비휘발성 메모리를 고려한 sendfile() 개선

황태호 ○ 박성순 원유집
한양대학교 글루시스/안양대학교 한양대학교
htaeh@hanyang.ac.kr, sspark@gluesys.com, yjwon@hanyang.ac.kr

Improving sendfile() on NVRAM

Taeho Hwang○ Sung-Soon Park Youjip Won
Hanyang Univ. Gluesys/Anyang Univ. Hanyang Univ.

요 약

STT-MRAM, 3D XPoint와 같은 비휘발성 메모리가 개발되고 있다. 비휘발성 메모리는 바이트 단위 접근성과 비휘발성 특성을 가지고 있고, 해당 특성을 고려하여 다양한 소프트웨어 계층이 개선되고 있다. 비휘발성 메모리가 적용된 환경에서 직접 바이트 단위로 접근이 가능한 파일을 페이지 캐시에 이동시키는 것은 불필요한 데이터 복사를 유발한다. sendfile은 read+send 형식으로 데이터를 전송하는 과정에서 데이터 복사를 줄이기 위해 사용되는 함수이다. 하지만, 기존 sendfile은 비휘발성 메모리가 적용된 환경에서도 페이지 캐시를 사용한다. 우리는 페이지 캐시를 bypass하는 DAX 기능을 페이지 캐시를 사용하는 sendfile에 적용하였다. 우리는 성능 측정을 통해 기존 sendfile보다 DAX 기능을 사용하는 sendfile이 최대 62%의 성능 향상을 보이는 것을 확인하였다.

1. 서 론

STT-MRAM [1], 3D XPoint [2], PCM [3]과 같은 비휘발성 메모리가 개발되고 있다. 비휘발성 메모리는 바이트 단위 접근이 가능하고, DRAM과 비슷한 성능을 제공한다. 비휘발성 메모리의 특성을 활용하여 블록 단위 인터페이스를 사용하던 파일시스템, 데이터베이스, 자료구조 등을 개선하려는 연구들이 진행되고 있다 [4]. 비휘발성 메모리 상의 데이터를 바이트 단위로 직접 접근이 가능하기 때문에 Ext4, XFS와 같은 파일시스템에서는 페이지 캐시를 사용하지 않고 파일 데이터 등을 직접 접근하는 DAX 기능을 제공하고 있다 [5].

서로 다른 호스트에서 동작하는 응용프로그램들이 파일을 공유하기 위해선 client 역할을 수행하는 응용프로그램이 read 시스템 콜을 통해서 파일을 사용자 버퍼로 읽어 들인 다음, send 시스템 콜을 통해서 파일을 server에 전송하였다. 해당 과정은 파일 데이터를 사용자 버퍼로 읽어 들인다는 과부하가 있다. 해당 과부하를 제거하기 위해서 sendfile 시스템 콜이 추가되었다 [6]. sendfile은 데이터를 사용자 버퍼로 읽어 들이지 않고, 페이지캐시까지 읽어 온 데이터를 네트워크로 전송하는 시스템 콜이다. sendfile은 zero-copy 방식으로 데이터 복사 과부하를 제거하여 데이터 전송 성능을 향상시킬 수 있다. 비휘발성 메모리가 적용된 환경에서, 바이트 단위 접근이 가능한 파일을 페이지 캐시로 읽어 들이는 과정은 과부하이다. 하지만, 기존 sendfile은 비휘발성 메모리를 고려하지 않았기 때문에, 데이터 전송 시 페이지 캐시까지 데이터를 읽어들인다.

본 논문에서 우리는 비휘발성 메모리가 적용된 환경에서 데이터 복사 과부하를 제거한 sendfile with DAX를 제안한다 (그림 1). DAX 기능은 페이지 캐시를 bypass하여 데이터를 접근하기 때문에 sendfile 시 DAX 기능을 사용할 경우 페이지 캐시로 데이터를 읽어 오는 과정을 제거할 수 있다. 우리는 sendfile with dax를 리눅스 커널에 구현하였고, 실험을 통해서 기존 sendfile 보다 sendfile with dax가 파일 전송 시 4KB 크기의 레코드를 기준으로 최대 62%의 성능 향상을 확인하였다.

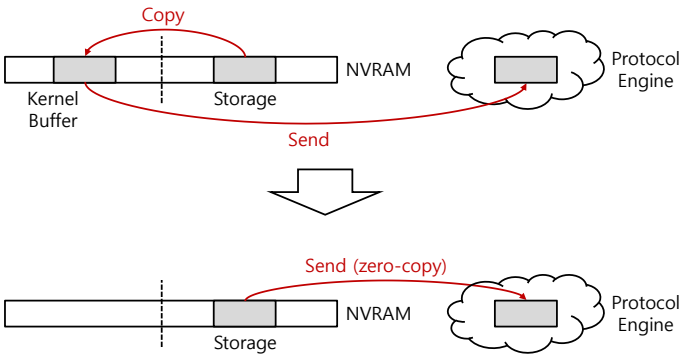

그림 1. Sendfile with DAX

2. Sendfile with DAX

DAX 기능은 dax_io 함수를 통해서 수행된다. dax_io 함수는 read 또는 write 할 파일, 파일 오프셋, 레코드 크기 등을 인자로 받는다. dax_io 함수는 크게 세 단계로

구분된다. 첫 번째로, read 또는 write의 대상이 되는 파일 블록이 파일시스템 상에서 어느 블록에 위치하고 있는지 확인한다. 해당 과정은 파일시스템 별로 다르기 때문에, ext4_get_block 과 같은 특정 함수가 사용된다. 두 번째로, 파일시스템 상의 블록이 메모리 상의 어느 블록에 위치하고 있는지를 확인한다. 해당 과정은 dax_get_addr 함수를 통해서 수행된다. 마지막으로, 획득된 메모리 블록과 사용자 버퍼 간의 read 또는 write를 memcpy를 통해서 수행한다. 해당 3단계는 블록 단위로 반복되어 수행된다.

우리는 DAX 기능을 sendfile에서 사용할 수 있도록, DAX 기능을 수행하는 dax_io 함수를 참조하여 sendfile용 dax_io_sendfile 함수를 구현하였다. dax_io에서 memcpy를 하는 부분은 사용자 버퍼와 파일 블록 간의 데이터 이동을 수행하는 것이다. 따라서, 우리는 해당 부분을 사용자 버퍼가 아닌 네트워크 소켓으로 대체하기위해 memcpy가 아닌 kernel_sendpage로 변경하였다. sendfile을 위해 두 파일 간의 데이터 이동을 수행하는 do_splice_direct 함수가 사용된다. 해당 함수에서 output file은 네트워크 소켓으로 지정된다. 해당 함수는 데이터를 읽고 네트워크 소켓으로 전송하는 역할을 수행하는 splice_direct_to_actor 함수를 호출한다. 우리는 DAX 기능을 sendfile에 적용하기 위해 splice_direct_to_actor 함수를 dax_io_sendfile로 대체하였다.

3. 실 험

실험에 사용된 CPU는 Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz이고, 32GB DRAM을 사용하였다. 32GB DRAM 중 16GB를 비휘발성 메모리로 모사하여 사용하였다. 우리는 sendfile with dax를 리눅스 4.3 버전에 구현하였고, 해당 버전에서 실험을 진행하였다. 실험을 위한 마이크로벤치마크를 작성하였다. 해당 벤치마크는 client에서 server로 sendfile을 통해 파일을 전송한다. 성능 측정은 client에서 파일을 전송 시에 각각의 sendfile에 소요되는 시간을 측정하였다. 모든 실험은 5번씩 진행되었고, 평균 값을 실험 결과로 사용하였다. 네트워크 지연시간을 제거하기 위해 client와 server는 동일한 호스트에서 실행하였다. sendfile은 전송할 파일 레코드의 크기를 지정할 수 있다. 우리는 4K, 16KB, 64KB, 256KB의 파일 레코드에 대해서 성능을 측정하였다 (그림 2). 페이지 캐시를 사용하는 기존 sendfile 보다 DAX를 통해 페이지 캐시를 거치지 않고 파일을 바로 전송하는 sendfile with dax가 각각의 레코드 크기에 대해서 62%, 40%, 41%, 57% 성능 향상을 나타내는 것을 확인하였다.

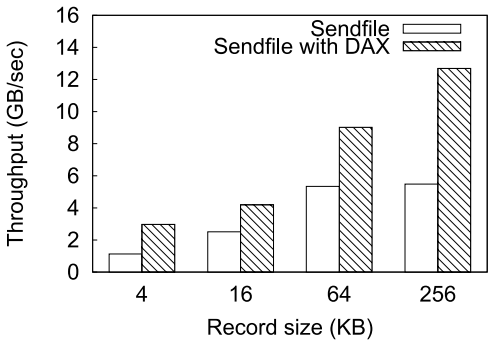

그림 2. sendfile 성능 비교

4. 결 론

비휘발성 메모리가 개발 및 상용화에 가까워짐에 따라 응용프로그램에서 커널까지 소프트웨어 계층이 비휘발성 메모리를 고려하여 개선되고 있다. 본 논문에서 우리는 비휘발성 메모리를 고려하여 sendfile을 개선하였다. DAX 기능을 sendfile에 적용 시 파일 전송 성능이 최대 62% 개선 되는 것을 확인하였다.

Acknowledgment

본 연구는 한국연구재단 기초연구실 지원사업(No. 2017R1A4A1015498), 정보통신기술진흥센터 지원사업[R7117-16-0232, 32Gbps 데이터 서비스를 위한 익스트림 스토리지 입출력 기술 개발], 그리고 IITP 전문연구실 지원사업(2018-0-00549)의 지원을 받아 수행되었음.

참고 문헌

[1] Khvalkovskiy, A. V., et al. "Basic principles of STT-MRAM cell operation in memory arrays." Journal of Physics D: Applied Physics 46.7 (2013): 074001.

[2] Intel. 2015. Intel and Micron produce breakthrough memory technology.

[3] Wong, H-S. Philip, et al. "Phase change memory." Proceedings of the IEEE 98.12 (2010): 2201-2227.

[4] Hwang, Taeho, Jaemin Jung, and Youjip Won. "Heapo: Heap-based persistent object store." ACM Transactions on Storage (TOS) 11.1 (2015): 3.

[5] Wilcox, Matthew, “DAX: Page cache bypass for filesystems on memory storage”, LWN (2014)

[6] Stancevic, Dragan. "Zero copy I: user-mode perspective." Linux Journal 2003.105 (2003): 3.