2B-LFS: Block logging과 byte logging이 가능한 log-structured file system

곽현호°, 신동군
성균관대학교 전자전기컴퓨터공학과
gusghrhkr@skku.edu, dongkun@skku.edu

2B-LFS: Block and Byte level Log-structured File System

Hyunho Gwak°, Dongkun Shin
Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

Byte-addressable SSD는 SSD 내부 battery-backed DRAM을 사용하여 byte 단위 접근이 가능한 persistent memory를 제공한다. 따라서, byte 접근이 가능하면서도 SSD의 큰 용량을 활용할 수 있는 독특한 특성을 가지고 있다. 그런데, SSD 내부에 존재하기 때문에 접근 가능한 persistent memory 영역 크기가 제한 되어있다. 본 논문에서는 이와 같은 byte-addressable SSD의 특성을 고려하여 작은 크기의 persistent memory를 효과적으로 활용 가능한 2B-LFS를 제안한다. 2B-LFS는 block log 뿐만 아니라 byte 크기의 log까지 SSD의 flash memory에 기록하는 것을 지원한다. 2B-LFS는 SSD 내부 persistent memory 크기보다 더 큰 크기의 byte log를 기록하여 byte-addressable SSD를 효과적으로 사용한다.

1. 서 론

Solid-State-Drives (SSDs)는 기존 주 저장장치로 사용되는 Hard disk drives (HDDSs) 보다는 높은 I/O 성능을 가지며, main memory로 사용되는 DRAM보다 낮은 가격을 가지고 있다. 이러한 특성은 SSD가 HDD를 대체하는 주 저장장치로 사용되게 하거나, DRAM과 HDD 사이의 cache로써 사용되게 한다. 하지만, SSD는 DRAM과 달리 block 단위 I/O를 수행해야만 하는 제약이 있으며 이로 인해 불필요한 I/O가 발생한다.

최근 연구들에서 제안되는 byte-addressable SSD [1, 2, 3]는 SSD의 block 단위 I/O 제약으로 인판 불필요한 I/O 문제를 해결할 수 있다. Byte-addressable SSD는 SSD 내부의 persistent memory를 사용하여, host에게 byte 단위 기록이 가능한 memory 영역을 제공한다. Byte-addressable SSD는 PCM, STT-RAM, and 3D-Xpoint같은 Non Volatile Memory (NVM)와는 달리 SSD 내부의 memory를 활용하므로 추가적인 DIMM slot 또는 PCIe slot을 요구하지 않고, 더 낮은 비용으로 byte-addressability를 활용할 수 있는 저장장치이다. 하지만, SSD 내부에 존재해야 하는 persistent memory는 사용 가능한 크기에 제한이 있고, 이전에 제안되었던 NVM file system과 같은 NVM 기법들을 충분히 활용할 수 없다. 따라서 이전 byte-addressable SSD 연구들에서는 file system의 metadata나 DB의 journal file등 제한된 크기만을 persistent memory에 할당하여 사용하고 있다.

본 논문에서는 byte-addressable SSD의 독특한 특성을 충분히 활용할 수 있는 2B-LFS를 제안한다. 2B-LFS는 작은 크기의 NVM만을 사용하여 byte 단위의 *delta log*가 block 단위 접근이 가능한 SSD에 기록되게 하는 file system이다. 따라서, 2B-LFS는 작은 크기의 persistent memory를 가진 byte-addressable SSD 환경에서 효과적으로 사용될 수 있다. 2B-LFS는 내부 persistent memory를 활용하여 flash memory에 delta log를 기록하고, 제한된 크기의 persistent memory보다 더 큰 크기의 delta log를 기록할 수 있다.

그런데, file system에서 이와 같은 delta logging을 지원하기 위해서는 다음 세 가지 문제를 해결해야 한다. 첫 번째 문제는, delta log에 대한 read 요청이다. Block 단위 I/O로 접근이 가능한 SSD에서 byte 단위의 delta log를 read하는 문제가 발생한다. 두 번째 문제는, delta log에 대한 cleaning이다. LFS의 block log와 마찬가지로 delta log가 기록될 공간을 확보하기 위하여 cleaning을 수행해야 한다. 그런데, block 단위로만 접근이 가능한 SSD에서 byte 단위의 delta log를 copy하기 위해 불필요한 I/O가 발생한다. 마지막으로, File system은 기록된 delta log에 대한 recovery를 보장해야 한다. Recovery를 위해서는 기록된 delta log에 대한 ordering이 보장되어야 한다. 그러나, byte 단위로 관리되는 delta log에 대한 ordering을 보장하기 위해서는 byte 단위의 log를 추적하는 metadata가 추가되어야 한다.

우리는 적은 overhead로 앞선 문제들을 해결 가능하도록 2B-LFS를 설계하였다. 2B-LFS는 delta log로 기록된 delta logged page에 대하여, *Delta Logged Clean Writeback* (DLCW) policy를 사용한다. DLCW는 DRAM page cache의 delta logged page에 대한 eviction policy로, delta logged page 중 clean page들은 eviction 과정에서 writeback을 수행하는 policy이다. 이때, writeback을 기록되는 clean page들은 모두 block log로 기록함으로써 user의 read request는 DRAM 또는 SSD에서 하나의 block read만으로 처리 가능하다. 또한, DLCW policy를

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.IITP-2017-0-00914, 지능형 IoT 장치용 소프트웨어 프레임워크)

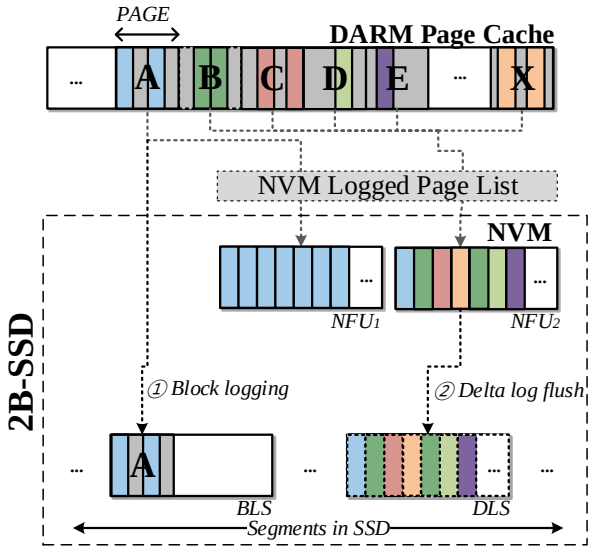


Figure 1 Design of 2B-LFS

사용함으로써 delta log를 cleaning을 위해 delta log에 대한 read가 발생하지 않는다. 대신, DRAM에 존재하는 delta logged page를 기록함으로써 cleaning을 수행할 수 있다. 그 외에, 2B-LFS는 기록되는 모든 delta log에 각 log의 page ID를 저장하는 delta log header를 같이 기록하고 이를 기반으로 recovery를 수행한다. Delta log header는 page ID를 포함하여 16B가 기록되기 때문에 큰 overhead 없이 관리할 수 있다.

우리는 실제 device, 2B-SSD [2]를 사용하여 2B-LFS를 구현하고 실험하였다. 우리는 2B-SSD의 8MB 크기의 persistent memory를 활용하여 실험하였으며, 실험 결과 기존 file system보다 최대 162%까지 성능을 증가시켰다.

2. 배경 지식과 관련 연구

2.1 Byte-addressable SSD

PebbleSSD [1], 2B-SSD [2], 그리고 FlatFlash [3]에서 제안하는 byte-addressable SSD는 SSD 내부 battery-backed DRAM을 사용하여 byte 단위 접근이 가능하게 한다. SSD 내부 memory를 활용하는 구조이기 때문에, 다른 저장장치와는 다른 독특한 특성을 가지고 있다. Byte-addressable SSD는 byte 단위 접근이 가능하면서도, NVM보다는 낮은 가격을 가지고 있다. 하지만, SSD 내부 memory 크기의 한계로 byte 단위 접근이 가능한 영역에 크기 제한이 있다. 따라서, byte-addressable SSD를 사용하는 연구들에서는 자주, 작은 크기로 update되는 file system이나 FTL의 metadata 또는 DB의 journal file을 SSD 내부 battery-backed DRAM에 저장하는 것으로 byte-addressable SSD를 활용하고 있다.

2.2 Heterogeneous storage system

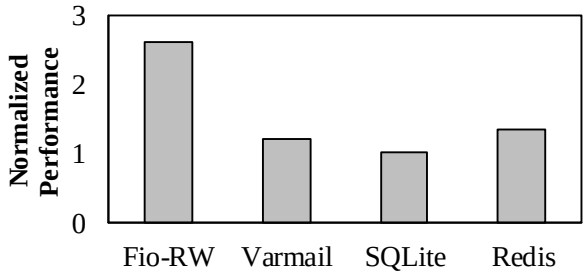
Strata [4], 그리고 Ziggurat [5]와 같은 heterogeneous storage system 에서는 NVM을 update log를 기록하기 위한 log 공간 또는 cache로 사용한다. 하지만, 이와 같은 기법들에서는 제한된 크기의 NVM만을 사용하며 NVM 공간이 부족할 경우 disk로 data를 옮기는 migration을 수행한다. Migration은 NVM에서 SSD로 data를 copy하는 overhead가 발생하는데, SSD에 block 단위로 기록하는 제약으로 인해 block log와 update log를 merge하는 overhead도 발생한다. 따라서, NVM 공간 부족으로 인한 migration은 추가 I/O를 유발시키고 NVM 크기가 작아질수록 성능이 감소된다. 그런데, byte-addressable SSD에서는 byte 접근이 가능한 persistent memory 크기에 제한이 있다. 따라서, heterogeneous storage system을 그대로 적용할 경우 잦은 migration이 유발되어 성능이 감소될 수 있다. 본 논문에서 제안하는 2B-LFS는 byte 단위의 update log, delta log를 merge 동작 없이 SSD에 기록하는 것을 지원한다. 이 과정에서 NVM은 수 MB의 작은 크기만을 사용하여 delta log를 block 단위로 SSD에 기록하기 위한 buffer로 사용된다.

3. 2B-LFS

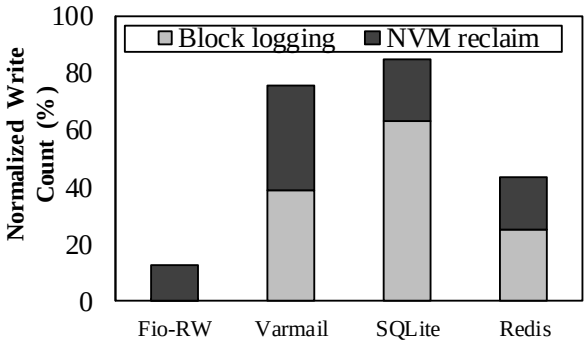
2B-LFS는 user의 write request를 block logging 뿐만 아니라 byte logging으로로 처리 가능한 LFS이다. 또한, 다른 heterogeneous storage system [4, 5]과 달리, 2B-LFS는 byte 단위 update log, delta log를 NVM 뿐만 아니라 block storage, SSD까지 기록한다. 이때 NVM은 delta logging이 가능한 write buffer로 사용하기 때문에 작은 크기의 NVM로도 동작이 가능하며, 내부 persistent memory 크기에 제한이 있는 byte-addressable SSD를 효과적으로 활용할 수 있다.

2B-LFS는 그림 1과 같이 *delta log flush*를 사용하여 NVM reclaim을 수행하고 NVM의 여유 공간을 확보한다. Delta log flush 동작은 NVM에 기록된 delta log를 NVM Flush Unit (NFU) 단위로 SSD에 기록한다. NFU의 크기는 SSD의 bandwidth를 활용할 수 있도록 충분한 크기를 사용하며, NFU가 SSD에 기록될 때 block 단위로 기록되기 때문에 delta log를 기록하기 위한 불필요한 쓰기가 발생하지 않는다. 그런데, 2B-LFS에서는 delta log flush 대신 block logging을 사용하여 더 효율적으로 NVM 여유 공간을 확보할 수 있다. 그림 1의 예와 같이, 특정 page A에 대한 delta log가 NFU 공간 대부분을 차지할 경우 NFU의 모든 delta log를 기록하는 대신 page A를 block log로 기록하는 것으로 여유 NVM 공간을 확보할 수 있다. Page A의 최신 data는 DRAM에서 유지되기 때문에 block log로 기록하는 것으로 이전에 기록되었던 page A의 old log는 무효화할 수 있다. 이 기법을 고려하여, 2B-LFS는 두 가지 NVM reclaim 방식의 cost를 계산하여 선택적으로 사용한다. 2B-LFS는 그림 1의 *NVM Logged Page List*를 DRAM에서 유지하며 NFU마다 기록되어 있는 page의 수를 관리한다. NVM Logged Page List는 delta log와 같이 기록되는 delta log header를 기반으로 recovery 가능한 structure이므로 DRAM에서만 유지한다. 이를 기반으로 block logging 방식의 cost를 page 수 X page 크기로 계산할 수 있다. 또한, delta log flush의 cost는 NFU 크기로 계산할 수 있다.

2B-LFS는 delta log를 SSD에 기록할 때 block log가 기록되는 block log segment (BLS)와 구분되는 delta log segment (DLS)에 저장한다. Segment는 LFS의 cleaning 단위이며 block log과 delta log의 cleaning 방식이 다르기 때문에 2B-LFS는 BLS와 DLS를 구분하여 관리한다. BLS를 cleaning 하기 위해서는 LFS의 copy and compact 방식을 사용한다. 반면



(a) Normalized write performance



(b) Normalized write traffic of SSD

Figure 2 Experiments result

DLS cleaning을 위해서는 DLS에 기록된 delta log의 page를 모두 block log로 기록하는 방법을 사용한다. 이 과정에서 block log로 기록해야 하는 page는 2B-LFS의 DLCW policy로 인해 DRAM에 존재하므로, delta log를 read하기 위한 추가 I/O가 발생하지 않는다.

모든 delta log는 page ID를 포함한 delta log header와 같이 기록된다. NVM에 먼저 기록되는 delta log header는 delta log flush 동작으로 SSD까지 저장된다. 또한, 2B-LFS는 DLS마다 다음 DLS의 주소를 저장함으로써 NVM에 기록되는 delta log 뿐만 아니라 SSD에 기록되는 delta log들에 대해서도 ordering을 보장한다. 2B-LFS는 recovery 과정에서 순차적으로 delta log header를 읽음으로써 recovery를 수행한다.

4. 실 험

4.1 실험 환경

2B-LFS는 linux kernel 4.15에서 구현되었다. F2FS를 기본 code로 사용하였으며, block logging만을 지원하는 F2FS에 byte logging 이 가능하도록 구현하였다. 또한, 2B-LFS는 실제 device, 2B-SSD를 활용하여 구현 및 실험을 진행하였다. 2B-SSD는 내부의 persistent memory를 활용하기 위한 API를 제공하므로, 이를 사용하여 2B-SSD byte logging이 가능하도록 구현하였다.

2B-LFS 실험을 위해서, Fio, Varmail, SQLite 그리고 Redis workload를 사용하였다. Fio는 1GB 크기 file에 대하여 1KB update와 fsync를 반복하는 것으로 delta logging을 적용하기 유리한 workload이다. Varmail과 SQLite 실험은 각각 filebench와 mobibench를 사용하여 진행하였다. Fio와 달리 큰 크기의 write가 발생하므로 2B-LFS는 block logging과 delta logging을 선택적으로 사용한다. 마지막으로 redis는 YCSB의 A workload를 사용하여 실험하였다. Redis 설정에서 journal file을 기록하도록 설정한 후 실험하였다.

4.2 실험 결과

그림 2는 4 가지 workload에서 측정한 성능과 write 양을 비교한 것이다. 각 수치는 block logging만을 사용하는 original LFS 수치에 normalization해서 나타낸 것이다. 그림 2의 (a)에서 확인할 수 있는 것처럼, 2B-LFS를 사용함으로써 성능이 최대 162%까지 증가하는 것을 확인하였다. 또한, 그림 2의 (b)에서는 각 workload에서 2B-LFS를 사용함으로써 감소되는 write 양을 확인할 수 있다. NVM reclaim은 NVM에 delta log가 가득 차서 NVM 공간 확보를 위해 SSD에 발생하는 write 양을 측정한 것이다. 2B-LFS는 그림 1에서 설명한 delta log flush를 사용하여 block logging으로 기록하는 것보다 더 적게 write를 발생시킬 수 있다. 실험 결과, update와 fsync를 반복하는 fio workload에서 write 양이 최대 8배까지 감소하였다. 이때, 4KB page에서 1KB update만을 기록하여 write 양이 4배로 감소되는 것이 아니라, file system에서 발생하는 metadata까지 delta logging이 적용되므로 write 양을 더 감소시킬 수 있었다.

5. 결론

우리는 byte-addressable SSD의 특성을 고려하여, 제한된 크기의 NVM을 효과적으로 사용할 수 있는 2B-LFS를 제안하였다. 2B-LFS는 byte 단위 log를 효과적으로 관리할 수 있는 file system 구조를 제안한다. 실제 device인 2B-SSD [2]를 사용하여 실험한 결과, 성능을 최대 162%까지 증가시킬 수 있었다.

참 고 문 헌

[1] Jin, Yanqin, et al. “Improving SSD lifetime with byte-addressable metadata.” Proceedings of the International Symposium on Memory Systems. ACM, 2017.

[2] Bae, Duck-Ho, et al. “2B-SSD: the case for dual, byte- and block-addressable solid-state drives.“ 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2018.

[3] Abulila, Ahmed, et al. “FlatFlash: Exploiting the Byte-Accessibility of SSDs within A Unified Memory-Storage Hierarchy.” Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. ACM, 2019.

[4] Kwon, Youngjin, et al. “Strata: A cross media file system.” Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 2017.

[5] Zheng, Shengan, Morteza Hoseinzadeh, and Steven Swanson. “Ziggurat: A Tiered File System for Non-Volatile Main Memories and Disks.“ 17th USENIX Conference on File and Storage Technologies (FAST 19) 2019.