[Ubuntu 13.10] SSD Trim 적용하기

A. 문서정보

- **최초작성일** : 2014-04-11
- **최초작성자** : simplism (song.chiseung@gmail.com)
- **변경이력**
  1. 2014-04-11 : 최초작성 by simplism
  2. 2014-04-22 : 1차 작성완료 by simplism

B. 시작하기 전에..

컴퓨터의 성능을 결정하는 요소는 아주 다양하다. 아주 기본적으로는 하드웨어의 성능이 중요하며, 동일한 하드웨어라고 해도 컴퓨터 자원을 사용하는 용도(프로그래밍, 게임, 3D영상 렌더링 등등…)에 따라서도 사용자가 체감하는 성능의 차이가 있고, 네트워크 속도에 영향을 미칠 수 있고, 동일한 용도라도 소프트웨어가 무엇이냐에 따라서도 체감성능이 달라질 수도 있다.

그 중에서 컴퓨터를 구성하는 하드웨어만을 고려한다면, 과거에는(실은 최근까지도..) CPU[1], 메인메모리[2]가 성능에 가장 큰 영향을 준다. 그런데, 두 부품이 성능향상을 주요 목표로 삼고 개발되었던 것에 반해서 실제로 컴퓨터의 작업결과물을 파일이라는 형태로 저장할 수 있는 보조기억장치의 주 개발방향은 성능향상보다는 용량증가를 하는 것이었다. 보조기억장치로 널리 사용되고 현재까지도 사용되고 있는 하드디스크는 KB, MB, GB, TB라는 단위로 용량을 늘려왔다.

그런데, 이 하드디스크는 성능향상에 걸림돌이 되는 요소이다. 비유를 통해서 설명해보면 아래와 같다.

> **사람A**는 **B책상**에서 작업을 하고 있다. B책상이 있는 방에는 **C책장**에 책이 많이 꽂혀있다.

- **사람A** : 작업의 두뇌역할을 하는 CPU
- **B책상** : 작업을 진행하는데 작업공간으로 사용하는 메인메모리
- **C책장** : 작업이 진행된 결과물을 저장하거나 저장된 자료를 꺼내오는 하드디스크

사람A가 두뇌회전이 빠르면(=CPU성능) 작업이 진행되는 속도가 빠르다. 그리고 사람A가 작업공간으로 활용하는 **B책상**의 크기(=메인메모리용량)가 매우 커서, 작업에 필요한 자료를 한 번에 많이 꺼내놓으면 꺼내놓을 수록 작업이 빨리 진행될 것이다. 그런데, **B책상**의 크기가 작아서, 작업에 필요한 자료 중에 일부만 꺼내놓아서 작업을 진행하다보면 B책상에서 책을 치우고 **C책장**(=하드디스크)에서 나머지 책을 꺼내서 B책상에 올려놓는 과정(=swapping)이 추가가 된다. 그런데, C책장이 너무 크다보니 필요한 자료를 찾아서 가져오는데 시간이 오래 걸린다. 그러므로 B책상이 크면 클 수록 그런 작업의 횟수가 줄어들테니 작업속도가 빠른 것이 당연한 사실이다.

위에서 비유를 한 것과 동일한 현상이 컴퓨터가 어떤 작업을 진행할 때 똑같이 발생하는데, 하드디스크의 주 개발목적이 속도향상이 아니다보니 swapping작업이 일어나는 횟수가 늘어날 수록 작업속도가 크게 저하된다. 이 속도향상에 발목잡는 해결사로 등장한 놈이 **고성능책장**(=SSD)이다. 책장에서 책을 찾고 꺼내는 속도가 기존의 책장보다 훨씬 빠르다보니 동일한 책상을 사용한다고 하더라도 성능저하가 많이 일어나지 않는거다.

C. SSD Trim의 필요성

SSD는 기계모터로 회전하면서 원하는 데이터를 찾는 하드디스크에 비해서 전기신호만으로 구동하다보니 단연 하드디스크보다는 Read/Write 속도가 우월하게 빠르다.[3] 그렇지만 이 SSD를 이루고 있는 메모리인 NAND 메모리의 특성에서 기존의 하드디스크와는 차이점이 발생하는데, 그 차이는 파일이 삭제될 때이다.

기존에 하드디스크를 사용할 때는 파일을 삭제할 때 해당 파일을 하드디스크 상에서 바로 삭제하지 않는다. 단지 해당 데이터가 의미가 없는 데이터이므로 이 공간은 다른 데이터로 덮어써도 좋다는 상태로 두게된다. **덮어쓰기가 가능한 하드디스크에서는 어차피 이제부터 무의미한 데이터인데 그것을 지우기 위한 작업을 하는 데에 리소스를 낭비**하지 않는다는 것이다. 그래서 실제로 고용량 파일을 하드디스크에 작성할 때는 시간이 오래 걸리는데, 고용

량 파일을 삭제할 때는 굉장히 빠르게 지워지는 것을 대부분의 사용자가 경험해봐서 알 것이다. 실제로 사용자가 보는 데에서는 지워졌지만, 그 부분을 다른 파일이 덮어쓰지 않는 한은 디스크 상에 존재하고 있는 것이다.[4)]

그런데 SSD에서는 **SSD를 구성하고 있는 NAND 플래쉬메모리 특성상 해당 영역을 사용하기 전에는 반드시 먼저 지워주는 작업이 선행**되어야 한다는 것이다. 하드디스크로 구성된 경우라면 해당 영역이 무휴(invalid)상태라면 기존 데이터와 무관하게 덮어쓰는 작업이 가능하지만, SSD의 경우에는 먼저 초기화를 하고 데이터를 작성하는 작업을 해야한다는 것이다.

이 데이터를 사용할 영역을 미리 초기화해두는 것이 Trim이다. 이 작업을 미리 해두면, 다음 해당 영역에 데이터를 작성할 때 초기화작업을 거치지 않으므로 더 빠르게 파일을 작성할 수 있는 것이다. 대부분의 SSD가 데이터 Read/Write 작업은 빠르지만 Delete가 느리기 때문에 미리 Trim을 해두지 않는다면 Write 속도에도 영향을 미치므로 Trim 작업이 SSD에서는 필수라고 할 수 있다.

D. SSD Trim 적용 전 점검사항

01. SSD TRIM 적용조건

1. Linux Kernel 2.6.33 이상
2. SSD가 TRIM을 지원
3. 파티션이 EXT4나 BTRFS[5)]인 경우

a) 리눅스커널버전 확인

1번 조건을 확인하기 위해서 리눅스 커널의 버전을 확인한다. 커널버전 확인을 위해서 사용하는 명령어는 **uname**[6)]명령어이다.

> 실행 예

```
$ uname -r
3.11.0-15-generic
```

- **-r**(–kernel-release)옵션은 리눅스커널 release 정보만 반환하도록 설정한다.

b) TRIM 지원여부 확인

2번 조건인 SSD가 TRIM을 지원하는지 확인한다. 확인을 위해서 사용하는 명령어는 **hdparm**[7)]명령어이다.

> 실행 예

```
$ sudo hdparm -I /dev/sda | grep TRIM
           *    Data Set Management TRIM supported (limit 2 blocks)
```

- **-I**옵션은 드라이브에 직접 드라이브의 식별정보를 요청하여 받는다.
- grep 파이프라인을 이용해서 TRIM 문자열이 포함된 라인만 추려내여 원하는 정보를 파악하기 용이하도록 한다.

limit 2 blocks라는 문구가 거슬려서 검색해보니… 숫자의 크기가 크면 클 수록 한 번에 TRIM작업을 진행할 수 있는 블록의 크기인 것 같다. 만약에 숫자가 적으면 여러 번에 나눠서 TRIM작업이 진행되므로 TRIM작업 시 소모되는 시간이 늘어날 것이다.

c) 파티션유형 확인

3번 조건에 파티션이 EXT4나 BTRFS이어야 한다고 했다. 시스템을 직접 설치한 경우라면, 파일시스템을 직접 선택했을테니 알고 있겠지만… 과정 상 확인하는 과정으로 확인해보려고 한다. 여기서 사용할 명령어는 **df**[8)]명령어를 사용한다.

실행 예

```
$ df -T /dev/sda1
Filesystem     Type 1K-blocks    Used Available Use% Mounted on
/dev/sda1      ext4   9711136 5205896   3988888  57% /
```

- **-T**(–print-type)옵션은 파일시스템의 유형까지 출력하도록 하는 옵션이다.
- 여기에서 디스크 사용율을 확인할 생각은 없지만, 파일시스템 유형을 확인하는데 가장 간단한 방법이라 이 명령어를 선택했다.

02. SSD Trim 테스트

Trim을 실행하는 명령어는 **fstrim**[9]이다. 명령어를 직접 실행해서 실제로 동작하는지 확인해본다.

```
$ sudo fstrim -v /
/: 1425326080 bytes were trimmed
```

- 1425326080 바이트가 TRIM되었다는 메시지가 출력된다. 정상적으로 동작한다는 소리…

E. SSD Trim 적용

01. 시스템 정보

- **장비명칭** : HP mini110-1111TU
- **운영체제** : Ubuntu 13.10 (Saucy Salamander)
- **커널버전** : Linux 3.11.0-15-generic i686 GNU/Linux
- **SSD** : ADATA Premier Pro SP600 (32GB) SSD

02. TRIM 방식결정

참고문서 1[10]에 따르면, TRIM을 적용하는 방식은 2가지이다.

1. 파일 삭제할 때마다 해당 블록 TRIM 실행
2. 주기적인 작업으로 전체 TRIM 실행(권장)

1번 방법은 파일이 삭제될 때마다 파일이 저장되어있던 해당 블럭이 재사용이 가능하도록 바로 TRIM처리를 하는 것이고, 2번 방법은 주기적으로 파일시스템 전체에 TRIM처리를 하는 것이다. 작업을 시작하기 전에 자료를 조사할 때도 그렇고 주로 주기적으로 TRIM을 실행하도록하는 2번 방법을 권장하고 있다.

2가지 방식으로 SSD Trim을 지원할 수 있는데, 시스템의 사용유형에 따라서 조금 달라질 수 있을 것 같다.

1. 파일생성만 빈번하게 발생하는 경우
   - 파일생성 시에 Trim을 하지 않아도 되므로 Online Trim(Automatic Trim)이 보다 효율적일 것
2. 파일삭제가 빈번하게 발생하는 경우
   - 삭제 시에 Trim을 하게되므로 삭제속도가 저하되므로 Manual Trim이 보다 효율적
3. 생성과 삭제가 균등하게 발생하는 경우
   - 디스크의 용량이 충분하다면, Manual Trim이 보편적으로 효율적일 것

시스템의 사용유형을 위 처럼 3가지로 판단할 수는 없지만, SSD Trim과 영향이 있는 것은 파일의 생성과 삭제이므로 그 두 가지 작업에 따른 유형으로 3가지로 추려본 것이다. 대부분의 시스템이 생성과 삭제가 균등하게 발생하는 경우일테니 아마도 Manual Trim이 보통의 경우에 권장하는 형태라고 생각해서 대부분의 가이드 문서들이 Manual Trim을 권장하는 듯하다.

03. OnLine Trim(Automatic Trim) 적용

> ***주의***
> 해당 작업은 시스템의 파티션 정보를 직접 수정하는 작업이 포함되어 있으므로 작업에 유의하고, 설정파일 백업은 반드시 해둬야 추후 복구에 보다 용이할 수 있습니다.

a) /etc/fstab 파일백업

유닉스계열(리눅스 포함) 운영체제에서 /etc/fstab파일은 시스템이 켜질 당시에 시스템이 사용할 디스크정보를 읽어오는 설정파일이다. 이 설정파일이 잘 못 편집될 경우에 시스템이 정상적으로 켜지지 않을 우려가 있으니 직접 해당 파일을 편집할 시에 유의해서 작업해야한다.

작업을 시작하기 전에 항상 파일을 백업해둔다.

```
$ sudo cp /etc/fstab /etc/fstab.20140417
```

b) /etc/fstab 편집

이제 해당 파일을 선호하는 편집기로 열어서 해당 파티션의 속성에 Online Trim 속성(discard)을 부여하려고 한다. 해당 파일을 시스템최고관리자인 root 권한으로 실행해야 편집이 가능하다.

01) vi나 vim으로 편집

```
$ sudo vim /etc/fstab
```

02) gedit 편집기로 편집

```
$ gksu gedit /etc/fstab
```

03) 편집 전 /etc/fstab 내용

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>

#Entry for /dev/sda1 :
UUID=40f6e757-5b5a-4166-9817-945365bddea0       /       ext4    errors=remount-ro       0
#Entry for /dev/sda5 :
UUID=b5d5ccaf-807c-4401-b0d9-82ac7e809cf1       none    swap    sw      0       0
```

04) 편집 후 /etc/fstab 내용

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>

#Entry for /dev/sda1 :
UUID=40f6e757-5b5a-4166-9817-945365bddea0       /       ext4    discard,errors=remount-ro
#Entry for /dev/sda5 :
UUID=b5d5ccaf-807c-4401-b0d9-82ac7e809cf1       none    swap    sw      0       0
```

05) 편집내용 비교

백업해둔 파일과 비교해보면 어떤 내용을 편집했는지 확인할 수 있다.

```
$ diff fstab fstab.20140417
8c8
< UUID=40f6e757-5b5a-4166-9817-945365bddea0     /        ext4    discard,errors=remount-r
---
> UUID=40f6e757-5b5a-4166-9817-945365bddea0     /        ext4    errors=remount-ro
```

두 파일의 차이점을 비교하는 **diff**명령어로 비교해보면 마운트 옵션에 **discard**가 추가된 것을 확인할 수 있다.

c) Online Trim 적용테스트

이제 정상적으로 적용되었는지 테스트를 해볼 차례이다. /etc/fstab 내용이 적용이 되려면 시스템을 재부팅을 해야 한다. 재부팅을 완료하고 아래와 같은 방식으로 테스트를 진행해본다. 테스트방법은 참고문서 6[11)]의 답변에 있는 내용을 참고하여 진행해봤다.

01) 테스트 전 Manual Trim 실행

Online Trim을 테스트해보기 전에 혹시 Trim이 되지 않은 블록이 있을 지 모르기 때문에 Manual Trim을 한 번 해준다.

```
$ sudo fstrim -v /
/: 11876442112 bytes were trimmed
```

02) 파일생성

파일을 생성하고 삭제할 예정이므로 임의의 파일을 생성한다.

```
$ dd if=/dev/urandom bs=1M count=50 of=bigfile
50+0 레코드 들어옴
50+0 레코드 나감
52428800 바이트 (52 MB) 복사됨, 19.3693 초, 2.7 MB/초
$ sync
$ ll
-rw-rw-r--  1 simplism simplism 52428800  4월 17 09:09 bigfile
```

03) 파일삭제

이제 Online Trim이 정상적으로 실행되는지 확인해봐야 하므로 삭제를 해서 테스트한다.

```
$ rm bigfile
$ sync
```

05) 점검

만약 Onlie Trim이 정상적으로 동작하고 있다면, Manual Trim을 실행했을 때 Trim되는 Byte의 수가 0이거나 작은 숫자일 것이다.

```
$ sudo fstrim -v /
/: 0 bytes were trimmed
```

위 처럼 0 byte가 Trim되었다는 것은 정상적으로 Online Trim이 적용된 것으로 판단할 수 있다.

04. Manual Trim 적용

Manual Trim은 Online Trim과 다르게 주기적으로 파티션 전체에 대해서 Trim을 적용하는 방법이다. 그렇다고 사용자가 명령어로 매번 입력할 수 없으므로, 스크립트를 만들어두고 Cron[12]에 등록하여 자동으로 실행하도록 시스템을 구현하도록 한다.

아마도 인터넷에서 사용자들이 안내하는 SSD Trim 가이드에 주로 Manual Trim을 추천하는 이유 중에 하나가 /etc/fstab파일을 직접 수정하는 위험성이 없어서 추천하는 것일지도 모른다는 생각이 들었다. 초보자들의 경우에는 가이드 문서에서 제공하는 스크립트를 복사 & 붙여넣기하여 진행하기에도 나쁘지 않아서 직접 /etc/fstab 파일을 수정하는 데에 어려움을 느끼는 사용자라면 Manual Trim이 보다 적절한 방법일지도 모르겠다.

a) 쉘스크립트 작성

사실 굳이 쉘스크립트를 작성할 필요는 없으나, 시스템에 문제가 생겨서 운영체제를 다시 설치한다고 해도 기존에 작성해둔 쉘스크립트만 잘 백업해두면 추후에 새로 설치한 운영체제에도 적용이 간단하기 때문에 개인적으로는 이런 작업들은 왠만하면 쉘스크립트로 작성해두는 편이다.

trim.cron

```
#!/bin/sh

LOG=/var/log/trim.log

echo "---------- $(date +%F_%H:%M:%S) Trim Result ----------" >> $LOG
fstrim -v / >> $LOG
```

- $HOME/.bin 디렉토리를 만들고 그 하위에 위 스크립트를 작성한다.

```
$ pwd
/home/simplism/.bin
$ ls -l
합계 4
-rw-rw-r-- 1 simplism simplism 124  4월 11 01:43 trim.cron
```

b) 쉘스크립트 권한부여

쉘스크립트가 실행(execute)될 수 있도록 실행권한을 부여한다.

```
$ chmod u+x trim.cron
$ ll
합계 16
drwxrwxr-x  2 simplism simplism 4096  4월 17 09:06 ./
drwxr-xr-x 29 simplism simplism 4096  4월 17 16:51 ../
-rwxrw-r--  1 simplism simplism  151  4월 11 08:39 trim.cron*
```

c) Cron 등록

Cron은 유닉스 계열의 운영체제에서 주기적인 작업을 자동화할 수 있는 시스템이다. 위에서 작성한 쉘스크립트를 하루를 주기로 실행하도록 등록할 생각이다. 시스템을 운영하면서 삭제되는 파일이 많다면, 좀 더 주기를 짧게 가지도록 Cron을 등록하면 되겠다.

지금 운영 중인 서버의 경우에는 /(root)로 할당된 파티션은 파일이 Write가 되는 일들은 많으나(로그성 데이터…) 삭제가 되는 일들은 거의 발생하지 않는다. 주로 파일들이 Write, Modify, Delete되는 작업이 발생되는 /home 파티션은 분리시켜뒀으므로 해당 파티션은 Online Trim 보다는 Manual Trim이 적합하다고 판단하였다.

Cron을 등록하는 방법에는 여러 가지 방법이 있으나, 현재 우분투 시스템에 적합한 방법으로 설정하려고 한다. Crontab -e[13] 명령어를 사용하여 직접 편집할 수 있으나, 설명하기에 쉽지 않고 설정파일들을 직접 편집하는 방법에는 역시나 매번 위험부담을 안고 있으니 심볼릭링크(Symbolic-Link)를 사용하여 1일 주기로 쉘스크립트를 실행하는 방법으로 설정하려고 한다.

```
$ cd /etc/cron.daily
$ sudo ln -s /home/simplism/.bin/trim.cron trim
$ ls -l trim
lrwxrwxrwx 1 root root 29  4월 22 10:01 trim -> /home/simplism/.bin/trim.cron
```

위 처럼 /home/simplism/.bin/trim.cron 파일에 trim이라는 링크파일로 연결을 했다. 우분투 시스템의 경우에는 /etc/cron.daily 폴더에 있는 쉘스크립트(또는 심볼릭 링크로 연결된 쉘스크립트)를 약 하루를 주기로 실행한다. 기존에 crontab -e 명령어를 통해서 직접 특정 시간 대를 지정하는 것에 비해서는 약간 느슨하게 동작하는 것이 차이라면 차이이다.(정확하게 특정 시간에 실행되도록 하려면 crontab -e 명령어를 이용해서 직접 cron 스크립트를 편집하는 것이 좋다.)

05. Trim 최적화

아마 대부분의 사용자들은 Manual Trim이 권장할 사항이겠지만, 시스템의 사용유형에 따라서는 Online Trim이 보다 효율적일 수도 있다. 시스템의 사용유형이 사용자마다 너무 차이가 크기 때문에 두 방법 중에 하나의 방법이 정답이라고 말할 수는 없다.

현재 운영 중인 넷북서버의 경우에도 주로 Write 위주인 /(root) 파티션의 경우에는 Manual Trim을 적용해뒀고, 파일의 Write, Modify, Delete가 계속 많이 발생하고 있는 다른 파티션의 경우에는 Online Trim을 적용해둔 상태이다. 뭐… 아직까지는 사용하면서 시스템의 속도가 저하되는 것을 발견하지 못했으니 적합하게 선택하여 설정한 것으로 판단하고 있다.

이 처럼 사용자의 시스템의 사용유형에 따라서 Online Trim 또는 Manual Trim 또는 Online + Manual Trim을 혼합하는 방법 등 여러 형태로 설정하여 사용해보면서 조정해나가는 방법을 추천한다.

F. 참고문서

1. Enable Trim On SSD (Solid-State Drives) In Ubuntu For Better Performance
2. HOWTO: Configure Ext4 to Enable TRIM Support for SSDs on Ubuntu and Other Distributions
3. BTRFS
4. SSD TRIM Support and Block Limits?
5. 리눅스에서 파티션의 파일 시스템 타입 확인하기
6. Automatic TRIM vs. manual TRIM
7. SSD 의 특성과 TRIM 기능의 이해, 자동 TRIM(트림) 기능의 작동 여부 확인과 설정 방법

---

[1] 인텔CPU의 경우에 펜티엄4까지는 주로 클럭스피드를 올리는 것에 주력을 삼았고 실제로 클럭스피드가 올라감에 따라서 성능향상에 많은 도움이 되었다. 그렇지만 펜티엄4 이 후에는 클럭스피드를 올리는 것이 큰 성능향상이 없자. 그 이후에 펜티엄D, 코어2시리즈, i시리즈로 진화하면서 멀티코어를 갖도록 개발해 다중처리를 지원하여 성능향상을 꾀했고, 친환경이 추세가 되면서 성능은 떨어뜨리지 않으면서 저전력을 동작할 수 있게 개발하는 추세이다.
[2] 흔히 RAM이라고 표현하는 메인메모리는 컴퓨터에서 필수적인 부품으로 **사람이 책상에 앉아서 어떤 작업을 하는 것**에 비유하면 CPU가 사람이고, 메인메모리는 책상의 역할에 해당한다. PC전원이 종료되면 내용이 사라지는 휘발성기억매체인 메인메모리는 현재 동작 중인 프로그램을 올려두는 기능을 담당한다. 메인메모리의 경우에는 주로 용량늘리기가 성능을 올리는 핵심전략이었다. SD-RAM, DDR, DDR2, DDR3와 같이 연결방식을 개선하면서 속도향상에도 노력했지만 역시 과거나 지금이나 용량이 큰 경우가 체감성능향상에는 최고이다.
[3] 하드디스크의 경우에는 디스크의 특정영역을 찾기 위해서 디스크를 회전시키고 데이터를 읽거나 쓰는 헤더를 해당 위치로 옮기는 등의 복잡한 알고리즘을 통해서 원하는 데이터를 찾는 작업을 진행한다. 그렇지만 SSD의 경우에는 원하는 데이터의 위치가 확인되면, 전기적인 신호로만 구동되니 해당 위치의 데이터를 바로 가져올 수 있기 때문에 우월하게 빠를 수 밖에 없다.
[4] 이런 이유로 하드디스크에 파일을 보관할 경우에 특정 파일을 휴지통에서 지워도 다른 파일로 덮어쓰지 않았다는 것이 전제가 되면 완벽하게 복구가 가능한 것이다.

[5] BTRFS(B-tree file system)는 컴퓨터 파일시스템 중 하나로, 현재 페이스북의 크리스 메이슨이 개발을 지휘하고 있다고 한다. GPL로 라이센스가 적용이 되어있으며, 2014년 2월 5일 현재 최신판은 3.13으로 리눅스 커널 2.6.32 이후에 릴리즈된 커널에 기본으로 포함된다.(출처:위키피디아)
[6] 시스템의 정보를 출력하는 명령어
[7] SATA/IDE 장치의 정보를 조회하거나 설정할 수 있는 명령어
[8] 파일시스템의 사용율을 확인하기 위한 명령어
[9] 마운트된 파일시스템의 사용하지 않는 블록을 폐기하는 명령어
[10] Enable Trim On SSD (Solid-State Drives) In Ubuntu For Better Performance
[11] Automatic TRIM vs. manual TRIM
[12] 유닉스 계열의 시스템에서 사용하는 스케쥴링 프로그램으로 주기적으로 자동실행되어야하는 작업에 활용한다.
[13] Cron 스크립트를 직접 편집하는 방법