매니코어 환경에서 F2FS 파일시스템의 단일 파일 읽기 성능 개선

서동주º 주용수 임성수 임은진
국민대학교 소프트웨어융합대학
{commisori, ysjoo, sslim, ejim}@kookmin.ac.kr

Improving Single File Read Performance of F2FS Filesystem in Manycore Environment

Dongjoo Seoº, Yongsoo Joo, Sung-Soo Lim, Eun-Jin Lim
School of Computer Science Kookmin University

요 약

Flash SSD 저장장치의 사용이 늘어남에 따라 HDD에 최적화된 기존 파일 시스템에 대한 대안으로 SSD에 최적화된 F2FS 파일 시스템이 등장하였다. 하지만 매니코어 환경에서의 F2FS의 성능 분석은 충분히 이루어지지 않았다. 본 논문에서는 다수의 CPU 코어가 F2FS 파일 시스템 상의 단일 파일에 대한 직접 읽기를 수행할 때 심각한 입출력 성능 저하가 발생함을 확인하고 그 원인으로 F2FS가 직접 읽기 요청시에 파일에 대한 잠금을 유지하는 문제가 있음을 파악하였다. 이를 개선하기 위해 읽기 요청시에는 파일 잠금을 적용하지 않도록 구현을 수정한 결과 최대 120개의 코어가 동시에 읽기 요청을 발생시키는 상황에서도 저장장치의 최대 성능을 확보할 수 있음을 보였다.

1. 서 론

매니코어 환경은 다수의 CPU 코어를 가지고 있는 단일 장치 환경을 뜻한다. 적은 수의 코어가 다수의 작업을 처리하던 기존 환경과는 다른 환경이므로 메모리, CPU 스케줄러, 파일 시스템 등 다양한 분야에 관한 확장성 연구가 이루어지고 있다. F2FS(Flash-Friendly File System)[1]는 Flash 기반의 SSD 장치에 대해 기존의 EXT4 파일시스템보다 더 좋은 성능을 얻는 것으로 알려져 있다. 하지만 F2FS가 매니코어 환경에서도 EXT4에 대해 성능 우위를 보이는지는 충분한 분석이 이루어지지 않았다.

본 연구에서는 파일시스템의 성능을 평가하는 다양한 성능지표 중에서 단일 파일에 대한 임의 읽기 성능을 집중적으로 분석하였고, 그 과정에서 입출력 요청을 발생시키는 CPU 코어의 개수가 증가함에 따라 F2FS의 읽기 대역폭이 급격하게 저하되는 현상을 확인하였다. 이에 대한 원인을 분석한 결과 F2FS가 입출력 요청을 생성하는 최종 단계에서 읽기 요청과 쓰기 요청에 대한 구분이 없이 해당 파일에 대해 항상 잠금을 적용하고 있음을 파악하였다. 이로 인해 읽기 요청을 발생시키는 CPU 코어 수가 증가함에도 불구하고 블록 계층과 저장 장치에서 제공하는 병렬성을 이용하지 못하게 되었다. 본 연구에서는 이와 같은 비효율성을 해결하기 위하여 읽기 요청에 대해서는 잠금을 사용하지 않도록 구현을 개선하였다. 120개의 CPU 코어로 구성된 매니코어 시스템에서 NVMe SSD에 대해 단일 파일 임의 읽기 성능을 측정한 결과 기존 구현에 비해 약 2배의 IOPS(IO operations Per Second) 향상을 보였으며 입출력 요청을 발생시키는 CPU 코어의 개수가 증가하는 경우에도 NVMe SSD의 최대 성능을 유지할 수 있음을 확인하였다. 제안된 방법은 Linux Kernel 5.7 버전의 F2FS에 적용되었다.[2]

* 이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2014-0-00035, 매니코어 기반 초고성능 스케일러블 OS 기초연구)
* 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2018R1D1A1B05044558)

2. 배경 지식

2.1 Flash 기반 저장장치

Flash 기반의 저장장치는 NAND, NOR 등을 이용한 셀에 데이터를 저장하거나 불러올 수 있는 장치를 일컫는다. 이 중, NAND 기반의 저장장치가 널리 사용되고 있다. 장치는 MTD(Memory Technology Device)와 FTL(Flash Translation Layer)을 통해 블록과 페이지라는 단위로 내부를 관리하며 운영체제로 통하는 인터페이스를 제공한다.

이러한 장치들의 대표 특성은 저장장치 내부에서 병렬성을 지원하는 것이다. 기존 디스크 기반 저장장치의 경우에 하나의 헤드가 움직이며 모든 요청을 처리하였지만, Flash 기반의 저장장치의 경우에 여러 요청을 동시에 처리하는 것을 FTL을 통해 지원한다. 이 외에도, 덮어쓰기가 불가한 특성이나 Flash 셀들의 생명주기가 같지 않는 등 다양한 특성이 있다.

2.2 F2FS(Flash-Friendly File System

F2FS는 로그 기반의 파일 시스템 중 하나이다. 로그 기반의 파일 시스템은 데이터를 로그에 기록하여 관리하고, 데이터가 바뀔 경우에 이전 데이터를 무효로 처리하고 사용하지 않는 로그를 가비지 컬렉션을 이용하여 지우는 파일 시스템들을 얘기한다.

CPU	Intel Xeon E7-4800/8800 v2
RAM	755GB
# of CPU	120
커널 버전	5.4
Nvme SSD	Samsung 960 pro

표 1: 실험 환경

# of request process	IOPS(k)	ratio of osq_lock
8	287.6	54.22%
15	150.6	72.1%

표 2: 요청 프로세스 숫자에 따른 F2FS 분석 결과

F2FS는 데이터를 관리하기 위하여 여러 개의 임의 읽기 및 쓰기 영역을 이용한다. 또한 데이터를 저장하기 위하여 데이터를 자주 쓰이는 정도에 따라 세가지(Hot/Warm/Cold)로 구분하여 정리하는 정책을 사용한다. 이를 통해 F2FS는 데이터를 세그먼트라는 단위로 구분하여 순차 읽기와 순차 쓰기 영역으로 사용한다.

2.3 EXT4

EXT4(EXTended file system)[3]는 리눅스에서 차용한 대표적인 파일시스템으로서 대형 파일을 지원하고 EXT3 에서 가지는 블록 사상을 대체하기 위한 Extent 기법, 지연 할당 기법, 멀티 블록 할당 기법등을 특징으로 가지고있다.

2.4 매니코어 환경에서의 파일시스템 확장성

매니코어 환경은 많은 숫자의 CPU 코어를 가진 컴퓨팅 환경을 뜻한다. 이때 Linux 시스템의 확인하기 힘들었던 다양한 확장성 문제들과 마주치게 된다[4][5]. 그 이유는 기존 시스템의 경우 작업 프로세스의 양이 증가하면 CPU 코어의 숫자가 부족할 것이라고 고려해서 만든 시스템이 대부분이기 때문이다.

파일 시스템 부분도 위와 같은 이유로 NOVA 파일 시스템에 대한 확장성 실험[6]이나 매니코어 환경에서의 입출력 확장성 분석[7]과 같은 연구들이 진행되었다. 특히 F2FS에 관한 기존 연구[8]에서도 단일 파일에 대한 쓰기 요청의 확장성에 관한 연구를 범위 기반 잠금을 이용해 크게 끌어올린 바 있다. 하지만 본 논문에서 다룰 문제에 대해서는 추후 연구로 남겨놓고 있다.

3. F2FS의 단일 파일 임의 읽기 성능 분석 및 개선

일반적인 Linux 운영체제의 입출력 요청 성능은 파일 시스템의 지연시간, 커널의 블록 계층의 성능[9] 그리고 저장장치가 제공하는 성능의 합으로 이루어져 있다. 또한 입출력 요청은 크게 4가지로 나누는데, 순차 읽기, 쓰기와 임의 읽기, 쓰기로 나누어서 구분한다. 본 논문에서는 임의 읽기 요청의 성능을 분석하였다.

매니코어 환경에서 입출력 요청에 대해 확장성을 가진다는 것은

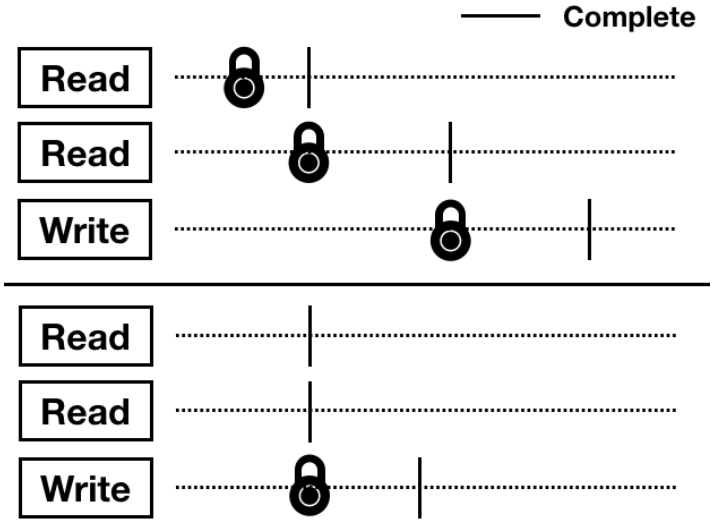

그림 1: F2FS 의 현재 구현방식 과 제안한 구현 방식

입출력 요청 프로세스의 숫자가 늘어날수록 초당 처리 가능한 입출력 요청의 개수가 증가하거나 저장장치의 최대 성능에서 저하되지 않을 경우 확장성이 있는 것으로 간주한다. 본 장에서는 F2FS의 단일 파일 임의 읽기에 대한 확장성을 검토하였다.

표 2는 단일 파일 임의 읽기 요청 프로세스가 8, 15개 일 때 초당 입출력 요청 처리 개수와 해당 프로세스들이 실행될 때에 전체 CPU 점유율 중 osq_lock 즉, 잠금이 차지하는 비율을 나타낸 표이다. 기존 구현에서는 프로세스의 숫자가 늘어날수록 잠금이 전체 실행 시간 중 차지하는 비율이 늘어남을 확인할 수 있다. 따라서 해당 잠금은 확장성 문제를 야기하며, 해결해야 될 문제라고 인식하였다.

그림 1의 윗부분은 F2FS의 현재 구현 방식인데 단일 파일에 대한 직접 읽기, 쓰기 요청에 대해 각 입출력 요청이 모두 잠금을 가지고 진행되기 때문에 타 계층에서 병렬성을 지원한다고 해도 직렬화가 되는 문제를 가지고 있다. 이것이 의미하는 것은 블록 계층이나 장치 자체에서 병렬성을 제공한다고 하더라도 파일 시스템 계층에서 병렬성을 최대로 활용하지 못하고 있음을 의미한다. 이 문제를 그림 1의 아래와 같이 읽기 시에는 잠금을 가지지 않도록 구현하여 해결하였다. 이러한 방식을 이용할 경우, 읽기 요청에 한해서는 병렬성을 확보해 확장성을 최대로 가지게 될 것이라고 예상된다.

4. 성능 평가

F2FS와 대부분의 시스템에서 기본으로 사용하고 있는 EXT4의 임의 읽기 성능 분석 비교를 위하여 본 논문에서는 FIO[10] 벤치마크를 활용하였다. 표 1은 실험을 위한 Nvme SSD 장치와 운영체제 및 대상 장치의 정보이다.

그림 2는 제안한 방법을 이용하여 Linux 커널에 적용한 결과이다. EXT4의 초당 읽기 요청 처리 개수에 비해 기존 F2FS의 성능

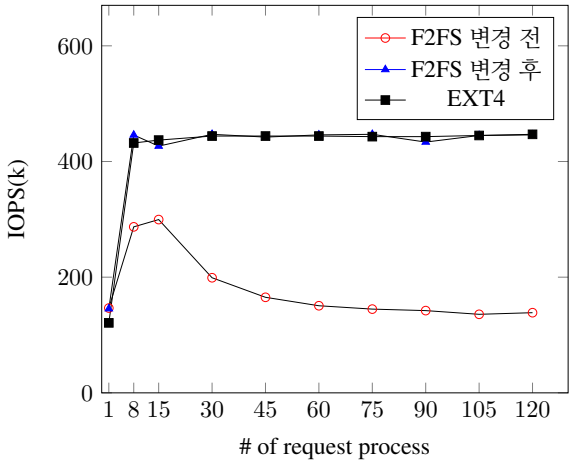

그림 2: 입출력 요청 프로세스 증가에 따른 단일 파일 임의 읽기 성능 비교

은 크게 떨어지는 것을 확인할 수 있다. 위에서 제안한 방법을 이용하여 변경 후에는 읽기 요청을 하는 프로세스의 숫자가 늘어나도 성능 저하가 일어나지 않는 것을 확인하였다.

5. 결론 및 향후 연구

본 논문은 매니코어 CPU와 NVMe 저장장치 위에서 F2FS 파일 시스템의 단일 파일에 대한 임의 직접 읽기 확장성 확인을 하였다. 이 결과로, EXT4에 비해 F2FS의 단일 파일 임의 읽기 확장성이 요청 프로세스가 증가함에 따라 저하되는 것을 확인하였다. 문제점은 F2FS 파일 시스템의 직접 입출력 실행 경로에 단일 inode에 대해 읽기 쓰기 모두가 동일하게 단일 잠금을 이용하는 것 때문인 것을 파악하였다. 이 문제점을 해결하기 위해 읽기 요청 시 잠금을 풀어줌으로써 성능이 저하되지 않음을 확인하였다. 본 연구결과는 Linux 5.7 버전의 F2FS 파일 시스템에 통합되었다.[2]

향후 계획으로는 같은 파일에 대해서 쓰기 요청과 읽기 요청이 동시에 발생하는 경우에 대한 성능 분석 및 로그 기반의 잠금 또는 RCU(Read-Copy-Update) 구조를 적용하여 구현을 확장할 예정이다.

참고 문헌

[1] 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, 2015.

[2] D. Seo, *f2fs: delete DIO read lock*, 2020. `https://github.com/torvalds/linux/commit/ad8d6a02d685ecf046c369d72285a5e69adaf66e`.

[3] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, "The new ext4 filesystem: current status and future plans," in *Proceedings of the Linux Sym..*, vol. 2, pp. 21–33, Citeseer, 2007.

[4] 서동주, 경주현, 임성수, "매니코어 환경에서 리눅스 커널의 공유 메모리 관리에 대한 문제점 분석," *정보과학회논문지*, vol. 35, no. 10, pp. 39–45, 2017.

[5] 노승우, 김서영, 남덕윤, 박근철, 김직수, "인텔 차세대 매니코어 프로세서에서의 다중 병렬 프로그램 성능 향상기법 연구," *정보과학회논문지*, vol. 44, no. 9, pp. 878–886, 2017.

[6] 이용섭, 김장웅, 박성용, "매니코어 NUMA 시스템에서의 파일 처리 능력 확장성을 위한 NOVA 파일시스템 특성 분석," *한국정보과학회 학술발표논문집*, pp. 1593–1595, 2017.

[7] 곽재혁, 변은규, 남덕윤, 황순옥, "매니코어 플랫폼에서 이종 저장 장치의 파일 I/O 특성 분석 연구," *한국정보과학회 학술발표논문집*, pp. 33–35, 2016.

[8] 노성현, 이창규, 김영재, "매니 코어 환경에서 F2FS 파일시스템의 단일 파일 I/O 병렬화," *한국정보과학회 학술발표논문집*, pp. 1073–1075, 2019.

[9] M. Bjørling, J. Axboe, D. Nellans, and P. Bonnet, "Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems," in *Proceedings of the 6th international Systems and Storage Conference (SYSTOR'13)*, pp. 1–10, 2013.

[10] J. Axboe, "Fio-flexible io tester," *URL http://freecode.com/projects/fio*, 2014.