비휘발성 메모리를 활용하는 Log-Structured File System의 설계 및 구현

강양욱[O1] 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

Design and Implementation of the Log-Structured File System Utilizing Nonvolatile 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

요 약

Log-Structured File system은 쓰기에 최적화한 파일 시스템으로 변경된 데이터를 최대한 모아서 순차적으로 기록하는 방식을 가지고 있다. 그러나 실제 시스템에서는 주기적인 동기화로 인해 작은 크기의 데이터들이 디스크로 쓰여지게 되면서 원래의 디자인 목표를 살리지 못하게 된다. 본 연구에서는 최근 급속도로 발전하고 있는 비휘발성 메모리(NVRAM)를 이용해서 주기적인 동기화를 없애고 작은 단위의 쓰기는 NVRAM을 통해 흡수하도록 하였다. 이를 통하여 DRAM만 있는 LFS에 비해 33% 가량 TPC-C 수행 성능이 향상되고, 더 빠르고 고른 응답 시간을 보일 수 있었다.

1. 서 론

파일 시스템 디자인에 있어서 디스크의 물리적 특징을 잘 활용하는 것은 매우 중요한 요소이다. 디스크는 헤드의 움직임을 최소화할 수록 더 빠른 응답이 가능하기 때문에 많은 파일 시스템들은 함께 접근될 데이터를 최대한 인접한 섹터에 기록할 수 있도록 디자인된다.

Fast File System (FFS)[1]는 이러한 특징을 잘 살린 파일 시스템으로 함께 접근될 가능성이 높은 정보들은 같은 혹은 인접한 실린더에 기록될 수 있게 함으로써 탐색 횟수와 거리를 줄여 성능을 높인다. 그러나 FFS는 메타 데이터의 연산이 동기화되어야 하기 때문에 작은 크기의 동기화된 쓰기 문제로 인한 성능의 한계가 있다[2].

Log-Structured File System(LFS)[3]은 이러한 작은 단위의 쓰기 문제를 해결하고자 디자인된 쓰기에 최적화된 파일 시스템이다. LFS는 메타데이터를 포함한 모든 쓰기 요청을 디스크의 마지막 기록 위치로부터 세그먼트 단위로 순차적으로 기록한다. 업데이트되는 블록들 또한 덮어쓰기를 하지 않고 새로운 위치에 순차적으로 기록한다. 이를 통해 쓰기 요청 시 발생하는 디스크 헤드의 움직임을 최소화하게 된다. 읽기 요청으로 인해 발생되는 헤드의 움직임은 많은 부분이 시간적 지역성의 특성에 의해 캐시에서 흡수된다. LFS는 덮어쓰기가 없기 때문에 디스크의 공간이 부족해지면 클리너가 세그먼트의 정보를 바탕으로 사용되지 않는 공간을 모아 빈 세그먼트를 만든다. LFS는 메타데이터조차 일반 데이터처럼 기록하여 기존의 메타데이터 연산이 유발하는 작은 동기화된 연산에 의한 오버헤드를 훌륭하게 해결한다. 또한 캐시의 도움으로 인해 읽기의 성능도 FFS에 비해서 큰 차이가 나지 않는다.

그러나 실제 시스템에서는 주기적인 동기화를 통해서 파일 시스템의 일관성을 유지하기 때문에 다시 작은 단위의 쓰기가 나타나게 된다. 또한 작은 단위의 쓰기를 줄이기 위해 데이터를 오래 모은다면 시스템의 이상으로 인한 데이터 손실의 가능성도 높아지게 된다.

LFS가 최대한 성능을 발휘하기 위해서는 쓰여질 데이터를 최대한 메모리에 모아두어 세그먼트 이상의 큰 단위로 쓰기가 이루어져야 하지만, 안정성의 문제 때문에 휘발성 메모리를 전제로 한 현대의 운영체제에서는 LFS가 목표했던 것처럼 동작하는데 한계가 있다. 이러한 한계를 극복하기 위한 방안으로 비휘발성 메모리의 활용을 고려할 수 있다.

비휘발성 메모리는 DRAM과 유사한 성능을 가지면서도 데이터를 영속적으로 저장할 수 있다. 기존에 배터리에 기반한 비휘발성 메모리는 오랫동안 사용되고 연구되어 왔지만[6], 최근 반도체 기술의 발전으로 배터리의 도움이 필요 없는 FeRAM(Ferro-electro RAM), PRAM(Phase-change RAM)과 같은 다양한 형태의 NVRAM이 개발되고 있다. 새로운 NVRAM은 DRAM과 비슷한 성능을 가지면서도 전원 공급이 없이 데이터를 영속적으로 저장 할 수 있다. 용량 면에서도 주요 반도체 회사 중 하나인 삼성전자는 2006년 512Mb NVRAM을 개발하였고 2008년부터 양산을 계획하고 있다[7]. 앞으로 점차 NVRAM이 현재의 휘발성 메모리의 일부를 대체하게 될 것으로 예상한다.

본 연구는 이러한 상황에서 NVRAM을 이용하여 LFS의 데이터의 안정성 문제와 주기적인 동기화 문제를 해결하고자 한다. 작은 단위의 쓰기를 비휘발성 메모리에서 흡수하고 변경된 데이터는 NVRAM에 위치하도록 하여 안정성을

보장할 수 있다.

이하 논문의 구성은 다음과 같다. 2장에서는 비휘발성 메모리를 이용한 LFS의 설계에 대해서 설명하며, 3장에서는 BSD 커널 상에 비휘발성 메모리를 활용하는 LFS의 구현과 실험, 4장에서는 결론 및 향후 연구 내용을 소개한다.

2. 비휘발성 메모리를 활용하는 LFS의 설계

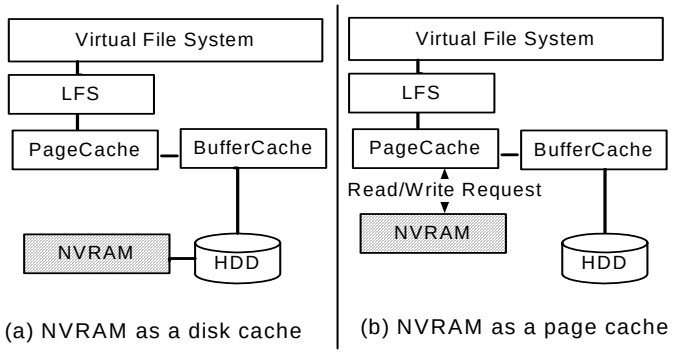
운영체제는 휘발성 메모리를 전제로 설계되었기 때문에 비휘발성 메모리를 활용하기 위해서는 운영체제의 기본 구조상의 변화가 불가피하다. 이 장에서는 LFS의 관점에서 변경되어야 할 운영체제의 서브시스템과 그 변화에 대해서 소개한다.

2.1 비휘발성 메모리의 활용

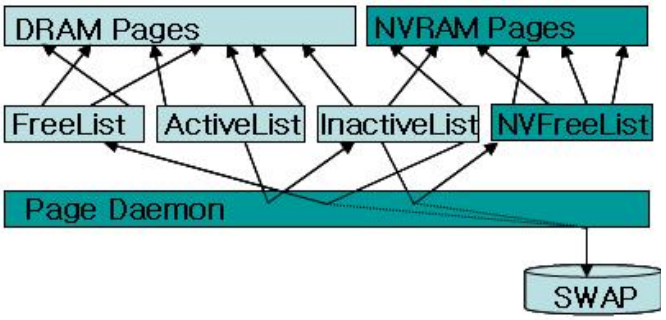
메타데이터 연산과 주기적인 동기화는 파일 시스템에 작은 단위의 쓰기 연산을 유발한다. 이러한 문제를 해결하는데 데이터의 안정성과 성능의 향상을 위해서 [그림 1](a)와 같이 비휘발성 메모리를 추가적인 디스크 캐시로 보는 방법을 사용할 수 있다. 이 방법은 버퍼 캐시를 거쳐 디스크로 내려오는 데이터를 한 번 더 캐싱 함으로써 데이터가 안전하고 더 큰 단위로 쓰여지게 한다. 그러나 LFS에서는 이미 데이터를 최대한 메모리에서 모아서 보내게 되므로 또 한번의 캐싱으로 인한 성능 향상의 효과는 다른 파일 시스템에 비해 적다. 뿐만 아니라 디스크로 전달되기 이전까지 메모리에 남아있는 데이터의 안정성도 보장할 수 없다.

또한 LFS에서는 디스크로 데이터를 기록할 때 마다 세그먼트에 관련된 메타데이터가 생성되어 함께 저장된다. 한 번 만들어진 메타데이터는 다시 업데이트가 될 수 없으므로, 세그먼트 사이즈 단위로 적는 것이 유리하다.

[그림 1](a)의 디스크 캐시로서의 NVRAM은 LFS가 디스크로 쓰는 시점을 늦출 수가 없기 때문에 LFS를 더 효율적으로 동작하게 하는데 도움을 주지 못 한다. 안전하게 작은 단위의 쓰기 요청을 제거하기 위해서는 변경된 모든 데이터들이 비휘발성 메모리 상에 존재해야 한다. LFS에서 파일 시스템 데이터는 페이지 캐시를 통해 저장되고 읽혀지기 때문에, [그림 1](b)와 같이 페이지 캐시가 비휘발성 메모리의 존재를 알고 활용할 수 있도록 설계하였다.



[그림 1] 비휘발성 메모리의 레이아웃



[그림 2] NVRAM을 인식하는 페이지 캐시의 구조

2.2 페이지 캐시의 구조 및 동작

페이지 캐시가 비휘발성 메모리를 효율적으로 활용하기 위해서는 기존의 페이지 리스트의 변화가 필요하다. DRAM만이 존재하는 환경에서는 페이지 하나를 할당할 때 빈 페이지 리스트에서 하나의 페이지를 가져오면 되지만, NVRAM이 존재하고 필요에 따라 NVRAM과 DRAM을 선택하여 할당하고자 한다면 리스트 검색이 필요하기 때문이다. 이러한 오버헤드를 피하기 위해서 [그림 2]와 같이 비휘발성 메모리를 관리하는 새로운 빈 페이지 리스트를 추가하고 페이지데몬이 활성화 혹은 비활성화 상태에 있는 페이지를 해제할 때 각각의 빈 페이지 리스트에 들어갈 수 있도록 설계하였다. 페이지 데몬은 적정 수준의 NVRAM이 항상 빈 페이지로 존재할 수 있도록 유지한다.

일반적으로 Read/Write 시스템 콜에 의해서 변경될 데이터들이 전달되면, LFS는 해당 데이터에 대한 페이지를 페이지 캐시로부터 할당 받고 요청된 작업을 수행한다. 본 논문에서 설계한 NVRAM을 인식하는 페이지 캐시는 모든 변경되는 페이지가 NVRAM에 있을 수 있도록 파일 시스템의 읽기 작업에는 휘발성 메모리를 할당하고, 쓰기 작업에는 비휘발성 메모리를 할당하도록 하였다. 쓰기 요청된 페이지가 휘발성 메모리에 있었다면, 그 페이지를 비휘발성 메모리로 복사하고, 이전 페이지를 해제한다. 또한 쓰기 요청된 페이지가 비휘발성 메모리에 있었다면 그 페이지를 그대로 리턴한다. 만일 쓰기 요청된 페이지가 어느 곳의 메모리에도 없었다면, 새롭게 NVRAM으로부터 페이지를 할당 받는다. NVRAM 페이지들이 디스크로 쓰여지는 시점은 NVRAM 내 모든 페이지들이 사용되어 공간에 여유가 없어지는 경우이다.

2.3 주기적인 동기화의 제거

주기적인 동기화는 데이터를 메모리에서 캐싱하면서 발생하는 디스크와 메모리 상의 데이터 불일치 문제를 해결하기 위해서 사용된다. 그러나 주기적으로 메모리의 내용을 디스크로 기록해야 하기 때문에 변경될 데이터가 많지 않은 경우 작은 단위의 쓰기가 빈번히 일어나게 한다.

비휘발성 메모리가 사용되면서 모든 변경된 데이터들은 비휘발성 메모리 상에 위치하게 된다. 이를 통해서 변경된 데이터가 이미 디스크에 쓰인 것처럼 생각할 수 있다. 따라서

NVRAM을 사용하는 경우 주기적인 동기화를 제거하여 작은 단위의 쓰기가 발생하는 일을 줄일 수 있다.

2.4 복구를 위한 자료구조

모든 변경된 데이터들이 비휘발성 메모리에 위치하기 때문에 변경된 부분까지의 완벽한 복구가 가능해질 수 있다. 본 연구에서 구현한 LFS는 NVRAM에 복구에 필요한 정보와 자료구조를 포함하였다.

3. 구 현 및 실 험

본 연구에서는 NetBSD 3.1 최신 커널 상에 NVRAM을 활용하는 LFS를 구현하였다. NetBSD는 LFS가 기본으로 탑재되는 유일한 운영체제이다. NetBSD에서 물리 메모리는 BSD의 가상 메모리 관리자인 UVM[4]에 의해서 관리되며, UVM과 디스크간의 데이터 교환은 UBC (Unified Buffer Cache)[5]에 의해서 이루어진다. 구현은 크게 3부분으로 나누어 질 수 있다.

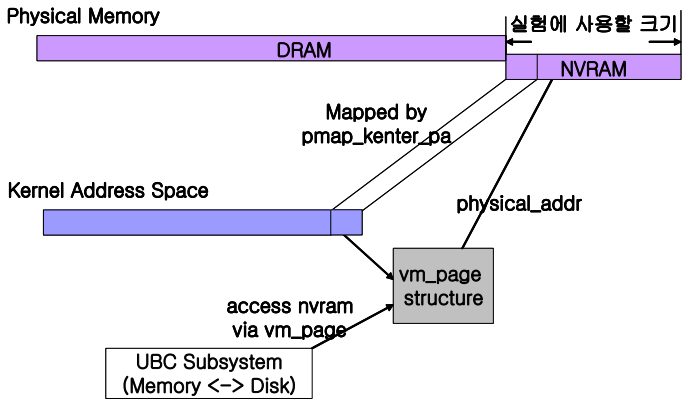
- 비휘발성 메모리 영역의 할당과 구조 - UVM이 물리 메모리의 일부를 NVRAM으로 가정하고 별도로 관리한다.
- NVRAM 페이지 관리 - NVRAM 페이지의 할당과 해제가 가능하다.
- LFS에서의 NVRAM 활용 - LFS는 쓰기 요청을 NVRAM 페이지를 할당 받아서 처리한다. NVRAM 공간이 가득 차는 경우를 체크하여 디스크로 적는다.

3.1 비휘발성 메모리 영역의 할당과 구조

실제 비휘발성 메모리는 페이지 캐시에서 사용될 만큼 큰 용량이 개발되지 않았기 때문에 운영체제의 물리 메모리(DRAM)의 일부를 비휘발성 메모리 영역으로 설정하였다. 추후 실제 NVRAM이 사용 가능한 경우를 대비해서 실제 NVRAM이 적용되더라도 디자인의 변경이 일어나지 않도록 설계하였다.

[그림 3]은 DRAM 영역의 일부를 NVRAM 영역으로 할당하는 방법을 보인다. 물리 메모리의 일정 공간을 커널에서 초기화하지 않고, 따로 NVRAM 영역으로 사용한다. 메모리 공간을 할당한 이후에는 커널의 주소 공간 상에서 NVRAM 영역을 접근할 수 있도록 헤더 부분을 맵핑한다. UVM은 페이지에 접근할 때 페이지 구조체로부터 실제 물리 주소를 구해서 접근하게 되므로 남는 페이지 공간은 맵핑하지 않는다.

할당된 NVRAM 영역은 [그림 4]에서 보이는 것처럼 구조화된다. NVRAM 영역의 앞 부분에는 Superblock과 Header가 포함되는데 NVRAM 페이지의 관리와 파일 시스템의 복구를 위해서 사용된다.

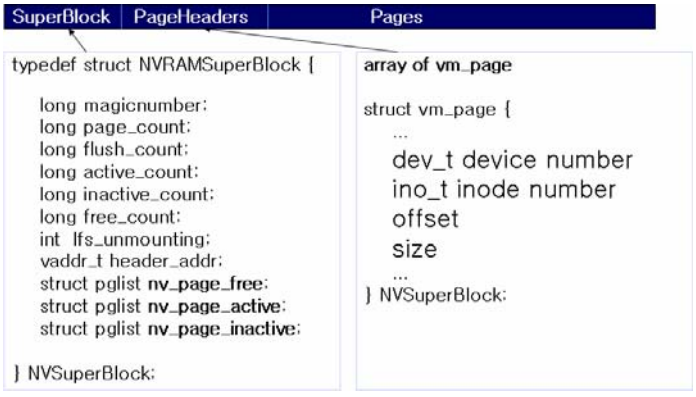


[그림 3] 비휘발성 메모리 영역(NVRAM)의 할당

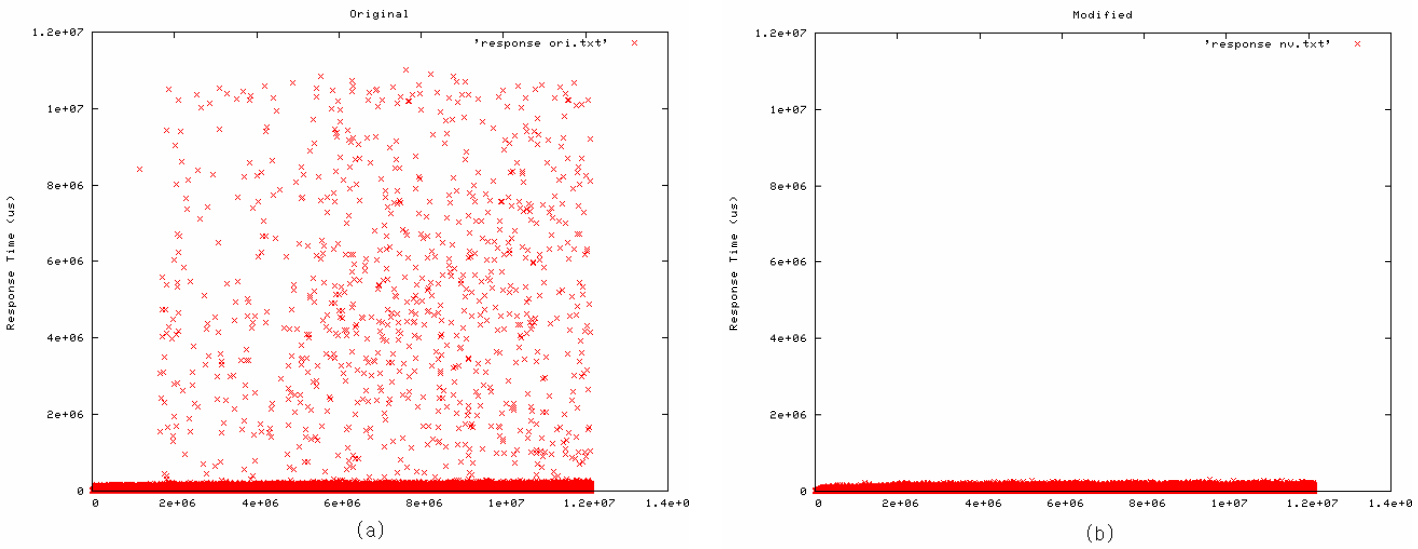
SuperBlock은 NVRAM 영역에 대한 정보를 가지고 있다. flush가 몇 번이 일어났는지 (flush_count), 현재 빈 페이지들은 얼마나 있는지 (free_count), 전체 NVRAM 페이지 수가 몇 개인지 (page_count), 페이지 헤더의 위치 (header_addr) 등이 포함된다.

PageHeaders는 BSD 시스템의 페이지 구조체의 배열이다. UVM은 이곳에 지정된 페이지 헤더를 통해 NVRAM 페이지의 위치를 찾을 수 있다. 또한 복구를 위해서 기존의 vm_page 구조체에 복구를 위한 정보를 추가하였다. 이 정보는 정상적인 상황에서는 페이지 구조체에 연결된 다른 구조체들로부터 추출할 수 있는 데이터이지만, 복구 상황에서는 이들에 접근할 수 없기 때문에 필수적인 데이터들을 중복해서 저장한다. LFS의 디스크 번호 (device_number), 소유하고 있는 inode의 번호 (inode_number), 파일내의 위치 (offset), 크기 (size)가 추가되어있다.

이렇게 초기화된 NVRAM 공간내의 페이지들은 커널의 빈 페이지 리스트에 포함되지 않고 SuperBlock 내에 존재하는 NVRAM의 nv_page_free 리스트에서 별도로 관리된다. 시스템 부팅 시에 NVRAM page 들은 nv_page_free 리스트에 추가된다. 파일 시스템의 요청에 따라 할당되었다가 페이지데몬에 의해 해제가 일어날 때 다시 이 리스트로 추가된다.



[그림 4] 비휘발성 메모리 영역의 구조



[그림 5] (a) NetBSD LFS의 응답 시간 분포 (b) 수정된 NetBSD LFS의 응답 시간 분포

3.2 NVRAM 페이지 관리

NVRAM 페이지의 할당은 두 가지 경우에 일어난다. 첫째로 현재 페이지 캐시 안에 포함되어있지 않은 페이지가 요청되었다면 새롭게 NVRAM 페이지를 할당한다. UVM에서 페이지는 uvm_pagealloc을 통해서 페이지가 할당되는데, 쓰기 요청인 경우 새롭게 NVRAM 페이지를 할당하도록 수정되었다. 둘째로 페이지가 DRAM에서 히트 되었다면, 변경될 페이지를 NVRAM으로 옮기기 위해서 NVRAM 메모리를 할당 받는다.

NVRAM 페이지의 해제는 uvm_pagefree 함수에서 이루어지는데 휘발성 메모리의 페이지는 시스템의 freelist로 가는 반면 NVRAM 페이지는 nv_freelist로 이동한다.

3.3 LFS에서의 NVRAM 활용

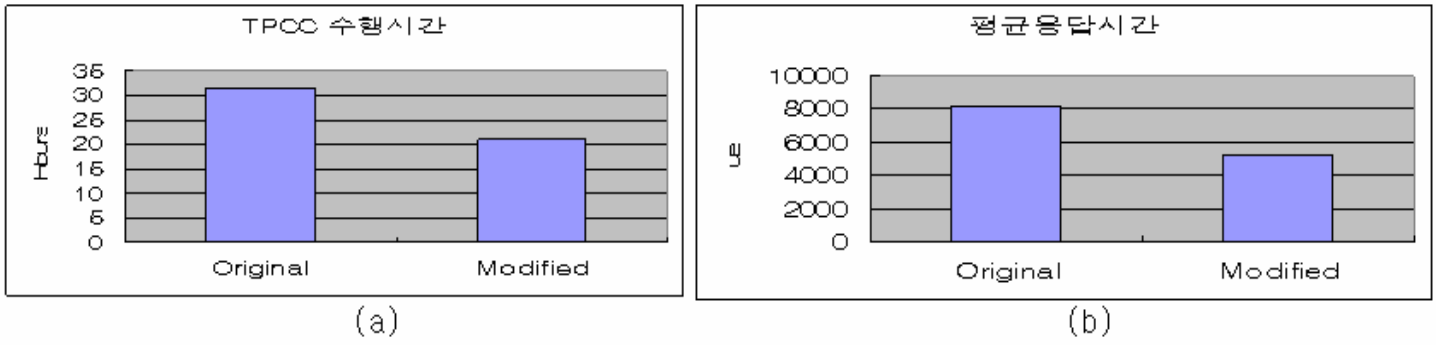
LFS는 데이터를 기록하는데 UVM의 도움을 받기 때문에 앞서 NVRAM 페이지 할당 루틴을 이용하여 새로운 데이터를 위한 비휘발성 메모리 공간을 얻는다. 매 읽기나 쓰기 요청 시 남은 NVRAM 메모리 공간을 확인하여 디스크로 쓰여져야 하는지 판단한다. 이 구현에서는 남은 NVRAM 메모리 공간이 캐싱된 총 데이터 크기보다 작을 때 디스크로 쓰도록 하였다.

3.4 실 험

주기적인 동기화를 제거한 효과를 알아보기 위해서 TPC-C 트레이스를 실행하면서 각 명령의 응답 시간을 측정해보았다. 응답 시간의 측정을 통해서 주기적인 동기화 작업의 유무가 사용자의 요청에 대한 응답에 어떠한 영향을 끼치는 지 알 수 있다.

TPC-C 트레이스는 약 1200 만개의 읽기와 쓰기 명령으로 구성되어있고, TPC-C 트레이스의 각 명령을 실행하기 전과 후의 시간 차이를 응답 시간으로 계산하였다. 응답 시간을 기록하는데 있어서 파일 시스템에 미치는 영향을 없애기 위하여 구해진 응답 시간은 네트워크를 통해 응답 시간 수집 서버로 전송하였다.

[그림 5](a)와 [그림 5](b)는 각각 LFS와 NVRAM을 활용하는 LFS의 응답 시간 분포 그래프이다. NetBSD LFS 시스템에서는 DRAM 512MB, 실험에 사용된 수정된 LFS에서는 DRAM 256MB, NVRAM 256MB를 함께 사용하였다. 전체 수행 시간도 [그림 6](a)에 보이는 것처럼 TPCC 전체를 실행하는데 걸린 시간이 NetBSD LFS에 비해 33% 정도 향상되었다. 또한 [그림 6](b)에 나타났듯이 NetBSD LFS와 수정된 LFS의 평균 응답 시간은 각각 8195us, 5273us로 수정된 LFS 시스템의 평균 응답 시간이



[그림 6] (a) TPC-C 수행 시간 (b) 평균 응답 시간

각 요청당 2922us정도 빨라졌다. 표준 편차 또한 수정되지 않은 시스템이 52509us, 수정된 LFS는 12528us로 [그림 5] 에서 보이는 것처럼 수정된 LFS의 응답시간이 평균 주변으로 더 고르게 분포함을 알 수 있다. 평균에 비해 표준 편차 값이 큰 이유는 BSD LFS의 경우 응답 시간이 최소 1us에서 11024929us까지로 큰 분포를 보이고 있기 때문이다. 이에 비해 수정된 LFS의 응답 시간은 1us에서 310622us까지 더 작은 분포 폭을 보이고 있다.

4. 결 론

본 연구는 LFS가 휘발성 메모리를 전제로 한 운영체제의 가정에 막혀 발생하는 문제점들을 해결하기 위해 NetBSD 상에 NVRAM을 활용하는 LFS를 디자인하고 구현하였다.

페이지 캐시 레벨에서 비휘발성 메모리의 사용으로 작은 단위의 쓰기 문제를 해결하고 메모리에 데이터를 오랫동안 보관해야 하는 부담을 줄여 안전한 파일 시스템을 만들 수 있게 하는데 의의가 있다. TPC-C 트레이스 실험 결과 전체 수행 시간이 33%, 응답 시간이 35% 향상되었다. LFS의 고질적인 문제인 클리너의 오버헤드는 파일 시스템의 동기화 명령이 줄어들면서 완화될 수 있다.

앞으로 주기적인 동기화를 없애고 이를 대체할 새로운 메커니즘을 개발하는 연구를 진행하고 있다. 더 지능적이고 효율적인 알고리즘을 통해서 성능을 향상하고 클리너의 부담을 줄이는 것을 목표로 한다.

감사의 글

본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업 [[2006-S-040-01, Flash Memory기반 임베디드 멀티미디어 소프트웨어 기술 개발]] 사업의 일환과 한국과학재단 특정시초연구(R01-2004-000-10188-0) 지원으로 수행하였음.

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 of a 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

[5] Chuck Silvers, “UBC: An Efficient Unified I/O and Memory Caching Subsystem for NetBSD”, In the Proceedings of the Freenix 2000 USENIX Annual Technical Conference, pp.285-290, 2000

[6] Mary Baker, Satoshi Asami, Etienne Deprit, John Ousterhout, Margo Seltzer, “Non-volatile memory for fast, reliable file systems”, In the Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.10-22, October 1992

[7] http://www.etnews.co.kr/newswire/press_view.html?id=0184587, April, 2007