# SSD 를 위한 로그 기반 파일시스템의 개선

기안호, 박성민, 강수용
한양대학교 전자컴퓨터통신공학과
e-mail : kyano@hanyang.ac.kr

# An Enhancement of Log-Structured Filesystem for SSD

Anho Ki, Sungmin Park, Sooyong Kang
Dept. of Electronics Computer Engineering, HanYang University

**요 약**

플래시메모리 기반 저장매체인 SSD 의 발전에 따라 로그 기반 파일시스템의 중요성이 증가되고 있다. 그러나 기존의 로그 기반 파일시스템은 여러 지점에 대해 동시에 쓰기 동작을 수행하여도 성능 저하가 발생하지 않는 SSD 의 특성을 제대로 활용하지 못하고 있다. 이에 대해 우리는 SSD 를 위한 다중 로그 지점을 가지는 로그 기반 파일시스템의 필요성을 보여주고, 그에 따른 여러 가비지 컬렉션 기법을 제안하고 비교하였다.

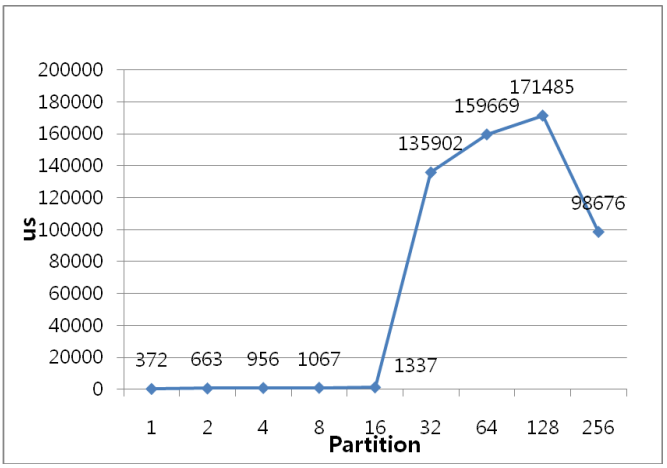
## 1. 서론

플래시메모리와 관련 기술의 발전에 따라 SSD(Solid State Drive)는 휴대용 랩탑 컴퓨터를 비롯하여 데스크탑 컴퓨터, 대용량 스토리지 서버에 이르기까지 널리 쓰이고 있다. 그러나 SSD 를 위해 특별히 최적화된 파일시스템에 대한 연구는 아직 이루어지고 있지 않다. 대신 이미 기록된 위치에 다시 재기록을 하기 어려운 플래시메모리의 특성 때문에 모든 쓰기 동작이 덮어 쓰기(overwrite)가 아닌 이어 쓰기(append)로 디스크에 전달되는 로그 기반 파일시스템(Log-Structured Filesystem)에 대한 관심이 다시 높아지고 있는 추세이다. 기존의 여러 로그 기반 파일시스템의 경우 HDD 만을 고려하여 개발되었기 때문에 SSD 의 새로운 특성을 모두 반영하고 있지는 않다. HDD 와 비교할 수 있는 SSD 의 새로운 특성으로는 동시에 여러 지점에 나누어서 쓰기 동작을 하여도 일정한 수준까지는 안정된 성능을 보여준다는 것이다.[1] 이 점을 이용하여 우리는 기존의 로그 기반 파일시스템을 개선할 수 있는 방법에 대해 연구하였다.

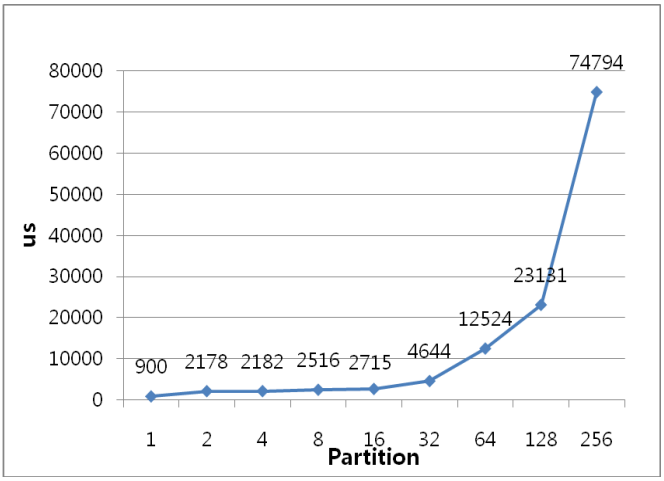
## 2. SSD 와 로그 기반 파일시스템

### 2-1. SSD 에서 다중 로그 지점

고전적인 로그 기반 파일시스템에서는 파일시스템의 메타데이터와 실제 파일데이터를 구분하지 않고 하나의 세그먼트로 묶어서 디스크에 저장하였다.[2] 이는 현재 SSD 에서 최적의 성능을 낼 것으로 기대되고 있는 NilFS 에서도 그대로 적용되고 있다.[3][4]
그러나 SSD 에는 기존의 로그 기반 파일시스템이 주목하지 못한 특성이 있다. 동시에 연속적이지 않은 여러 위치에 쓰기를 시도하여도 일정한 수준까지는 속도가 보장이 된다는 것이다.[1] 이를 확인하기 위해 우리는 현재 널리 사용되고 있는 일부 SSD 제품을 이용하여 실험을 하였다.




(a) 모델 S




(b) 모델 J


(c) 모델 H

(그림 1) 각각의 SSD 제품 모델에 따른 다중 지점 쓰기시 반응속도

위 그림 1 에서 살펴보면 SSD 모델 S 와 모델 J 의 경우에는 16 개의 서로 떨어진 지점에 동시에 쓰기를 수행할 경우까지는 안정된 속도를 보여주고 있음을 알 수 있다. 모델 H 의 경우에는 여러 지점에 쓰기를 수행함에 따라 아주 작지만 그 반응속도가 빨라짐을 알 수 있다. 이 두 가지 경우는 각각의 SSD 모델들이 사용하는 사상 방법에 차이가 있기 때문이다. 그림 1(a)와 (b)에서 보여지는 두 모델은 블록 단위 사상(Block-level mapping)을 사용하고 있지만, (c)에서 보여지는 모델 H 는 페이지 단위 사상(Page-level mapping)을 쓰고 있기에 서로 떨어진 지점에 대한 쓰기일 지라도 하나의 플래시메모리 블록으로 내려 보낼 수 있기 때문에 전형적인 동시성 향상에 따른 성능 향상의 모습을 보여준다. 물론 위 3 가지 경우 모두 동시에 서로 떨어진 여러 지점에 쓰기를 진행하여도 일정 수준의 지점 개수에 대해서는 성능이 떨어지지 않음을 보여준다.

위의 실험에서 보여지는 결과는 모두 여러 로그 지점을 갖는 파일시스템의 가능성을 보여주고 있다.

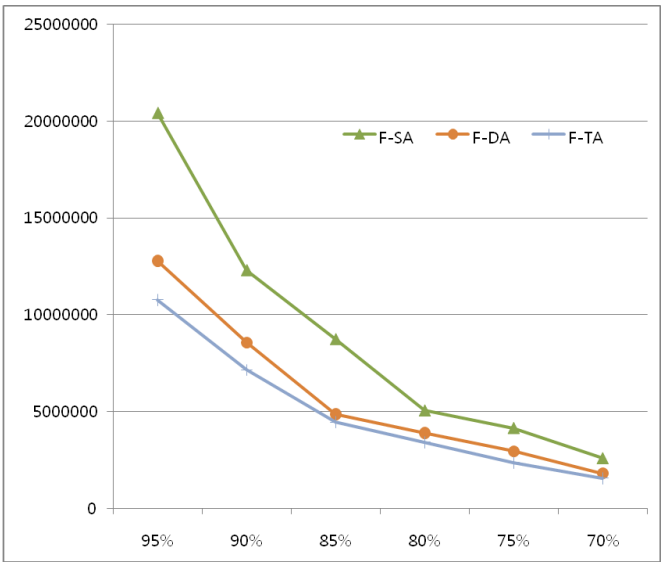
### 2-2. 다중 로그 지점의 필요성

로그 기반 파일시스템의 가장 큰 문제는 가비지 컬렉션이다. 이는 실제 데이터의 쓰기 동작과 상관없이 추가적으로 발생하는 디스크 접근으로 이를 효율적으로 수행함은 곧 로그 기반 파일시스템의 성능에 큰 영향을 미치게 된다.[1] 그러나 지금까지의 가비지 컬렉션 정책들은 모두 하나의 로그 지점을 가지고 순차적으로 로그를 기록하는 방식만을 고려해 왔다. 그에 따라 데이터 접근/갱신의 주기가 다른 여러 데이터들이 한 세그먼트에 동시에 기록되는 것을 가정하고 있으며 이들 데이터의 분리를 위해 여러 기법을 시도하였다.

그러나 다중 로그 지점을 사용하게 되면 가비지 컬렉션 정책은 다양한 시도를 해볼 수 있게 된다. 애초에 세그먼트 구성을 각각의 데이터의 특성을 고려하여 유사한 데이터끼리 되도록 한다면 가비지 컬렉터는 자신이 동작하는 시점에서 복잡한 연산을 하지 않게 되고 로그 기반 파일시스템의 성능에 가장 큰 영향을 끼치는 가비지 컬렉션을 더욱 빠르게 수행할 수 있다. 이는 특히 플래시메모리 기반의 디스크에서 큰 의미를 가지는데, 기록 횟수의 제한이 있는 플래시메모리의 특성 상 자주 쓰이지 않는 데이터끼리 모아진 세그먼트는 재쓰기가 발생하지 않아 전체적인 플래시메모리의 수명을 늘릴 수 있다.[5]

### 2-3. 다중 로그 지점에 따른 가비지 컬렉션 정책

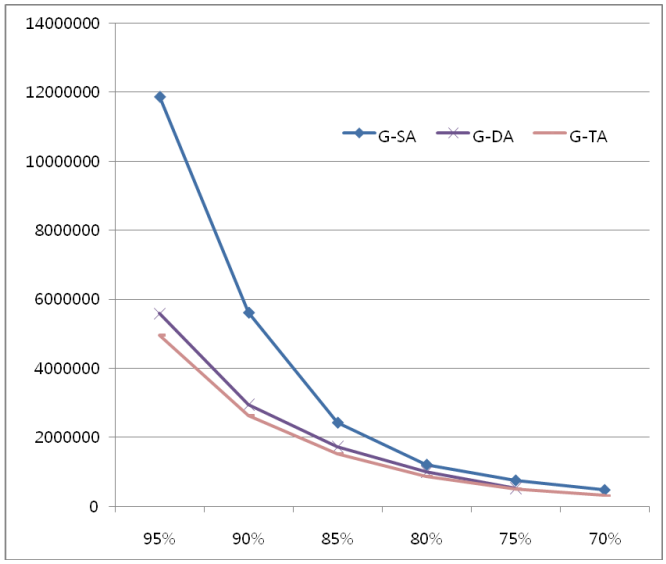
다중 로그 지점을 사용하는 것이 실제 가비지 컬렉션에 어떤 영향을 주는 지 확인하기 위해 간단한 시뮬레이션을 수행하였다.




(그림 2) FIFO 를 이용한 가비지 컬렉션 정책에서의 쓰기 횟수.

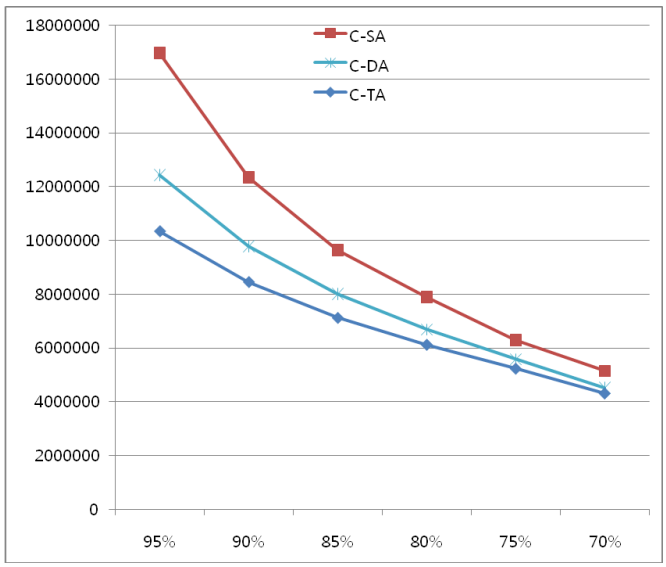
그림 2 는 FIFO 를 이용하여 가비지 컬렉터가 Victim 을 선택할 경우 디스크 사용 범위(LBA Utilization)에 따른 전체 쓰기량을 나타낸다. 이때 SA, DA, TA 는 각각 다중 로그 지점의 수가 1, 2, 3 인 경우이다. 이 결과를 통해 우리는 로그 지점의 개수가 많은 것이 실

제로 디스크에 쓰여지는 양을 줄일 수 있다는 것을 확인할 수 있다. 특히 디스크를 넓게 쓰는 경우 그 이득이 더 크며, 이는 디스크를 처음부터 끝까지 채우게 되는 로그 기반 파일시스템에서 이러한 접근이 맞음을 보여준다.



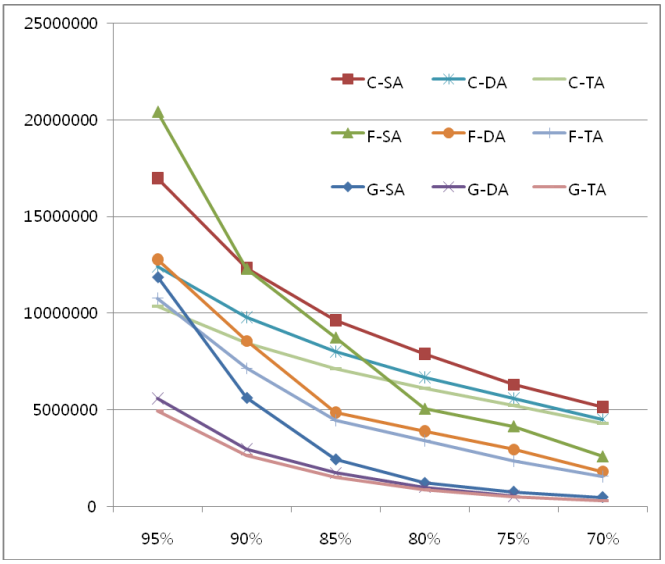

(그림 3) Greedy 를 이용한 가비지 컬렉션 정책에서의 쓰기 횟수.

그림 3 은 앞선 그림 2 에서 나타내고 있는 실험에서 Victim 선택을 위한 가비지 컬렉터의 정책을 Greedy 로 변경한 후의 결과를 보여준다. 앞서와 마찬가지로 다중 로그 지점이 더 효과적으로 쓰기를 줄여주고 있다는 것을 확인할 수 있다. 그리고 전체적인 쓰기량은 FIFO 보다 더 적음을 알 수 있다.




(그림 4) Cost-Benefit 를 이용한 가비지 컬렉션 정책에서의 쓰기 횟수.

그림 4 는 가비지 컬렉터의 정책을 Cost-Benefit[6]으로 했을 경우의 결과이다. 역시 마찬가지로 다중 로그 지점을 사용했을 경우 쓰기량이 줄어듬을 알 수 있다.




(그림 5) 여러 가비지 컬렉션 정책에서의 쓰기 횟수.

그림 5 는 앞의 결과들을 하나로 모은 그래프를 보여준다. 전체적으로 Greedy 정책을 이용했을 경우 성능이 좋음을 알 수 있다.

## 3. 추후 연구 과제

지금까지 살펴본 것에서 더 나아가 Greedy 정책보다 더 향상된 가비지 컬렉션 정책이 필요하다고 본다. 앞서 우리가 실험한 정책들은 기존의 로그 기반 파일시스템에서 실험되고 평가된 방법이기 때문에 새로운 다중 로그 지점을 활용할 수 있는 정책이 연구되고 검증되어야 한다. 그리고 다중 로그 지점을 사용함에 있어 어떤 종류의 데이터들끼리 모여서 한 지점에서 기록될지에 대한 연구가 필요하고 더 나아가 실제 이를 기반한 파일시스템의 개발이 요구된다.

## 4. 결론

SSD 는 HDD 와 달리 여러 지점에 동시에 쓰기를 하여도 자신의 설계에 따라 일정한 성능을 보장해 줌을 알 수 있다. 그에 따라 단지 덮어쓰기(overwrite)를 하지 않는다는 이유로 SSD 용 파일시스템으로 주목 받고 있는 로그 기반 파일시스템을 개선하여 동시에 여러 지점에서 세그먼트들이 기록되는 방법을 사용하게 되면 더 많은 성능 향상을 이끌 수 있으며 로그 기반 파일시스템의 약점인 가비지 컬렉션으로 인한 성능 하락도 줄일 수 있다. 특히 우리의 이러한 접근은 상대적으로 가격이 저렴한 블록 단위 사상을 사용하는 SSD 에서 더 큰 이득을 얻을 수 있기 때문에 물리적으로 많은 SSD 를 소비하는 대용량 스토리지 서버를 구축하거나 SSD 의 가장 큰 수요인 넷북/랩탑, 양 쪽 모두에서 유용한 파일시스템으로 쓰일 수 있다.

## 참고문헌

[1] L. Bouganim, B.Þ. Jónsson, and P. Bonnet. 2009. uFlip: Understanding flash IO patterns. *4th Biennial Conference on Innovative Data Systems Research (CIDR)* January 4-

7, 2009, Asilomar, California, USA.
[2] Mendel Rosenblum and John K. Ousterhout. 1992. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems (TOCS) Volume 10, Issue 1* (February 1992). 26-52.
[3] Dongjun Shin. 2008. *Linux Storage & Filesystem Workshop (LSF '08)*.
[4] Ryusuke Konishi, Yoshiji Amagai, Koji Sato, Hisashi Hifumi, Seiji Kihara, and Satoshi Moriai. 2006. *The Linux implementation of a log-structured file system. ACM SIGOPS Operating Systems Review Volume 40, Issue 3* (July 2006). 102-107.
[5] Hu, X., Eleftheriou, E., Haas, R., Iliadis, I., and Pletka, R. 2009. Write amplification analysis in flash-based solid state drives. In *Proceedings of SYSTOR 2009: the Israeli Experimental Systems Conference* (Haifa, Israel). SYSTOR '09. ACM, New York, NY, 1-9.
[6] A. Kawaguchi, S. Nishioka, and H. Motoda,\A Flash Memory based File System," *Proceedings of the USENIX Technical Conference*, 1995.