

Discard 명령의 F2FS 성능 영향

이인수[◦], 이재훈[◦], 이상원

성균관대학교

{insu301, wogns1602, swlee}@skku.edu

Effect of Discard Command on F2FS Performance

Insu Lee[◦], Jaehun Lee[◦], Sang-Won Lee

Sungkyunkwan University

요약

NAND 플래시메모리 기반 SSD (Solid State Drive)가 주류 저장장치로 자리 잡으면서 SSD 최적화된 파일시스템 및 응용들이 제안/발전되어 왔다. 특히, 대표적인 플래시친화 파일시스템 F2FS (Flash-Friendly File System)는 OLTP 서버 워크로드에서 Ext4/XFS 등 기존 파일시스템보다 나은 성능을 보이는 것으로 알려져 있다. 본 논문에서는 SSD와 F2FS상에서 RocksDB를 이용해서 YCSB 벤치마크 수행 시, 예상과 달리 in-place update 방식의 Ext4와 XFS 대비 성능이 상당히 저조한 점을 보이고, 원인 규명 및 해결책을 제시한다. 구체적으로, 유휴 시간(idle time)이 없이 동작하는 워크로드에서 F2FS가 discard 명령을 적절히 내리지 못하기 때문에 성능저하가 유발되는 현상을 설명하고, 벤치마크 수행시 유휴 시간을 적절하게 줌으로써 F2FS가 discard 명령을 활용해서 결과적으로 F2FS의 성능을 약 두 배까지 증가시킴을 보인다.

1 서론

SSD의 가격 하락으로 대용량 데이터 센터와 같은 다양한 분야에서 HDD가 SSD로 대체되며 그에 최적화된 파일 시스템 및 응용이 개발되어 왔다. F2FS 또한 플래시 친화적으로 설계된 파일 시스템으로, F2FS가 SSD 상의 OLTP 서버 워크로드에서 기존의 Ext4/XFS보다 뛰어난 성능을 낸다는 선행 연구가 있다 [1]. 그러나 RocksDB에서 YCSB 벤치마크로 파일 시스템 별 성능 측정을 수행한 결과, F2FS가 Ext4/XFS에 비해 저조한 성능을 보였다. 본 논문에서는 F2FS의 성능 저하 원인을 discard 명령의 부재로 상정하고 실험 결과를 제시, 분석하였다. YCSB 벤치마크 중 유휴 시간을 준 결과 F2FS가 정상적으로 discard 명령을 내릴 수 있었고, WAF(쓰기 증폭 정도)가 감소했다. 최종적으로 디스크 사용률 20%에서 F2FS의 성능이 OPS 기준 95% 향상되었다. 그와 동시에, 디스크 사용률이 높아질수록 더 짧은 유휴 주기가 요구되며, 그렇지 않을 시 Ext4/XFS에 비해 낮은 성능을 보이는 F2FS의 한계점을 확인했다.

2 배경

2.1 TRIM

SSD의 쓰기는 페이지 단위로 이루어지며, 덮어쓰기가 불가능하다. 즉 반드시 페이지가 삭제된 후에 쓰기가 가능하다. 그러나 쓰기와 달리 삭제는 블록 단위로 이루어지며 이러한 단위 차이로 인해 가비지콜렉션(Garbage Collection) 대상 블록에 유효한 페이지와 그렇지 않은 페이지가 섞여있을 수 있다. 이 때 유효한 페이지의 삭제를 방지하기 위해 해당 페이지를 다른 블록으로 복사하는

카피백(Copyback)이 필요하다. 즉, 호스트에서 요청한 쓰기 외의 추가적인 쓰기(쓰기 증폭)가 카피백으로 인해 발생하게 되며, 이는 SSD 성능 저하의 원인이다. 따라서 이를 최소화하기 위해 TRIM 명령어가 도입되었다. 파일 시스템에서 삭제 가능한 데이터에 대한 정보를 discard 명령 내리면, SSD가 해당 페이지를 TRIM 하고, 카피백을 수행하지 않는다. 따라서 불필요한 카피백이 줄어들고, SSD의 성능이 크게 개선될 수 있다 [2].

2.2 RocksDB

RocksDB는 SSD에 최적화된 Key-Value 형태의 로그 구조 데이터베이스 엔진이다 [3]. RocksDB의 데이터 저장 구조는 Log-Structured Merge Tree(LSM-tree)를 기반으로 한다. RocksDB는 쓰기 요청 시 메모리의 Active Memtable이란 이름의 임시 버퍼에 데이터를 쓴다. 해당 Memtable이 일정 크기가 되면 읽기 전용 Memtable이 되고, 일정 개수 이상의 Memtable이 모이면 저장 장치에 내려가(flush) SST 파일 형태로 저장된다. 따라서 RocksDB는 SST 파일 단위로(기본 64MB) I/O를 진행하고, 데이터를 append-only 로그에 저장함으로써 순차 쓰기를 보장한다.

2.3 F2FS

유닉스 계열의 파일 시스템은 FFS(Fast File System)와 LFS(Log-Structured File System)로 나뉜다. Ext4와 XFS는 주로 리눅스에서 사용되는 대표적인 FFS이며 디스크를 그룹으로 나누어 각 그룹마다 파일을 저장한다. FFS는 데이터 저장 시에 위치가 변하지 않고 공간 지역성을 따르는 in-place update 방식을 사용한다. 한편 F2FS는 플래시 메모리 친화적으로 설계된 LFS 기반의 파일시스템이다. LFS는 파일 시스템 내부의 모든 세그먼트를 로그 구조로 저장하며, 따라서 FFS에 비해 쓰기의 성능이 좋다. 그

*이 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2015-0-00314, 비휘발성 메모리 기반 고성능 DBMS 개발)

러나 계속 새로운 공간에 데이터를 쓰기 때문에 파일시스템 logical volume이 모두 소모되면 가비지콜렉션 요구되며, 가비지콜렉션 오버헤드가 파일 시스템의 성능에 많은 영향을 끼친다. 이러한 단점을 보완하기 위해 F2FS는 SSR(Slack Space Recycling)을 사용한다. SSR은 logical volume의 여유 공간이 5% 이하가 되어 가비지콜렉션이 요구될 때 F2FS가 로그 구조로 쓰지 않고, 빈 공간에 데이터를 할당하는 기법이다. 또한 F2FS는 최적의 성능을 위해 5초 이상의 유후 시간이 주어지면 discard 명령을 내린다 [4].

3 성능 평가

F2FS가 OLTP 서버 워크로드에서 성능이 우수하다는 선형 연구 검증을 위해 Ext4, XFS, F2FS 총 3개의 파일 시스템에서 성능 평가를 진행하였다. 그 후, F2FS의 성능 저하 원인 분석을 위해 10분의 벤치마크 수행마다 1분의 유후 시간을 주는 방식의 추가 실험을 진행하였다. 디스크 사용률에 따라 파일 시스템의 성능이 다를 수 있기 때문에, 250GB 중 Over-Provisioning(OP) 영역을 제외한 233GB 용량 기준 디스크 사용률 20%, 40%, 60%로 실험을 진행하였다. 성능 측정의 지표로는 SSD와 파일 시스템 사이의 WAF(Write Amplification Factor)를 1분 단위로 측정한 Run WAF, OPS(Operation Per Second)를 활용하였다. 벤치마크 툴로는 읽기 50%, 쓰기 50%를 수행하는 YCSB의 워크로드 A를 사용하였으며, 각 실험마다 벤치마크를 2시간 수행하였다.

3.1 실험 환경

표 1: 실험 환경

타입	사양
OS	Ubuntu 18.04.6 LTS
CPU	Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz
Memory	1.56TB
Kernel	Linux 5.4.0-84-generic
Data Device	CT250 MX500 SSD

본 논문에서 저장장치로 사용한 SSD는 마이크론 Crucial MX500 CT250 이다. 자세한 실험 환경은 [표 1]과 같다.

3.2 파일 시스템 별 RocksDB 성능 평가

SSD상의 OLTP 서버 워크로드에서 F2FS가 Ext4/XFS 대비 어느 정도의 성능 차이를 보이는지 확인하기 위하여 각각의 파일 시스템에서 RocksDB를 이용하여 성능 평가를 진행 하였다. 실험 결과는 [표 2]와 같다.

RocksDB는 SST 파일 단위로 I/O를 진행하기 때문에 F2FS의 logical volume에 SSR을 할 작은 빈 공간이 발생하지 않는다. 이에 따라 F2FS가 SSR 기능을 수행하지 않는 것을 확인 하였다. [표 2]에서 보여지듯이, 모든 디스크 사용률에서 F2FS의 평균 OPS가

표 2: 디스크 사용률에 따른 파일시스템별 YCSB OPS

디스크 사용률	Ext4	XFS	F2FS
20%	52.4K	46.0K	27.4K
40%	43.6K	40.0K	23.1K
60%	29.9K	28.2K	17.5K

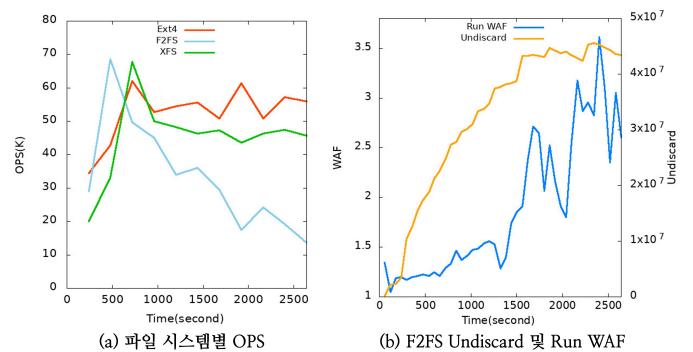


그림 1: 2600초까지의 OPS, Undiscard, Run WAF (디스크 사용률 20%)

Ext4와 XFS에 비해 매우 낮았다. 한가지 주목할 만한 점은 디스크 사용률 20%에서 [그림 1]과 같이 F2FS의 OPS가 대략 500초부터 2000초 까지 급격하게 감소 해 다시 복구하지 못한다는 것이다. OPS뿐만 아니라 Run WAF도 동시간대에 급격하게 증가해 2000초 부근에서 Ext4/XFS 대비 2배 이상을 기록 했다. 다른 디스크 사용률에서도 증감 시간대만 다를 뿐 똑같은 패턴을 보여주었다. F2FS에서 높은 쓰기 증폭의 원인을 파악하기 위해 blktrace 로그를 분석한 결과, 벤치마크 수행 중 지속적으로 discard 명령을 내리는 Ext4 및 XFS와 달리, F2FS는 discard 명령을 전혀 내리지 않는 점을 발견하였다. 이로 인해 Undiscard 지표가 급격히 증가하여 최대 치값으로 수렴한 후 지속하게 된다. F2FS에서 Undiscard 지표는 무효화된 페이지중 어떤 이유로 discard되지 못한 블록(4KB) 수를 나타내는데, 이러한 블록들은 SSD 내부에서 가비지콜렉션 시 불필요한 카피백을 유발하게 된다. 따라서, SSD의 Run WAF도 비례해서 증가하게 되고, 궁극적으로 OPS는 급격히 감소하게 된다.

3.3 F2FS 유후 시간 실험 결과

위 분석 결과에 따라 F2FS의 discard 명령이 정상적으로 내려가지 않은 것을 성능 저하의 주 원인으로 추정하고 이 문제점을 개선하기 위해 YCSB 수행 중 유후 시간을 주는 방식의 추가 실험을 진행하였다.

F2FS 기본 설정에서는 5초 이상의 유후 시간이 주어질 때 discard 명령을 내린다. 따라서 RocksDB 로그와 Memtable이 모두 디스크로 내려가기까지의 여유 시간을 고려하여, YCSB 프로세스의 10분 실행마다 정지 시그널(SIGSTOP)을 주어 1분 정지 하였다. 유후 시간의 OPS가 0으로 계산되어 Ext4/XFS는 평균 OPS가

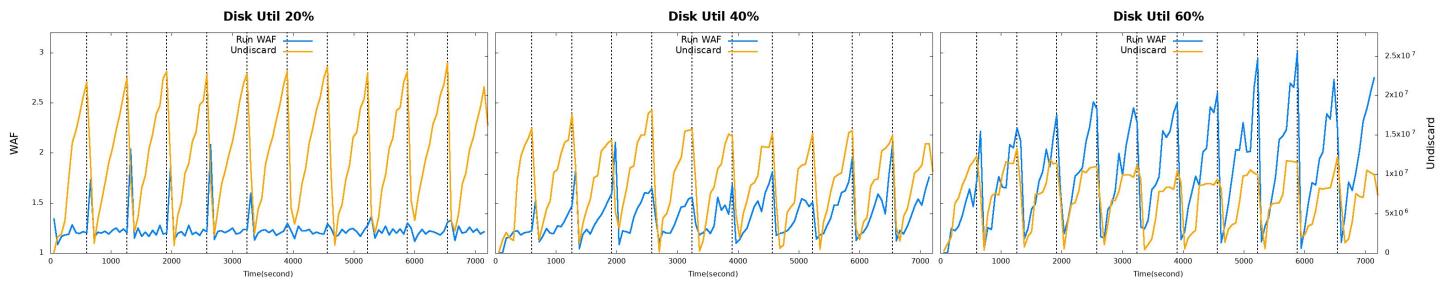


그림 2: 디스크 사용률에 따른 F2FS Undiscard 및 Run WAF

오히려 감소한 경우도 있었지만 F2FS는 그럼에도 불구하고 기존의 실험 결과 대비 월등한 성능을 보였다. [그림 2]에서 점선으로 표시된 시점에 유휴 시간을 제공했다. 기존에 유휴 시간을 주지 않은 경우, F2FS가 discard 명령을 내리지 못하여 Run WAF가 증가하고 OPS가 감소 후 회복하지 못하였다. 반면 유휴 시간을 준 경우 F2FS가 discard 명령을 내려 Undiscard 지표가 감소하고, 이에 따라 Run WAF 또한 감소하였다. 디스크 사용률이 낮을 때는 SSD에 free 블록과 무효 페이지가 상대적으로 많고, 따라서 Undiscard 지표의 최대치가 높다. 하지만 디스크 사용률이 높아질수록 SSD에 free 블록과 무효 페이지가 상대적으로 적어져 Undiscard 지표의 최대치가 낮아지게 된다.

로그 구조를 사용하는 F2FS의 특성상 SSD의 블록을 소모하는 속도가 빠르고, 디스크 사용률이 높아질 수록 free 블록과 무효 페이지가 많은 블록이 적어, 단기간에 카피백 비용이 큰 블록만 남게 된다. 그 결과 [그림 2]에서 보이듯 Run WAF가 더욱 이른 시점에, 많이 증가한다.

주기가 짧아져야 최적의 성능 유지가 가능하다는 것을 확인하였다. 따라서 디스크 사용률에 따라 유휴 시간의 주기 및 길이를 조정하여 성능의 개선이 가능할 것이다.

4 결론

본 논문에서는 파일 시스템 별 RocksDB 성능 평가 결과 F2FS의 성능이 Ext4, XFS에 비해 상당히 저조한 것을 확인하였다. 이에 F2FS의 성능 저하 원인을 discard의 부재로 추정하고, YCSB 벤치마크 중 유휴 시간을 주어 성능 향상을 확인하였다.

실험 결과 F2FS에 충분한 유휴 시간을 준다면, 유휴 시간이 없을 때에 비해 큰 성능 개선을 보였다. 이를 통해 파일 시스템의 적절하지 않은 discard 정책이 SSD의 불필요한 카피백을 늘릴 수 있고 결국 DBMS의 성능에 큰 영향을 미치는 것을 확인 했다. 또한 유휴 시간 제공 시, 낮은 디스크 사용률에서는 Ext4 및 XFS와 비슷한 성능을 보였으나, 디스크 사용률이 높아질수록 Ext4 및 XFS에 비해 낮은 성능을 보였다. 즉, 높은 디스크 사용률과 끊임없는 I/O가 요구되는 OLTP 서버 워크로드에서의 F2FS는 최적의 성능을 내지 못할 수 있다. 따라서 DBMS와 같은 응용에서의 F2FS 사용을 위해서는 F2FS discard 정책의 변경이 요구된다.

향후 연구에서는 로그 구조인 RocksDB뿐만 아니라 in-place update 방식을 사용하는 MySQL 등 다른 DBMS에서의 F2FS의 discard 정책으로 인한 성능 저하를 분석할 예정이다. 더 나아가 F2FS가 유휴 시간이 없는 워크로드에서도 안정적인 성능을 낼 수 있도록 적절한 discard 정책을 연구할 예정이다.

참고 문헌

- [1] C. Lee, D. Sim, J. Hwang, and S. Cho, “F2FS: A new file system for flash storage,” in *13th USENIX Conference on File and Storage Technologies (FAST 15)*, (Santa Clara, CA), pp. 273–286, USENIX Association, Feb. 2015.
- [2] C. Hyun, J. Choi, D. Lee, and S. H. Noh, “To trim or not to trim : Judicious trimming for solid state drives,” 2000.
- [3] Rocksdb, <https://rocksdb.org/>.
- [4] F2FS Documentation, <https://www.kernel.org/doc/Documentation/filesystems/f2fs.txt>.

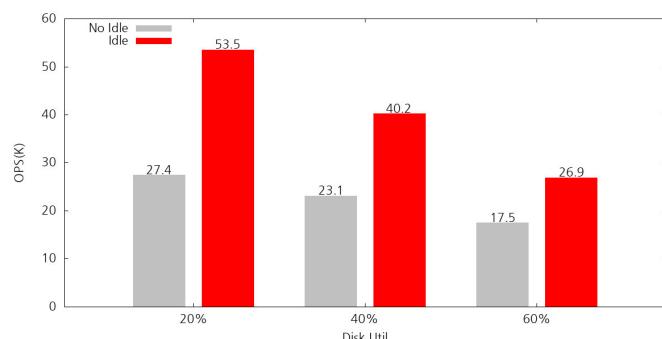


그림 3: 디스크 사용률 별 유휴 시간 전/후 F2FS RocksDB OPS

[그림 3]에서 보이듯, F2FS에 유휴 시간 제공 시 모든 디스크 사용률에서 성능의 개선을 보였다. 따라서 5초 이상의 유휴 시간이 있어야만 discard 명령을 내리는 F2FS의 정책이 Rocksdb에서 성능 하락의 주 요인임을 확인하였다. 또한 디스크 사용률이 20%인 경우에는 유휴 시간을 주지 않은 경우에 비해 95%의 성능 향상이 있었지만, 디스크 사용률이 60%인 경우, 53%의 성능 향상밖에 이 루지 못하였고 결국 Ext4 및 XFS에 비해 저조한 성능을 보였다. 이를 통해 디스크 사용률이 높아질수록 F2FS의 discard 주기, 즉 유휴