특집 02

NVM/DRAM 기반 하이브리드 메인 메모리를 위한 파일 시스템 동향

조중석·조두산 (국립순천대학교)

목 차

1. 서 론

DRAM은 실제 프로세스들이 작업을 수행하는 메인 메모리 장치로 디스크 다음으로 넓은 용량이 사용되고 있다. 지금까지 10년에 약 100배의 비율로 30년 이상 그 용량을 확대하여 개발되어왔다. 프로세서 속도 향상을 나타내는 무어의 법칙과 더불어 IT기술 발달을 지탱해온 기둥이었다. 하지만 엘피다메모리사의 CTO가 말했던 것처럼 4~5년전의 DRAM 제조 기술로는 30nm 이후 공정에서는 미세 공정에서의 개발이 매우 어려워진다고 전망되었었다. 콘덴서와 트랜지스터 조합으로 데이터를 기록하는 DRAM은 콘덴서 때문에 미세화가 어렵게 된다. 데이터 저장을 위해서는 전위차를 만들 콘덴서가 필요한데, 30nm 이후 미세공정에서는 최소 용량인 20 femto farad을 확보할 수 없게 되기 때문이다. 귀금속을 재료로 사용하는 몇가지 해결책이 있으나 당시의 메모리 단가에서 수배 가격 상승을 발생시키기 때문에 제품 경쟁력이 떨어진다. 이러한 미세화에 대한 해결책으로 많은 기업에서 비휘발성 메모리 기술을 선택하였다. 왜냐하면 4~5년 전에는 공정 미세화의 한계로 비휘발성 메모리와 같은 차세대 메모리 기술이 기존 DRAM을 대체 할 것이라고 예상했기 때문이다. 그러나 이러한 메모리 기술이 예상을 깨고 14나노까지 상용화되었고, 10나노 이하 기술이 등장할 전망인데다 3차원 반도체 적층 기술과 멀티 패터닝 기술 등 집적도를 높이기 위한 설계 · 공정 · 소재 기술이 발전함에 따라 기존 메모리의 '대체'가 아닌 '보완' 형태가 상용화될 것으로 보인다. 즉, 상호보완적인 하이브리드 메모리의 형태로 개발되는 것이다.

삼성전자와 SK하이닉스는 이미 D램과 낸드플래시 등 기존 메모리의 성능을 높이고 단점을 보완할 수 있는 새로운 메모리 제품군을 개발하는 데 속도를 내고 있다. 새로운 제품을 개발해 시장을 창출하는 형태가 아니라 시장 요구를 맞추기 위한 보완

적인 형태인 것이다. 상용화에 성공하면 새로운 성장동력으로 빠르게 자리매김할 전망이다.

과거와 달라진 것은 차세대 메모리가 이미 시장에서 확고한 위치를 차지한 D램, S램, 낸드플래시를 대체하기보다 이들의 성능을 보완해주는 데 초점이 맞춰있다는 것이다. 현재 가장 주목받는 차세대 메모리 기술은 3D XPoint, R램(저항변화형 메모리), P램(상변화 메모리), STT-M램(스핀주입 자화반전 메모리) 등인데 우리는 여기서 이러한 메모리 기술을 통칭 비휘발성 메인 메모리 (Non-Volatile Main Memories, NVMM)으로 표기하겠다. 각각의 메모리 기술들이 서로 다른 장단점을 갖고 있는데, STT-MRAM은 DRAM보다 빠르기 때문에 온칩 메모리 혹은 하단 캐시 (last-level cache)로 쓰일 수 있으나 큰 셀 크기 때문에 용량에 제약을 받아 DRAM을 대체하기는 어렵다고 전망되고 있다. PCM과 ReRAM은 DRAM에 비하여 집적도가 높으나 접근속도가 느리기 때문에 DRAM을 단일 기술로 대체하기에 용이하지 않다. 3D XPoint 메모리 기술은 최근 인텔과 마이크론사에 의하여 발표되었다. 낸드플래시에 비하여 1000배까지 속도 향상이 가능한 것으로 알려져 있다. 조만간 SSD를 대체하는 기술로 자리매김 할 것으로 전망된다.

우리는 여기서 이러한 NVMM과 DRAM의 단점을 보완하고 상호 장점을 살려주는 하이브리드 메인메모리에 대해서 살펴보기로 한다. 앞서 언급된 바와 같이 NVMM은 미세 공정에서 집적도를 충분히 활용할 수 있도록 하는 메모리 기술이며, 현재 이를 활용한 저전력 연구도 많이 진행되고 있으나 산업계에서는 실질적으로 집적도 미세화에 따른 수율을 맞추기 위한 방안으로 선택하여 활용되고 있다. 하이브리드 메모리가 개발되면 결국 상용화의 성공을 결정할 가장 중요한 요소는 파일 시스템이다. 기존 파일 시스템은 메모리 계층 성능 최적화를 위하여 단일 디스크와 DRAM 사이의 대역폭을 효율적으로 활용하는데 초점을 맞추어 개발되어 있다. 하지만 하이브리드 메모리에서는 대역폭의 활용과 더불어 NVM의 장점을 최대한 활용한 데이터 일관성 (consistency) 유지가 중요한 문제로 대두된다. 우리는 여기서 하이브리드 메모리의 파일 시스템에서 요구되는 주요한 문제들을 살펴보고자 한다. 비휘발성 메모리의 장점 때문에 NVMM 기반의 하이브리드 메인메모리가 곧 시장을 점유하기 시작할 것으로 예상되며, 이때 하이브리드 메모리를 구성하는 각 메모리의 장점을 충분히 활용하도록 설계된 파일 시스템의 특징에 대해서 기존 연구[1]를 기반으로 살펴보고자 한다.

2. DRAM과 NVM으로 구성된 하이브리드 메모리에 적합한 파일 시스템

상대적으로 빠른 휘발성 메모리인 DRAM과 집적도가 높고 약간 상대적으로 느린 상변화메모리 같은 비휘발성 메모리 결합은 상호 기술의 장점을 극대화하는 최적의 방안으로 그 가능성을 주목받고 있다. 하이브리드 메인 메모리 시스템은 매우 강력한 일관성 (consistency) 보장을 위한 부분이 필요하며 동시에 소프트웨어 오버헤드도 최소화해야 하이브리드 메모리의 장점을 충분히 활용할 수 있게 된다. 이러한 관점에서 기존의 파일시스템은 하이브리드 메모리 시스템에 적합하지 않다. 기존 기법은 디스크의 성능에 초점을 맞추어 개발되어 있으며 일관성 또한 디스크의 정확성 (correctness)에 의존하기 때문이다. 강력한 일관성 보장은 특히 NVMM 기반 파일시스템에서 도전적인 요소이다. NVMM에서 쓰기 동작은 큰 비용을 요구하기 때문이다. 현대의 CPU와 메모리 시스템은 성능 개선을 위하여 쓰기 동작 순서를 변경하기도 한다. 이때 시스템 오류가 발생하면 일관성이 유지되지 못한다. 이

러한 경우를 방지하기 위하여 파일시스템은 CPU 캐시에서 데이터를 순서를 명확히 유지하여 플러시 (flush) 되도록 할 필요가 있다. 이때 순서 유지를 위한 오버헤드가 추가되고 NVMM을 통한 성능 개선을 포기하더라도 일관성 유지를 위해 이러한 부분들을 희생해야하는 문제점이 발생한다.

많은 애플리케이션들은 원자적 파일 시스템 동작 (atomic file system operation)에 그들의 정확성을 의존하기 때문에 이러한 문제를 해결하는 것은 중요하다. 현존하는 주요 파일 시스템은 저널링 (journaling), 쉐도우 페이징 (shadow paging), 로그 구조 (log structuring) 기술들을 이용하여 원자성 (atomicity)를 제공한다. 하지만 저널링은 쓰기 연산의 증가에 따라 대역폭을 낭비하게 되고, 쉐도우 페이징은 잎 노드 (leaf node)에서 루트(root)까지 연속적인 업데이트를 요구하기 때문에 양쪽 기술 모두 쓰기 순서 제약 (ordering constraint)을 유지하기 위한 성능 저하를 피할 수 없게 된다.

하드디스크 혹은 NAND 플래시 기반 SSD에서 효과적으로 사용할 수 있도록, 로그 구조화 파일 시스템 (log-structured file system, LFS)은 적은 수의 랜덤 쓰기 요청을 더 많은 순차 쓰기로 그룹화할 수 있다. 하지만 현재의 LFS는 가용한 연속된 자유 공간 (contiguous free region)에 의존적이다. 연속된 자유 공간을 유지하기 위해서는 비싼 비용을 요구하는 가비지 콜렉션 동작 (garbage collection operation)을 빈번히 실행해야 한다. 결과적으로 최근의 연구[2] 에서는 NVMM에서 LFS가 저널링 파일시스템보다 성능이 저하될 수 있음을 보고하였다. 이러한 여러 가지 어려운 문제들을 해결한 결과가 최근 학계에 보고되었다 [1]. 우리는 여기서 해당 케이스를 살펴보고 향후 흐름에 대한 전망을 논해보기로 한다.

2.1 NOn-Volatile memory Accelerated (NOVA) 로그구조 (log-structured) 파일 시스템

NOVA는 여러 부분에서 기존의 로그 구조 파일 시스템과 다르다. NOVA는 각각의 inode를 분리된 로그로 구성하여 보통 동작과 복구 (recovery) 실행동안 동시성 (concurrency)을 최대화 하도록 하였다. NOVA는 로그들을 링크리스트 (linked list)로 저장하기 때문에 연속된 메모리 공간이 필요없다. 로그의 끝 (tail) 포인터에 원자적 정보 갱신을 수행하기 때문에 경량 저널링이 가능하다. NOVA는 데이터 로그를 기록하지 않기 때문에 복구 프로세스는 NVMM의 일부만을 스캔하는 것으로 경량화 된다. 이로써 가비지 콜렉션의 오버헤드를 줄일 수 있고 메모리가 거의 차있는 상황에서도 좋은 성능을 제공할 수 있게 된다.

2.2 배경 지식

NVMM 기술은 파일 시스템 디자이너에게 여러 가지 도전적 과제를 안겨주었다. 가장 중요한 것 중 하나는 소프트웨어 오버헤드에 대한 메모리 성능의 밸런싱이다. 이와 관련된 요소들은 다음과 같다.

- **성능 :** NVMM의 낮은 지연시간 (latency)은 하드웨어와 소프트웨어 지연시간 사이의 상충관계 (trade-off)를 변화시켰다. 보통 저장 시스템은 높은 지연시간을 갖는 디스크와 같은 저장장치가 접근 시간을 결정하는 주요요인이다. 그래서 소프트웨어 효율은 중요한 요소가 아니었다. 하지만 최근 몇몇 연구결과 [3, 4]가 보였듯이 빠른 NVMM에서는 소프트웨어 비용이 접근시간을 결정하는 주

요요인이 된다. STT-MRAM 기반의 NVMM은 낮은 지연시간을 제공하기 때문에 프로세서의 메모리 버스에 직접 연결되고 로드/스토어 (load/store) 명령어로 직접 사용된다. 최근의 NVMM 기반 파일시스템은 DRAM 페이지 캐시를 거치지 않고 직접 접근 기술 (Direct Access, DAX) 혹은 eXecute In Place (XIP)로 불리는 기술로 NVMM과 DRAM 사이의 저장 스택에 추가 복사본을 만들지 않고 NVMM에 직접 접근된다. NOVA의 경우 DAX 파일시스템이고 향후의 NVMM 파일시스템도 유사한 유형으로 개발 및 사용될 것이 명확하다.

- **쓰기 재정렬 (Write reordering):** 현대의 메모리 계층에서 프로세서와 캐시는 쓰기 명령 순서를 재정렬하여 성능 개선을 할 수 있다. CPU의 메모리 일관성 프로토콜은 메모리 쓰기 순서에 대한 보장을 제공한다. 그러나 현재 사용되는 이러한 기술이 NVMM에는 적용되지 않는다. NVMM에서 전력 손실에 의한 일관성 복구 절차는 메모리 쓰기 순서 변경을 완전히 복구할 수 없기 때문에 NVMM에 한해서는 쓰기 순서 변경에 의한 성능 개선 부분은 희생해야한다. 이러한 문제를 해결하기 위한 NVMM 인지 소프트웨어는 명확한 캐시 플러싱 (cache flushing) 명령어를 사용하여 쓰기 순서를 유지할 수 있다. x86 아키텍처는 CPU 캐시라인을 플러시 하도록 clflush 명령어를 제공한다. clflush 명령어는 캐시라인들이 메모리에 올바른 순서에 따라 기록되는 것을 보장하지만 해당 캐시라인들은 무효화되기 때문에 캐시의 효율적인 활용에 따른 장점을 잃게되고 이에 따라 전체적으로 성능이 현저하게 저하된다. 게다가 명확히 말하면 메모리에 기록되는 것이 보장되는 것이 아니라 메모리 제어기 (controller)에 전달되는 것만 보장하기 때문에 데이터가 메모리에 기록되는 것을 완전히 보장하지는 못한다. 이러한 문제를 해결하기 위해서 인텔은 clflush 명령어의 효율화 버전인 clflushopt를 추가하였다. NOVA는 이러한 개선 명령어에 기반하여 구성되었다.
- **원자성 (atomicity):** POSIX 스타일의 파일 시스템 시맨틱 (semantic)은 여러 동작들이 원자적 (atomic)이 되는 것을 요구한다. 즉, 모든 작업이 완전하게 실행되거나 아니면 전혀 실행되지 않거나 양자택일되는 결과가 발생하도록 동작되는 것이다. 예를 들어, POSIX rename 명령은 명령이 실패하였을 때 해당 파일 이름이 원래의 이름으로 남아 있거나 혹은 새로운 이름으로 변경되거나 양자 경우 중에서 어느 한가지 결과로 종료되는 것이 보장되어야 한다. 파일이름 변경은 메타데이터만을 변경하는 작업이지만 다른 원자적 명령은 데이터와 메타데이터 모두를 수정하는 경우도 있다. 때로는 원자성을 제공하기 위하여 프로그래머는 새로운 복잡한 기법을 직접 개발해서 사용해야 한다. 현재 원자성 보장은 기본적인 작업에서만 제공되고 있다. 예를 들면, 디스크에서 정렬된 섹터 쓰기 (aligned sector writes)는 8바이트 이하 단위에서만 제공된다. 하이브리드 메모리에서는 이것을 유지하기 위한 새로운 방안을 모색해야 할 것이다.

2.3 복잡한 원자적 동작 보장

현존하는 파일 시스템들은 저널링, 쉐도우 페이징, 로그 구조화 등 다양한 기술들을 사용하여

원자성 보장을 제공한다. 이와 관련하여 살펴보도록 하자.

- **저널링:** 저널링 시스템은 저널에 모든 변경사항을 실제 변경이 이루어지기 이전에 기록한다. 전력 손실 오류가 발생하는 경우에 저장된 저널에 따라 시스템의 일관성을 유지하도록 복구하는 것이 가능하게 된다. 따라서 저널링은 데이터 쓰기를 저널 로그에 한번 변경 대상에 한번 이렇게 총 두번 수행하게 된다.
- **쉐도우 페이징:** 기존의 다양한 파일 시스템이 쓰기 복사 (copy-on-write) 메커니즘을 사용하고 있다 [5, 6]. 쉐도우 페이징 파일 시스템은 원자성을 제공하는 트리 구조에 강하게 의존한다. 쓰기 시 해당 데이터를 바로 수정하는 대신에 쉐도우 페이징 쓰기는 새로운 복사본에 수정을 수행한다. 새로 생성한 복사본은 저장장치의 빈공간에 생성한다. 이때 새롭게 생성된 페이지들이 파일 시스템 트리를 갱신하게 만드는데, 이 추가된 페이지인 잎노드에서 루트 노드까지 트리 자료구조의 연속된 업데이트 작업은 잠재적인 오버헤드 비용으로 연결된다.
- **로그 구조 파일 시스템:** 로그 구조 파일 시스템 (log-structured file system, LFS)는 본래 하드 디스크 드라이브의 순차접근에 의한 성능 개선을 이용하기 위하여 설계되었다. LFS는 메모리의 랜덤 쓰기 연산을 그룹화 (grouping)하고 결과적으로 더 많은 순차쓰기 동작으로 이들을 변환하여 대역폭을 충분히 사용함으로써 하드디스크 성능을 최적화하게 된다. LFS는 명쾌한 아이디어임에 분명하지만 이것을 효과적으로 구현하는 것은 어려운 일이다. 왜냐하면 LFS는 순차쓰기를 디스크의 연속된 공간에 진행해야 하고, 이러한 공간 확보를 위하여 빈번한 가비지 콜렉션을 수행하여야하기 때문이다. 이러한 오버헤드를 줄이기 위한 몇몇 시도들이 현재 학계에 보고되어있다. F2FS [7]는 여러개의 로그 헤드를 사용하여 메타데이터와 데이터 로그를 분리한다. 새로운 데이터는 현재 있는 자유공간 (free space)에 바로 쓰도록하여 가비지 콜렉션 오버헤드를 줄일 수 있었다. RAMCloud [8]는 DRAM 기반 저장장치에서 모든 로그 구조를 기록하고 관리하도록 하여 DRAM 사용율을 높이도록 설계되었다. 또한 백업 데이터는 디스크에 두고 자유공간 확보가 원활하도록 지원하였다.

2.4 NVMM을 위한 파일 시스템

몇몇 연구 그룹에서 NVMM 기반 파일시스템의 문제점들을 고려하고 그에 대한 해결책을 발표하여 왔다. BPFS [9]는 메타데이터와 데이터의 일관성을 제공하는 쉐도우 페이징 파일시스템 이다. BPFS는 쓰기 복제와 순서 일관성을 제공하기 위한 하드웨어 메카니즘을 사용하였다. 하드웨어 쉐도우 페이징은 복사 및 관리에 대한 소프트웨어 오버헤드를 제거할 수 있으나, 고정된 하드웨어 버퍼 크기 때문에 거대한 파일시스템에서는 이러한 오버헤드가 다시 문제로 재발할 수 있는 단점이 존재한다. PMFS [10]는 경량화된 DAX 파일시스템이다. 메타데이터 수정에 대한 저널링을 사용하지만 데이터 바로 쓰기를 허용하여 원자성 제공을 못하고 결과적으로 데이터 일관성이 보장되지 않는다. Ext4-DAX [11]는 NVMM에 직접 접근 가능하도록 Ext4를 DAX에 적합하게 확장하였다. 또한 원자성이 보장된 저널링으로 메타데이터 수정을 지원한다. DAX로 확장되지 않은 본래의 Ext4 파일시스템은 데이터 원자성을 보장하는 데이터 저널 모드를 포함하고 있으나, 확장된 파일시스템은 해당 모드를 지원하지 않아 데이터 수정에 원자성이 제공되지 못하고 있다. SCMFS [12]는 운영체제의

가상 메모리 관리 모듈을 사용한다. 이로써 가상의 연속된 주소 영역으로 파일을 매핑하고 관리 및 접근을 단순화하고 경량화 할 수 있었다. 다만 메타데이터와 데이터의 일관성 보장을 위한 기능이 제공되지 않았다. Aerie [13]은 NVMM 데이터 접근을 위한 기능과 파일 시스템 인터페이스를 구현하였다. POSIX 시맨틱을 완화하여 성능을 개선하였으며 메타데이터 저널링을 제공하였다. 하지만 데이터 원자성과 mmap 동작에 일관성 지원을 하지 못하였다.

3. NOVA 설계안

NOVA는 하이브리드 메모리에서 LFS의 장점을 충분히 활용하도록 POSIX 파일시스템으로 구성되어있다. 두 가지 서로 다른 메모리 기술로 구성된 하이브리드 메모리에는 기존의 디스크 대역폭을 최대로 활용하도록 구성된 기존의 파일 시스템과는 다른 구조의 특징들이 요구된다. NOVA는 3가지 주요 분석결과를 바탕으로 설계되었다. 첫째, NVMM에서 원자적 수정을 지원하는 로그는 구현이 쉽지만 디렉토리 검색과 파일 액세스와 같은 검색 동작은 효율적이지 않다. NVMM에서 빠른 검색을 지원하는 트리와 같은 자료구조는 더욱 정확한 구현이 어렵다 [14, 15]. 둘째, 로그 삭제의 복잡도는 연속된 자유 공간 확보를 지원하는 방식에 따라 결정되는데, NVMM에서는 랜덤 접근 비용이 저렴하기 때문에 복잡한 방식이 요구되지 않는다. 셋째, 단일 디스크에서는 공간 지역성을 개선하기 위하여 단일 로그를 사용하는 것은 효과적인 방법이지만, NVMM에서는 빠른 광대역 동시 접근이 가능하기 때문에 복수의 로그를 사용하는 것이 성능에 긍정적인 효과를 가져올 수 있다.

이러한 3가지 관찰을 바탕으로 NOVA는 다음과 같은 5가지 특징을 갖는 파일 시스템으로 설계되었다.

- NOVA는 로그와 파일 데이터를 NVMM에 저장하며 빠른 검색을 수행하기 위한 radix tree [16]를 DRAM에 저장한다. radix tree는 리눅스 커널에서 다양하게 사용되었고, 많은 테스트를 거쳤으며, 충분히 완성도가 높아 NVMM에서 자료구조를 단순하고 효율적으로 구성하여 준다.
- 각각의 inode가 자신의 로그를 갖도록하여 동기화 없이 동시 수정이 가능하도록 한다. 이렇게 함으로써 NOVA는 복구시 다중 로그를 동시에 재현 (replay) 할 수 있으며 파일 접근 또한 높은 동시성 (concurrency)을 제공할 수 있게 된다.

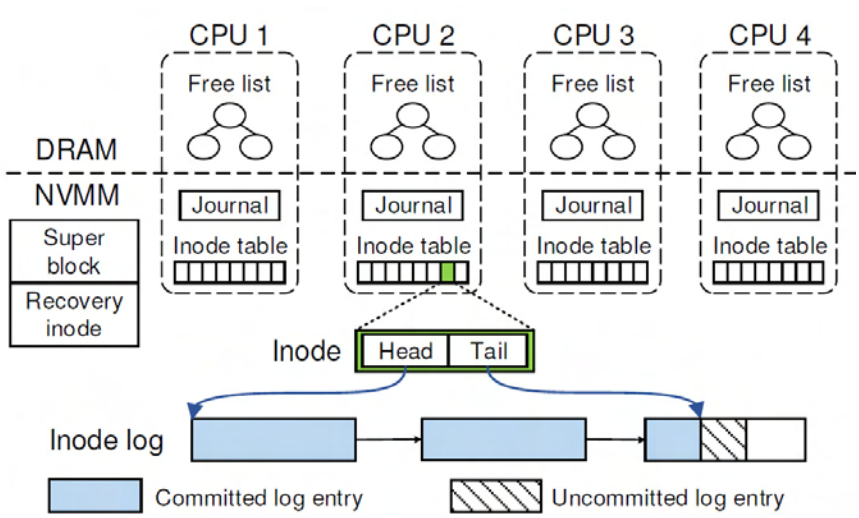
NOVA는 로그 구조 파일 시스템인데 이점은 저널링과 쉐도우 페이징 보다 더 저렴한 비용의 원자적 수정을 제공할 수 있게 한다. 로그에 원자적 데이터 쓰기를 하기 위해서 NOVA는 우선 데이터를 로그에 끼워두고 그 다음에 실제 수정을 실행하기 위해서 로그 끝 (tail)을 수정하게 된다. 이렇게 하면 저널링의 쓰기 중복 오버헤드와 쉐도우 페이징의 연속된 자료구조 갱신 오버헤드를 피할 수 있게 된다. move와 같은 디렉토리 관련 동작 실행시 여러 inode들을 선회하게 되는데, NOVA의 경우 복수의 로그들을 저널링을 통하여 원자적 (atomically) 업데이트를 수행한다. NOVA의 저널링은 데이터를 각 inode의 로그 끝에 쓰기 때문에 오직 로그 끝 (tail)만을 업데이트하면 된다. 이러한 점은 파일 데이터 혹은 메타 데이터와는 다르게 NOVA의 저널링을 상당히 경량화 시켜주는 요인이 된다.

디스크와는 다르게 NVMM에서는 지역성에

기반한 순차 로그의 장점이 상대적으로 작다. 광대역 동시 랜덤 접근이 가능하기 때문이다. 따라서 NOVA는 4KB NVMM 페이지를 위한 하나의 링크리스트에 로그와 다음 페이지를 위한 포인터를 저장한다. 이처럼 비순차적인 로그 구조는 3가지 이점을 제공하는데, 우선 대량의 연속된 공간이 필요없기 때문에 로그 공간 확보가 용이해 진다. 다음으로 로그 삭제가 페이지 크기 단위로 정밀하게 진행될 수 있다. 마지막으로 오래된 엔트리를 포함하는 로그 페이지들의 환원(reclaiming)을 단지 몇개의 포인터 할당만으로 수행할 수 있는 이점이 있다.

NOVA의 inode 로그들은 파일 데이터를 포함하지 않는다. NOVA는 수정된 페이지들을 위한 쓰기 복사 (copy-on-write)를 사용하고 로그에 그 쓰기에 대한 메타데이터를 첨부한다. 이 메타데이터는 해당 데이터 페이지와 업데이트를 기록한다. 파일 데이터의 쓰기 복사는 여러 이유로 상당히 유용하다. 우선 복구 절차를 가속할 수 있는 더 짧은 로그를 구성할 수 있게 하고, 가비지 콜렉션을 더 단순하고 효율적으로 만들어 준다. NOVA는 로그 페이지를 복원할 때 해당 로그로 부터 파일 데이터를 복사하지 않기 때문이다. 다음으로 DRAM 프리리스트 (free list)에서 페이지를 추가 삭제만으로 이전 페이지 복구와 새로운 데이터 페이지 할당이 가능하기 때문에 구현이 용이하다. 마지막으로 데이터 페이지 복구를 바로 할 수 있기 때문에 대량의 쓰기 부하 혹은 NVMM 접근 부하에서도 성능 유지가 가능하다.

이러한 특징들을 갖는 NOVA는 리눅스 커널 4.0에서 구현되었다. 또한 해당 커널에 있는 모든 NVMM 후크 (hooks)를 그대로 사용한다. 현재 리눅스 POSIX 파일시스템 테스트 수트 [17]를 통과하여 소스코드가 GitHub: https://github.com/NVSL/NOVA에 공개되어 있다. NOVA 파일시스템의 전체적인 구조, 원자성, 쓰기 순서 결정 메카니즘이 그림 1에 나타나 있다. NOVA는 확장성 제공을 위해 CPU마다 프리리스트, 저널, inode테이블을 갖는다. 각 inode는 4KB 로그 페이지의 단일 링크리스트로 구성된 분리된 로그를 갖는다. inode에서 끝 (tail) 포인터는 가장 최



(그림 1) NOVA 자료구조[1]

근에 반영된 로그 엔트리를 기록한다. 보다 세부적인 구현 상세는 [1]의 4장을 참조하기 바란다.

4. 결 론

하이브리드 메인 메모리를 위한 파일시스템은 기존의 메모리를 위한 것과는 다른 각 메모리 기술의 특징 및 장점을 충분히 살릴 수 있는 형태로 구성되어야 한다. 기존의 파일시스템은 새로운 메모리 기술에 대한 고려가 구현되어 있지 않기 때문에 그대로 적용하여 사용하면 성능이 저하되는 문제가 발생한다. NOVA는 가장 최근에 발표된 LFS로 하이브리드 메모리의 장점을 충분히 고려하여 설계되었다. 이러한 고려사항과 특징들은 향후 개발되는 하이브리드 메모리를 위한 파일시스템에서도 그대로 적용될 것으로 예상된다. 보다 단순하면서도 고성능을 제공할 수 있는 효과적인 가비지 콜렉션과 복구를 제공하는 기능은 모든 파일시스템이 제공해야 할 기본이기 때문이다. NVMM기반 하이브리드 메모리는 이러한 장점들로 인하여 조만간 메인메모리 시장을 크게 점유할 수 있을 것으로 예상된다. 다만 임베디드 시스템에 적용 가능한 하이브리드 메인 메모리에서는 이러한 기본 기능에 대한 요구조건이 크게 다를 수 있을 것이다. 다른 제약에 따라 복구 기능을 포기할 수도 있는 환경이기 때문이다. 이러한 환경에 적합한 파일시스템을 새롭게 고려하는 것도 향후 필요하다고 판단된다.

참 고 문 헌

[1] Jian Xu, Steven Swanson, “NOVA: A Log-structured File System for Hybrid Volatile/Non-volatile Main Memories”, 14th USENIX Conference on File and Storage Technologies (FAST ’16), 2016.

[2] P. Sehgal, S. Basu, K. Srinivasan, and K. Voruganti, “An Empirical Study of File Systems on NVM”, In Proceedings of the 2015 IEEE Symposium on Mass Storage Systems and Technologies (MSST’15), 2015.

[3] M. S. Bhaskaran, J. Xu, and S. Swanson, “Bankshot: Caching Slow Storage in Fast Non-volatile Memory”, In Proceedings of the 1st Workshop on Interactions of NVM/FLASH with Operating Systems and Workloads, INFLOW ’13, pp. 1:1–1:9, New York, USA, 2013.

[4] A. M. Caulfield, T. I. Mollov, L. A. Eisner, A. De, J. Coburn, and S. Swanson, “Providing safe, user space access to fast, solid state disks”, In Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS XVII, pp. 387–400, New York, USA, 2012.

[5] J. Condit, E. B. Nightingale, C. Frost, E. Ipek, B. Lee, D. Burger, and D. Coetzee, “Better I/O through byteaddressable, persistent memory”, In Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, SOSP ’09, pp. 133–146, New York, USA, 2009.

[6] J. Bonwick and B. Moore. ZFS: The Last Word in File Systems, 2007.

[7] 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, pp. 273–286, Santa Clara, CA, Feb. 2015.

[8] J. Ousterhout, A. Gopalan, A. Gupta, A. Kejriwal, C. Lee, B. Montazeri, D. Ongaro, S. J. Park, H. Qin, M. Rosenblum, S.

Rumble, R. Stutsman, and S. Yang, “The RAMCloud Storage System”, ACM Trans. Comput. Syst., 33(3), pp. 7:1–7:55, Aug. 2015.

[9] J. Condit, E. B. Nightingale, C. Frost, E. Ipek, B. Lee, D. Burger, and D. Coetzee, “Better I/O through byte addressable, persistent memory”, In Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, SOSP ’09, pp. 133–146, New York, USA, 2009.

[10] S. R. Dulloor, S. Kumar, A. Keshavamurthy, P. Lantz, D. Reddy, R. Sankaran, and J. Jackson, “System Software for Persistent Memory”, In Proceedings of the Ninth European Conference on Computer Systems, EuroSys ’14, pp. 15:1–15:15, New York, USA, 2014.

[11] M. Wilcox. Add support for NV-DIMMs to ext4. https://lwn.net/Articles/613384/.

[12] X. Wu and A. L. N. Reddy, “SCMFS: A File System for Storage Class Memory”, In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC ’11, pp. 39:1–39:11, NY, USA, 2011.

[13] H. Volos, S. Nalli, S. Panneerselvam, V. Varadarajan, P. Saxena, and M. M. Swift, “Aerie: Flexible File-system Interfaces to Storage-class Memory”, In Proceedings of the Ninth European Conference on Computer Systems, EuroSys ’14, pp. 14:1–14:14, NY, USA, 2014.

[14] S. Chen and Q. Jin, “Persistent B+-trees in Non-volatile Main Memory”, Proc. VLDB Endow., 8(7), pp. 786–797, Feb. 2015.

[15] I. Moraru, D. G. Andersen, M. Kaminsky, N. Tolia, P. Ranganathan, and N. Binkert, “Consistent, Durable, and Safe Memory Management for Byte-addressable Non Volatile Main Memory”, In Proceedings of the First ACM SIGOPS Conference on Timely Results in Operating Systems, TRIOS ’13, pp. 1:1–1:17, NY, USA, 2013.

[16] Trees I: Radix trees. https://lwn.net/Articles/175432/.

[17] Linux POSIX file system test suite. https://lwn.net/Articles/276617/.

저 자 약 력

조 중 석

이메일 : icaroosion@naver.com

- 2006년~2012년 국립순천대학교 전자공학과 학사
- 2012년~2014년 국립순천대학교 전자공학과 석사
- 2014년~현재 국립순천대학교 전자공학과 박사과정
- 관심분야 : 임베디드 시스템, 저전력 메모리 시스템, 시스템 소프트웨어, 머신러닝

조 두 산

이메일 : mew26@snu.ac.kr

- 2010년~2012년 국립순천대학교 전자공학과 전임강사
- 2012년~2014년 국립순천대학교 전자공학과 조교수
- 2014년~현재 국립순천대학교 전기전자공학부 부교수
- 관심분야 : 최적화 컴파일러, 임베디드 시스템, 저전력 메모리 시스템, 시스템 소프트웨어, 머신러닝, 인공지능