ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 42, No. 4, pp. 427-433, 2015. 4
http://dx.doi.org/10.5626/JOK.2015.42.4.427

뉴메모리를 이용한 고성능 및 호환성을 위한 I/O 변환 계층 기술

(I/O Translation Layer Technology for High-performance and Compatibility Using New Memory)

송 현 섭 † 문 영 제 † 노 삼 혁 ††
(Hyunsub Song) (Young Je Moon) (Sam H. Noh)

요 약 컴퓨팅 시스템이 급속도로 발전함에 따라 빠른 데이터 처리가 요구되고, 이를 위한 고성능 저장 장치 기술이 요구되고 있다. 차세대 메모리인 뉴메모리는 고성능 저장장치에 활용될 수 있는 장점을 가진다. 뉴메모리는 비휘발성을 가지고 있으며, DRAM (Dynamic Random Access Memory)에 가까운 접근 속도를 가지고 있어서, 업계 및 학계로 하여금 새로운 저장장치의 역할을 할 수 있을 것이라는 기대를 받고 있다. 본 연구는 뉴메모리를 저장장치로 활용하기 위한 기술로 NTL (New memory Translation Layer)을 제시한다. NTL은 기존의 디스크 파일시스템을 뉴메모리에서 사용할 수 있게 하여 높은 호환성을 제공하며, 블록 단위가 아닌 바이트 단위로 입출력 데이터를 처리하여 높은 데이터 처리량을 제공한다. 본 논문에서는 NTL의 설계에 대해 서술하며, NTL을 통해 얻는 성능 이점을 보여주기 위한 실험 결과를 제시한다.

키워드: 뉴메모리, 파일 시스템 호환성, 바이트 레벨의 입출력 처리, 고성능 입출력

Abstract The rapid advancement of computing technology has triggered the need for fast data I/O processing and high-performance storage technology. Next generation memory technology, which we refer to as new memory, is anticipated to be used for high-performance storage as they have excellent characteristics as a storage device with non-volatility and latency close to DRAM. This research proposes NTL (New memory Translation layer) as a technology to make use of new memory as storage. With the addition of NTL, conventional I/O is served with existing mature disk-based file systems providing compatibility, while new memory I/O is serviced through the NTL to take advantage of the byte-addressability feature of new memory. In this paper, we describe the design of NTL and provide experiment measurement results that show that our design will bring performance benefits.

Keywords: new memory, file system compatibility, storage, byte-level I/O, high-performance I/O

· 본 논문은 2012년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2012R1A2A2A01045733).
· 이 논문은 2014 한국컴퓨터종합학술대회에서 'NTL: 뉴메모리를 저장장치로 활용하기 위한 운영체제 소프트웨어 계층'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 홍익대학교 컴퓨터공학과
thdgustjq@gmail.com
lemonkenya@gmail.com
†† 종신회원 : 홍익대학교 컴퓨터공학과 교수(Hongik Univ.)
samhnoh@gmail.com
(Corresponding author임)

논문접수 : 2014년 8월 29일
(Received 29 August 2014)
논문수정 : 2014년 12월 12일
(Revised 12 December 2014)
심사완료 : 2015년 1월 13일
(Accepted 13 January 2015)

정보과학회논문지 제42권 제4호(2015. 4)

1. 서 론

오랜 기간 컴퓨팅 시스템의 저장장치였던 HDD (Hard Disk Drive)는 느린 기계적 작동방식으로 인해 미래의 고성능 저장장치에 활용되기에는 한계점을 가지고 있다. 저장장치 업계 및 학계에서는 차세대 메모리 기술인 뉴메모리를 활용하여 앞서 언급한 한계점을 극복하기 위한 방법을 찾고 있다[1,2]. 뉴메모리는 바이트 단위의 임의 접근이 가능하고, DRAM (Dynamic Random Access Memory)과 유사한 나노 초 단위의 접근 속도를 가지고 있다. 또한 비휘발성, 높은 내구성을 가지기 때문에 고성능 저장장치의 매체로 활용되기에 적합한 매체이다.

기존의 DRAM 및 플래시 메모리 기반 SSD (Solid State Drive)도 고성능 저장장치의 역할을 할 수 있는 매체로서 평가 받고 있다. 하지만, DRAM의 경우에는 휘발성 매체라는 특성 때문에 데이터의 지속적 보존이 어렵다는 단점을 가지고 있다. 그리고 SSD의 경우에는 HDD보다 높은 접근 속도를 가지고 있지만, 쓰기 증폭 현상(Write Amplification)으로 인한 입출력 성능 하락과 내구성의 문제점을 가지고 있다. 뉴메모리는 비휘발성 특성을 가지면서도 플래시 기반의 SSD에 비하여 상대적으로 빠른 접근속도와 높은 내구성을 가지고 있어서, 기존 매체들의 한계를 극복하는 것을 가능케 해준다. 따라서, 뉴메모리를 저장장치로 활용하는 주제의 연구가 필요하게 되었으며, 업계 및 학계에서는 그 필요에 따른 다각도의 뉴메모리 활용 연구를 진행하고 있다.

본 논문은 뉴메모리를 저장장치로 활용하는 NTL (New memory Translation Layer)기술을 제안한다. 이 기술은 최소한으로 운영체제를 수정하고, 독립적인 모듈로 제공되기 때문에 기존의 뉴메모리 전용 파일시스템에 비하여 호환성을 제공한다. 그리고 블록 변환 과정 없이 바이트 단위로 입출력 데이터를 처리하기 때문에 높은 데이터 처리량을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 언급한다. 3장에서는 NTL의 구조 및 동작 매커니즘을 설명하고, 4장에서는 구현 측면에서 고려되어야 할 사항들을 실험적으로 증명한 내용들을 보여준다. 마지막으로 5장에서 결론을 맺고 향후 연구에 대해 언급한다.

표 1 메모리 기술들의 성능 비교[3]
Table 1 Comparison of memory technologies[3]

	DRAM	STT-MRAM	PCM	NAND
Maturity	Product	Prototype	Product	Product
Read Latency	10ns	10ns	20~50ns	25us
Write Latency	10ns	10ns	80~500ns	200us
Erase Latency	-	-	-	200us
Endurance (writes/bit)	10^{16}	10^{16}	10^{6}~10^{8}	10^{5}
Cell Size	6~8F^2	>6F^2	5~10F^2	4~5F^2

2. 관련 연구

2.1 뉴메모리

뉴메모리는 PCM 또는 PRAM (Phase Change RAM)[4], STT-MRAM (Spin Transfer Torque MRAM)[5], 그리고 RRAM (Resistive RAM)[6]과 같은 다양한 형태로 개발되고 있는 차세대 메모리 기술이다. 뉴메모리는 비휘발성 매체이며, 바이트 단위의 임의 접근이 가능한 매체이다. 표 1은 뉴메모리와 기존의 메모리 기술과의 성능 비교를 보여준다. STT-MRAM과 PCM은 NAND 플래시 메모리보다 높은 접근 속도를 가지고 있으며, 특히 STT-MRAM은 DRAM과 유사한 속도를 보유하고 있다. 내구성 또한 STT-MRAM과 PCM이 NAND 플래시 메모리보다 뛰어난 것을 볼 수 있다. 뉴메모리가 가지고 있는 이러한 특성들은 미래의 저장장치의 역할을 뉴메모리가 담당할 것이라는 주장에 뒷받침 되는 근거이다.

2.2 기존의 뉴메모리를 저장장치로 활용한 연구

기존의 뉴메모리를 저장장치로 활용하는 연구는 크게 두 가지로 구분할 수 있다. 첫 번째는 소프트웨어 관점으로 접근한 뉴메모리 전용 파일시스템이다[7-10]. 뉴메모리 전용 파일시스템은 뉴메모리의 특성을 고려하여 개발되었기 때문에 바이트 단위의 임의 접근이 가능하다는 것이 특징이다. 그러나, 전용 파일시스템을 사용하기 위해서는 사용자들이 기존에 사용하고 있던 시스템 환경의 변경이 요구되었고, 기존의 디스크 기반 파일시스템에서 제공하는 스냅샷과 같은 기능들도 제공하지 못하였다. 두 번째는 하드웨어 관점으로 제시한 뉴메모리 기반 저장매체이다[11,12]. 뉴메모리 SSD라고도 불리는 이 기술은 운영체제의 변경 없이 기존의 블록 소프트웨어 계층을 통해 입출력이 수행되기 때문에 호환성을 보장해준다. 하지만, 블록 소프트웨어 계층을 기반으로 수행되는 입출력 방식으로 인해 뉴메모리의 장점인 바이트 단위 접근 가능성을 이용하지 못한다는 단점을 갖고 있다. 이는 결국 뉴메모리 매체의 성능을 최대로 끌어 올리지 못한다는 한계로 이어진다.

본 논문에서 제안하는 기술인 NTL은 앞서 언급한 두 가지 문제점을 해결한다. NTL은 기존의 시스템 환경과의 호환성을 위하여 기존 운영체제의 수정을 최소화하고, Ext4, NTFS, FAT등의 기존의 디스크 기반 파일시스템을 뉴메모리 저장장치에서 구동할 수 있게 해준다. 또한 NTL은 성능을 위하여 바이트 단위의 접근을 보장한다. 바이트 단위의 접근을 위해 운영체제의 소프트웨어 계층 중 블록 소프트웨어 계층을 우회하는 방식으로 입출력 계층을 구현하다. 이는 뉴메모리의 바이트

단위 접근성을 제공함과 동시에 불필요한 블록 변환 계층의 오버헤드도 없앨 수 있기 때문에 입출력 성능에 큰 이점을 가져다 줄 것이라 기대한다.

3. NTL 개요

3.1 NTL 구조

그림 1은 NTL의 시스템 구조이다. 그림 1에서 'The Upper Layer'는 기존의 파일시스템 상위의 계층을 통칭한다. 그림 1 좌측의 구조는 기존의 블록 기반 소프트웨어 계층 구조이며, 그림 1 우측은 뉴메모리를 위하여 추가된 소프트웨어 계층 구조이다. 오른쪽의 NTL 구조에서 볼 수 있듯이, NTL은 'The Upper Layer' 아래 즉, 기존의 파일시스템 계층 아래 존재한다. 이와 같이 NTL은 파일시스템 자체가 아닌, 특정 파일시스템 아래에 하나의 계층으로서 존재하기 때문에 기존의 디스크 기반 파일시스템을 그대로 사용할 수 있게 해준다. 이를 통해 NTL은 사용자들에게 호환성을 제공해준다.

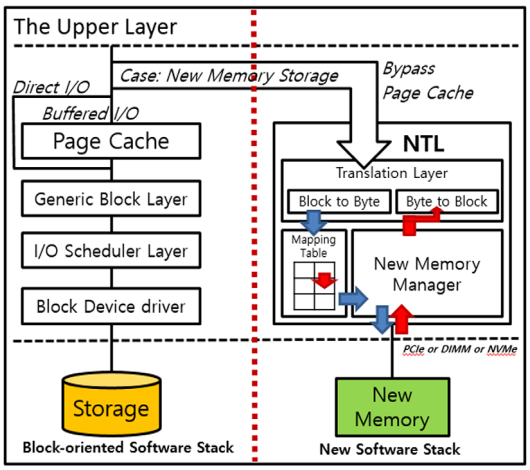


그림 1 NTL의 시스템 구조
Fig. 1 System architecture of NTL

NTL을 통해 뉴메모리로 접근하는 데이터는 블록 소프트웨어 계층을 우회하여 접근하기 때문에 성능적 부하를 줄일 수 있다. 또한 뉴메모리 매체 자체의 접근 속도가 빠르기 때문에 느린 장치로의 전송을 위한 버퍼링을 사용할 필요가 없다. 이는 버퍼링에 의한 성능적 부하를 줄일 수 있다.

3.2 NTL의 내부 구현

NTL은 그림 1 우측에서 볼 수 있듯이, 크게 세 가지 컴포넌트로 구성된다. 변환 계층, 매핑 테이블, 뉴메모리 관리자 이렇게 세 가지로 구성되는데, NTL을 구현하기 위해 우선적으로 필요한 부분은 뉴메모리 관리자이다. 뉴메모리 관리자는 뉴메모리를 문자 장치로 인식시키는 기능과 뉴메모리 장치의 데이터를 할당하고, 해제할 수 있는 자원 관리를 제공한다. 뉴메모리 관리자는 모듈 형태로 제공됨으로써 운영체제에 대한 호환성을 가진다. 내부는 바이트 단위로 데이터를 처리하는 문자 디바이스 드라이버 형태로 구현된다. nm_alloc(), nm_free()와 같은 자원 관련 인터페이스 및 기타 여러 뉴메모리 관리 연산들을 처리 할 수 있는 함수들도 제공한다.

변환 계층은 블록 단위로 입출력 되는 데이터를 바이트 단위로 변환해주는 컴포넌트이다. 파일 시스템으로부터 넘어오는 블록 단위의 I/O 데이터를 바이트 단위로 변환해주는 역할을 한다. 이 변환 계층을 통해 뉴메모리 본연의 특성인 byte-addressable을 가능케 한다.

매핑 테이블은 파일 시스템에서 넘어온 데이터의 변환 되기 전 메타 데이터 정보와 변환 후의 바이트 단위로 뉴메모리에 저장될 주소를 매핑 해 놓은 자료 구조이다. 자료 구조는 Radix tree를 이용하여 구현하였다.

3.3 NTL 동작 매커니즘

그림 1에서 보여주듯이 NTL의 데이터 입출력 방식은 페이지 캐시를 우회하는 Direct I/O와 유사하다. 이와 같은 입출력 방식을 설계한 이유는 뉴메모리와 같이 매체 자체의 접근 속도가 빠른 환경에서는 오히려 소프트웨어 계층의 부하가 전체 시스템 성능에 더 크게 영향을 미칠 수 있기 때문이다[11,13].

NTL의 내부에서 수행하는 데이터 입출력의 매커니즘을 입력 과정과 출력 과정으로 나누어 설명한다. 뉴메모리 장치에 대한 입력이 수행되면, 다음에서 블록 소프트웨어 계층을 우회하여 NTL 내부로 접근하게 된다. 'The Upper Layer'에서 받은 데이터는 변환계층에 접근하게 되는데, 그 중 'Block to Byte' 컴포넌트에 의해 바이트 단위의 데이터로 변환된다. 변환된 후 블록 단위 주소와 바이트 단위 주소를 담고 있는 매핑 테이블에 블록 단위 주소와 그에 해당하는 뉴메모리의 바이트 단위 주소를 입력한 후 데이터는 뉴메모리 관리자에게 전달된다. 그 후 뉴메모리 관리자는 바이트 단위로 뉴메모리에 데이터를 입력한다.

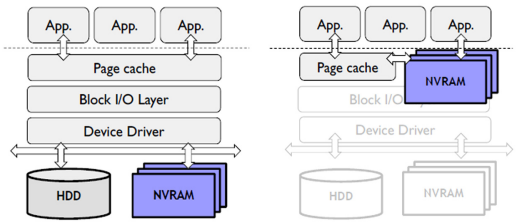
뉴메모리 장치에 대한 출력이 수행 될 때에는, 매핑 테이블에서 읽고자 하는 데이터의 블록 단위 주소에 해당하는 바이트 단위 주소를 찾아 뉴메모리 관리자에게 넘겨 준다. 뉴메모리 관리자는 그 주소를 이용하여 뉴메모리로부터 데이터를 읽은 후에 그 데이터를 변환계층의 'Byte to Block' 컴포넌트로 보내준다. 'Byte to Block' 컴포넌트는 바이트 단위의 데이터를 블록 단위의 데이터로 변환하여 'The Upper Layer'에서 수용할 수 있는 데이터인 블록 단위 데이터를 보내 주고 출력 과정을 마무리 한다.

4. 실 험

4.1 실험 개요

본 논문은 NTL의 높은 입출력 성능을 위해 블록 소프트웨어 계층과 페이지 캐시를 우회하는 데이터 입출력 방식을 제시하였다. 이 장에서는 앞서 언급한 데이터 입출력 방식들의 이점을 실험적으로 증명한 내용을 다루고, 기존의 뉴메모리를 저장장치로 활용한 연구와의 성능 비교에 대한 내용을 언급한다.

먼저, 실험에 이용한 기술은 그림 2에서 볼 수 있는 Ramdisk [14]와 Ramfs [15]이다. 이 두 기술은 DRAM을 저장장치로 활용하기 위한 기술이다. Ramdisk 는 DRAM을 블록 장치로 설정하는 기술인데, 이는 DRAM을 기존의 HDD와 같이 블록 소프트웨어 계층을 통해 입출력을 가능하게 만든 기술이다. 반면, Ramfs는 DRAM에 페이지 캐시 매커니즘을 입출력 처리로 활용하여 블록 소프트웨어 계층의 통과 없이 입출력이 수행될 수 있게 만든 DRAM 전용 파일시스템 기술이다. Ramfs가 파일시스템인 것과 페이지 캐시 매커니즘을 활용한다는 것은 우리의 NTL 기술과 다른 점이지만, 블록 소프트웨어 계층을 우회한다는 점은 유사하다고 할 수 있다.



(a) Ramdisk: block storage (b) Ramfs: byte-addressable storage

그림 2 뉴메모리를 저장장치로 활용하는 두 가지 가능한 구조[13]

Fig. 2 Two possible architectures for making use of new memory storage [13]

4.2 실험 내용

다음은 수행한 각각의 실험들에 대한 간략한 설명이다. 실험을 통해 얻고자 하는 내용에 대해 언급하고, 비교 대조군 및 실험 환경에 대해 설명한다.

4.2.1 블록 소프트웨어 계층의 우회

첫 번째 실험은, 뉴메모리를 저장장치로 활용하는 환경에서 블록 소프트웨어 계층을 우회하는 데이터 입출력 방식의 이점을 알아보기 위해 블록 소프트웨어 계층을 통해 입출력이 수행되는 Ramdisk와 블록 소프트웨어 계층을 우회하여 입출력이 수행되는 Ramfs의 성능을 비교하였다. Ramdisk, Ramfs를 이용하여 각각 총 DRAM의 용량 4GB중에서 2GB를 메인 메모리로 남겨놓고, 나머지 2GB를 저장장치 공간으로 구성하였다. 이 기술들의 기본적인 입출력 대역폭을 측정하기 위해 FIO [16]의 마이크로 벤치마크 툴인 순차 쓰기, 임의 쓰기, 순차 읽기, 임의 읽기를 이용하였다.

4.2.2 페이지 캐시의 우회

두 번째 실험은 뉴메모리와 같이 접근 속도가 빠른 매체를 저장장치로 활용하는 환경에서 페이지 캐시를 우회하는 데이터 입출력 방식의 이점을 알아보기 위한 실험이다. 실험을 위해 기존의 HDD를 저장장치로 구성한 환경과 Ramdisk 환경에서 각각 Buffered I/O와 Direct I/O의 성능을 비교하였다. 이 실험에서 Ramfs와 Ramdisk중에서 Ramdisk를 비교 군으로 설정한 이유는 Ramdisk의 환경이 매체를 제외하고는 HDD를 저장장치로 구성한 환경과 동일하기 때문이다. Ramdisk와 Ramfs의 용량은 위의 실험과 동일하게 총 메모리 용량 4GB중에서 2GB로 구성하였다. 이 실험에서도 기본적인 입출력 대역폭을 측정하기 위해 FIO의 마이크로 벤치마크 툴들을 이용하였다. 또한, 실제 워크로드에서의 성능을 측정하기 위해 Filebench [17]의 매크로 벤치마크 툴인 File Server, Mail Server, Web Server를 이용하였다.

4.2.3 NTL 저장장치 및 기존의 저장장치 기술 비교

세 번째 실험은, NTL을 적용시킨 뉴메모리 저장장치와 기존의 저장장치 기술들의 성능을 비교하기 위한 실험이다. 비교 대조군은 NTL 저장장치, BPFS, Ramdisk, 플래시 메모리 기반 SSD이다. BPFS는 뉴메모리 전용 파일 시스템이며, Ramdisk는 뉴메모리 기반 SSD와 같이 DRAM을 블록 장치로 사용한 기술이다. 이 둘은 본 연구의 동기가 되는 기술이다. 플래시 메모리 기반 SSD는 현재 고성능을 위한 저장장치로서 사용되는 대표적인 매체라고 할 수 있는데, 결과에서 보여주는 처리량의 객관성을 보여주기 위해 비교 대조군으로 넣었다.

NTL 저장장치는 뉴메모리 관리자를 통해 DRAM을 뉴메모리 영역으로 에뮤레이트하여 사용하였다. DRAM을 뉴메모리로 에뮤레이트하여 실험한 이유는, 본 연구팀에서 보유하고 있는 뉴메모리의 용량은 대역폭 실험의 당위성을 보여주기에는 적절하지 않다는 판단하에 결정하였다. 앞선 뉴메모리 관련 동향에서도 살펴보았듯이, 뉴메모리는 DRAM과 거의 근접한 빠른 속도를 가지고 있기 때문에 DRAM을 뉴메모리로 에뮤레이트하여 도출해낸 결과도 신뢰성이 있다고 판단한다.

실험은 FIO의 마이크로 벤치마크 툴을 이용하였으며, 각각 비교 대조군의 총 용량은 모두 10GB로 설정하였다. Total I/O는 5GB이며, 하나의 스레드를 이용하여 100개의 파일로 나누어 실험을 수행하였다.

4.3 실험 결과

4.3.1 블록 소프트웨어 계층의 우회

그림 3은 Ramfs와 Ramdisk의 성능 비교 결과 그래프이다. Ramfs는 본 논문에서 제안하는 NTL과 같이 블록 소프트웨어 계층을 우회하여 입출력이 일어난다. 반면에, Ramdisk에서는 블록 소프트웨어 계층을 통해 데이터의 입출력이 일어난다. 이 실험을 통해 각각의 입출력 상황에서 블록 소프트웨어 계층의 오버헤드가 미치는 성능적 영향을 알아보고자 하였다.

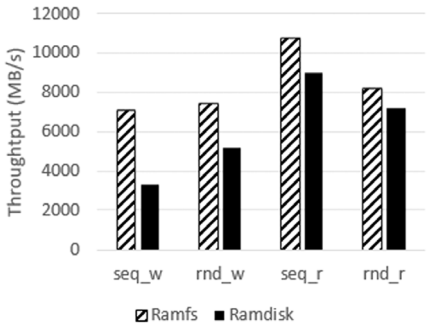


그림 3 블록 소프트웨어 계층으로 인한 성능 저하

Fig. 3 Overhead of block software layer

실험 결과를 보면, 모든 입출력 상황에서 블록 소프트웨어 계층을 우회하여 입출력이 일어나는 Ramfs의 성능이 블록 소프트웨어 계층을 통해 입출력이 일어나는 Ramdisk의 성능 보다 앞서는 것을 볼 수 있다. 이 결과를 통해 뉴메모리와 같이 자체의 접근 속도가 빠른 매체를 저장장치로 활용할 경우 블록 소프트웨어 계층과 같이 오버헤드가 높은 소프트웨어 계층은 불필요한 성능 저하를 일으킨다는 것을 알 수 있다. 본 연구팀은 논문에서 제안하고 있는 NTL의 블록 소프트웨어 계층을 우회하는 입출력 구조가 NTL의 성능적인 측면에서 상당한 이점을 가져다 줄 수 있다고 기대한다.

4.3.2 페이지 캐시의 우회

페이지 캐시가 성능에 미치는 영향을 알아보기 위한 실험은 두 가지로 나누어 수행하였다. 첫 번째는, 단일 입출력 상황에서 즉, 캐시 히트 및 캐시 미스 등 페이지 캐시 매커니즘이 발생하지 않는 상황에서 단순한 페이지 캐시 계층이 성능에 미치는 영향을 알아보기 위한 실험이다. 두 번째는, 실제 워크로드에서 페이지 캐시 매커니즘이 일어날 경우 그 매커니즘의 오버헤드가 성능에 미치는 영향을 알아보기 위한 실험이다.

그림 4는 첫 번째 내용의 결과를 보여준다. HDD와 Ramdisk에서 Buffered I/O와 페이지 캐시를 우회하는 Direct I/O의 성능을 단일 입출력 상황에서 비교한 실

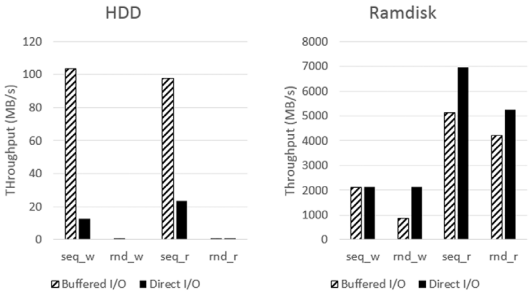


그림 4 메모리 기반 저장장치에서 페이지 캐시가 미치는 영향(Micro Benchmark)

Fig. 4 Overhead of page cache in memory-based storage (Micro Benchmark)

험 그래프이다. 결과에서 볼 수 있듯이, 매체 자체의 접근 속도가 느린 HDD에서는 페이지 캐시를 이용하는 입출력 방식이 큰 이점을 가져다주지만, 접근 속도가 빠른 DRAM에서는 페이지 캐시를 우회하는 것이 더욱 높은 성능을 가져다준다.

Ramdisk의 실험 결과는 단일 입출력 상황이기 때문에 페이지 캐시 매커니즘이 발생하지 않는 경우이다. 이 경우에서도 페이지 캐시를 사용하지 않는 Direct I/O가 더 빠른 결과를 보이는데, 이는 페이지 캐시의 계층적 오버헤드도 빠른 매체를 저장장치로 이용할 때는 성능에 영향을 미칠 수 있다는 것을 보여준다.

그림 5는 두 번째 내용의 결과를 보여준다. 그림 4의 실험과 매체 및 입출력 방식은 동일하게 비교하였다. 단일 입출력 상황을 실제 워크로드 상황의 실험으로 바꾼 것이 차이점이다. 이 실험에서는 실제 워크로드를 이용하였기 때문에 앞선 실험에서 보지 못했던 캐시 히트 및 캐시 미스와 같은 페이지 캐시의 활동이 활발할 때 생기는 오버헤드가 입출력 성능에 미치는 결과를 알 수 있다. 각각의 워크로드에 대한 특성을 간단하게 설명하면, File Server는 write와 read의 비율이 2:1인 워크로드이다. Mail Server는 write와 read가 골고루 분포된 것이고, Web Server 90%가 read로 구성된 read 집중 워크로드이다.

결과를 살펴보면 HDD의 경우에는 역시나 페이지 캐시를 이용하는 입출력 방식이 높은 성능을 가지는 것을 볼 수 있지만, Ramdisk의 경우에는 페이지 캐시를 이용하지 않는 입출력 방식이 페이지 캐시를 이용하는 입출력 방식에 비해 거의 같거나 격차가 크지는 않지만 조금 씩 높은 결과를 보이는 것을 볼 수 있다. 이는 접근 속도가 느린 저장 매체에서는 페이지 캐시의 활동이 가시적일 수 없는 오버헤드이지만, 접근 속도가 빠른 매체에서는 그 오버헤드가 전체 입출력 성능에 가시적인

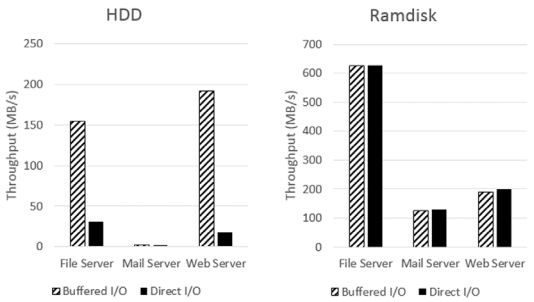


그림 5 메모리 기반 저장장치에서 페이지 캐시가 미치는 영향(Macro Benchmark)

Fig. 5 Overhead of page cache in memory-based storage (Macro Benchmark)

영향을 미칠 수 있다는 것을 보여주고 있다.

앞선 페이지 캐시의 두 가지 실험을 통해 페이지 캐시의 계층적 오버헤드, 매커니즘적 오버헤드 모두가 접근속도가 빠른 매체에서는 불필요한 성능 저하의 결과가 될 수 있다는 것을 알 수 있다. NTL은 페이지 캐시 자체를 우회하기 때문에, 블록 소프트웨어 계층을 우회하면서 얻는 성능적 이점에 페이지 캐시의 오버헤드까지 줄일 수 있다는 장점을 가지고 있다. 또한 이는 운영체제 소프트웨어 계층 관점에서도 페이지 캐시 계층이 없는 더욱 간결화된 계층 구조를 가질 수 있게 하였다.

4.3.3 NTL 저장장치 및 기존의 저장장치 기술 비교

그림 6은 NTL을 적용시킨 뉴메모리 저장장치와 기존의 저장장치 기술들의 성능을 비교한 결과이다. NTL 저장장치, Ramdisk, 플래시 메모리 기반 SSD는 모두 Ext4 파일 시스템으로 포맷하였다. 그래프의 BPFS의 처리량은 관련 논문에서 제시한 수치이다. BPFS의 실험 환경은 본 연구에서 실험하는 다른 비교 대조군의 환경과는 다소 차이가 있을 것으로 예상한다. 결과에서 볼 수 있듯이 본 연구팀에서 구현한 NTL 저장장치가 BPFS, Ramdisk, 플래시 메모리 기반 SSD 보다 높은 처리량을 보이고 있다. 기존 기술들 보다 높은 처리량을 보이는 이유는 앞선 실험에서 증명하였던 불필요한 계층들을 우회하는 것에 대한 이점, 그리고 NTL 내부 컴포넌트에서 byte-addressable I/O를 가능케 하여 매체 본연의 장점을 보장했기 때문이라고 생각한다.

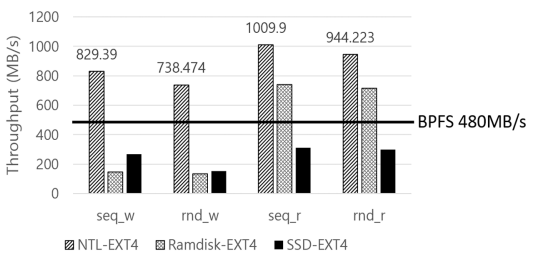


그림 6 NTL 저장장치와 기존 저장장치 기술의 성능 비교

Fig. 6 Performance comparison of NTL storage and traditional storage technologies

5. 결론 및 향후 연구

NTL은 뉴메모리를 차세대 저장장치로서 효율적으로 활용하기 위하여 제안된 새로운 입출력 계층이다. NTL은 기존의 디스크 기반 파일시스템을 사용할 수 있는 호환성을 제공한다. 그리고 나노 초 단위의 바이트 단위 접근성을 이용하여 높은 입출력 처리 성능을 제공한다. 또한 NTL을 통해 뉴메모리로 데이터가 접근할 때 블록 소프트웨어 계층을 거치치 않기 때문에 성능 부하를 줄일 수 있으며, 입출력 계층 내에 있는 불필요한 캐시 및 버퍼링을 제거하여 뉴메모리 본연의 접근 속도를 보장할 수 있도록 하였다.

향후 연구로는 저장장치로 사용되는 뉴메모리에 적합한 데이터 할당 정책과 데이터의 신뢰성을 제공하기 위한 매커니즘을 연구할 것이다.

References

[1] M. H. Kryder, and C. Kim, "After Hard Drives-What Comes Next?," *IEEE Transactions on Magnetics*, 2009.

[2] R. F. Freitas, and W. W. Wilcke, "Storage-class memory: The next storage system technology," *IBM Journal of Research and Development*, 2008.

[3] S. Eilert, M. Leinwander, and G. Crisenza, "Phase Change Memory: A new memory technology to enable new memory usage models," *Proc. the 1st IEEE Int'l Memory Workshop (IMW)*, 2009.

[4] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," *Proc. the 36th Annual International Symposium on Computer Architecture (ISCA)*, 2009.

[5] E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," *Proc. the 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013.

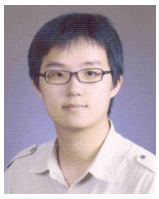
[6] M. Jung, J. Shalf, and M. Kandemir, "Design of a Large-Scale Storage-Class RRAM System," *Proc. the 27th International Conference on Supercomputing (ICS)*, 2013.

[7] J. Condit, E. B. Nightingale, C. Frost, E. Ipek, B. Lee, D. Burger, and D. Coetzee, "Better I/O through byte-addressable, persistent memory," *Proc. ACM Symposium Operating Systems Principles (SOSP)*, 2009.

[8] X. Wu, and A. L. Reddy, "SCMFS: a file system

for storage class memory," *Proc. ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2011.
[9] S. R. Dulloor, S. Kumar, A. Keshavamurthy, P. Lantz, D. Reddy, R. Sankaran, and J. Jacksonand, "System Software for Persistent Memory," *Proc. the Ninth Eurapean Conference on Computer Systems (EuroSys)*, 2014.
[10] H. Volos, S. Nalli, S. Panneerselvam, V. Varadarajan, P. Saxena, and M. M. Swift, "Aerie: Flexible File-system Interfaces to Storage-class Memory," *Proc. the Ninth European Conference on Computer Systems (EuroSys)*, 2014.
[11] A. M. Caulfield, A. De, J. Coburn, T. I. Mollov, R. K. Gupta, and S. Swanson, "Moneta: A High-Performance Storage Array Architecture for Next-Generation, Non-volatile Memories," *Proc. IEEE/ACM Symposium Microarchitecture (Micro)*, 2010.
[12] A. Akel, A. M. Caulfield, T. I. Mollov, R. K. Gupta, and S. Swanson, "Onyx: a prototype phase change memory storage array," *Proc. USENIX Hot topics in Storage and File systems (HotStorage)*, 2011.
[13] E. Lee, H. Bahn, S. Yoo, and S. H. Noh, "Empirical Study of NVRAM Storage: An Operating System's Perspective and Implications," *Proc. IEEE 22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2014.
[14] Ramdisk [Online]. Available: http://lxr.free-electrons.com/source/Documentation/blockdev/ramdisk.txt?v=3.10
[15] Ramfs [Online]. Available: http://lxr.free-electrons.com/source/Documentation/filesystems/ramfs-rootfs-initramfs.txt?v=3.10
[16] Fio [Online]. Available: http://freecode.com/projects/fio
[17] Filebench [Online]. Available: http://filebench.sourceforge.net

송 현 섭
2014년 홍익대학교 컴퓨터공학과(학사)
2014년~현재 홍익대학교 컴퓨터공학과 석사과정. 관심분야는 운영체제, 차세대 비휘발성램 소프트웨어, 고성능 스토리지



문 영 제
2008년 홍익대학교 컴퓨터공학과(학사)
2010년 홍익대학교 컴퓨터공학과(석사)
2010년~현재 홍익대학교 컴퓨터공학과 박사과정. 관심분야는 운영체제, 내장형시스템, 차세대 비휘발성램 소프트웨어

노 삼 혁
정보과학회논문지
제 42 권 제 2 호 참조