The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 23, No. 3, pp.107-111, Jun. 30, 2023. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2023.23.3.107
JIIBC 2023-3-15

인메모리 파일시스템을 위한 효율적인 메타데이터 저널링 기법

An Efficient Metadata Journaling Scheme for In-memory File Systems

반효경*

Hyokyung Bahn*

요 약 저널링 기법은 파일시스템을 크래쉬 상황으로부터 보호하여 일관성 있는 상태로 유지하기 위해 널리 사용되고 있다. 한편, 기존의 저널링 기법들은 하드디스크와 같은 블록 스토리지를 위해 설계되었기 때문에 바이트 단위 접근이 가능한 영속 메모리 상에서의 저널링에 활용하기에는 비효율적이다. 본 논문은 크래쉬 상황으로부터 파일시스템의 일관성이 깨어지는 것을 방지하는 기능을 가진 메타데이터 저널링 기법을 인메모리 파일시스템에 기반해 설계하는 방법을 제안한다. 제안하는 기법은 바이트 단위 접근이 가능한 메모리 미디어의 특성을 활용하여 저널링이 발생시키는 많은 쓰기량을 줄일뿐 아니라 입출력 시 통과해야 하는 무거운 소프트웨어 스택을 제거하는 장점을 가진다. IOzone 벤치마크를 이용한 성능 측정 실험을 통해 제안하는 저널링 기법이 Ext4의 저널링과 비교해서 평균 49.2%의 성능 개선 효과가 있음을 보인다.

Abstract Journaling techniques are widely used to maintain a consistent file system state in the event of a system crash. As existing journaling techniques are designed for block storage such as HDDs, they are not efficient for byte-addressable persistent memory media. This paper proposes a metadata journaling technique for in-memory file systems that has the ability of avoiding inconsistent file system states in crash situations. The proposed journaling technique reduces a large amount of writing by making use of the byte-addressable feature of memory media and bypasses heavy software I/O stack. Experimental results with the IOzone benchmark show that the proposed journaling technique improves the performance of Ext4 by 49.2% on average.

Key Words : File System, Journaling, Metadata, Persistent memory, In-memory file system.

Ⅰ. 서 론

파일시스템의 설계에서 데이터의 신뢰성 보장은 매우 중요한 고려 사항 중 하나이다[1, 2]. 특히 배터리 기반의 모바일 기기에서는 갑작스런 전원 오류가 발생할 수 있으며, 시스템 크래쉬 발생 시의 데이터 복구는 반드시 필요한 핵심 기능 중 하나이다. 저널링은 Ext4와 같은 파일 시스템에서 이러한 데이터의 신뢰성 확보를 위해 사용되는 기술이다[3, 4]. 저널링은 수정된 데이터를 저널 영역이라 불리는 스토리지 상의 위치에 먼저 기록한 후 파

*정회원, 이화여자대학교 컴퓨터공학과
접수일자 2023년 5월 14일, 수정완료 2023년 5월 30일
게재확정일자 2023년 6월 9일

Received: 14 May, 2023 / Revised: 30 May, 2023 /
Accepted: 9 June, 2023
*Corresponding Author: bahn@ewha.ac.kr
Dept. of Computer Engineering, Ewha University, Korea

일 시스템의 원위치에 나중에 반영한다. 저널 영역에 수정 데이터를 먼저 기록할 경우 시스템 크래쉬 상황에서 유의미한 데이터가 저널 영역과 파일시스템 영역 중 적어도 한 곳에는 존재하므로 데이터가 유실되지 않도록 보호하는 역할을 한다. 이와 달리 데이터를 파일시스템의 원위치에 직접 수정할 경우 전원 오류 시 수정 내용 중 일부만 반영되어 예전 데이터도 수정된 데이터도 아닌 일관성이 깨어진 상태에 이를 가능성이 있다[4, 5].

저널링은 파일시스템의 신뢰성을 높이는 역할을 하지만 스토리지의 쓰기량을 늘려 성능을 저하시킨다. 하드디스크의 경우 쓰기량보다 헤드의 이동에 성능이 크게 좌우되므로 저널링의 오버헤드는 상대적으로 낮은 편이다. 그러나, 성능이 데이터의 쓰기량에 의존적인 저장 매체에서는 이러한 저널링의 오버헤드를 줄이는 것이 중요하다. 본 논문에서는 바이트 단위 접근이 가능한 영속 메모리(byte-addressable persistent memory)를 사용하는 인메모리 파일시스템을 위한 저널링 기법을 제안한다. 영속 메모리에서는 전통적인 블록 디바이스인 하드디스크를 위해 설계된 저널링 기법을 적용하기에 2가지 측면에서 비효율적이다. 첫째, 블록 디바이스용 저널링 기술을 영속 메모리에 적용할 경우 1바이트가 변경되더라도 블록 전체를 저널링해야 한다. 대부분의 파일시스템에서 사용하는 저널링의 기본 세팅은 작은 수의 바이트만을 저널링하는 메타데이터 저널링이므로 바이트 단위 접근이 가능한 영속 메모리에서는 상당한 비효율성을 초래한다. 둘째, 전통적인 블록 디바이스에서 I/O 요청이 발생할 경우 스토리지 디바이스의 접근을 위해 일련의 소프트웨어 스택을 통과해야 하는 오버헤드가 뒤따른다. 영속 메모리는 하드웨어의 접근 속도가 빠르므로 이러한 소프트웨어 스택의 상대적인 오버헤드가 훨씬 증가한다. 일부 연구에 의하면 하드디스크의 경우 이러한 소프트웨어 스택의 오버헤드가 전체 스토리지 접근 시간의 1% 정도로 미미하나 영속 메모리에서는 90% 이상을 차지할 수 있다는 분석이 있다[6].

본 논문에서는 영속 메모리 상에 존재하는 파일시스템에 저널링 기능을 어떻게 효율적으로 구현할 수 있는지에 대해 기술한다. 영속 메모리를 블록 디바이스로 간주하여 기존의 스토리지용 저널링 기법을 적용할 경우 매체 자체의 고속 접근 성질은 활용할 수 있으나 바이트 단위 접근 특성을 활용할 수 없으며, 소프트웨어 스택을 통과해야 하는 오버헤드가 뒤따른다[7, 8]. 이러한 오버헤드를 없앨 수 있는 방식으로 인메모리 파일시스템이 있다. 인메모리 파일시스템은 메모리 상의 버퍼 캐쉬 일부를 파일시스템으로 매핑해서 사용하는 방식으로 바이트 단위 접근이 가능하고 입출력을 위한 일련의 소프트웨어 스택을 통과하지 않는 효율성을 얻을 수 있다. 그러나, 인메모리 파일시스템은 전원이 나갈 때 내용이 사라지는 휘발성 메모리에 존재하는 임시 파일시스템으로 데이터의 신뢰성뿐 아니라 영속성을 제공하지 않는다. 물론 매체가 휘발성 D램에서 영속 메모리로 변경될 경우 전원이 나가더라도 내용이 유지될 수는 있다. 그러나, 시스템 크래쉬 발생 시 파일시스템 자체가 깨어져서 복구가 불가능해질 수 있다는 문제점이 있다.

본 논문에서는 이러한 문제점을 해결하여 인메모리 파일시스템에 저널링 기능을 추가하여 바이트 단위 접근성과 소프트웨어 스택의 오버헤드를 제거하면서 크래쉬 발생 시 복구가 가능한 높은 신뢰성을 제공하고자 한다. 특히, 본 논문에서는 기존의 저널링 파일시스템 대부분이 기본 세팅으로 제공하는 메타데이터 저널링에 집중하여 메타데이터 수정본을 영속 메모리 상의 저널 영역에 바이트 단위로 기록할 수 있도록 한다. IOzone 벤치마크를 이용한 실측 실험을 통한 성능 평가 결과 제안하는 저널링 기법의 성능이 동일한 영속 메모리 매체에 Ext4 파일시스템을 설치했을 때보다 평균 49.2%의 성능 개선이 있음을 보인다.

본 논문의 이후 구성은 다음과 같다. II장에서는 제안하는 메타데이터 저널링 기법에 대해 설명한다. III장에서는 실험을 통해 제안하는 저널링 기법의 성능을 검증한다. 끝으로 IV장에서는 본 논문의 결론을 제시한다.

II. 제안하는 메타데이터 저널링 기법

본 장에서는 본 논문이 제안하는 인메모리 파일시스템의 저널링 기술에 대해 기술한다. 제안하는 기술은 전통적인 파일시스템 설계와 달리 메모리 주소 공간의 일부를 파일시스템 트리에 직접 매핑해서 사용하는 방식으로 그림 1과 같이 버퍼 캐쉬 계층을 사용하지 않는다. 이러한 파일시스템 구조는 읽기 쓰기 연산 시 소프트웨어 스택을 통과하지 않고 불필요한 데이터 복제 오버헤드를 없애는 장점을 가진다. 또한, 파일시스템의 접근을 블록 단위로 해야 하던 전통적인 방식의 비효율성을 해소할 수 있다. 그러나, 이러한 파일시스템 구조에서 파일을 기록하는 도중에 전원 오류 발생 시 데이터의 일관성이 깨어질 수 있다.

이러한 문제를 해결하기 위해 저널링 기법이 널리 사

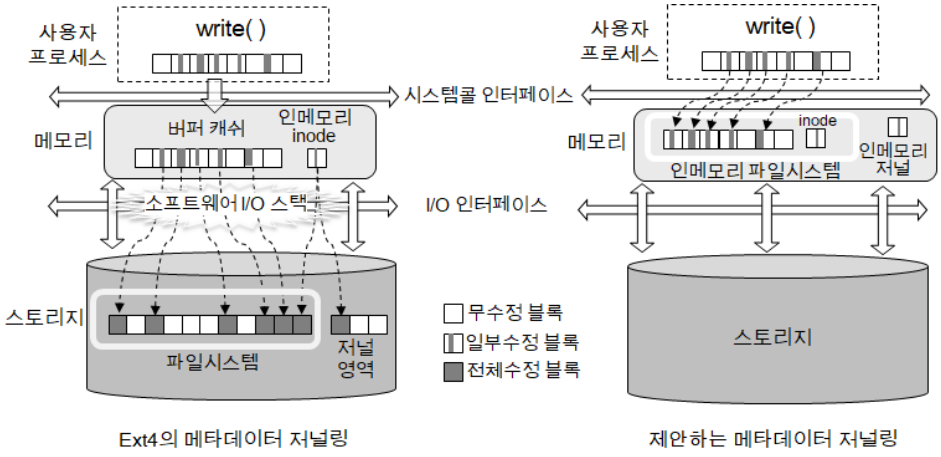


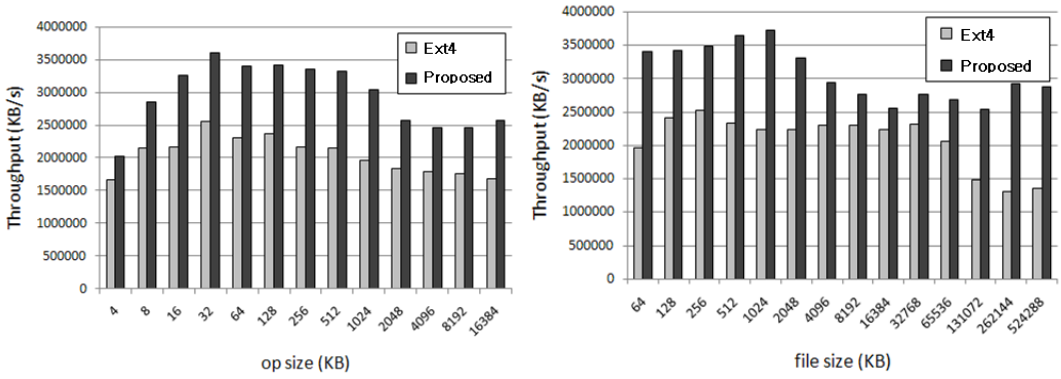
그림 1. 기존의 저널링 기법과 제안하는 저널링 기법의 비교
Fig. 1. Comparison of the existing and the proposed journaling techniques.

용되고 있으나 이는 기존의 블록 디바이스용 파일시스템에서 사용하던 방식으로 인메모리 파일시스템의 저널링에 직접 사용하는 것이 불가능하다. 본 논문에서는 기존의 저널링 파일시스템과 동일 수준의 신뢰성을 제공하는 인메모리 파일시스템의 저널링 기법에 대해 기술한다.

한편, 저널링 파일시스템이 전체 데이터를 저널링할 경우 그 오버헤드가 크기 때문에 Ext4 등 대부분의 파일시스템은 그림 1의 좌측에서 보는 것과 같이 메타데이터만을 저널링하는 방법을 기본으로 채택하고 있다. 이와 유사하게 본 논문에서도 그림 1의 우측과 같이 인메모리 파일시스템의 저널링 방법으로 메타데이터를 저널링하는 방식을 채택하였다. 이는 기존 저널링과 신뢰성 보장 수준은 동일하지만 보장 시간 간격이 더 짧아 최신 정보의 유실 가능성이 줄어드는 반면 오버헤드는 상대적으로 적은 장점을 가진다.

사용자 프로세스가 쓰기 시스템 콜을 하는 경우 본 논문의 인메모리 파일시스템은 이를 버퍼 캐쉬에 기록하지 않고 해당 데이터의 파일시스템 위치에 직접 덮어쓰기 방식으로 기록한다. 이는 기존의 파일시스템이 버퍼 캐쉬에 데이터를 먼저 기록한 후 파일시스템 위치에 일정 시간 간격을 두고 반영하는 방식과 차별화된다. 따라서, 인메모리 파일시스템의 신뢰성을 보장하기 위해서는 사용자 프로세스로부터 쓰기 요청이 도착할 때마다 저널링이 이루어져야 하며 이는 상당한 오버헤드를 발생시킨다. 비록 입출력 과정에서 발생하는 소프트웨어 스택의 오버헤드가 제거되는 장점이 있지만, 인메모리 파일시스템의 저널링은 매 쓰기 시스템콜 마다 2번 기록을 해야 하는 부담이 발생하며, 이는 기존의 저널링 파일시스템이 주기적으로 파일시스템에 반영하는 것과 비교하여 쓰기량 및 쓰기 빈도를 증가시키는 결과를 초래한다. 그러나, 저널링 대상을 메타데이터로 한정할 경우 크기가 작은 inode 부분만 저널링되므로 대부분의 쓰기 오버헤드를 줄일 수 있다. 또한, 인메모리 파일시스템은 매 쓰기 연산 시마다 원자성을 보장하므로 주기적인 트랜잭션을 제공하는 전통적인 저널링 방식과도 차별화된다.

제안하는 저널링 기법의 동작 방식을 좀 더 자세히 살펴보면 다음과 같다. 사용자 프로세스로부터 쓰기 요청이 도착했을 때 먼저 일반 데이터를 파일시스템의 해당 위치에 직접 기록한다. 이때, 기존의 파일시스템과 달리 쓰기 연산은 수정된 파일 위치에 바이트 단위로 반영되며, 이는 블록 단위 쓰기 연산 대비 쓰기량이 적을 뿐 아니라 입출력을 위한 소프트웨어 스택을 통과하지 않으므로 효율적이다. 이는 사실상 전통적인 파일시스템에서 버퍼 캐쉬에 기록하는 것과 유사한 수준의 작업이라 할 수 있다. 일반 데이터에 대한 쓰기가 완료되면 메모리의 특정 위치에 inode에 대한 저널링을 수행한 후 저널링이 끝나면 완료 마크를 기록한다. 완료된 쓰기 연산의 의미는 시스템이 크래쉬되더라도 복구 가능한 상태임을 의미한다. 저널링이 끝난 후에는 inode의 수정사항을 파일시스템의 원래 위치에 반영한다. 이러한 모든 연산이 메모리 접근만으로 구성되기 때문에 그 오버헤드는 전통적인 파일시스템에서 버퍼 캐쉬에 저장하는 것과 유사한 수준에 불과하다. 즉, 제안하는 저널링 기법은 전통적인 파일시스템에서 블록 단위 입출력으로 인해 발생하는 쓰기량



(a) 연산 크기에 따른 성능

(b) 파일 크기에 따른 성능

그림 2. Ext4와 제안하는 인메모리 파일시스템의 저널링 성능 비교
Fig. 2. Comparison of journaling in Ext4 and the proposed in-memory file system.

을 줄일 뿐 아니라 입출력 경로로 인한 소프트웨어 스택을 줄이는 장점을 가진다. 그럼에도 저널링 주기 사이에 발생할 수 있는 기존의 신뢰성 유실 구간 없이 모든 쓰기 요청 즉시 신뢰성이 최신으로 보장되는 특성을 가진다.

III. 성능 평가

제안하는 저널링 기법의 성능을 평가하기 위해 Ext4 파일시스템과 벤치마킹을 통한 성능 비교를 수행하였다. 성능 측정은 인텔 코어 i5-3570 프로세서에 8GB의 메모리를 사용했다. D램을 사용해서 바이트 단위 접근이 가능한 영속 메모리를 에뮬레이션하였으며, 비교 대상 역시 동일한 D램을 사용해서 공정한 비교가 가능하도록 하였다.

Ext4 파일 시스템의 경우 비록 바이트 단위 접근이 가능한 매체라 하더라도 이를 블록 디바이스로 관리하기 때문에 램디스크 디바이스 드라이버를 설치해서 블록 디바이스로 인식되도록 한 후 파일시스템을 설치하여 성능을 측정하였다. Ext4의 저널링은 디폴트 옵션인 순서 모드(ordered mode)를 사용하여 제안한 저널링과 마찬가지로 메타데이터 저널링이 이루어지도록 하였다. 성능 평가를 위한 벤치마크로는 일련의 입출력 접근 연산을 발생시켜 스토리지 성능을 평가하는 IOzone을 사용하였다[9]. IOzone이 지원하는 다양한 입출력 시나리오 중 저널링의 성능 확인을 위해 쓰기 시나리오를 수행하여 성능을 비교하였다. 입출력 파일의 크기는 64KB에서 512MB까지 변화시켰으며, 연산의 크기는 4KB에서 16MB까지 변화시키며 실험하였다.

그림 2는 IOzone에서 파일의 크기 및 연산의 크기가 변함에 따라 Ext4의 저널링과 제안하는 저널링 방식의 처리량을 비교해서 보여주고 있다. 그림에서 보는 것처럼 제안하는 저널링 방식은 모든 경우에 있어 Ext4에 비해 우수한 성능을 나타내었으며, 성능 개선 효과는 평균 49.2%, 최대 131.5%에 이름을 확인할 수 있었다. 이는 제안하는 저널링 방식이 스토리지 쓰기량을 줄일 뿐 아니라 소프트웨어 스택의 오버헤드를 줄이기 때문에 얻을 수 있는 효과이다. 특히, 제안하는 기법은 커널 주소 공간 중 버퍼 캐쉬의 일부를 파일시스템으로 직접 사용하여 입출력 연산을 줄이는 장점을 가진다. 이를 통해 제안하는 저널링 방식은 메모리와 스토리지 간의 동기화 비용을 크게 줄일 수 있다. 반면, Ext4 파일시스템은 버퍼 캐쉬에 먼저 수정 내용을 기록한 후 입출력을 위한 소프트웨어 스택을 통과해서 영속 메모리에 블록 단위로 기록하기 때문에 상당한 오버헤드를 발생시킨다.

IV. 결 론

저널링 기술은 갑작스런 전원 오류로 인한 크래쉬 발생시 파일시스템이 깨어지지 않고 복구될 수 있도록 하는 기능을 제공한다. 그러나, 전통적인 저널링 기술은 바이트 단위 접근이 가능한 영속 메모리에 사용하기에 비효율적이다. 본 논문에서는 영속 메모리에 블록 디바이스용 파일시스템이 아닌 인메모리 파일시스템을 설치하여 입출력의 효율성을 높이고 여기에 저널링 기능을 추

가하는 기법을 제안하였다. 제안하는 기법은 기존의 인메모리 파일시스템이 임시 파일시스템 역할만 하여 데이터의 신뢰성을 제공하지 못하던 것을 개선하여 돌발적인 전원 오류가 발생하더라도 파일시스템이 깨어지지 않는 기능을 제공한다. 벤치마크를 이용한 성능 측정 실험을 통해 제안한 저널링 기법이 Ext4 파일시스템 대비 평균 49.2%의 성능 개선 효과가 있음을 확인하였다.

References

[1] S. Park, "RFJ: A reliable and fast journaling mechanism," Journal of the Korea Academia-Industrial cooperation Society(JKAIS), vol. 20, no. 7, pp. 45-51, 2019. DOI: https://doi.org/10.5762/KAIS.2019.20.7.45.

[2] H. Bahn and K. Cho, "Development of a distributed file system for multi-cloud rendering," The Journal of The Institute of Internet, Broadcasting and Communication, vol. 23, no. 1, pp. 77-82, 2023. DOI: https://doi.org/10.7236/JIIBC.2023.23.1.77

[3] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, "The new ext4 filesystem: Current status and future plans," Proceedings of Linux Symposium, vol. 2, pp. 21-33. 2007.

[4] E. Lee, S. Yoo, and H. Bahn, "Design and implementation of a journaling file system for phase-change memory," IEEE Transactions on Computers, vol. 64, no. 5, pp. 1349-1360, 2015. DOI: https://doi.org/10.1109/TC.2014.2329674

[5] Y. Choi and S. Ahn, "Separating the file system journal to reduce write amplification of garbage collection on ZNS SSDs," Journal of Multimed. Inf. Syst., vol. 9, no. 4, pp. 261-268, 2022. DOI: https://doi.org/10.33851/JMIS.2022.9.4.261

[6] E. Lee, H. Bahn, S. Yoo and S. H. Noh, "Empirical study of NVM storage: an operating system's perspective and implications," Proc. IEEE MASCOTS Conf., pp. 405-410, 2014. DOI: https://doi.org/10.1109/MASCOTS.2014.56.

[7] T. Cai, Y. Ma, P. Liu, D. Niu, and L. Li, "A new NVM device driver for IoT time series database," Micromachines, vol. 13, no. 3, article 385, 2022. DOI: https://doi.org/10.3390/mi13030385

[8] Y. Park and H. Bahn, "Modeling and analysis of the page sizing problem for NVM storage in virtualized systems," IEEE Access, vol. 9, pp. 52839-52850, 2021. DOI: https://doi.org/10.1109/ACCESS.2021.3069966

[9] W. Norcutt, the IOzone Filesystem Benchmark, http://www.iozone.org/

저 자 소 개

반 효 경(정회원)

- 1997년 2월 : 서울대학교 계산통계학과 학사
- 1999년 2월 : 서울대학교 전산과학과 석사
- 2002년 2월 : 서울대학교 컴퓨터공학부 박사.
- 2002년 9월 ~ : 이화여자대학교 컴퓨터공학과 교수.
- 주관심분야 : 운영체제, 스토리지시스템, 임베디드시스템

※ This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1009275) and the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub).