Flash-SSD 데이터 중복 제거를 위한 사용자 파일 시스템 설계

명재희* · 권오영*

*한국기술교육대학교

Design Deduplication User File System for Flash-SSD

Jae-hui Myeong* · Oh-young Kwon**

*Korea University of Technology and Education

E-mail : audwogml@koreatech.ac.kr

요 약

급격한 데이터의 증가로 인해 효율적으로 데이터를 관리하기 위한 다양한 연구가 진행되고 있다. 2025년 데이터의 총량은 163 ZB 이상으로 증가하고, 그 중 1/4 이상의 데이터는 실시간 데이터가 될 것이라 전망한다. 대용량의 저장장치가 HDD에서 SSD로 바뀌고 있는 추세로, SSD에서 데이터를 효과적으로 관리하기 위한 별도의 방안이 필요하다. 본 논문은 현재까지의 Flash-SSD 관련 시스템 구조 및 데이터 관리 방법 중 중복 제거 관리 방법에 관련한 연구들을 탐색한다. 그리고 중복 제거 기법을 적용한 어플리케이션 레벨의 사용자 파일 시스템을 제안하여, 저장 장치의 용량 확보, 성능 저하 및 불필요한 트래픽 최소화 등의 효과를 가져 올 수 있음을 기대한다.

ABSTRACT

Due to the rapid increase in data, various studies are being conducted to efficiently manage the data. In 2025, the total amount of data will increase to more than 163 ZB, and more than a quarter of the data will be a real-time data. As mass storage devices is changed from HDD to SSD, SSD needs own way to manage their data effectively. In this paper, we study the SSD system structure and deduplication management methods of data management related to Flash-SSD. We also propose an application level user file system using deduplication. It is anticipated that it saves storage capacity and minimize reducing performance by unnecessary traffic.

키워드

파일 시스템, SSD, 데이터 중복 관리, 플래시 파일 시스템

Ⅰ. 서 론

데이터의 양이 폭발적으로 증가하고 있다. 최근 발표된 IDC의 자료에 의하면 2025년까지 전 세계 데이터 총량이 163 ZB 까지 늘어날 것이라고 전망했다[1]. 이에 따라 개인 사용자, 정부, 기업이 어떤 데이터를 생성할 것이며, 어떻게 활용하고 관리할 것인지에 대한 중요성이 높아지고 있다. 빅 데이터는 단순히 데이터의 양이 많은 것을 의미하는 것이 아니라, 많은 양의 데이터 속에서 신뢰할 수 있는 핵심 데이터를 찾아내고 효율적으로 관리하는 것이 중요하다.

데이터를 관리하는 다양한 방법 중 하나로 데이터 중복 관리 방법이 연구되고 있다. 데이터 중복을 최소화함으로써, 중복 데이터를 저장 혹은 호출하는 불필요한 트래픽을 최소화할 수 있고, 저장 장치의 효율적인 용량 확보를 통한 비용 절감을 할 수 있다. 통상적으로 데이터 중복 제거를 통해 40% 이상의 용량 절감 효과를 얻을 수 있으며, 데이터에 따라 저장 장치의 소요 비용을 최대 90% 까지 절감할 수 있다. 또한 중복 제거를 통

한 일반 데이터 백업 비용 절감을 넘어 가상화 이미지 백업 비용 절감에도 큰 효과를 가져올 수 있어, 개인 사용자 뿐만 아니라 클라우드 서비스 제공자에게도 비용 절감 및 효율성 향상에 도움을 줄 수 있다[2].

데이터의 양이 많아지면서, 이를 수용할 수 있는 용량 및 성능을 갖춘 저장 장치가 빠르게 발전하고 있다. 특히 SSD는 저전력, 저소음, 저발열 등의 특성을 가지고 있는 차세대 저장장치로 평가되고 있다. 개인 PC에도 SSD+HDD 형태의 하이브리드 스토리지가 사용되는 경우가 많아졌으며, 많은 데이터 센터도 하이브리드 스토리지 시스템으로 핫 데이터와 콜드 데이터를 별도 관리하거나, 데이터에 대한 기본 정보와 실 데이터를 별도 관리하는 방식으로 사용하고 있다. 최근에는 하이브리드 스토리지 시스템에서 All-flash 스토리지 시스템으로 확장되고 있는 추세이다[3]. SSD는 Nand Flash 혹은 DRAM 기반이며, 디스크를 가지고 있는 HDD와는 다른 동작 방식을 가지고 있기 때문에 SSD 전용의 데이터 중복 관리 방법을 적용해야 한다.

본 논문에서는 Flash-SSD의 기본적인 특성을 기반으로 데이터 중복 관리 방법에 대한 관련 연구들을 살펴보고, 파일 시스템 레벨에서 데이터의 중복을 관리하는 방법에 대한 설계를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 데이터 중복 제거 관점에서 Flash-SSD의 각 동작 수준에서의 데이터 관리 방안 관련 연구를 살펴본다. 3장에서는 데이터 중복 관리를 적용한 어플리케이션 레벨의 플래시 파일 시스템 설계 내용을 기술하고, 4장에서 추후 연구 방향 및 결론을 제시한다.

II. Flash-SSD 동작 수준에 따른 데이터 중복 관리 방안

1. 중복 제거 방식에 따른 분류

중복 제거를 하는 시점에 따라 inline, post-process, source 중복 제거 방식으로 구분할 수 있다. inline 중복 제거 방식은 데이터 수신 직후 중복 제거가 이루어지며 별도의 임시 스토리지가 필요하지 않지만, 데이터 송수신시 오버헤드가 있다. 반면 post-process 중복 제거 방식은 데이터 저장 후, 추후 시점에서 데이터의 중복 제거를 진행한다. 데이터 송수신 시점에서는 오버헤드가 발생하지 않아 빠른 송수신이 가능하다. 하지만, 매 시점마다 중복 검사를 하기 때문에 중복이 많지 않은 시스템에서는 오버헤드로 작용할 수 있다는 단점이 있다. 마지막으로 source 중복 제거 방식은 데이터 송신 시점에서 중복 데이터에 대해 실제 데이터를 전송하지 않고 포인터만 전송하는 방식이다. 포인터만 전송하기 때문에 트래픽 감소에 효과적이다.

일반적으로 hash를 생성해 중복 검사를 하는 방식이 대부분이며, 파일 수준에서 중복 제거를 하는 방식과 일정 블록 단위로 나누어 각각의 해시를 생성해 중복 제거를 수행하는 방식이 있다. 블록 단위로 나누어 중복 제거를 하는 방식은 다시 VLC(Virtual Length Chunking), FLC(Fixed Length Chunking) 으로 나눌 수 있다[4].

2. SSD 동작 시스템 분류

하드디스크는 섹터 기반의 저장장치로 디스크가 마모되지 않고 계속해서 사용할 수 있다는 특징이 있다. 하지만, SSD 등 플래시 메모리 기반의 저장장치는 전기적인 특성을 가지며, 각 셀은 P/E(Program/Erase) Cycle이 정해져 있다. 대부분의 파일 시스템과 어플리케이션은 HDD 기반으로 설계되어 사용하고 있으며, FTL(Flash Translation Layer)이 해당 기능을 가능하게 한다. FTL은 논리 주소를 물리 주소로 바꾸어 주는 역할인 논리 블록 맵핑(Logical Block Mapping)을 수행한다. 현재의 LBA(Logical Block Address) Array는 HDD에서만 적합한 요소이기 때문이다. 그 외에 Garbage Collection, Wear Leveling 등의 SSD의 성능 향상을 위한 주요 알고리즘이 적용된다. 그리하여 FTL에 의해 SSD가 HDD와 동일한 인터페이스를 이용할 수 있고, 개인 사용자들은 SSD나 HDD의 특성에 의존하지 않고 비슷한 사용 방식을 취한다.

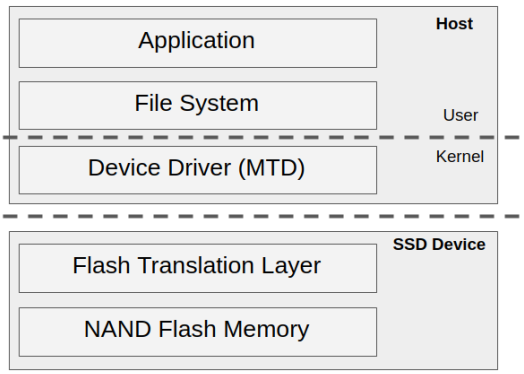

그림 1. SSD 시스템 레이어

SSD가 동작하는 시스템 레이어는 그림 1과 같이 FTL 레벨이 추가되어 있다. 구조의 최하단은 SSD를 구성하는 메모리와 메모리의 용도를 결정하는 Hardware 레벨이다. 하드웨어가 결정되면 FTL 레벨에서 SSD의 성능 최적화를 위한 알고리즘을 적용하고, 파일 시스템 및 어플리케이션 레벨에서 사용자가 작업을 수행하는 구조로 되어있다. 다음 장에서는 각 수준에서 연구되고 있는 데이터 중복 제거 방법에 대해 살펴본다.

3. 레벨 별 중복 제거

HW level에서는 데이터 중복 제거를 위해 실 데이터와 중복 검사를 위한 해시 테이블을 별도의 저장 장치에서 관리하는 방법을 취한다. 이를 통해 해시값 검색에서 발생하는 오버헤드를 최소화하여 성능을 최대화한다.

관련하여 PRAM(Phase-change Random Access Memory)과 NAND Flash Memory를 결합한 SSD에서 자주 참조되는 데이터에 대한 해시값을 PRAM의 해시 테이블에 저장해 해시값 검색에 드는 오버헤드를 최소화한 데이터 중복 제거 방법이 연구되었다[5]. 이와 같이 입출력 속도가 빠르고 덮어쓰기가 가능한 PRAM 등의 메타 데이터 처리가 유리한 메모리를 캐시로 사용하는 것과 같이, 해시를 처리하는 메모리를 별도로 두어 기존의 중복 제거 방식을 보완하는 방법이 있다.

FTL에서 중복을 제거하는 방식은 online 방식과 offline 방식으로 구분된다. online 방식은 런타임에 중복 제거 여부를 검사하는 방식이며, 실제 쓰기를 줄여 용량 확보에 효율적이다. 하지만 갑작스러운 시스템 정지가 발생할 경우 데이터의 복구가 어렵다는 한계가 있다. offline 방식은 저장 장치의 유휴 시간에 중복을 제거한다. 이 경우, 중복 데이터에 대한 응답 시간을 줄이지는 않지만, 가비지 컬렉션 비용을 줄여 전체적인 응답 시간을 감소할 수 있다. 하지만 중복 제거 시간이 상대적으로 길기 때문에, 해당 처리가 끝나기 이전에 가비지 컬렉션이 발생하는 경우 불필요한 페이지 복사가 일어날 수 있다. 이런 단점을 보완하기 위해 런타임 시에 경량 해시키인 CRC32 값을 생성해 중복 가능성이 있는 페이지를 구분해 두고, 해당 페이지만 직접 비교하는 최소한의 오프라인 중복제거 프로세스를 수행해 성능을 개선한 사례가 있다[6]. 또한, online 중복제거 방식인 CAFTL은 중복이 발생될 때 중복된 데이터의 논리 주소를 중간의 가상 페이지 테이블을 통해 하나의 물리 주소를 가리키도록 하고, 중복된 데이터는 삭제하는 방식을 사용하였다[7].

파일 시스템 레벨에서 중복 제거를 적용한 파일 시스템은 LessFS(Data Duplication for Less), ZFS(The Z File System), SDFS(A User Space Deduplication File System) 등이 있다. LessFS는 inline 중복 제거 방식을 취하며, 고정된 블록 파일 수준의 중복제거를 수행한다. LZ4 압축 기능을 지원하고, 중복 제거 시간이 짧고 자원 관리가 효율적이라는 장점이 있다. ZFS도 LessFS와 같이 inline 중복 제거 방식을 취한다. 다양한 환경에서 COW, Snapshot 등을 지원하는 것이 장점이지만, 상대적으로 중복 제거 시간이 길고 효율성이 좋지 않다. SDFS는 오픈 소스 기반의 프로젝트이며, TigerHash 기반의 inline 중복 제거 방식과 post-process(batch) 중복 제거 방식 모두 가능하고 snapshot 기능을 지원한다. Windows와 Linux 플랫폼을 모두 지원하고, 실제 스토리지 사용률을 최대 90%까지 감소시킬 수 있다[8].

일반 개인 사용자가 사용할 수 있는 중복 제거 방법으로 linux에서는 fdupes, rsync 패키지나, 오픈소스 혹은 어플리케이션 형태로 다양한 중복 제거 프로그램들이 배포되고 있다. 각 파일에 대해 hash을 생성해 중복 파일에 대한 관리를 한다. 프로그램을 실행하는 시점에서 단발적으로 지정된 사용자 영역의 스토리지만 중복 관리를 하는 형식이고 사용자의 선택에 따라 이루어지기 때문에 효율적인 방식은 아니다. [9]에서는 프로그램을 데몬화해 파일 이벤트 발생 시 hash list 검색을 통해 중복 및 버전 관리를 어플리케이션 레벨에서 구현하였다. 이 논문은 파일 이벤트 발생 이후에 중복 제거 관리가 이루어지기 때문에 성능 면에서는 효과를 얻지 못했다.

III. 데이터 중복 관리를 적용한 어플리케이션 수준 파일 시스템 설계

응용 수준에서 파일 중복을 확인하기 위해, 각 파일에 해시 함수를 적용해 파일의 중복을 확인한다[9]. 이 경우 파일 이벤트가 발생하는 시점에서 해시를 생성하거나 검색하는 프로세스가 추가된다. 하드디스크를 사용하는 경우, 전체 저장 장치의 용량에 대한 효율성을 가지게 관리를 할 수 있다. 하지만, 불필요한 메타 데이터 접근 및 쓰기 작업이 수반되기 때문에 성능 면에서는 오히려 오버헤드를 가져온다.

SSD 등의 플래시 메모리 기반의 저장장치에 해당 시스템을 사용하는 경우, 대용량 파일에 대한 불필요한 쓰기 작업이 수명 문제와 연관이 될 수 있다. 셀 당 P/E 횟수는 평균 1000(TLC), 1만(MLC), 10만(SLC) 정도를 웃돈다. 웨어 레벨링 기능으로 대체용 셀을 5% 정도 남겨두고 있기도 하고, 이론상 SSD 전체의 수명은 256GB 기준으로 수백TB 기록이 가능하다. 현재 수명에 대한 우려는 거의 없는 편이지만, 'Data Age 2025'에 따르면 임베디드 시스템과 사물인터넷(IoT)에 의해 2025년에는 네트워크 기기 간의 상호 정보 교환 횟수가 하루 평균 4800번, 즉 18초에 한 번 씩 정보가 교환이 될 것이라고 한다. 또한 생성 데이터의 1/4 이상이 실시간 데이터로 구성된다고 예측하고 있다[1]. 비록 현재 SSD가 수용 가능한 수명을 가지고 있지만, 실시간 데이터가 급속도로 늘어나고 있는 실정이기 때문에 데이터 및 스토리지에 대한 효율적 관리의 중요성이 커질 것이다.

본 논문에서는 오픈 소스인 FUSE(Filesystem in USEr space)를 사용해 중복 제거 관리를 위한 사용자 수준의 파일 시스템 설계 방식을 제안한다. FUSE는 기존 커널 및 파일 시스템을 수정하지 않기 때문에 이식성이 높아 VFS(Virtual File System)을 작성하는데 용이하고, MAC OS,

Windows, Solaris 등의 OS에서 사용 가능하다. 중복 제거 관련 프로세스만 추가하여 성능 저하 및 불필요한 메타 데이터 접근을 최소화하고, 중복 데이터의 읽기/쓰기를 제한하여 용량 확보를 할 수 있다. 본 논문은 [9]을 기반으로 특정 파일 이벤트가 발생하는 시점에서의 성능 저하 및 수명 저하를 최소화하는 파일 시스템을 제안한다.

본 시스템에서 구현하는 중복 제거 방식은 inline 중복 제거 방법으로 파일이 생성되는 시점에서 md5 기반의 파일 단위 해시를 생성해 해시 리스트에서 비교하는 방식을 취한다. md5는 128bit 해시 함수로, 주로 파일의 무결성 검사에 사용한다. CRC32, SHA-256/512 등의 알고리즘을 사용하기도 하나, inline 방식에서는 hash 생성 시간을 최소화하는 것이 중요하기 때문에 md5를 선택하였다. 배포되어있는 대부분의 파일 중복 관련 유틸리티들도 md5를 사용하는 경우가 많기 때문에 호환이 용이하다.

해시 리스트의 구조는 중복 관리에 대해서만 구현하도록 한다. 이전 연구에서는 데이터 구조에 버전 정보를 같이 구현하여 관리하는 방식을 취했지만, 특정 시점에서 스냅샷 기능을 제공하는 것으로 대체할 것이다. 해시 리스트가 무한정 증가하는 것을 방지하기 위함과, 최근 활발하게 연구되고 있는 대용량 분산 파일 시스템에서는 버전 정보보다는 일정 시점의 스냅샷을 통해 데이터를 백업해 관리하는 방식을 취하고 있기 때문이다.

Ⅳ. 결 론

플래시 메모리 기반의 저장장치에 중복 방지 프로세스가 추가된 어플리케이션 수준의 파일 시스템을 사용하는 경우 다음의 이점들을 가져온다. 첫째, 중복 제거를 통해 전체 저장장치의 용량을 효율적으로 관리할 수 있다. 플래시 메모리 기반의 저장장치들(SSD, UFS 등)은 아직까지는 고비용이고, 용량이 크지 않다는 특징이 있다. 둘째, 1차적으로 중복 검사를 한 뒤에 메타 데이터에 접근하도록 지정함으로써, 메타데이터에 접근하는 불필요한 트래픽을 줄일 수 있다. SPEC의 연구 결과에 따르면, 파일 시스템에 발생하는 트래픽의 60% 이상이 메타 데이터에 대한 요청에 의해 발생한다고 한다[10]. 셋째, 플래시 메모리 기반의 저장장치는 한정된 P/E 횟수가 있다. 데이터의 양 및 파일의 크기가 급속하게 증가하고, 실시간 데이터가 많아지는 것에 대비해 불필요한 IO를 줄여 수명 저하를 최소화한다.

추후 본 논문을 바탕으로 파일 시스템을 구현하여 다른 중복 제거 파일 시스템과의 성능 비교를 통해 연구의 타당성을 검증하고, 다양한 데이터 입출력 환경에서 성능을 평가해 볼 예정이다.

Acknowledgement

이 논문은 2015년도 정부(미래창조과학부)의 재원으로 연구개발특구진흥재단의 과학벨트기능지구지원사업 [공동연구법인 설립 및 운영 지원] 2차 사업의 지원을 받아 수행하였음.

참고문헌

[1] David Reinsel, John Gantz, John Rydning, "Data Age 2025", IDC, 2017.4

[2] IT on the Wheel IT Common Data Deduplication page, [Online]. Available: http://bigstory.tistory.com/117

[3] SKTelecom, "SKT SDDC white paper", 2016.8

[4] Venish, A., & Sankar, K. S. “Study of Chunking Algorithm in Data Deduplication.” In Proceedings of the International Conference on Soft Computing Systems, 2016, 13-20

[5] 이승규, 김주경, 김덕환, "SSD 스토리지 시스템에서 PRAM 기반 계층구조 해시테이블을 이용한 데이터 중복제거 기법", 대한전자공학회 학술대회, 2012.11, 836-838

[6] 박은수, 신동군, "경량 해시 키를 이용한 SSD에서 오프라인 중복제거 기법", 한국정보과학회 학술발표논문집, 2014.12, 1495-1497

[7] F.Chen et al, "CAFTL:A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives", FAST 2011

[8] 정성욱, 최훈, "오픈 소스 기반 데이터 분산 중복제거 파일 시스템의 성능 분석", 정보과학회 컴퓨팅의 실제 논문지, 20(12), 623-631.

[9] 명재희, 권오영, "응용 수준의 파일 중복 및 버전 관리", 한국항행학회 종합학술대회, 2017.10

[10] SPEC, [Online]. Available: www.storageperformance.org/