드에서 약 9% 정도로 가장 많이 감소한 것을 알 수 있다.

V. 결 론

본 논문에서는 SSD의 블록 당 쓰기 가능 횟수 제한 문제와 I/O 성능을 향상시키기 위하여 유효데이터 중복제거 및 무효데이터 재활용 기법을 제안하였다. 제시된 기법은 파일시스템에서 유효데이터 중복제거를 하고 FTL에서 무효데이터 재활용을 함으로써 쓰기 횟수를 감소시켰고 그에 따라서 가비지 컬렉션 횟수도 감소하였다. 그 결과 SSD의 마모도가 향상되었고 I/O 성능이 향상되었음을 확인 할 수 있었다.

참 고 문 헌

[1] N. Agrawal, V. Prabhakan, T. Wobber, J. D. Davis, M. Manasse and R. Panigrahy, “Design Tradeoffs for SSD Performance,” USENIX’08 ATC, 57~70p, 2008.

[2] G. Wu, X. He and B. Eckart, “An Adaptive Write Buffer Management Scheme for Flash-Based SSDs,” ACM Transactions on Storage, Vol.8, No.1, 1~24p, 2012.

[3] J.-Y. Shin, Z.-L. Xia, N.-Y. Xu, R. Gao, X.-F. Cai, S. Maeng, F.-H. Hsu, “FTL Design Exploration in Reconfigurable High-Performance SSD for Server Applications,” ACM ICS’09, 338~349p, 2009.

[4] A. Berman, Y. Birk, “Integrating De-duplication and Write for Increased Performance and Endurance of Solid-State Drives,” IEEE 26th IEEEI, 821~823p, 2010.

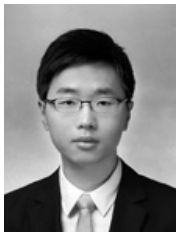
[5] J. Lee, Y. Kim, G. M. Shipman, S. Oral, F. Wang and J. Kim, “A Semi-Preemtive Garbage Collector for Solid State Drives,” IEEE ISPASS, 12~21p, 2011.

[6] J. Kim, S. Lee, P. Mehdi, D. Kim, “Recycling Invalid Data Method for Improving I/O Performance in SSD Storage System,” KIISE KCC 2012, Vol.39, No.1(A), 230~232p, 2012.

[7] F. Chen, T. Luo, X. Zhang, “CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives,” USENIX FAST‘11, 2011.

[8] B. Debnath, S. Sengupta, J. Li, "ChunkStash: Speeding up Inline Storage Deduplication using Flash Memory," USENIX ATC'10, 2010.
[9] D. Meister, A. Brinkmann, "dedupv1: Improving Deduplication Throughput using Solid State Drives(SSD)," IEEE MSST, 1~6p, 2010.
[10] D. Bhagwat, K. Eshghi, D. D. E. Long, M. Lillibridge, "Extreme Binning: Scalable, Parallel Deduplication for Chunk-based File Backup," IEEE MASCOTS'09, 1~9, 2009.
[11] H. E. Michail, A. P. Kakarountas, A. Milidonis, C. E. Goutis, "Efficient Implementation of the Keyed-Hash Message Authentication Code(HMAC) Using the SHA-1 Hash Function," IEEE ICECS, 567~570p, 2004.
[12] Q. He, Z. Li, X. Zhang, "Data Deduplication Techniques," IEEE FITME, 430~433p, 2010.
[13] C.-H. Wu, H.-S. Wu, "A Data De-duplication Access Framework for Solid State Drives," ACM SAC'11, 600~604p, 2011.
[14] G. Wu, X. He, "△FTL: Improving SSD Lifetime via Exploiting Content Locality," ACM EuroSys'12, 253~265p, 2012.
[15] O. Kwon, K. Koh, "Swqp Space Management Technique for Portable Consumer Electronics with NAND Flash Memory," IEEE Transactions on Consumer Electronics, Vol.56, No.3, 1524~1531p, 2010.
[16] J.-S. Song, J.-M. Huh, Y.-S. Yang, D.-H. Kim, "SSD-based RAID-6 System Architecture for Reliability and Performance Enhancement," IEEK, Vol.47, CI, No.6, 589~598p, 2010.
[17] Y.-S. Yang, D.-H. Kim, "Data allocation and Replacement Method based on The Access Frequency for Improving The Performance of SSD," IEEK, Vol.48, CI, No.5, 528~536p, 2011.

저 자 소 개



김 주 경(학생회원)
2009년 인하대학교 전자공학과 학사 졸업.
2012년~현재 인하대학교 전자공학과 석사과정
<주관심분야 : 임베디드 시스템, 스토리지 시스템>

이 승 규(학생회원)
2011년 대진대학교 통신공학 학사 졸업.
2011년~현재 인하대학교 전자공학과 석사과정
<주관심분야 : 임베디드 시스템, 스토리지 시스템>

김 덕 환(정회원)-교신저자
2003년 한국과학기술원 컴퓨터 공학 박사
2006년~현재 인하대학교 전자공학부 교수
<주관심분야 : 시각정보처리, 스토리지 시스템, 임베디드 시스템>