비휘발성 메모리 파일 시스템 설계와 공간 효율성 분석1)

현철승*○ 백승재† 최종무† 이동희* 노삼혁‡
*서울시립대학교 컴퓨터과학부
†단국대학교 정보컴퓨터학부
‡홍익대학교 컴퓨터공학과
{cshyun*○, dhlee*}@venus.uos.ac.kr, {ibanez1383†, choijm†}@dandook.ac.kr, ‡samhnoh@hongik.ac.kr

Design of Non-Volatile RAM File System and Analysis of Space Effectiveness

Choulseung Hyun*○, Seungjae Baek†, Jongmoo Choi†, Donghee Lee*, Sam H. Noh‡
*Department of Computer Science, University of Seoul
†Division of Information and Computer Science, Dankook University
‡Department of Computer Engineering, Hong-Ik University

요 약

최근 차세대 메모리 기술이 급격히 발전하여 FeRAM과 PRAM같은 비휘발성 메모리의 상품화가 진행중이다. 이러한 차세대 비휘발성 메모리는 메모리와 저장장치의 속성을 모두 만족시켜 주지만, 용량/가격 면에서 비효율적이다. 따라서 다양한 크기의 객체를 효율적으로 표현하고 네이밍과 같은 영속성을 제공하면서 공간 효율성이 뛰어난 관리기법이 필요하다. 비휘발성 메모리에서 공간 효율성을 높이기 위하여 새로운 메모리 파일시스템을 설계하였으며, 본 논문에서는 파일 시스템을 설계하면서 파일 시스템의 공간 효율성을 측정하기 위한 공간 비용 분석 모델과 그 결과를 제시한다. 분석 모델은 다양한 파일 시스템의 공간 효율성을 수치로 제시하여 파일 시스템 설계 단계부터 공간 효율성을 예측하고 설계를 구체화 하는데 매우 큰 도움이 되었다. 또한 분석 모델은 파일 시스템의 공간 효율을 최대화 하는 블록 크기를 결정하는데 근거를 제시하였으며, 아울러 공간 효율을 최대화 하는 블록 크기는 파일 시스템에 존재하는 파일의 평균 크기에 의존적임을 보여주었다.

1. 서 론

최근 차세대 비휘발성 메모리(Non-Volatile RAM: NVRAM) 기술의 급격한 발전으로 인하여 FeRAM(Ferro-electric RAM), MRAM(Magneto-resistive RAM), PRAM(Phase-change RAM) 등의 NVRAM 용량이 비약적으로 증가되고 있다. 이에 따라 운영체제의 메모리 계층 구조에 대한 변화와 기존 구조상에서의 새로운 가능성에 대한 활발한 연구가 진행되고 있다. NVRAM의 용량이 점차 커지고 있으며, 실용화되어가는 추세를 감안하여 대용량의 NVRAM에 정보를 체계적으로 저장하고 관리하는 기술이 필요하다.

NVRAM은 비휘발성 저장장치와 휘발성 저장장치의 특성을 모두 가지고 있다. 그렇기 때문에 휘발성 저장장치인 RAM처럼 작은 객체들을 효율적으로 관리할 수 있고, 비휘발성 저장장치인 디스크처럼 큰 파일 데이터도 효율적으로 저장 가능하다. 따라서 공간 효율성이 뛰어나면서 네이밍과 같은 영속성을 제공할 수 있는 관리 기법이 필요하다. 또한 NVRAM은 용량/가격이 비효율적이기 때문에 저장할 때 낭비되는 공간을 최소화할 필요가 있으며, 이와 동시에 다양한 크기의 객체를 효율적으로 저장할 수 있어야 한다.

비휘발성 메모리에서 공간 효율성을 높이기 위하여 공간 효율이 높은 메모리 파일 시스템을 설계하였으며, 본 논문에서는 파일 시스템을 설계할 때 사용한 공간 비용 모델과 그 결과를 설명한다. 분석 모델은 다양한 파일 시스템의 공간 효율성을 수치로 제시하여 파일 시스템 설계 단계부터 공간 효율성을 예측하고 설계를 구체화 하는데 매우 큰 도움이 되었다. 또한 분석 모델은 파일 시스템의 공간 효율을 최대화 하는 블록 크기를 결정하는데 근거를 제시하였으며, 아울러 공간 효율을 최대화하는 블록 크기는 파일 시스템에 존재하는 파일의 평균 크기에 의존적임을 보여주었다.

본 논문은 다음과 같이 구성되어 있다. 2장 연구배경 및 관련연구에서는 새로운 NVRAM용 파일시스템에 대한 연구가 필요한 이유에 대해서 설명한다. 3장에서는 NVRAM용 파일시스템에서 요구 조건 및 설계 원칙에 대해서 설명하고 실제 파일시스템 내부구조에 대해 설명한다. 4장에서는 파일 시스템의 공간 비용 모델에 대해 설명하고 5장에서는 공간 비용 모델이 제시하는 결과를 통하여 여러 파일 시스템을 비교 평가한 후 6장에서 본 논문을 끝맺는다.

2. 연구배경 및 관련연구

NVRAM을 블록장치로 보고 그 상위에 전통적인 파일시스템을 사용한다면 아주 손쉽게 NVRAM을 저장장치로 활용할 수 있을 것이다. 그러나 이러한 접근 방법은 디스크 특성에 최적화되어 있는 파일시스템을 사용함으로써 NVRAM의 장점을 최대로 활용할 수 없는 문제점이 있다.

대표적인 디스크기반 파일시스템으로는 EXT2와 FAT 파일시스템 등이 있다. EXT2 파일시스템은 파일시스템 생성 시 많은 양의 메타데이터를 저장장치에 기록한다. 또한 파일데이터를 가리키기 위해 간접블록을 사용함으로써 작은 파일에는 효과적이지만 큰 파일일 경우 메타데이터 비용이 크다.

또한 FAT 파일시스템의 경우, 저장장치를 최대로 사용했을 때 각 디렉토리 계층구조를 위해 할당되는 클러스터에서 사용하지 않은 공간과 파일데이터의 마지막 클러스터만 낭비된다. 그러나 클러스터 크기가 작아지게 되면 파일데이터를 가리키기 위한 FAT의 크기가 커지게 되어 메타데이터를 저장하기 위한 공간이 커진다.

1) 본 연구는 한국과학재단 특정기초연구(R01-2004-000-10188-0)지원으로 수행되었음.

NVRAM용 파일시스템으로는 PRAMFS[7]와 MRAMFS[8]등이 있다. PRAMFS는 EXT2 파일시스템의 하나의 블록그룹을 파일시스템 전체로 확장한 파일시스템이다. 이 파일시스템은 디렉토리 엔트리를 위한 블록을 할당하지 않으며 파일데이터를 가리키기 위해 EXT2의 간접블록 형태를 사용한다. 따라서 저장할 수 있는 최대 파일 크기는 (블록크기/4) * 블록크기이며, 한 파일을 저장할 때 평균적으로 하나의 블록이 낭비되게 된다.

MRAMFS는 파일시스템의 공간을 효율적으로 사용하기 위해 압축 기법을 사용하였다. 압축을 사용할 경우 공간적으로는 이득이 발생하지만 파일 탐색같은 연산 시 많은 비용이 소요되며, CPU의 점유율이 높아지게 된다. 또한 압축하여 데이터를 저장하더라도 블록 기반으로 저장하기 때문에 낭비되는 공간은 압축하지 않은 경우와 동일하다.

3. NVRAM용 파일시스템 구조 설계

3.1 NVRAM용 파일시스템 설계의 기본 원칙

첫째, 가변블록의 장점과 고정블록 할당의 장점을 활용하기 위해 할당단위를 가능하면 작게 하고, Extent기반 할당정책을 사용하여 큰 공간을 할당할 때 필요한 메타데이터의 크기를 줄인다.

둘째, 파일시스템을 유지하기 위한 최소한의 메타데이터만 사용하고 파일을 위해 필요한 메타데이터는 필요할 때마다 할당한다.

셋째, 파일의 메타데이터를 한곳으로 집중시킴으로써 효율적인 메타데이터 연산을 제공할 수 있도록 한다.

넷째, 성능과 공간 효율성의 트레이드오프를 제공하기 위해 유동적으로 블록의 크기를 조절할 수 있도록 한다.

다섯째, NVRAM의 특성을 효과적으로 이용하기 위해 사용자가 파일열기를 시도하였을 때 가상메모리 페이지크기로 파일의 데이터에 대해 조각모음을 수행하고 파일데이터의 시작위치를 통하여 직접 접근할 수 있는 인터페이스를 제공한다.

마지막으로, 데이터의 일관성 유지를 위한 복구기법을 제공해야 한다.

3.2 파일시스템의 내부 구조

제안한 파일시스템(NEBFS)의 내부 구조는 FAT 파일시스템과 유사하다. FAT 파일시스템은 MBR(Master Boor Record)에 해당되는 섹터 다음에 FAT(File Allocation Table) 테이블, 그 다음에 루트 디렉토리 정보가 존재한다. NEBFS는 FAT 파일시스템에서 FAT 테이블을 비트맵으로 대체한 구조이다. 또한 FAT 파일시스템에서는 파일데이터 블록을 가리키기 위한 정보를 FAT 테이블이 가지고 있지만, NEBFS에서는 이러한 정보를 디렉토리 내의 엔트리에 파일의 속성정보와 함께 Extent형태로 저장한다. 이러한 구조는 초기 메타데이터를 최소화 시킬 수 있고, 메타데이터정보를 한곳으로 집중시킬 수 있게 된다.

NEBFS에서 사용한 Extent의 구조는 4바이트 블록번호와 2바이트 오프셋으로 구성되어 있다. 이는 시작 블록번호로부터 연속으로 몇 개의 블록을 가리키는지를 나타낸다. 만약 한 블록이 128바이트이고 블록이 모두 연속으로 할당된다면 하나의 Extent는 최대 8M바이트의 공간을 가리킬 수 있고 하나의 엔트리는 최대 40M바이트의 블록을 가리킬 수 있다. 파일이나 디렉토리가 커져 하나의 엔트리로 표현하지 못하면 추가적인 블록을 가리킬 수 있는 인덱스 엔트리를 할당받아 확장할 수 있다. 인덱스 엔트리의 하나의 최대 표현 범위는 블록 크기가 128바이트일 때 72M바이트이다.

이러한 구조는 하나의 블록 크기부터 Extent가 허용하는 최대 크기까지의 블록을 하나의 Extent로 표현할 수 있기 때문에 가상적으로 가변 크기의 블록을 할당하는 것과 같은 효과를 낸다. 또한 용량이 큰 파일이나 작은 파일 모두 효과적으로 기록할 수 있는 장점이 있다.

4. 파일시스템의 공간 효율성 분석

파일시스템의 공간 효율성 분석을 위하여 파일이 생성될 때 비용을 모델링하였다.[1] 각 파일시스템의 비용은 파일의 속성과 파일 데이터를 가리키는 인덱스를 포함하는 디렉토리 비용과 순수한 파일데이터를 저장하기 위한 비용으로 표현할 수 있다. 일반적으로 파일을 생성하면 파일 데이터가 차지하는 공간과 마지막 블록 내에서 낭비되는 공간은 블록할당을 기반으로 하는 파일시스템에서 동일하기 때문에 메타데이터의 비용을 비교하였다.

[표1]은 제안한 파일시스템인 NEBFS와 비교대상 파일시스템의 비용 모델이다. 공간 비용은 블록 크기 b를 기준으로 계산된다. 실험에서는 파일 시스템이 가장 간단한 상태일 때 비용

파일 시스템	비용 모델
파라미터 및 함수	A^c는 파일데이터 저장을 위해 사용한 블록 수 B는 블록의 크기 N_f는 파일시스템에 존재하는 파일 수 N_d는 동일한 디렉토리에 존재하는 파일 수 Ceiling(A, B)는 A를 B의 배수로 올림하는 함수 Ceil(A)는 A와 같거나 큰 정수를 반환하는 함수 Bitmap = Ceil(A^c / 8) File = B * A^c inode = 128 바이트
NEBFS	C = D + Bitmap + File D = Ceiling(Ceil((A^c/E^{avg} + 5)/9) * dentry size, B) / N_d 단, $1 < A^c <= 5$, D = B E^{avg}는 평균 Extent 크기
EXT2	C = D + inode + Bitmap + F/N_f + File D = Ceiling(Sum(Ceiling(4, 8+namelen(i))), B) / N_d F/N_f = 고정 비용을 전체 파일수로 나눈 값
FAT	C = D + FAT + File D = Ceiling(N_d * 32, B) / N_d FAT = FAT Entry size * A^c
PRAMFS	C = D + Bitmap + File D = inode + 1 block

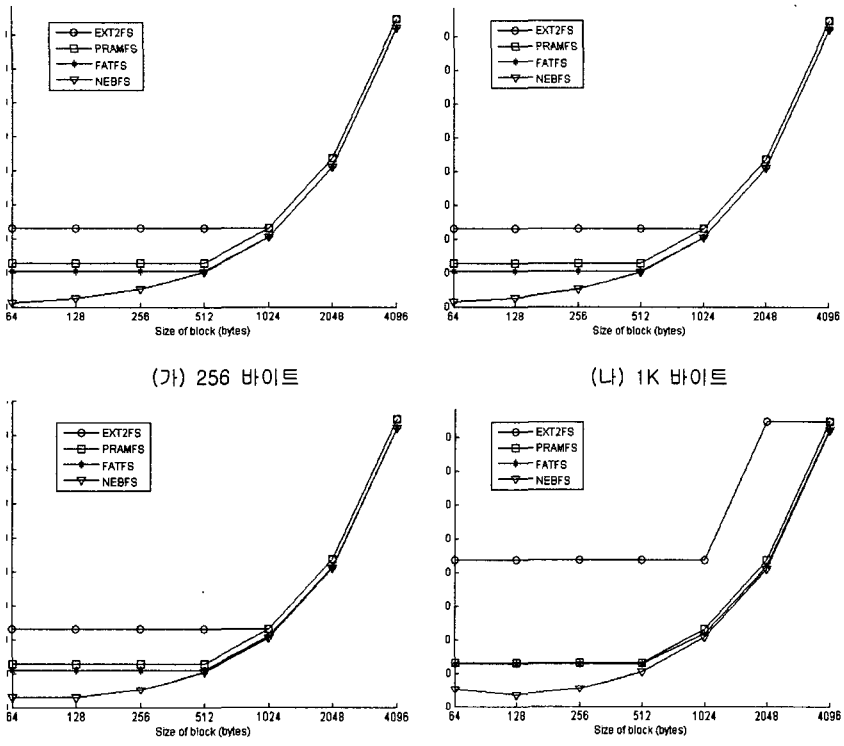
표 1. 파일시스템 비용 모델

을 계산하기 위하여 Nf와 Nd를 1로 설정하였다. Ext2 파일 시스템에서 파일 이름 길이를 나타내는 namelen()를 11로 설정하였으며, 수퍼블록과 그룹 디스크립터는 개개의 파일 오버헤드와는 무관하므로 수퍼블록과 그룹 디스크립터 비용을 나타내는 F/Nf는 0으로 설정하였다. NEBFS에서 Extent 가 표현하는 연속적인 블록의 평균 개수를 나타내는 Eavg는 30으로 설정하였다. 모든 인자가 주어지면 공간 비용 C는 디렉토리 오버헤드 (D), 메타데이타 오버헤드 (Inode, Bitmap, F/Nf, FAT), 그리고 데이터 파일 자체 (File)를 더하여 구한다.

5. 구현 및 실험 결과

공간 비용 모델을 사용하여 블록 크기를 변경하면서 각 파일시스템의 공간 비용을 그래프로 표현하였다[그림 1]. 그림 1은 (가) 256 바이트, (나) 1K 바이트, (다) 8K 바이트, (라) 32K 바이트 크기의 파일을 생성할 때 실제 파일 시스템에서 소모된 메타 데이터 크기를 나타낸 것이다.

EXT2 파일 시스템은 블록 크기를 1K 바이트 미만으로 설정하는 것이 불가능하며, PRAMFS와 FAT 파일 시스템의 경우 블록 크기를 512바이트 미만으로 설정하는 것이 불가능하다. 이 경



(가) 256 바이트 (나) 1K 바이트

(다) 8K 바이트 (라) 32K 바이트

그림 1 파일 시스템의 공간 비용

우 해당 파일 시스템은 가능한 최소 블록 크기로 설정하였다.

실험 결과는 NEBFS의 블록 크기를 64 또는 128 바이트로 설정할 때 가장 좋은 공간 효율을 보여준다. 이러한 결과는 공간 효율이 높은 메모리 파일 시스템을 설계할 때 최소 할당 단위인 블록 크기를 64바이트 또는 128 바이트 처럼 매우 작은 크기로 설정해야 함을 보여준다. 생성하는 파일의 크기가 32K바이트인 경우 Ext2 파일 시스템은 인덱스 블록 하나를 할당해야 하므로 다른 경우보다 매우 큰 메타데이타를 필요로 한다. 그리고 NEBFS를 보면 파일 크기가 32K 바이트 미만인 경우 공간 효율이 가장 좋은 지점은 블록 크기가 64 바이트일 때이며, 파일 크기가 32K 바이트일 때 공간 효율이 가장 좋은 지점은 블록 크기가 128 바이트일 때이다. 이 결과는 가장 좋은 공간 효율을 보이는 블록 크기는 파일 시스템의 평균 파일 크기에 의존적임을 보여준다.

6. 결론

본 논문에서는 데이터를 저장할 때 낭비되는 공간을 최소화시키는 동시에 다양한 크기의 객체를 효율적으로 저장할 수 있는 새로운 구조의 NVRAM용 파일시스템을 설계하였다. 그리고 파일 생성 비용 모델에 따른 분석을 통해 제안한 파일시스템의 공간 효율성을 분석하였으며 파일 시스템 설계에 필요한 다양한 인자들을 분석하였다. 이러한 분석 결과는 설계 목표를 만족시키는 메모리 파일 시스템을 설계할 때 공간 소모량을 예측할 수 있도록 하여 설계를 구체화하는데 많은 도움이 되었다.

7. 참고 문헌

[1] Alessandro Forin and Gerald R. Malan, "An MS-DOS File System for UNIX", In Proceedings of the winter 1994 USENIX Conference, 1994

[2] L. W. McVoy and S. R. Klieiman, "Extent-like Performance from a UNIX File System", In Proceedings of the winter 1991 USENIX Conference, 1991

[3] M. Rosenblum and J. 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] M. I. Seltzer, K. Bostic, M. K. McKusick and C. Staelin, "An Implementation of a Log-Structured File System for UNIX", Proceedings of the 1993 USENIX Winter Conference, pp. 307-326, 1993

[5] M. K. McKusick, W. Joy S. Leffler and P. S. Fabry, "A Fast File System for UNIX", ACM Transactions on Computer Systems, vol. 2, no. 3, pp. 181-197, 1984

[6] An-I A. Wang, Peter Reiher, Gerald J. Popek and Geoffrey H.Kuenning, "Conquest: Better Performance Through A Disk/Persistent-RAM Hybrid File System", In The Proceedings of the USENIX Annual Technical Conference, 2002

[7] PRAMFS, http://pramfs.sourceforge.net

[8] Nathan K. Edel, Deepa Tuteja, Ethan L. Miller and Scott A. Brandt "MRAMFS: A Compressing File System for Non-Volatile RAM", 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004