비휘발성 메모리를 이용한 로그 구조 파일 시스템의 성능 향상

강양욱[01] 최종무[2] 이동희[3] 노삼혁[1]
[1]홍익대학교 컴퓨터공학과 [2]단국대학교 정보컴퓨터학부 [3]서울시립대학교 컴퓨터과학부
{ywkang, samhnoh}@cs.hongik.ac.kr choijm@dankook.ac.kr dhlee@venus.uos.ac.kr

Improving Log-Structured File System Performance By Utilizing Non-Volatile Memory

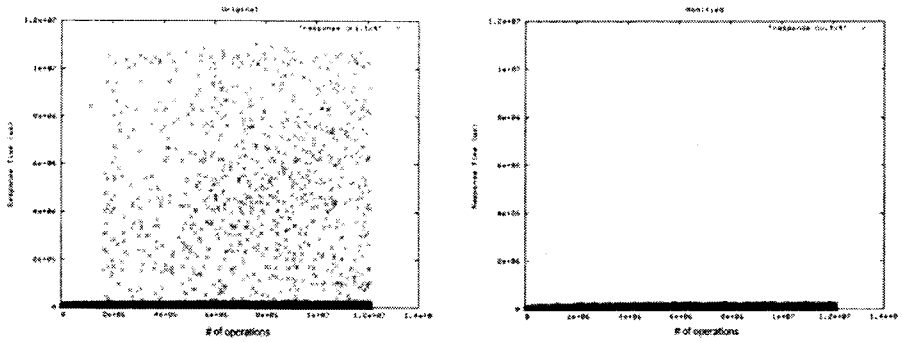
Yangwook Kang[1], Jongmoo Choi[2], Donghee Lee[3], Sam H. Noh[1]
[1]Department of Computer Engineering, Hong-Ik University
[2]Division of Information and Computer Science, Dankook University
[3]Department of Computer Science, University of Seoul

지난 십 수 년 간, 파일 시스템의 성능에 대한 연구들은 탐색 시간과 회전 지연 시간을 지닌 디스크의 물리적인 단점을 적절한 할당 정책과 휘발성 캐시를 바탕으로 해결하는 것에 초점이 맞춰져 왔다. Fast File System(FFS)[1]은 인접한 실린더에 함께 탐색될 가능성이 높은 데이터들을 저장하여 디스크 헤드의 움직임을 최소화하고자 하였으며 로그 구조 파일 시스템(Log-Structured File System, LFS)[2,3]는 큰 단위로 데이터를 모아서 세그먼트란 구조를 만들어 디스크로 쓰게 함으로써 최대한 순차 쓰기가 일어날 수 있게 설계되었다. 읽기 요청시 발생할 수 있는 탐색 오버헤드는 대부분이 메모리 캐시에 의해서 흡수된다. 이렇게 함으로써 LFS는 기존의 파일 시스템들에 비해서 쓰기 성능을 높이고 유사한 읽기 성능을 유지함으로써 더 높은 성능을 얻고자 하였다.

그러나 메타 데이터와 같은 파일 시스템의 안정성과 관련된 데이터들을 보호하기 위해서 실제 시스템의 구현에서는 동기화가 일어날 수밖에 없다. FFS는 작은 단위의 메타데이터 쓰기에 의해서 성능이 하락하고, LFS는 메타데이터들의 기록은 비동기화 되어 있지만, 메타 데이터 연산이 요청되면 가능한 빠르게 디스크로 모든 변경된 데이터를 기록한다. 또한 LFS는 덮어 쓰기가 없기 때문에 파일이 변경됨에 따라서 순차 쓰기를 위해 만들어지는 새로운 메타 데이터들이 늘어나고, 이전 블록들을 재활용해야 하는 클리너의 부담도 늘어나게 된다. 따라서 실제 환경에서 디스크의 사용량이 적을 때는 다른 파일 시스템에 비해서 높은 성능을 보이고 있으나 디스크의 사용량이 늘어남에 따라 점차 성능이 하락하는 경향을 보이게 된다. 본 연구는 이러한 상황에서 NVRAM을 활용하기 위한 운영 체제의 설계와 LFS 파일 시스템에 적용하여 동기화 문제를 해결하고 항상 안전한 파일 시스템을 만들 수 있도록 하였다.

기존 비휘발성 메모리에 대한 연구는 주로 추가 적인 쓰기 캐시로서의 활용에 집중되어왔다. 이 캐시는 운영체제를 거쳐 디스크로 쓰이는 데이터들을 한 번 더 모아서 큰 단위의 쓰기로 바꾸어 내려 보낼 수 있게 한다. FFS를 비롯한 많은 파일 시스템들에서 이러한 방식을 통해서 메타데이터의 동기화 연산으로 인한 작은 쓰기 오버헤드를 효과적으로 줄일 수 있었다. 그러나 이러한 활용은 NVRAM을 활용하고 있음에도 파일 시스템의 안정성을 개선하는데 도움을 줄 수 없다. 휘발성 메모리상에는 아직 쓰이지 않은 데이터들이 존재하기 때문이다.

본 논문에서는 새로운 비휘발성 램의 활용 방식으로 페이지 캐시의 일부분으로 사용하는 방법을 제안한다. 페이지 캐시 내의 변경된 데이터들이 모두 NVRAM에 포함되게 함으로써 어떠한 상황에서도 데이터 손실을 방지할 수 있으며 기존 Inode 구조에 복구를 위한 정보를 추가로 넣음으로써 재부팅 시에 변경된 데이터들이 다시 디스크로 쓰여질 수 있도록 한다. 데이터를 안전하게 저장할 수 있게 됨으로써 시스템내의 모든 동기화를 제거할 수 있다. 읽기 요청의 경우 정책에 따라 DRAM 페이지에 할당하는 방법과 NVRAM에 할당하는 방법을 선택할 수 있다. 전자의 방법은 쓰기 요청을 NVRAM 페이지가 가능한 만큼 미루어 쓸 수 있기 때문에 디스크 I/O를 최대한 줄일 수 있다는 장점이 있다. 그러나 이 방법은 읽기 요청에 의해 휘발성 캐시에 올라온 블록을 변경하고자 할 때 이 블록들이 다시 NVRAM으로 복사되어야 한다는 오버헤드가 발생한다. 후자의 방법은 읽기, 쓰기 요청을 모두 NVRAM에서 활용하기 때문에 페이지 복사 오버헤드가 발생하지 않는다. 그러나 쓰여지는 시점에 읽기 요청이 쓰기 요청에 비해 NVRAM 공간을 많이 차지하는 경우 성능상 불이익이 있어 높은 성능을 위해서 상대적으로 더 많은 NVRAM 공간이



[그림 1] (a) NetBSD LFS의 응답 시간 분포 (b) 수정된NetBSD LFS의 응답 시간 분포

필요할 수 있다. 두 가지 방법이 모두 장단점을 포함하고 있는데 본 연구에서는 NVRAM에 읽기, 쓰기 요청을 모두 포함하는 방법을 선택하였다. 복사 오버헤드를 고려하지 않을 수 있고, 다른 서브시스템들과 함께 사용되는 DRAM 량의 의존성을 낮출 수 있어 NVRAM의 크기에 따른 효과를 살펴보기에 적합하다. 또한 NVRAM공간을 실제 DRAM을 이용하여 커널 내에서 에뮬레이션하기 때문에 큰 NVRAM을 사용하는데 용이하다.

NVRAM을 커널 내에서 활용하기 위해서는 가상 메모리 관리자[4]와 LFS의 변화가 필요하다. 우선 가상 메모리 관리자는 부팅 시에 DRAM의 일부 공간을 떼어내어 NVRAM 공간으로 만들고, 이 두 가지 공간을 각각 관리할 수 있도록 구현하였다. LFS는 NVRAM 공간이 부족해진 경우 디스크로 기록하도록 한 뒤, 동기화를 제거하였다. 그림1은 NVRAM 크기를 변화시켜가면서 TPC-C 트레이스를 수행한 결과이다. TPC-C 트레이스는 약 1200만개의 읽기/쓰기 명령으로 구성되어있으며 DB 트레이스로 동기화 연산들로 구성되어있다. 전체 수행 시간 [그림 2](a)에 보이는 것처럼 TPC-C 전체를 실행하는데 걸린 시간이 NetBSD LFS에 비해 2.5배 향상되었다. 또한 [그림 2](b)에 나타났듯이 NetBSD LFS와 수정된 LFS의 평균 응답 시간은 각각 8195us, 5273us로 수정된 LFS 시스템의 평균 응답 시간이 각 요청 당 2922us정도 향상되었다. 또한 그림[1]에서 보이는 것 처럼 응답 시간의 분포가 평균 주변으로 고른 것을 알 수 있다.

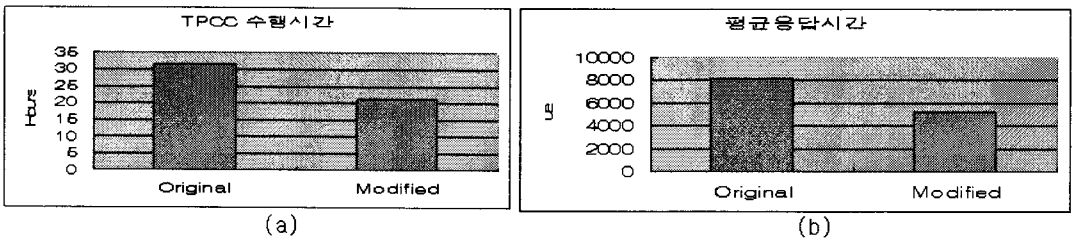
참고 문헌

[1] Marshall Kirk McKusick, Marshall K. Mckusick, William N. Joy, Samuel J. Leffler, Robert S. Fabry, "A Fast File System for UNIX", Computer Systems, vol.2, no.3, pp.181-197, 1984

[2] Margo Seltzer, Keith Bostic, Marshall Kirk McKusick, Carl Staelin, "An Implementation ofa Log-Structured File System for UNIX", In the Proceedings of the Winter 1993 USENIX Conference, pp.307-326, 1993

[3] Mendel Rosenblum and John K. Ousterhout, The Design and Implementation of a Log-Structured File System, ACM Transactions on Computer Systems, vol.10, no.1, pp.26-52, 1992

[4] Charles D. Cranor and Gurudatta M. Parulkar, "The UVM Virtual Memory System", In the Proceedings of the USENIX Annual Technical Conference, pp.117-130, 1999



[그림 2] (a) TPC-C 수행 시간 (b) 평균 응답 시간