파일 시스템 마운트 단계의 제거: NV-RAM을 이용한 메모리 영역과 파일 시스템 영역의 융합

신형종 , 김은기 , 전병길 , 원유집
한양대학교 전자통신컴퓨터공학부 분산멀티미디어 연구실
{newbell, zerobit, ugfmain, yjwon}@ece.hanyang.ac.kr

Merging Memory Address Space and Block Device using Byte-Addressable NV-RAM

Hyung-Jong Shin , Eun-Ki Kim , Byung-Gil Jeon , You-Jip Won
DMC Lab, Dept. of Electronics and Computer Engineering, Han-Yang University

요 약

본 논문은 낸드 플래쉬 디바이스의 고질적인 문제인 마운트 지연시간을 바이트 접근성을 가지는 비휘발성 저장소자를 이용하여 해결하는 기법을 다룬다. 낸드 플래쉬 디바이스를 사용하기 위해서는, 마운트시에 낸드 플래쉬 디바이스의 전 영역에 걸쳐 분산되어 저장되어 있는 메타 데이터를 스캔하여, 해당 파일 시스템 파티션의 사용-구성정보 자료를 주기억장치에 생성해야 한다. 이러한 과정은 대용량 낸드 플래쉬 디바이스를 사용하는 경우 매우 긴 시간을 필요로 하게 되어 실제 환경에서는 낸드 플래쉬 디바이스를 채용하기가 어렵다. 본 논문에서는 차세대 비휘발성 저장장치의 바이트 단위의 접근 가능성을 활용한다. 낸드 플래쉬 디바이스 마운트시에 생성되는 최종 자료구조를 직접 NVRAM에 저장함으로써 낸드 플래쉬 디바이스의 메타 데이터를 스캔 하는 절차를 완전히 제거하였다. 즉, 낸드 플래쉬 디바이스의 마운트에 필요한 메타 데이터의 In-memory Data Structure를 NVRAM상에 저장하여 두면 이 후 NVRAM상에는 그 정보가 계속 유지되어 있기 때문에 낸드 플래쉬 디바이스의 마운트 동작은 단순히 Memory Pointer Mapping 정도의 간단하고 빠른 동작만으로도 충분하다. 본 논문에서는 비휘발성 메모리 소자가 블록 디바이스와 메모리 영역에 동시에 사상되어 있는 융합 파일 시스템을 성공적으로 개발하였다. 마운트 시간의 측정결과 효율적인 기존의 낸드 플래쉬 파일 시스템인 YAFFS에 비해 파티션의 크기나 파티션내 File의 개수에 관계없이 그 값이 매우 작고 고정적인 수치를 갖는다는 것을 확인하였다.

1. 서 론

오늘날 많은 부분에서 HDD, CD등 대용량 저장장치의 영역을 낸드 플래쉬 디바이스가 대신하고 있다. 반도체 산업의 눈부신 발달로 인해 낸드 플래쉬의 용량은 과거 Mega Byte급에서 이제는 Giga Byte급으로 그리고 멀지 않은 미래에는 Tera Byte급까지 증가하리라 예상된다. 낸드 플래쉬는 육중한 크기의 HDD에 비해 크기가 작을 뿐 아니라, HDD와 마찬가지로 Random Access가 가능하고, 비교적 충격에도 강하기 때문에, 오늘날 MP3, 디지털 카메라, PMP, 노트북 등 휴대용 장치에 특히 적합하여 그 사용이 폭발적으로 증가하고 있다.[1]

그러나, 이렇게 유용한 낸드 플래쉬 디바이스도 몇 가지 제약사항을 가지고 있다. 낸드 플래쉬 디바이스는 메모리의 제작 특성상 쓰기 횟수에 제약이 있다. 그리고, 동작 특성상 Write를 하기 위해서는 해당 영역에 대해 Erase 동작이 먼저 실행되어야 한다. 한가지 더 말하자면, Write 단위와 Erase 단위가 서로 다르다는 특성도 가지고 있다. 이러한 낸드 플래쉬 디바이스의 고유한 특성으로 인하여 낸드 플래쉬 디바이스를 HDD와 같은 블록 디바이스로 사용하기 위해 여러 가지 S/W들이 제작되어왔다. FTL[2], JFFS[3], YAFFS[4] 등이 그것이다. 이 들은 낸드 플래쉬 디바이스의 쓰기 회수 제한을 극복하기 위해 Wear-Leveling이라는 기술을 도입하였는데, 어떤 특정 블록에 대해서만 계속해서 Erase/Write 동작이 발생하면 그 블록이 열화되기 때문에 이를 방지 하기 위해 낸드 플래쉬 전 영역이 고르게 사용되도록 하는 기술이다. FTL(Flash Translation Layer)은 Block의 주소를 매번 새로운 맵핑을 통해 전체 블록을 돌려가며 쓸 수 있는 가법을 제공하며, Bad 블록이 발생하면 그 블록을 새로운 블록으로 대체하는 기능까지를 가지고 있다. 다만, FTL은 특허로 인해 사용상 많은 제약을 가지고 있다. JFFS나 YAFFS는 사용상 제약을 가진 낸드 플래쉬의 특성을 극복하기 위해 LFS(Log-Structured File System)[5]을 채택하였다. 같은 위치를 지우고 다시 쓰는 동작은 낸드 플래쉬의 특성에 맞지 않지만, 수정사항을 계속해서 Log처럼 새로운 영역에 붙여나가는 방식의 LFS의 개념은 wear-leveling에 적합하였기 때문이다. LFS은 낸드 플래쉬에 있어 여유 공간의 비효율적 사용이라는 또 다른 문제를 가져왔지만, 이에 대한 해결책으로 Garbage-Collection이란 기술을 도입하였다. 여유 공간이 부족할 때 Dirty Page들을 모아서 새로운 여유공간을 만드는 동작을 하는 것이다. 이로 인해 LFS기반의 낸드 플래쉬 디바이스는 낸드

플래쉬 디바이스를 성능면에서 효율적으로 사용할 수 있는 기법을 제공할 수 있게 되었다. 그러나 LFS는 그 외에 치명적인 단점이 있다. LFS를 사용하는 낸드 플래쉬 디바이스를 마운트하기 위해서는 낸드 플래쉬 전체 영역을 스캔하여 전 영역에 분산되어 있는 메타 데이터를 수집하고 그에 기반하여 In-memory File System Data Structure를 구성하여야 하기 때문이다. 이러한 마운트 과정은 낸드 플래쉬 용량이 커질수록 매우 긴 시간을 소모하게 되며, 신속한 가용성이 요구되는 제품에 있어 낸드 플래쉬 파일 시스템의 치명적인 약점으로 부각될 수 밖에 없다.

본 논문에서는 먼저 YAFFS를 타겟으로 선정하고, NVRAM을 이용하여 이를 해결하고자 한다. 본 논문과 관련하여 3 단계에 걸쳐 연구가 진행되어 왔는데, 첫 번째 단계는 낸드 플래쉬 디바이스의 Spare 영역과 Object Header를 FRAM에 저장하여 마운트시 낸드 플래쉬에 비해 빠른 FRAM만을 스캔하도록 하여 성능을 향상시켰고[6], 두 번째 단계는 FRAM에 저장되는 메타 데이터의 저장 구조를 개선하도록 하였다[7]. 세 번째 단계로써 본 논문에서는 전혀 새로운 접근 방식으로 낸드 플래쉬의 메타 데이터를 스캔하여 구성되는 In-memory File System Data Structure를 FRAM에 저장함으로써 낸드 플래쉬 디바이스의 메타 데이터 저장 공간으로써 FRAM을 활용할 뿐 아니라, 메타 데이터의 형식이 Run-Time시 낸드 플래쉬를 바로 사용할 수 있는 In-memory File System Data Structure 형태로 존재하게 되어 마운트라는 동작이 매우 간단하게 처리되도록 할 수 있었다.

이에 다음 단락인 본론에서 논문 설계 및 구현에 앞서 먼저 관련 연구들에 대해 알아보고, NVRAM과 YAFFS 에 대해 좀더 자세하게 조사해보도록 하겠다. 그리고, 성능 평가를 통해 본 논문이 이루고자 했던 사항과 결과들에 대해 알아보도록 하겠다. 결론에서는 본 논문의 성과와 함께 앞으로 지속적인 개선이 이루어 질 수 있도록 개선 방향 및 미래 연구 과제 등에 대해 언급하고자 한다.

2. 본 론

2.1 관련 연구

본 논문과 관련하여 다루어져야 할 소재들로는 LFS 구조의 플래쉬 파일 시스템, Hybrid 파일 시스템 정도라고 할 수 있겠다.

LFS구조의 플래쉬 파일 시스템은 구조적으로 마운트할 때 많은 시간이 걸리기 때문에 이에 대한 연구가 진행되어 왔다. Snapshot Boot, RFFS[8], MNFS[9] 등이 이에 속한다 하겠다.

Hybrid 파일 시스템은 하드 디스크 기반으로 메타 데이터의 저장 용도 혹은 Read/Write 버퍼 용도로 NVRAM을 사용하는 방안에 대한 연구들이 있었다. HeRMES[10], MRAMFS[11], Conquest[12] 등의 논문은 NVRAM을 메타 데이터 저장 공간으로써 제안하여 성능 향상을 도모하고 있으며, 관련 몇몇 논문은 Memory Hierarchy 구조 연구의 일환으로 NVRAM을 Read/Write 버퍼로 사용하도록 제안하여, Power Failure시에도 정상적인 복구가 가능하도록 제안하고 있었다.

본 논문은 이러한 연구들에 공감하며, 낸드 플래쉬 디바이스 기반에서 NVRAM을 사용하여 효율적인 공간 활용과 HDD와는 다른 낸드 플래쉬 디바이스 단점의 극복을 주요 연구 대상으로 삼았다.

2.2 NVRAM (Non-Volatile RAM)

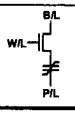
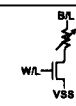
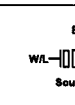
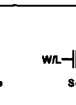
Items	FRAM	PRAM	NOR	NAND
Byte Addressable	Yes	Yes	Yes (read only)	No
Non-volatility	Yes	Yes	Yes	Yes
Read	85ns	62ns	85ns	16us
Write/Erase	85ns / none	300ns / none	6.5us / 700ms	200us / 2ms
Power Consumption	Low	High	High	High
Capacity	Low	Middle	Middle	High
Endurance	1E15	>1E7	100K	100K
Unit Cell				

그림 1. NVRAM의 비교

NVRAM이란 전원이 인가되지 않더라도 저장된 데이터가 사라지지 않으면서 Random Access가 가능한 메모리를 일컫는다. NVRAM은 각각 1bit를 저장할 수 있고 전원에 상관없이 그 정보를 유지할 수 있는 Unit Cell들로 이루어져 있으며, 낸드 플래쉬도 단 한 개의 Floating Gate를 포함하는 Unit Cell로 이루어진 NVRAM이라고 할 수 있다. 단순하고 작은 Unit Cell의 구조로 인해 낸드 플래쉬는 NVRAM중 가장 Density가 높다. 그러나 단순하고 작은 그 Unit Cell의 구조로 인해 Byte-addressable Read/Write는 불가능하다. 낸드 플래쉬뿐만 아니라 그림 1 에서 보듯이 여러 가지 NVRAM은 모두 Non-volatile이고, Random Access라는 특성을 갖지만 Unit Cell의 구조에 따라 각각의 세부 특성은 매우 다르다는 것을 알 수가 있다. 현재 가장 널리 쓰이고 있는 NVRAM중의 하나인 노어 플래쉬는 빠른 Read 성능과 꽤 높은 수준의 용량을 가지고 있지만, 느린 Write 성능과 Page Write 특성만을 가지고 있으며, PRAM은 Read/Write 모두 Byte-addressable 한 특성을 가지며 빠른 Read 성능을 가지고 있으나, Read 에 비해 Write 성능이 매우 느린 단점이 있다. 반면 FRAM은 Byte-addressable Read/Write 특성 및 동등한 Read/Write 성능을 가지는 좋은 특성을 가지고 있으나, 용량이 다른 NVRAM에 비해 작다는 단점을 가진다. 그러나 가장 완성도 높은 Byte-addressable NVRAM으로써 본 논문에서는 FRAM을 채용하여 NVRAM의 가능성도 검증해 볼

수 있으리라 생각한다.

2.3 YAFFS

YAFFS(Yet Another Flash File System)는 낸드 플래쉬 전용으로 개발된 첫 번째 파일 시스템이다. JFFS가 낸드 플래쉬를 위한 파일 시스템으로 개발되어 있었으나, 이는 원래 노어 플래쉬용으로 개발된 파일 시스템을 낸드 플래쉬용으로 포팅한 것으로 저널링에 사용되는 메모리의 소모량이 크고, 마운트 시간이 플래쉬 메모리의 용량이 커짐에 따라 큰 폭으로 증가하는 단점을 가지고 있었다. YAFFS는 JFFS의 이러한 단점을 보완하기 위해 Charles Manning (Aleph One) 에 의해 개발되었다.

YAFFS는 낸드 플래쉬의 각 Page를 Chunk와 Tag로 관리하며, Chunk에는 File, Directory등의 Object의 Object Header와 Data를 저장하고, Tag에는 object ID, chunk ID 등 각각의 Chunk를 관리하기 위해 필요한 메타 데이터들을 저장한다. 그리고, 마운트시 YAFFS는 전체 Tag들을 차례로 스캔하면서 Tag의 chunk ID가 0인 Object Header를 찾아내어 Object Structure를 구성하고, Tag의 object ID와 chunk ID를 이용하여 해당 Object에 속한 데이터를 관리하기 위해 Tnode Tree를 구성하여 Object Header에 넘겨주도록 한다. 즉, 각각의 Object(File, Directory, Symbolic Link, Hard Link)는 1개의 Object Structure와 1개의 Tnode Tree로 관리가 된다고 볼 수 있고, 전체 구조는 Yaffs Device Structure에 의해 관리되고 있다. YAFFS가 이러한 마운트 과정을 거치는 이유는 YAFFS가 Chunk와 Tag를 기본 단위로 하는 Log-Structured 낸드 파일 시스템이기 때문이다. 본 논문에서는 YAFFS의 Object Structure와 Tnode라는 Small Size의 In-memory Data Structure의 강점을 이용하여 적은 용량의 NVRAM을 이용하여 Log-Structured 낸드 파일 시스템의 단점을 극복하고자 한다.

2.4 설계 및 구현

YAFFS의 In-memory File System Structure를 분석한 결과, YAFFS는 파일 시스템을 운영하기 위해 Run-time 시에만 필요로 하는 Run-time In-memory Data Structure와 낸드 플래쉬 디바이스의 메타 데이터를 가공한 정보를 가지고 있는 Meta-data In-memory Data Structure로 나눌 수 있었다. Run-time In-memory Data Structure는 파일 시스템의 마운트시에 초기화되어야 하는 변수들로 항시 저장할 필요가 없으며, 초기화 시간은 매우 짧다. 반면, Meta-data In-memory Data Structure는 낸드 플래쉬의 스캔을 통해 얻어져야 하는 데이터이며, 이를 얻기 위해서는 마운트 시 상당히 긴 시간을 소모한다. Meta-data In-memory Data Structure에 포함되는 데이터를 분석하여 다음 표 1과 같은 요소들이 필요하다는 것을 알 수 있었다.

데이터 타입	설 명
Device data structure	파티션에 대한 상태정보를 포함하며 파티션의 크기에 따라 고정적인 크기를 가지는 데이터.
Block Information	낸드 플래쉬의 각 블록별 상태정보를 가지고 있으며, 파티션의 크기에 따라 고정적인 크기를 가지는 데이터.
Chunk Bits	각 Page의 사용유무를 나타내고 있는 Bit Map 정보를 포함하며, 파티션의 크기에 따라 고정적인 크기를 가지는 데이터.
Object data structure	Object Header에 포함된 정보를 가공한 데이터로 Object 하나당 1개씩 필요한 데이터.
Tnode data structure	Object가 가지는 Chunk Data 하나당 할당되는 데이터.

표 1. NVRAM에 포함되는 메타 데이터 구성 요소

따라서, NVRAM에 구축되는 In-memory 메타 데이터 구조는 그림 2 와 같이 구성할 수 있었다.

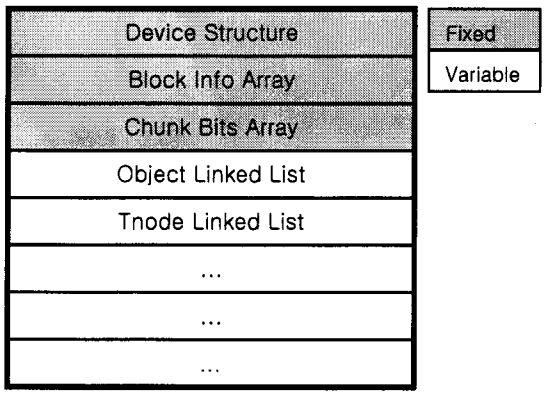


그림 2. NVRAM내 Data Structure 구성.

그림 2 에서 Device Structure, Block Info Array, Chunk Bits 까지는 고정된 파티션에 대해서 고정 크기를 가지고 있으므로, NVRAM의 첫 부분에 고정 할당하였고, Object List와 Tnode List는 필요에 따라 동적으로 할당을 필요로 하는 데이터이기 때문에 NVRAM의 나머지 공간을 두 가지 타입에 대해 동적 할당이 가능하도록 Linked List로 구성하여 별도로 관리되도록 설계하였다.

2.5 성능 테스트

본 논문에서 이루고자 한 것은 마운트 시간의 획기적인 단축이었다. 우선 본 연구에서 개발한 파일 시스템을 가칭 FRASH2 라고 명명하자. 그리고 실제 측정을 위해 SMDK 2440[13] 보드와 별도의 FRAM을 위한 소켓용 Daughter Board로 구성된 Target Board 환경을 구성하였다. FRASH2을 올려서 테스트하기 위해서는 별도의 Controller를 거치지 않는 낸드 플래쉬 디바이스를 필요

로 하기 때문에 낸드 플래쉬 디바이스로는 SMDK 2440 보드의 SMC Card 소켓을 이용하여 SMC(Smart Media Card) 128MB를 사용하기로 하였다.[14] 구성 환경은 그림 3과 같다.

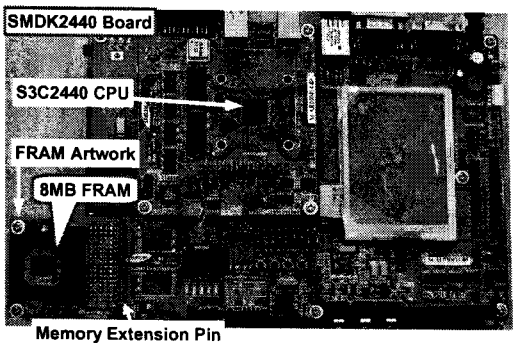
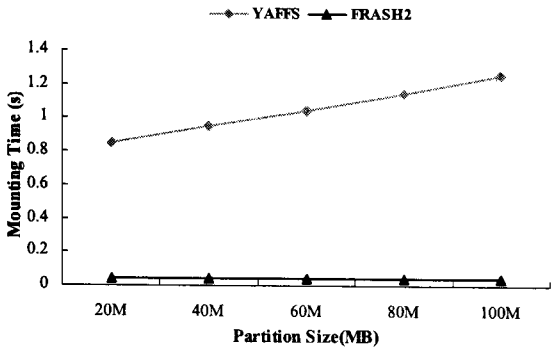


그림 3. FRAM을 연결한 SMDK 2440 보드

먼저 낸드 플래쉬 디바이스의 마운트 시간에 가장 큰 영향을 줄 수 있는 요소로서 파티션의 크기와 File의 개수가 있고, 각각의 요소의 변화에 따른 마운트 시간을 측정하였다.



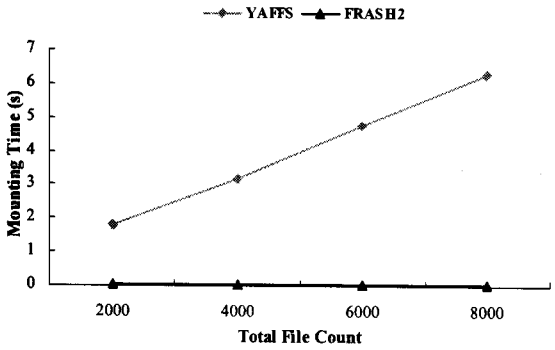
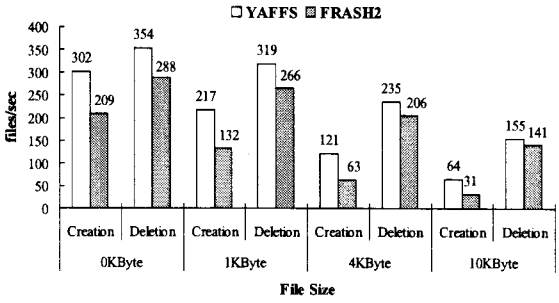


그림 4. 마운트 시간 측정 테스트

그림 4에서 첫 번째 그래프는 1000개의 File을 포함하며 10MB 정도 채워진 가변 파티션에 대해서 YAFFS와 FRASH2의 마운트 시간을 측정한 것이고, 두 번째 그래프는 100MB로 고정된 파티션상에서 1000개당 10MB정도 되는 File Density로 File의 개수가 증가함에 따라 변하는 YAFFS와 FRASH2의 마운트 시간을 측정하였다. 그래프에서 알 수 있듯이 YAFFS는 마운트 시간이 File 개수 및 파티션의 크기에 비례하여 증가하는 것을 알 수 있으며, Giga Byte 단위의 낸드 플래쉬의 경우를 생각하면 매우 긴 시간이 마운트에 소모됨을 생각해 볼 수 있다. 반면, FRASH2의 경우는 파티션의 크기나 File의 개수에 상관없이 0.04초라는 매우 작고 고정된 마운트 시간을 갖는 것을 알 수 가 있었다. 이것은 낸드 플래쉬 Access시 전혀 지체 없이 사용 가능하다는 것을 의미한다. 또한, 파티션의 사용량에 따른 FRAM의 사용량을 체크해 본 결과, 100MB 당 대략 2MB 정도의 적은 Memory Footprint를 차지하는 것을 알 수 있었다. 동적 할당하는 Tnode와 Object Structure 중 Tnode도 파티션의 크기에 따라 거의 고정된다고 볼 수 있기 때문에 Object Structure의 공간에 대해서만 고려해 본다면 8MB FRAM 에서 대략 최대 3만개 정도의 File만을 수용할 수 있다. 따라서 본 연구의 파일 시스템은 최대 파일 개수가 FRAM의 크기에 따라 정해지며, 이 이상은 만들 수 없다. 그러나 대부분의 낸드 플래쉬가 멀티 미디어 분야 및 임베디드 분야에서 사용된다고 볼 때, 이는 커다란 제약이 되지 않으리라 생각된다.



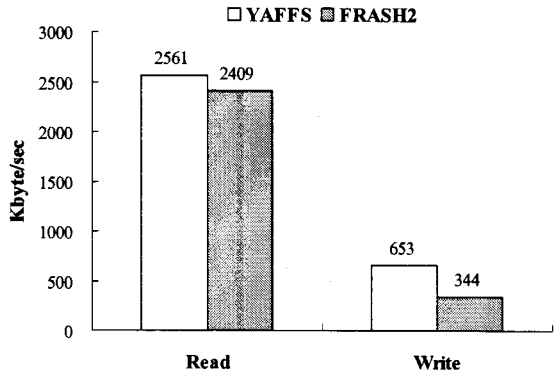


그림 5. lmbench 성능 측정 테스트

이번에는 FRASH2가 파일 시스템으로서 기본적인 Run-Time 성능이 어느 정도인지 측정하여 보았다. Benchmark Tool로 널리 알려진 lmbench[15]를 이용하여 파일 시스템의 Creation/Deletion 시 성능 (lat_fs)과

Read/Write 성능(Imdd)을 측정하여 보았다. 각 파티션은 거의 비어있는 Empty 상태로 진행되었다. 그 결과는 그림 5와 같다.

그래프에서 보면 대체적으로 FRASH2의 성능이 YAFFS에 비해 Run Time 성능이 좋지 못한 것을 알 수 있다. 특히, Creation 혹은 Write 같은 동작의 경우 Deletion 혹은 Read에 비해 성능이 더욱 안 좋아지는 것을 볼 수 있다. 본 논문에서 측정을 위해 얻은 FRAM 샘플은 현재 SMDK 2440에서 사용중인 DRAM에 비해 거의 10배정도 차이가 나는 낮은 동작 클럭을 사용하고 있다. 따라서, FRAM을 사용하는 FRASH2가 DRAM만을 사용하는 YAFFS에 비해 Run-time 성능이 좋지 않을 수 밖에 없다. 특히, FRAM에 대한 Access가 빈번한 동작일수록 성능은 더욱 떨어지는 것처럼 나올 수 밖에 없는 것이다. 하지만, FRAM의 구조상 현재의 DRAM에 비해 동작 클럭에 있어 제약이 없기 때문에 앞으로 반도체 기술의 발달과 더불어 동작 클럭이 빨라지면 Run-Time 성능의 열세는 사라지게 될 것이다.

3. 결 론

본 논문은 휴대용 기기, CE, 산업용 기기 등 거의 모든 전자 제품에서 그 사용영역을 넓혀가고 있는 낸드 플래쉬 디바이스의 마운트 시간을 줄여 신속한 가용성을 확보하고, 더불어 메모리 공간 활용을 높이고자, Byte-addressable NVRAM인 FRAM을 이용하여 낸드 플래쉬 파일 시스템 용 In-memory Data Structure를 처리할 수 있도록 설계 구현하여 보았다.

FRAM의 H/W 성능이 DRAM에 비해 떨어지기 때문에 Run-Time 성능이 떨어지긴 했지만, 마운트 시간이 기존 YAFFS와 비교가 되지 않을 정도로 향상되었다. 마운트 시간에 영향을 줄 수밖에 없는 File 개수나 파티션의 크기에도 전혀 영향을 받지 않고 항상 매우 작고 고정된 마운트 시간을 갖는 다는 것은 낸드 플래쉬 디바이스가 적용되는 모든 임베디드 제품에 대해 매우 획기적인 결과인 것이다.

본 연구는 NVRAM을 활용한 낸드 플래쉬 디바이스의 성능 향상에 중점을 두었기에 NVRAM내의 자료구조 관리 및 동적 할당 그리고 NVRAM의 효율적인 공간 이용 등에서 아직 개선의 여지가 있다. 뿐만 아니라, NVRAM의 특성으로 인해 발생하는 동기화 등의 여러 가지 문제에 대해서도 역시 고민해 볼 필요가 있다고 생각한다.

참고 문헌

1. SamsungElectronics, *Living in NAND Flash World.* http://www.samsung.com/global/business/semiconductor/products/flash/Products_NANDFlash.html.

2. Intel, C., *Understanding the flash translation layer(FTL) specification.* http://www.intel.com, 1998.

3. Woodhouse, D., *JFFS: The Journaling Flash File System.* Ottawa Linux Symposium, 2001.

4. AlephOne, C., *Yet Another Flash File System.* 2002.

5. Mendel, R. and K.O. John, *The design and implementation of a log-structured file system.* ACM Trans. Comput. Syst. %@ 0734-2071, 1992. **10**(1): p. 26-52.

6. Eun Ki Kim, B.G.J., Hyung Jong Shin, You Jip Won, Seok Hee Han, Jae Min Jung, *FRASH: Hierarchical File System for FRAM and Flash.* In Proceedings of International Workshop on Data Storage Device and Systems, Kuala Lumpur, Malaysia, Aug. 2007(As a part of International Conference on Computational Science and Its Applications), 2007.

7. Byung Gil Jeon, E.K.K., Hyung Jong Shin, You Jip Won, Seok Hee Han, Jae Min Jung, *Boosting Up the file system mount latency using Byte Addressable NV-RAM.* plan to submit, 2007.

8. TaeHoon Kim, K.S., TaeHoon Lee, KiDong Jung, *A Flash File System to Support Fast Mounting and Reliability in NAND Flash Memory.* CSICC 2, 2006: p. 87~91.

9. Hyojun, K. and W. Youjip. *MNFS: mobile multimedia file system for NAND flash based storage device.* in *Consumer Communications and Networking Conference, 2006. CCNC 2006. 2006 3rd IEEE.* 2006.

10. Ethan L. Miller, S.A.B., Darrell D.E. Long:, *HeRMES: High-Performance Reliable FRAM Enabled Storage.* In Proceedings of the 8th IEEE Workshop on HotOS-VIII, 2001.

11. Nathan K. Edel, D.T., Ethan L. Miller, 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: p. pp. 596-603.

12. H.Kuenning, A.-I.A.W.a.P.R.a.G.J.P.a.G., *Conquest: Better Performance through a Disk/Persistent-RAM Hybrid File System.* In The Proceedings of the USENIX Annual Technical Conference (USENIX '02), 2002.

13. Meritech, C., *SMDK2440.* http://www.meritech.co.kr/eng/.

14. SamsungElectronics, *K9D1G08V0A:128MB Smart MediaTM Card.* http://www.samsungsemi.com.

15. Larry McVoy, C.S., *lmbench: Portable Tools for Performance Analysis.* USENIX Annual Technical Conference, 1996.