레거시 파일 시스템을 이용한 Persistent Memory 기반 저장 장치의 효율적 구동

송현섭O 문영제 이세권 노삼혁
울산과학기술원 전기전자컴퓨터공학부
{hssong1987, yjmoon, sekwonlee, samhnoh}@unist.ac.kr

Lightweight Adaptation of Legacy File Systems for Persistent Memory based Storage

Hyunsub SongO, Young Je Moon, Se Kwon Lee, Sam H. Noh
UNIST, School of Electrical and Computer Engineering

요 약

비휘발성 및 바이트 단위 접근성, 그리고 높은 접근 속도를 가지는 Persistent Memory (PM)의 도래에 따라 PM을 컴퓨팅 시스템의 저장 장치로서 사용하고자 하는 연구가 활발해졌다. 이런 대부분의 연구들은 오랜 기간 저장 장치를 관리하기 위해 사용되었던 기존 파일 시스템을 차선책으로 두고, 메모리 인터페이스에서 PM 기반 저장 장치를 구동시킬 수 있는 PM 전용 파일 시스템에 초점을 두고 있다. 하지만, 오랜 기간을 통해 성숙해져 온 기존 파일 시스템들을 단순히 차선책으로만 간주하는 것은 고찰해봐야 할 문제이다. 이 논문에서, 우리는 기존의 파일 시스템의 성숙된 특성을 보장하면서 PM 기반 저장 장치를 효율적으로 구동시킬 수 있는 방법에 대해 고려한다. 이 목적을 위해, 우리는 대표적인 기존 파일 시스템인 Ext4 파일 시스템과 PM 전용 파일 시스템인 PMFS를 비교 대상으로 설정하였으며, 기존 파일 시스템에 간단히 추가되어 PM 저장 장치를 효율적으로 구동시킬 수 있는 모듈인 Persistent Memory Adaptation Layer (PMAL)를 설계하였다. 성능 평가를 위해 수행된 실험에서 PMAL을 적용시킨 Ext4 파일 시스템은 기존의 특성들을 보장하면서, PMFS와 5% 이내의 성능 차이를 보였다.

1. 서 론

*차세대 메모리 기술인 Persistent Memory (PM)는 DRAM과 유사한 접근 속도를 보유함과 동시에 비휘발성을 가진다. PCM, STT-MRAM, 3D XPoint [1], 등으로 개발되고 있는 PM은 컴퓨팅 시스템에서 특히 저장 장치로서 적용될 것을 기대받고 있다. 이런 흐름에 따라, 많은 학계 및 업계에서는 PM 기반 저장 장치를 위한 새로운 파일 시스템에 대해서 연구하고 있다 [2, 3, 4, 5]. 이 연구들은 기존의 블록 기반 파일 시스템은 PM 기반 저장 장치에 효율적이지 않다는 전제 하에, PM 매체에 특화된 PM 전용 파일 시스템을 고려하고 있다. 하지만, 오랜 기간 검증된 성숙도 있는 파일 시스템을 단순히 차선책으로만 간주하는 것은 고찰해봐야 할 문제이다.

표 1은 대표적인 기존 파일 시스템인 Ext4 파일 시스템과 PM 전용 파일 시스템인 PMFS [5]에 대한 특성을 보여준다. 표 1에서 볼 수 있듯이, 새로운 PM 전용 파일 시스템의 구조는 기존 파일 시스템의 구조와 크게 다르지 않다는 것을 알 수 있다. 하지만, 기존 파일 시스템은 블록 기반 저장 장치를 위해 설계되었기 때문에, PM 기반 저장 장치에서는 불 필요한 성능 부하를 야기할 수 있는 블록 소프트웨어 계층 및 페이지 캐시 계층을 가지고 있다. 또한, 파일 시스템 일관성을 보장하기 위해 제공되는 저널링 작업도 매우 큰 부하를 가지고 있다.

표 1 파일 시스템들의 특성 비교

		Ext4	PMFS
Performance Scalability	Max. Vol.	1EB	NA
	Max. File Size	16TB	NA
	Max. # of Files	4 Billion	NA
	Metadata, Block	HTree	B-tree
Reliability	FS Consistency	Journaling, Checksum	Logging
	Data Durability†		clflush &sfence, pm_wbarrier

새로운 PM 전용 파일 시스템들은 이 계층들을 제거하는 방향으로 파일 시스템의 구조를 설계한다. 이런 파일 시스템들은 PM을 저장 장치로 사용하는 환경에서 성능적 이점을 가질 수 있지만, 기존 파일 시스템의 성숙도 있는 특성들을 보장할 수는 없다.

이 논문에서, 우리는 기존 파일 시스템의 성숙된 특성들을 보장하면서, 간단한 수정을 통해 PM 기반 저장 장치를 효율적으로 구동시킬 수 있는 방법에 대해서 고려한다. 이를 위해, 우리는 기존 파일 시스템에서 높은 부하를 가지는 블록 소프

* 이 논문은 삼성전자 미래기술육성센터의 지원을 받아 수행된 연구임 (과제번호 SRFC-IT1402-09).

† PMFS는 CPU 캐시와 영속매체인 PM 기반 저장 장치 사이에서 발생되는 데이터 내구성 문제를 해결하기 위해 입력 단위 마다 명시적으로 **clflush**, **sfence** 명령을 호출한다.

트웨어 계층 및 페이지 캐시 계층의 흐름을 조절하여 성능 하락을 줄일 수 있는 Persistent Memory Adaptation Layer (PMAL) 모듈을 제안한다. 이 모듈은 앞선 두 계층 대신 가볍게 수정된 PMAL 내의 컴포넌트로 입출력 작업을 수행할 수 있게 해준다. 이 모듈은 파일 시스템의 전체적인 수정 없이, 모듈로 삽입되기 때문에, PM 전용 파일 시스템에 비해 구현적으로 용이할 뿐 아니라, 기존 파일 시스템의 성숙도 있는 특성들을 그대로 보장할 수 있다는 장점을 가진다. PMAL을 적용 시킨 파일 시스템의 성능을 평가하기 위해 우리는 Linux 플랫폼에서 기존 파일 시스템인 Ext4에 PMAL 모듈 삽입을 구현하였다. Filebench 벤치마크 프로그램을 이용한 실험에서 PMAL을 적용한 Ext4 파일 시스템은 PM 전용 파일 시스템인 PMFS에 육박하는 성능을 보였다.

2. 관련 연구

2.1 PM을 고려한 파일 시스템에 대한 연구

PM 저장 장치에 특화되어 설계된 대표적인 PM 전용 파일 시스템은 PMFS [5]이다. 이로 인해, PMFS는 PM 매체 본연의 성능을 효율적으로 보일 수 있다. 하지만, 새로운 파일 시스템이기 때문에 설계 및 구현에 상당한 시간과 어려움이 따른다. 또한, 다양한 실제 환경에서 오랜 기간 검증되어 온 파일 시스템이 아니기 때문에, 기존 컴퓨팅 시스템에 적용되기에는 어려움이 따른다.

최근 Linux 플랫폼에는 PM 저장 장치에 직접적으로 데이터를 입출력 시킬 수 있는 DAX (Direct Access eXciting) [6] 기능이 추가되었다. 이 기술은 POSIX I/O 시스템 콜을 통해 구동되며, 기존 파일 시스템인 Ext4, XFS의 파일 시스템 계층 내부에서 바이트 단위로 넘어온 입출력 데이터를 DAX 인터페이스 (**direct_access()**) 이용하여 블록 단위로 변환하지 않고, PM 기반 저장 장치에 직접적으로 데이터를 접근시킴으로써 불 필요한 성능 부하의 문제를 해결하였다.

하지만, DAX는 페이지 캐시를 우회시키는 Direct I/O 흐름을 기반으로 설계되었기 때문에, 기존의 운영체제의 페이지 캐시를 이용한 성숙된 특성들을 사용할 수 없게 한다 [7]. 페이지 캐시의 사용으로 인한 이점은 **read()** 시스템 콜에 대한 빠른 응답 시간뿐만 아니라 파일 간의 복사, 압축, 암호화 등 여러 측면에서 존재하기 때문에, DAX를 사용하게 된다면, 이러한 이점들을 사용할 수 없게 된다.

3. Persistent Memory Adaptation Layer

앞서 언급한 것처럼, 본 논문의 목적은 기존 파일 시스템의 성숙도 있는 특성들을 보장하는 동시에, PM 매체의 성능을 최대한 보장하는 것이다. 이 목적에 따라, 우리는 기존 파일 시스템의 입출력 흐름을 조정하여 기존 파일 시스템의 블록 소프트웨어 계층, 페이지 캐시 계층, 파일 시스템 일관성을 위한 계층의 부하를 줄일 수 있는 모듈인 Persistent Memory Adaptation Layer (PMAL)을 제안한다.

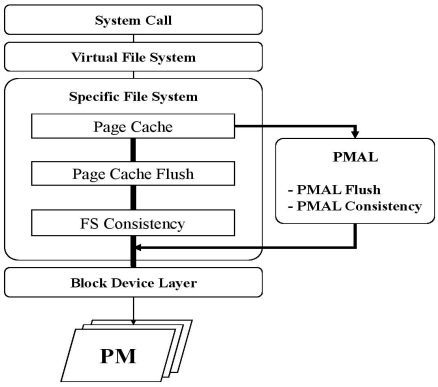

그림 1 PMAL을 적용시킨 파일 시스템의 구조

3.1 PM 기반 저장 장치 설정

우리는 PMAL이 적용되는 시스템의 메모리 (페이지 캐시 영역) 및 저장 장치 (파일 시스템 영역)가 모두 비휘발성 매체인 PM으로 구성되는 환경을 가정한다. PM 기반 저장 장치는 램-디스크 소프트웨어 드라이버를 통해 블록 장치로 보여질 수 있도록 하여, 수정 없이 PM 매체 위에서 기존 파일 시스템을 구동 할 수 있도록 해준다. 또한, 램디스크 소프트웨어 드라이버에는 기존 블록 장치 드라이버와 다르게 입출력 스케쥴러 및 블록 장치 입출력에 대한 작업 없이 메모리 입출력 명령을 (**load**/**store**)을 이용하여 데이터의 입출력을 수행하기 때문에, 기존 블록 소프트웨어 계층의 부하를 없앨 수 있다.

3.2 PMAL의 설계 및 구현

PMAL이 삽입되는 위치는 그림 1에서 볼 수 있듯이 기존 파일 시스템의 페이지 캐시 계층 이후에 데이터가 저장 장치로 플러시 되기 전 부분이다. 이 시점에서, PMAL은 저장 장치로 내려가는 입출력 흐름을 인터셉트하여 PMAL의 컴포넌트들을 통해 플러시 작업 및 파일 시스템 일관성을 위한 작업을 바로 수행하게 되고, 수행이 완료되면 사용자에게 완료 신호를 보내게 된다. 이럼으로써, PMAL을 적용한 파일 시스템은 기본적으로 동기 입출력을 수행하게 된다.

그림 1에서 볼 수 있듯이, PMAL은 두 가지 컴포넌트로 구성된다. 첫 번째로, PMAL Flush 컴포넌트는 기존의 페이지 캐시 플러시 계층의 메커니즘과는 다르게, 현재 쓰려고 하는 데이터에 대해서 쓰기 시마다 플러시 작업을 수행한다. 이런 방법을 채택한 이유는 기존에는 느린 블록 장치의 입출력 횟수를 줄이기 위해, 쓰려고 하는 데이터가 속한 파일 전체에서 더티 페이지들을 찾아 한 번에 플러시 시키는 방법을 이용하였지만, 접근 속도가 빠른 PM 기반 저장 장치에서는 오히려 더티 페이지를 찾기 위한 소프트웨어적 부하가 더 큰 성능 저하의 요인이 될 수 있기 때문이다.

두 번째로, PMAL Consistency 컴포넌트는 성능 부하가 매우 큰 파일 시스템의 일관성을 위한 계층을 대신하여, 앞서 가정한 영속적인 페이지 캐시 영역에서 페이지 캐시 기능과 저널링 기능을 함께 수행할 수 있게 한다. 이 방법을 통해, 추가적인 부하 없이 페이지 캐시 영역 (동시에 저널 영역)에 새로운 데이터를 씀과 동시에 멀티-버저닝이 이루어지게 된다.

표 2 Filebench 워크로드들의 특성

	R:W	File Size	# of Files	# of Threads
File Server	1:2	128K	100K	50
Web Server	10:1	32K	500K	50
Mail Server	1:1	16K	800K	50
OLTP	1:1	1.5GB	10	W:10 R:200

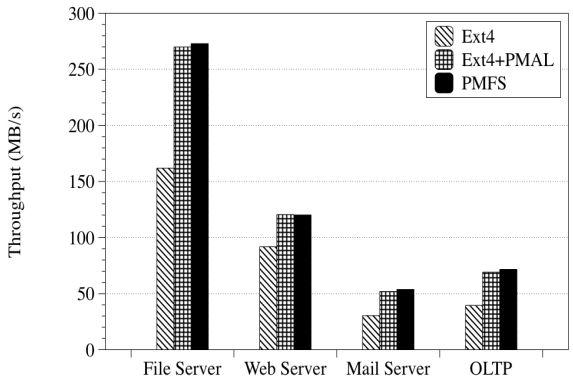

그림 2 파일 시스템들의 성능 비교

이 방법은 Lee 등이 제안한 PM 영역에서 버퍼 캐시와 저널링을 함께 제공하는 기술과 유사하다고 볼 수 있다 [8]. 작동 방식은 Ext4의 Ordered 모드 저널링 기법과 유사하다.

작동 흐름을 살펴보면, 우선 페이지 캐시 영역에 쓰여진 데이터는 PMAL Flush 컴포넌트를 통해 동기적으로 파일 시스템 영역에 쓰여지게 된다. 그런 후에, PMAL 시스템은 이미 캐시에 쓰여진 메타데이터에 대한 쓰기가 완료되었다고 통지한다. 이 단계가 파일 시스템의 일관성이 보장되는 시점이다. 그 후에, 일정한 간격으로 수행되는 페이지 캐시 플러시 데몬은 자동적으로 캐시의 메타데이터를 파일 시스템 영역으로 플러시 시켜주는 체크포인팅 역할을 하게 된다.

PMAL Flush 컴포넌트에서 추가적으로 고려해주어야 하는 사항은 쓰여진 데이터를 업데이트하는 과정에서 이전 버전의 메타데이터를 보존해 주어야 하는 것이다. 이를 위해, 현재 수행되고 있는 쓰기 작업이 업데이트일 경우, 새로운 버전의 메타데이터에 대해서는 로깅을 통해 멀티-버저닝을 한다. 또한, PM을 저장 장치로 사용하게 될 경우 발생하는 데이터의 내구성 문제를 해결하기 위해, 우리는 PMFS에서 제시하는 **clflush**, **sfence** 명령을 이용한 방법을 그대로 적용하였다.

4. 성능 평가

4.1 실험 환경

PMAL의 성능을 평가하기 위해, 우리는 Linux 커널 3.11.0 버전의 Ext4에 PMAL을 적용하였으며, 이 버전에서 제공하는 PMFS를 비교 대상으로 선정하였다. PMAL이 적용된 Ext4는 앞서 언급한 램디스크 소프트웨어 드라이버를 이용하여 에뮤레이트 된 DRAM의 30GB 영역 위에 마운트 시켰으며, PMFS는 **ioremap()**을 이용하여 에뮤레이트된 DRAM의 30GB 영역 위에 마운트 시켰다. 실험은 Filebench의 워크로드들을 이용하였으며 종류 및 특성은 표 2에서 보여준다.

4.2 실험 결과

그림 2에서 볼 수 있듯이, PMAL을 적용한 파일 시스템은 기존 파일 시스템보다 상당히 높은 성능을 가지며, PMFS에 거의 육박하는 성능을 보인다. 이 결과의 이유는 PMAL 모듈의 삽입을 통해 기존의 페이지 캐시 플러시 계층과 파일 시스템의 일관성을 위한 계층의 부하를 줄였기 때문이다. 이를 통해, PMAL을 적용시킨 파일 시스템은 기존 파일 시스템의 성숙도 있는 특성들을 보장함과 동시에, PM 매체 본연의 성능도 효율적으로 사용할 수 있다는 것을 보여준다.

5. 결론 및 향후 연구

Persistent Memory를 저장 장치로 활용하는 연구들 중 많은 연구들이 PM에 적합한 새로운 파일 시스템을 설계하는데 초점을 맞추고 있다. 하지만, 오랜 기간 동안 검증되어 온 성숙된 기존의 파일 시스템들을 단순히 차선책으로 두는 것은 다시 한번 생각해봐야 할 문제이다. 우리는 기존 파일 시스템을 이용하여 PM 기반 저장 장치를 효율적으로 관리할 수 있는 모듈인 PMAL을 제안한다.

PMAL은 기존 파일 시스템의 부하 계층들을 제거하지 않고, 입출력 요청을 PMAL 내의 컴포넌트들이 대신하여 부하 없이 수행할 수 있도록 해준다. 또한, PMAL은 기존 파일 시스템을 거의 수정하지 않고 모듈 삽입을 통해 구현되기 때문에, PM 전용 파일 시스템 보다 구현적 용이함을 가진다. Linux 플랫폼에서 수행한 성능평가에서 PMAL을 적용한 Ext4 파일 시스템은 PMFS에 거의 육박하는 성능을 보였다.

향후 과제로서, 우리는 시스템 장애 후 파일 시스템을 복구하는 과정에 대해 연구할 것이다. PMAL Consistency의 설명에서 보였듯이, 기존 페이지 캐시의 기능들을 많은 수정 없이 복구 메커니즘을 위해 사용할 수 있을 것으로 예상한다.

참 고 문 헌

[1] Micron Technology, “3D XPoint Technology,” http://www.micron.com/about/innovations/3dxpoint-technology.

[2] X. Wu, et al., “SCMFS: A File System for Storage Class Memory,” In SC, 2011.

[3] J.Condit, et al., “Better I/O Through Byte-Addressable, Persistent Memory,” In SOSP, 2009.

[4] H. Volos, et al., “Aerie: Flexible File-System Interfaces to Storage-Class Memory,” In EuroSys, 2014.

[5] S. R. Dullor, et al., “System Software for Persistent Memory,” In EuroSys, 2014.

[6] LWN.net, “DAX: Page cache bypass for file systems on memory storage,” https://lwn.net/Artivles/588218/.

[7] LWN.net, “Directly mapped persistent memory page cache,” http://lwn.net/Articles/644120/.

[8] E. Lee, et al., “Unioning of the Buffer Cache and Journaling Layers with Non-volatile Memory,” In FAST, 2013.