KIPS Tr. Comp. and Comm. Sys.
Vol.4, No.9 pp.307~314 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2015.4.9.307

Selective Recovery of the SSD TRIM Command in Digital Forensics

Hwang Hyun Ho† · Park Dong Joo††

ABSTRACT

Recently, market trends of auxiliary storage device HDD and SSD are interchangeable. In the future, the SSD is expected to be used more popular than HDD as an auxiliary storage device. The TRIM command technique has been proposed and used effectively due to the development of the SSD. The TRIM command techniques can be used to solve the problem of Freezing SSD that operating system cooperates with the SSD. The TRIM command techniques are performed in the idle time of the internal SSD that are actually deleted when a user deletes the data. However, in the point of view of computer forensics, the digital crime is increasing year by year due to lack of data recovery. Thus, this rate of arrest is insufficient. In this paper, I propose a solution that selectively manages data to delete based on advantage of the stability and the write speed of the TRIM command. Through experiments, It is verified by measuring the performance of the traditional method and selected method.

Keywords : Digital Forensics, Freezing Solution, Solid State Drive, TRIM Command

디지털 포렌식 관점에서 SSD TRIM 명령의 선별적 복구

황 현 호† · 박 동 주††

요 약

최근 보조기억장치 스토리지 시장 추세는 HDD와 SSD가 혼용되어 사용되고 있다. 미래에는 HDD보다 SSD가 보조기억장치 역할로 더욱 많이 사용될 것으로 예상되고 있다. SSD 기술이 발달하면서 이를 효율적으로 사용하기 위해 TRIM 명령기법이 나오게 되었다. TRIM 명령기법은 과거 SSD의 문제점인 Freezing 현상을 해결하기 위해 나온 명령으로 운영체제와 SSD가 협동하여 작동한다. TRIM 명령기법은 사용자가 데이터를 삭제하였을 때 SSD 내부에서도 실제로 삭제하는 기법으로 유휴시간에 수행한다. 하지만, 디지털 포렌식 관점에서 본다면 디지털 범죄는 매년 급증하지만 데이터 미복구로 인한 검거율은 미흡하다. 본 논문에서는 기존 TRIM 명령의 장점인 안정성(Freezing Solution)과 쓰기 속도를 최대한 지원하며 선별적으로 데이터를 관리하여 삭제하는 기법을 제안한다. 실험을 통해 기존 기법과 선별된 기법의 성능을 측정하여 검증한다.

키워드 : 디지털 포렌식, 안정성, SSD, TRIM 명령

1. 서 론

최근 디지털 제품의 사용량이 급속히 증가하여 범죄와 관련된 정보들도 디지털화되고 있다. 기존에 주로 사용되었던 보조기억장치인 HDD(Hard Disk Drive)에서 플래시메모리를 사용하는 SSD(Solid State Drive) 보조기억장치로 급격히 변화되고 있다. SSD는 크기가 작고 가벼우며 충격에 강하고 소비전력이 낮다. 또한 HDD에 비해 빠른 데이터 접근 성능을 가진다. HDD는 물리적 한계 때문에 발전의 폭이 크지 않으며 한계에 도달했다고 평가된다. 반면 SSD는 반도체를 이용하여 데이터를 저장하는 것이 특징이며 물리적인 한계가 거의 없기 때문에 최근 지속적으로 사용되고 발전되고 있다.

디지털 포렌식(Digital Forensic)[1]은 범죄수사에서 적용되고 있는 과학적 증거 수집 및 분석기법의 일종이다. 각종 디지털 데이터, 이메일 접속기록, 통화기록 등의 정보를 수집 및 분석하여 핏자국, 지문, DNA 등 범행과 관련된 증거를 확보하는 수사기법을 말한다. 따라서 실생활에서 많이 사용되는 디지털제품이 급증함에 따라 디지털 범죄뿐만 아니라 일상 범죄에서도 중요 증거를 디지털 제품에 활용하여 보관할 수 있는 가능성이 높아진다. 그 결과 SSD에 기록된 데이터는 디지털 포렌식 수사 대상이 될 가능성이 높다.

SSD가 비약적으로 발전하면서 효율적으로 사용하기 위해 TRIM 명령[2]기법을 사용하고 있다. TRIM 명령기법은 과거 SSD의 문제점인 Freezing 현상을 해결하기 위해 나온

† 준 회 원 : 숭실대학교 컴퓨터학과 석사
†† 정 회 원 : 숭실대학교 컴퓨터학부 부교수
Manuscript Received : July 22, 2015
First Revision : August 31, 2015
Accepted : August 31, 2015
* Corresponding Author : Park Dong Joo(djpark@ssu.ac.kr)

명령으로, 운영체제와 SSD가 협동하여 작동한다. TRIM 명령기법은 사용자가 데이터를 삭제하였을 때 SSD 내부에서도 실제로 삭제하는 기법으로, 기존 방법은 TRIM 명령 즉시 데이터를 삭제하는 방식이었지만, 효율적으로 사용하기 위해 즉시 수행이 아닌 유휴시간(Idle Time)에 수행한다.

TRIM 명령기법은 TRIM 명령이 적용되지 않은 SSD보다 논리적으로 무효화된 파일에 페이지를 정확하게 소거연산을 할 수 있으며, 쓰기 전 삭제 문제점을 회피할 수 있는 장점이 있다. 하지만 디지털 포렌식 관점에서 본다면 TRIM 명령기법은 많은 문제를 초래할 수 있다. 우선, 기본적으로 TRIM 명령이 기본(Windows7 이후 기본동작)으로 활성화 되어있다면 복구가 불가능하다[3].

복구가 불가능하다면 디지털 범죄가 급증하는 현 시대에 많은 문제가 발생하게 된다. 디지털 증거는 정황증거만으론 증거 채택이 어렵고 확실한 물증이 있어야 채택 가능성이 높다. 간단히 예를 들자면, 공공기업이나 업체가 불법 행위의 증거를 기록해놓은 자료를 자신들이 수사 대상이라는 것을 알아차리고 경찰이 들이닥치기 30분 전 단순 삭제(Shift Delete, 또는 휴지통 비우기 명령)를 하여도 복구가 불가능하다[4]. 현재 디지털 범죄는 급증하고 있는 추세이나 검거율은 현저히 떨어지고 있는 상황이다. 데이터 미복구로 인한 검거율 감소도 한몫하고 있다.

Table 1은 공공데이터를 기반으로 하여 서울 중앙지방검찰청에서 공개한 컴퓨터 범죄의 죄명에 따른 기소 현황이다. 전자문서 관련 죄와 음란물 유포에 관한 죄가 가장 많으며, 이를 대상으로 하여 선별적으로 데이터를 관리한다면 데이터 미복구로 인한 검거율 향상에 상당한 도움을 줄 수 있다. 본 논문에서는 디지털 포렌식 관점에서 SSD TRIM 명령의 선별적 복구 기법을 제안한다. 공공데이터를 기반으로 하여 선별적으로 데이터를 관리하고 제안한 기법으로 데이터를 선별한 경우와 특별한 상황에 대처하여 사용하는 경우를 나누어 성능을 평가하고 검증한다.

본 논문의 구성은 다음과 같다. 2절에서는 TRIM 명령에 대한 배경 지식과 기존 TRIM 명령에 대한 장점에 대해 기술한다. 3절에서는 기존 TRIM 명령기법과 제안한 선별적 TRIM 명령기법에 대해 기술하고, 4절에서는 실험과 성능을 평가한다. 마지막으로, 5절에서는 결론 및 향후 계획에 대해 기술한다.

2. 배경 지식 및 관련 연구

2.1 낸드 플래시메모리(NAND Flash Memory)

낸드 플래시메모리는 블록의 크기에 따라 소블록(Small Block) 구조와 대블록(Large Block) 구조로 분류하고, 소블록과 대블록은 페이지 수와 크기가 각각 다른 특성을 가지고 있다. 플래시메모리는 블록(Block), 페이지(Page), 섹터(Sector), 여유영역(Spare Area)으로 구성된다. 한 Flash Array에는 여러 개의 블록이 있고, 한 블록은 다수의 페이지로 구성된다. 한 페이지는 데이터가 기록되는 섹터와 페이지의 메타정보가 저장되는 여유영역으로 구성되어있다. 여유영역은 LBN(Logical Block Number), LSN(Logical Sector Number), ECN(Erase Count Number), ECC(Error Correction Code), 블록 및 페이지 상태 등 여러 가지 유용한 정보를 저장할 공간으로 사용된다[5]. 플래시메모리는 하드디스크와 동일한 연산인 읽기, 쓰기연산 이외에 소거연산이 필요하다. 소거연산은 덮어쓰기(Overwrite)가 불가능한 플래시메모리에서 이전에 기록한 데이터를 지우기 위한 것으로, 같은 위치에 데이터를 쓰기 위해선 사전 작업인 쓰기 전 소거(Erase-Before-Write)연산을 수행해야 한다. 하지만 기록된 페이지를 다시 사용하기 위해 매번 소거연산[6]을 수행하게 된다면 플래시메모리에서는 소거연산이 읽기연산과 쓰기연산에 비해 가장 느리고 읽기, 쓰기연산은 페이지 단위이지만 소거연산은 블록 단위이기 때문에 Freezing 현상이 일어나게 된다. Freezing 현상은 데이터를 기록하는 작업과 쓰기작업 속도가 10배 이상 차이를 보이는 플래시메모리 특성 문제로, 사용 중인 장치가 정지(Freezing)하는 현상이다.

1) FTL(Flash Translation Layer)

플래시메모리는 제자리 덮어쓰기(In-Place Update)가 되지 않아 소거연산을 수행해야 하는 물리적인 제약을 가지고 있다. 이러한 제약을 해결하기 위해 플래시메모리와 파일시스템 중간 사이에 FTL계층을 사용한다. FTL은 파일시스템에서 호출되는 논리주소(Logical Address)를 물리주소(Physical Address)로 사상(Mapping)하는 작업을 한다. 사상정보는 사상 테이블에 저장하고 관리하며, 덮어쓰기 연산

Table 1. Computer Crime

Division \ Crimes		Criminal Law					
		total	digital records secret invasion	public electronic history damage	electronic document	Computational disruption	Computer fraud
Indictment	Total	2,570	2	12	1,719	20	806
	Trial	1,013	-	12	573	3	423
	Summary	1,557	2	-	1,146	17	383

Division \ Crimes		Information Network Law					
		total	leak private information	Defamation	Spreading pornography	Information Networks infringement	Etc
Indictment	Total	6,912	51	1,160	2,903	510	2,183
	Trial	795	7	118	120	162	347
	Summary	6,117	44	1,042	2,783	348	1,836

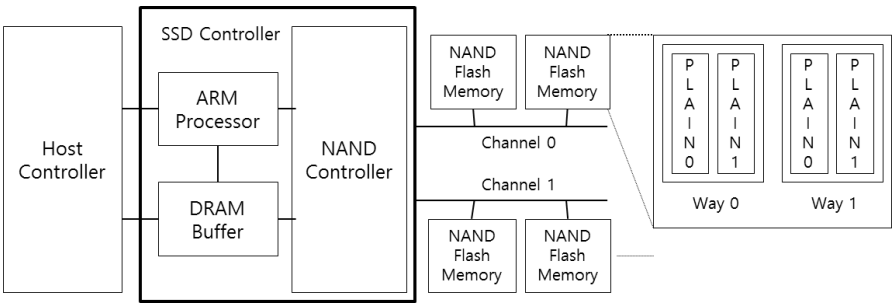


Fig. 1. Internal Architecture of SSD

이 일어날 때마다 사상 테이블의 정보를 갱신한다. 또한 다른 자리 덮어쓰기가 일어나면 새로 덮어쓴 데이터는 유효데이터(Valid Data)로 사상되고, 덮어쓰기 이전 데이터는 무효데이터(Invalid Data)로 사상되어 플래시메모리에 존재하게 된다. FTL은 파일시스템과 플래시메모리 사이에 위치하는 소프트웨어 계층이다. 응용프로그램은 파일시스템을 통해 데이터를 사용할 수 있으며, 파일시스템은 FTL을 통해서 플래시메모리에 쓰기 및 읽기 동작을 요청할 수 있다. FTL은 플래시메모리의 특징으로 인해 여러 저장장치 관리 기법을 요구하고, 이를 효율적으로 구현하기 위해 계층적 구조를 가진다.

2) GarbageCollection & Wear-Leveling

가비지 컬렉션(Garbage Collection, GC)[7]은 플래시메모리에 빈 블록이 없을 경우 가비지 컬렉터가 호출되어 사용하지 않는 무효블록(Invalid Block)을 수집하여 관리한다. 유효블록(Valid Block)의 페이지 중에서 유효데이터가 기록된 페이지를 다른 곳으로 옮긴 후 해당 블록을 지움으로써 무효데이터가 차지하던 공간을 소거연산하는 것이다. 가비지 컬렉션 작업을 자주하면 할수록 블록 소거연산이 반복되므로 파일시스템에서 수행할 연산이 지연된다. 플래시메모리는 블록당 소거 횟수가 제한되어있으며, 만약 특정 블록에 연산이 자주 수행되면 특정 블록은 배드블록(Bad Block)이 되어 사용할 수 없다. 배드블록이 되면 플래시메모리의 내구성이 낮아진다는 것을 의미하기 때문에 데이터의 신뢰성이 낮아진다. 웨어 레벨링(Wear Leveling)[8]은 가비지 컬렉션 영역에서 블록당 소거 횟수(Erase Count Number, ECN)가 가장 적은 블록을 선정하여 수행한다. 즉, 플래시메모리의 소거 횟수를 골고루 분포하여 플래시메모리의 신뢰성과 내구성을 향상시킨다.

2.2 SSD의 구조와 특성

Fig. 1은 SSD의 내부구조를 도식화한 것이다. 그림과 같이 다수의 플래시메모리와 그 외 4가지 요소들로 구성되어 있다. SSD는 FTL을 포함한 펌웨어(Firmware)를 수행하기 위한 ARM Processor, Host로부터 들어온 데이터 및 Host로 보낼 데이터를 저장할 DRAM 버퍼, 플래시메모리를 제어하기 위한 낸드 컨트롤러(NAND Controller), Host와의 Interface를 위한 SATA 또는 PCIe Host Controller로 구성된다. Host Controller는 상위 파일시스템에서 보낸 요청들을 SSD로 전달하거나 SSD에서 Host로 데이터를 전달하는 역할을 하며, 낸드 컨트롤러는 SSD 내부 소프트웨어가 보낸 요청의 정보를 받아 플래시메모리에 읽기/쓰기/소거연산을 직접 수행한다. 낸드 컨트롤러에 직접 연결된 버스는 다수의 플래시메모리 칩에 의해 공유되고 있으며 여기에서 각 버스를 채널이라 한다. 하나의 칩은 다시 독립적으로 읽기/쓰기/소거연산이 가능한 Way로 나눈다. Way는 다시 페이지 크기의 레지스터를 갖고 있는 Bank(Plain)로 나누어진다. 만약 하나의 Way가 2개 이상의 Bank를 포함하는 경우, Bank에 포함된 페이지 크기의 레지스터를 이용하여 2개 이상의 Bank에서 동시에 동일한 연산이 가능하다. 하나의 Bank는 수백~수천 개의 블록으로 나누어져 있고, 하나의 블록은 다시 수백 개의 4KB~16KB 크기 페이지로 나누어져 있다.

2.3 TRIM 명령

SSD는 HDD와 비교하여 월등한 성능을 가지고 있다. 덮어쓰기가 불가능하여 소거/쓰기작업을 몇 단계 이상 수행하더라도, SSD의 빠른 성능으로 사실 별 차이 없을 것이라는 생각을 가질 수 있다. 그러나 플래시메모리 특성 중 하나가 바로 상대적으로 소거 속도가 많이 느리다는 것이다. 일반적으로, 비어있지 않은 페이지에 데이터를 기록하는 작업은 비어있는 페이지에 데이터를 쓰기작업하는 것에 비해 무려 10배 이상 속도 차이가 난다. SSD가 아무리 빠르더라도 갑자기 쓰기 속도가 10배 정도로 느려지면 Freezing 증상 등 많은 문제들이 있을 수 있다. OS는 파일을 삭제하면 디스크에서 해당 파일을 실제로 삭제하는 것이 아니고, 파일의 메타데이터를 삭제하여 파일시스템과의 연결만 끊은 후 디스크에는 파일을 그대로 남겨놓는 방식을 사용한다[9]. 즉, SSD는 HDD와 마찬가지로 사용할수록 이러한 실제로 사용되지 않는 쓰레기 데이터가 점점 늘어나 그만큼 공간을 차지하고 있게 되는데, 덮어쓰기가 가능한 HDD에서는 문제가 되지 않았지만 SSD에서는 큰 문제가 된다. SSD를 구입한 직후에는 모든 페이지와 블록이 깨끗하게 비워져 있기 때문에 좋은 쓰기 속도를 보여주지만 이후 사용할수록 소거/쓰기 수행의 증가로 인해 SSD의 전체적인 쓰기 성능이 큰 폭으로 하락한다[10].

이러한 문제를 해결하기 위해 SSD에서는 TRIM 명령이

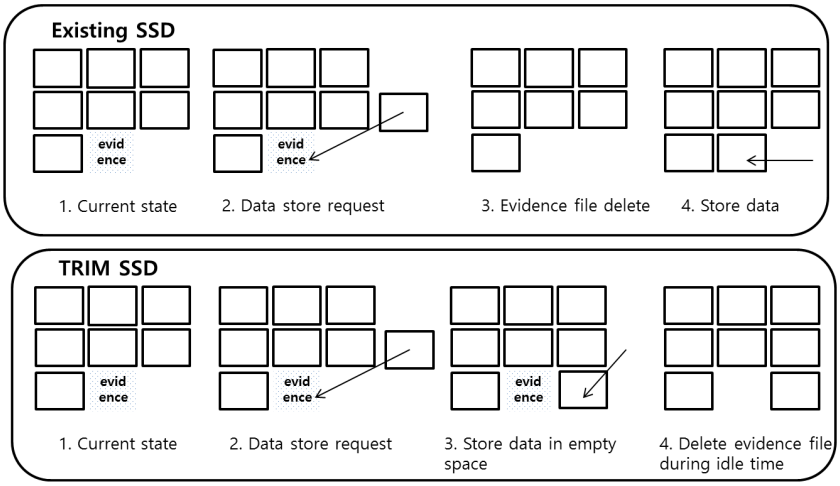


Fig. 2. Existing SSD and TRIM Applied SSD

새롭게 등장하게 되었다. Fig. 2는 TRIM 명령의 간단한 모식도이다[11]. TRIM 명령은 운영체제에서 삭제된 파일들을 파악하여 SSD 상에서도 실제로 해당 파일들을 지우는 기능이다.

간단히 말하면, TRIM 명령은 나중에 쓰기작업 수행 시 소거될 데이터를 미리 적절한 방법으로 소거시켜 윈도우에서 파일을 지우면 SSD 내부에서도 실제로 파일을 지우는 기능이다. SSD의 TRIM 명령이 효율적으로 운영될 수 있도록 운영체제와 SSD가 손을 잡고 협동하여 수행된다. 위 기법을 사용하기 위해 윈도우7부터 등장한 것이 DisableDelete Notify Option이며, 이 Option은 윈도우에서 해당 파일을 삭제하면 SSD에게 해당 파일은 더 이상 사용하지 않으니 유휴시간에 삭제하라고 실시간으로 알려주는 것을 의미한다[12].

3. 선별적 TRIM 명령기법

SSD의 발달과 더불어 TRIM 명령 또한 효율적인 알고리즘을 도입하여 발전되고 있다. TRIM 명령이 나온 초창기에는 다른 문제점들이 많이 발견되어 TRIM 명령 자체를 쓰지 않아야 한다는 주장도 많았다. 하지만 효율적인 알고리즘 구현을 통해 점차 안정되어, 현재는 TRIM 명령을 쓰지 않은 SSD와 TRIM 명령을 사용한 SSD의 성능 차이는 상당히 크다.

현재 S-ATA 규격에서는 더욱 효율적인 TRIM 명령의 지원을 위해 노력하였으며 차세대 규격인 S-ATA 3.1 규격에 Queued Trim Command를 추가하여 S-ATA 규격 자체에 최적화를 준비하고 있다. 그러나 디지털 포렌식을 다루는 주요한 사이트[13]에서 TRIM 명령에 대한 문제점을 꾸준히 제기하였다. 보안이 중요한 기관이나 사용자의 개인정보 등을 가지고 있는 기업과 같이 완전 삭제를 꼭 해야 하는 경우도 있지만, 서론에서의 예와 같이 완전 삭제를 하지 말아야 하는 경우에는 데이터 미복구로 인해 심각한 문제를 초래할 수 있다.

3.1 아이디어 및 전제조건

선별적으로 데이터를 관리하기 위해서 우선, TRIM 명령의 동작 시점을 파악해야 한다. TRIM 명령은 사용자가 데이터를 삭제하였을 때 운영체제에서 명령을 내려준다. 운영체제에서 명령을 내린 후 이후의 동작은 관여하지 않는다. 이 시점에서 선별적으로 데이터를 관리하여 TRIM LIST를 만든 후 보관하여 데이터의 완전 삭제를 방지한다. 선별된 LIST들은 TRIM이 적용되지 않은 기존 방법과 같이 GarbageCollection & Wear-Leveling 이하 영역을 수행하지 않고 실제 데이터는 남아있지만 사용자에게는 보이지 않는다. 선별된 TRIM LIST를 계속 가지고 있어 실제 데이터들이 계속 남아있다면 사용자가 사용하는 실제 사용량은 급격히 줄어들 것이다.

SSD는 구조상 블록들을 관리하는 Bank(Plain) 단위로 구성되어있다. 그래서 Bank 단위로 블록들이 가득 찬 상황에 직면했을 때 빈 공간 부족 이벤트를 발생시켜 해당 Bank를 TRIM 명령이 적용된 기법처럼 GarbageCollection &Wear-Leveling 이하 영역으로 보내준다. GarbageCollection & Wear-Leveling 이하 영역으로 보내주면 작업한 Bank에 해당되는 블록들은 기존 TRIM 명령이 동작하는 방식과 같이 사용 가능하다.

3.2 TRIM 명령의 소거연산

Table 2는 기존 SSD와 TRIM 명령이 적용된 SSD의 소거연산을 비교한 표이다. 기존 SSD는 삭제를 하여도 실제 데이터는 남아있고 특정 시점에 GarbageCollection & Wear-Leveling 이하 영역을 수행하여 실제 데이터를 삭제 시킨다. GarbageCollection & Wear-Leveling 이하 영역은 GarbageCollection, Wear-Leveling, ECC Check, Bad Block handling으로 구성되며, ECC Check 영역은 읽히거나 전송되고 있는 데이터에 오류가 생겼는지를 검사하고, 필요하면 전송 중에 정정할 수 있는 영역이다. Bad Block handling 영역은 Bad Block을 관리하며, 정상적인 Block에 영향을 주

지 못하게 제어하는 영역이다. GarbageCollection & Wear-Leveling 이하 영역은 전반적으로, SSD의 수명과 성능 향상에 도움을 주는 역할을 한다. 데이터 삭제 후 동일한 블록에 쓰기작업을 수행한다면 기존 SSD는 데이터 쓰기 바로 전 해당 블록에 소거연산을 수행한다. 이때 Freezing현상이 일어날 수 있으며 이 문제가 초창기 SSD의 가장 큰 문제점으로 꼽혔다. 쓰기작업을 하기 위해선 필요한 만큼 빈 블록의 개수가 필요하므로 플래시메모리 특성상 블록의 낭비가 있을 수 있다. 하지만 TRIM 명령이 적용된 SSD는 운영체제에서 데이터 삭제명령이 전달될 시 유휴시간에 미리 소거연산을 수행하므로 무효화된 파일의 블록을 정확하게 소거연산할 수 있기 때문에 블록의 낭비와 Freezing 현상을 예방할 수 있다.

Table 2. Comparison between Existing SSD and TRIM SSD Regarding Delete Operation

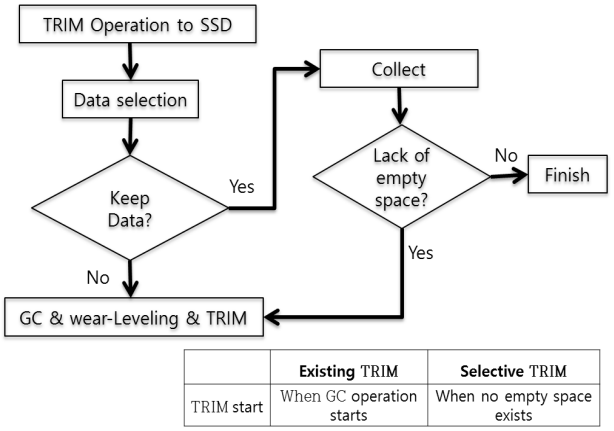
	Existing SSD	TRIM SSD
Erase Start Time	Before data write	Idle time
Erase Unit	# Empty blocks required	Blocks of a invalid file

3.3 TRIM 명령의 시점

Fig. 3은 TRIM 명령 시점에 대한 모식도이다. 운영체제에서 사용자가 삭제를 하면 운영체제는 TRIM 명령을 SATA Interface 이하 영역에 전달한다. SATA Interface는 보조기억장치를 위한 고속의 연결 방식으로 속도와 편의성, 안정성 등을 위해 규격화되었다. TRIM 명령은 단순히 명령만 전달하며, TRIM 명령이 실제로 동작하는 영역은 GarbageCollection & Wear-Leveling 이하 영역이다. 실제로 데이터를 삭제한 후 플래시메모리 내부의 작업도 함께 수행되어야 하기 때문이다.

3.4 TRIM LIST

Fig. 4는 선별적으로 데이터를 관리하여 수집하고, 빈 공간이 부족하여 빈 공간 부족 이벤트 발생 시 원래의 방법으로 수행하는 흐름도이다. 목적에 맞게 데이터를 선별한 후 선별되지 않은 데이터는 기존의 TRIM 명령기법을 따른다. 선별된 데이터는 수집되어 TRIM LIST로 관리하고 작업을 종료한다. 한 Bank가 가득 차 더 이상 해당 Bank에 블록을 사용할 수 없을 때 빈 공간 부족 이벤트를 발생시킨다. 빈 공간 부족 이벤트가 발생하면 TRIM LIST로 수집하였던 블록들 중 가득 찬 Bank에 해당하는 LIST를 원래의 영역으로 전달시켜 작업을 수행하게 한다.



	Existing TRIM	Selective TRIM
TRIM start	When GC operation starts	When no empty space exists

Fig. 4. Flow Chart for Selective TRIM Command

Fig. 5는 TRIM LIST 수집을 수행하는 과정을 나타낸 것이다. 사용자로부터 삭제 명령을 전달받으면 SSD로부터 TRIM 명령을 전달받게 된다. 무효블록 번호 22, 23, 24, 27, 29, 30 중 선별되지 않은 데이터가 속해있는 블록들은 GarbageCollection & Wear-Leveling 이하 영역으로 보내 기존 TRIM 명령기법과 동일하게 처리한다. 선별된 데이터들은 GarbageCollection & Wear-Leveling 이하 영역으로 보내지 않고 TRIM LIST를 만들어 관리한다. Fig. 5와 같이 무효화(Invalid)된 블록 번호 22, 23, 24, 27, 29, 30 중 3개의

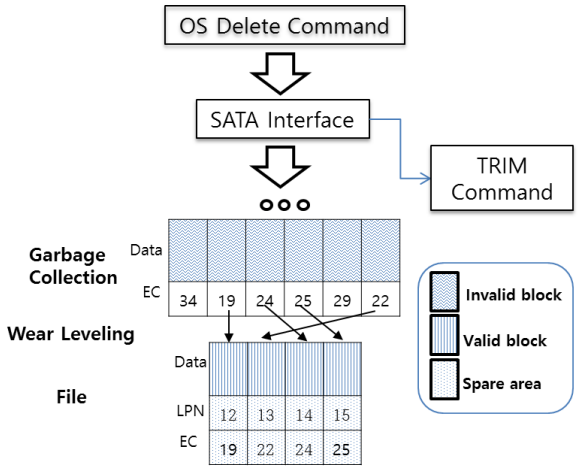


Fig. 3. Start Time of TRIM Command

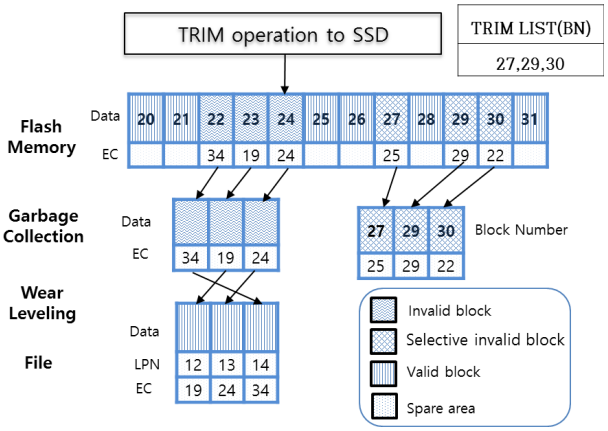


Fig. 5. TRIM LIST

블록은 기존 TRIM기법처럼 삭제하고 나머지 3개의 블록들은 선별되어 삭제되지 않고 관리한다면 플래시메모리 특성상 가장 속도가 느린 소거연산을 3번 회피할 수 있으므로 기존 TRIM 명령기법보다 성능이 향상된다.

3.5 빈 공간 부족 이벤트

Fig. 6은 빈 공간 부족 이벤트 발생 시 수행하는 과정을 나타낸 것이다. 한 Bank에 모든 블록들이 유효블록들로 가득 찼을 경우 빈 공간 부족 이벤트를 발생시킨다. 빈 공간 이벤트가 발생되면 Fig. 6과 같이 선별된 무효블록 번호 33, 34, 35, 37, 39, 40을 기존 TRIM 명령기법과 같이 GarbageCollection & Wear-Leveling 이하 영역으로 보낸 후 기존과 같이 수행한다. SSD Controller 및 FTL 내부구조는 제조사별로 다르고 성능과 직관적으로 연결되므로 빈 공간 부족 이벤트 발생 시 알고리즘을 수정하거나 다른 방법을 사용하지 않고 그대로 전달 역할을 한다. 하지만 빈 공간 이벤트 즉시 Bank 안에 있는 선별된 무효블록을 동시에 처리해야 하므로 기존 TRIM 명령기법보다는 수행시간이 길다.

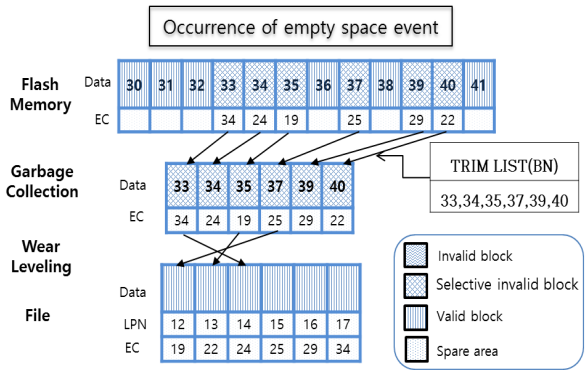


Fig. 6. Event Alert on Lack of Empty Blocks

4. 실험 및 평가

4절에서는 실험을 통하여 제안한 선별 TRIM 명령기법이 기존 TRIM 명령기법의 장점인 안정성(Freezing Solution)과 쓰기 속도를 최대한 지원하며, 모든 블록이 가득 차지 않을 경우에는 기존 TRIM 명령 다 성능이 우수함을 증명한다. 또한 모든 블록이 가득 찰 경우 성능 저하의 폭을 측정한다. 실험 환경은 Jasmin OpenSSD Platform을 사용하였으며, 페이지 맵핑(Page Mapping) 기반의 제공된 Greedy FTL을 사용하였다. 메모리는 삼성 K9LCG08U1M 8GB로 실험하였으며, 실험 데이터는 HWP, JPG, DOC파일을 대상으로 하였고 파일크기는 무작위로 선정하였다. 현재 Jasmin OpenSSD Platform은 TRIM 명령을 지원하지 않는다. TRIM 명령을 지원하지 않기 때문에 가상으로 TRIM 명령이 동작한다는 함수를 추가한 후 진행하였다. TRIM 명령이 동작하는 함수는 FTL 계층인 Ftl_Flush() 안에서 호출하여 사용하였는데, Ftl_Flush()는 POR/SPOR을 위해 반드시 보존되어야 할 FTL 메타데이터 정보를 플래시메모리에 기록하는 API이다. SATA Idle/Standby 시 주기적으로 호출하여 FTL 메타데이터의 일관성 및 POR을 가능하게 한다. TRIM 명령의 동작 방식인 유휴시간에 작업을 수행하는 방식과 같아 Ftl_flush() 단계에서 호출하여 실험하였다. 선별된 TRIM 명령과 기존 TRIM 명령의 수행시간 차이 실험을 위해, 블록을 2가지의 특정 블록으로 나누어 실험하였고, 500MB로 블록을 각각 고정하였다. 첫 번째 특정블록은 선별된 데이터의 블록으로 TRIM 명령이 동작 하지 않은 블록이며, 선별된 데이터 블록은 TRIM LIST로 관리하여 수집한다. 두 번째 특정블록은 선별되지 않은 데이터의 블록으로 기본적으로 동작하는 방식과 같이 TRIM & GarbageCollection & Wear-Leveling 이하 영역을 수행 한다.

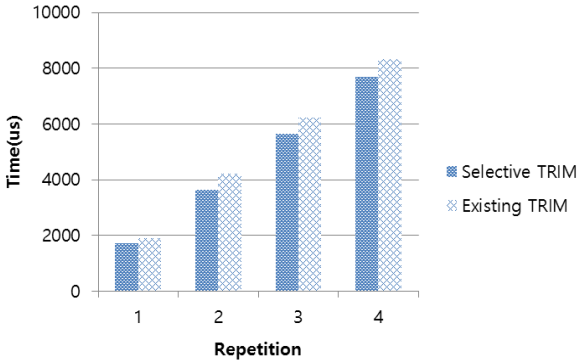


Fig. 7. Time Cost After 4 Times Repetition

Fig. 7은 고정된 500MB의 블록을 각각 1회, 2회, 3회, 4회 반복한 결과이다. 제안한 선별된 TRIM 명령기법은 기존 TRIM 명령기법과는 다르게 운영체제에서 삭제 명령을 내렸을 때, 바로 TRIM 명령을 전달하여 유휴시간에 처리하는 것이 아니라, 선별된 데이터가 있다면 TRIM LIST로 관리하며 다른 작업은 하지 않는다. 1회 수행 결과, 선별된 데이터가 있는 블록은 1713 마이크로초이며 선별되지 않은 기존 TRIM 기법으로 동작한 경우에는 1896 마이크로초로 수행되었다. 2회, 3회, 4회 결과 선별된 TRIM 명령기법은 기존 TRIM 명령기법보다 수행시간이 짧았다. Fig. 8은 5회, 6회, 7회, 8회 반복한 결과이다. 앞서 1회~4회 반복한 결과와 비슷한 패턴을 보이며 수행시간이 증가하였고, 8회까지 모든 메모리를 가득 채워도 특별한 문제점을 가지지 않고 정상적으로 반복되었다. 이는 명세하지는 않았지만, Jasmin OpenSSD Platform도 자체적으로 오버프로비저닝(Over Provisioning, OP) 공간을 확보하고 있음을 알 수 있다. 오버프로비저닝 공간이란 GarbageCollection, Wear-Leveling, Bad Block Control 등 SSD를 운영하는 데 필요한 핵심적인 기능들이 어떠한 상태에서도 원활하게 작동할 수 있도록 미리 예약된 예비 공간이다.

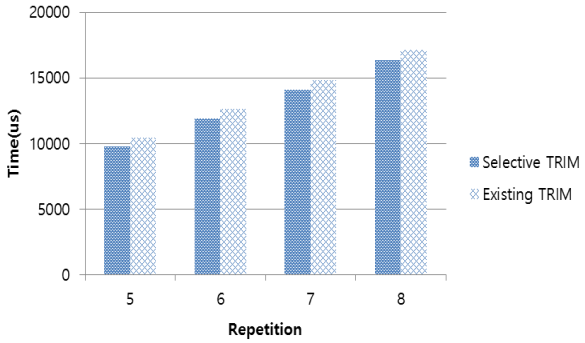


Fig. 8. Time Cost After 5~8 Times Repetition

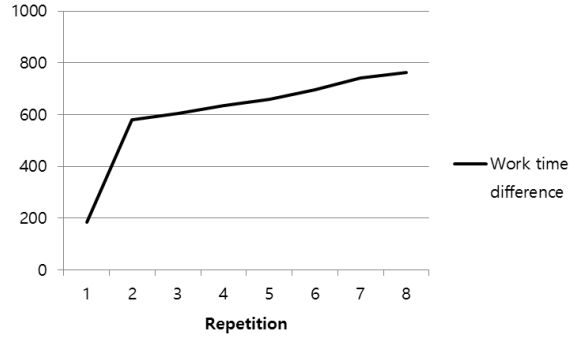


Fig. 9. Time Cost with Varying the Number of Repetitions

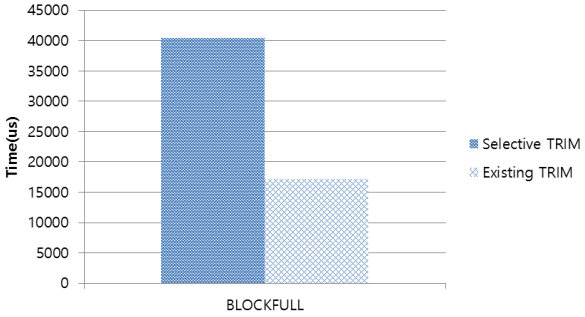


Fig. 10. Time Cost under the Condition that All Blocks are Full

Fig. 9는 1회부터 8회까지 반복하여 수행시간 차이를 나타낸 결과이다. 1회, 2회 반복 시 그래프가 급상승한 것은 메모리를 완전 포맷한 상태에서 작업을 하였기 때문이다. 2회~8회 결과 제안한 선별된 TRIM 명령기법이 기존 TRIM 명령기법보다 수행 시간이 점점 감소하는 것을 알 수 있다. 즉, 메모리가 가득 차기 전까진 선별된 데이터의 양이 많으면 많을수록 기존 TRIM 명령기법보다는 수행시간이 짧다는 것을 확인할 수 있다.

Fig. 10은 모든 데이터 블록이 가득 차 더 이상 메모리를 사용하지 못할 때, 선별된 TRIM 명령기법은 빈 공간 부족 이벤트를 발생시켜 기존의 TRIM LIST로 관리하여 수집하였던 블록들을 GarbageCollection & Wear-Leveling 이하 영역으로 보내주어 성능을 측정한 것이다. TRIM LIST로 관리하고 있던 블록들을 한 번에 처리한다면, 기존 TRIM 명령기법처럼 꾸준히 유휴시간에 처리한 기법보다는 수행시간이 길다. 하지만 일반 사용자가 데이터의 모든 블록을 사용하거나, 선별된 데이터가 주를 이루는 경우는 드물 것이다. 선별된 데이터를 주로 사용한다 해도 Fig. 7~Fig. 9와 같이 기존 TRIM 명령기법보다 이점이 있기 때문에 디지털 포렌식 관점에서 선별적으로 TRIM 명령을 사용하여도 기존 TRIM 명령의 안정성과 쓰기 속도를 유지하며 사용이 가능하다.

5. 결론 및 향후 계획

디지털 제품의 사용량이 급속히 증가하면서 범죄와 관련된 정보들도 디지털화되고 있다. 디지털 증거는 변조가 용이하고 메타데이터나 정황 증거만으로는 증거 채택이 어렵다. 이러한 특성 때문에 TRIM 명령기법은 많은 장점이 있지만 선별적으로 데이터를 관리하는 기법이 필요하다. 이에 기존 TRIM 명령기법의 안정성(Freezing Solution)과 쓰기 속도를 유지하며 선별적으로 데이터를 관리하는 TRIM 명령기법을 제안하였다. 제안 기법은 공공데이터를 기반으로 하여 데이터를 선별하고 증거 채택이 이루어질 가능성이 높은 것을 선택하여 기존 TRIM 명령기법을 그대로 적용하여 사용할 수 있는 방법이다. 본 논문의 실험을 통해 선별적으로 TRIM을 관리하면 특정한 상황을 제외하고는 성능이 더 우수함을 확인 하였다. 향후 실제 TRIM 명령이 적용된 내장형 플랫폼에서 다양한 실험을 진행하여 특정한 상황이 여러 번 반복되었을 때 성능을 측정하여 기존 TRIM 명령기법과의 차이를 비교하고 최악의 상황에 대처하는 기법을 연구할 계획이다.

References

[1] I. R. Jeong, D. W. Hong, and K. I. Chung, “Technologies and Trends of Digital Forensics”, *Electronics and Telecommunications Research Institute, Electronics and Telecommunications Trends*, Vol.22, No.1, pp.97-104, 2007.

[2] D. H. Kang and Y. I. Eom, “Analyses of the Characteristics of Trim Command in Solid State Drives,” *Proc. of the Korean Information Science Society Conference*, Vol.2014, No.6, Jun., 2014.

[3] H. S. Shin, “A Study on the possibility of logical data recovery of SSD consequential enable TRIM,” Dongguk Graduate School of International Affairs & Information G, Aug., 2013.

[4] Forensic-proof, “SSD Forensics: TRIM Command,” [Internet], http:// forensic-proof.com.

[5] S. H. Lee, M. S. Shin, and D.-J. Park, “Preventing Deleted File Recovery using Block Permutation on NAND Flash Memory,” KIISE: Database, Vol.39, No.6, Dec., 2012.

[6] B. Dipert and M. Levy, “Designing with Flash Memory,” Annabooks Publisher Poway, CA, USA, 1993.
[7] S.-W. Lee, D.-J. Park, et al., “A log buffer-based flash translation layer using fully-associative sector translation”. *ACM Transactions on Embedded Computing Systems*, Vol.6, No.3, Jul., 2007.
[8] L.-P. Chang, T.-W. Kuo, and S.-W. Lo, “Real-Time Garbage Collection for Flash-Memory Storage Systems of Real-Time Embedded Systems,” *ACM Transactions on Embedded Computing Systems*, Vol.3. No.4, Nov., 2004.
[9] X. Wang and J. Wang, “A Wear-Leveling Algorithm for Nand Flash in Embedded System,” *5th IEEE International Symposium on Embedded Computing,* Oct., 2008.
[10] M. Saxena and M. M. Swift, “FlashVM: Virtual memory management on flash,” *Proc. of the 2010 USENIX Annual Technical Conf. (USENIX ATC)*, Jun., 2010.
[11] J. Kim, H. Kim, S. Lee, and Y. Won, “FTL design for TRIM command,” *Proc. of 5th Int’l Workshop on Software Support for Portable Storage (IWSSPS)*, 2010.
[12] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. “Design tradeoffs for SSD performance,” *Proc. of USENIX Annual Technical Conference*, Jun., 2008.
[13] [Internet], http://articles.forensicfocus.com/2012/10/23/why-ssd-drives-destory-court-evidence-and-what-can-be-done-about-it/xhsl.

황 현 호

e-mail : fireicesss7@ssu.ac.kr
2015년 숭실대학교 컴퓨터학과 석사
관심분야 : 플래시메모리, 데이터베이스 등

박 동 주

e-mail : djpark@ssu.ac.kr
1995년 서울대학교 컴퓨터공학부(학사)
1997년 서울대학교 컴퓨터공학과(석사)
2001년 서울대학교 전기전자컴퓨터공학부 (박사)
2001년~2003년 삼성전자 책임연구원
2004년~2005년 숭실대학교 컴퓨터학부 전임강사
2006년~2009년 숭실대학교 컴퓨터학부 조교수
2010년~현 재 숭실대학교 컴퓨터학부 부교수
관심분야 : 플래시메모리, 임베디드 데이터베이스, 멀티미디어 데이터베이스 등