낸드 플래시 파일 시스템을 위한 결함 복구 시간 제한 기법의 설계 및 구현

(Design and Implementation of Crash Recovery Technique with Bounded Execution Time for NAND Flash File System)

강 승 엽 † 박 현 찬 † 김 기 만 †† 유 혁 †††
(Seungyup Kang) (Hyunchan Park) (Kiman Kim) (Chuck Yoo)

요 약 현재 낸드 플래시 메모리는 다양한 이동형 기기에 활용되면서 급격하고 지속적인 성장을 하고 있다. 사용자들은 더욱 용량이 크고 빠른 기기들을 요구하지만, 현재의 낸드 플래시 파일 시스템은 결함 상황에서 복구 및 초기화 시간이 미디어의 용량에 비례하여 지연되는 문제가 있으며 이는 기기의 부팅 시간을 지연시켜 사람들의 불편을 초래한다. 우리는 이를 해결하기 위해 파일 시스템 초기화 시간이 미디어의 용량과 무관하도록 '작업 영역 기법'을 제안한다. 작업 영역 기법은 한 시점에서 낸드 플래시 메모리의 일부 영역만을 작업 공간으로 정의하고 유지하며 쓰기 명령을 작업 공간 안에서만 수행시킨다. 따라서 결함 상황에서 복구시 검색해야 할 크기를 최근 작업 영역으로 제한하므로써 결함 복구 수행 시간이 낸드 플래시 메모리 용량과 비례하지 않도록 한다. 우리는 이러한 작업 영역 기법을 YAFFS2를 기반으로 구현하였으며 기존 기법들과 초기화 성능을 비교해 보았다. 그 결과 1기가 낸드 플래시 메모리상에서 결함 복구 수행 시간이 기존의 로그 기반 기법에 비해 25배 가량 단축됨을 확인했다. 이러한 격차는 낸드 플래시 메모리 용량이 늘어날수록 커질 것이다.

키워드 : 작업 영역 기법, 결함 복구 기법, 낸드 플래시 파일 시스템, 낸드 플래시 스토리지

Abstract Flash storage devices are very popularly used in portable devices such as cell phones, PDAs and MP3 players. As technology is improved, users want much bigger and faster storage system. Paradoxically, people have to wait more and more time proportionally to the capacity of their storage devices when these are trying to be recovered after file system crash. It is serious problem because booting time of devices is dominated by crash recovery of flash file system. In this paper, we design a crash recovery mechanism, named 'Working Area(WA hereafter)' technique, which has bounded crash recovery execution time. With WA technique, write operations to flash memory are only performed in WA. Therefore, by simply scanning the latest WA, we can recover a file system crash because every change for flash memory is occured only in latest WA. We implement the WA technique based on YAFFS2 and evaluate by comparing with traditional techniques. As a result, WA technique shows that its crash recovery execution time is 25 times faster than Log-based Method when we use 1 giga bytes NAND flash memory in worst case. This gap will be futher and futher as storage capacity grows.

Key words : Working Area Technique, Crash Recovery, NAND Flash File System, NAND Flash Storage

· 본 연구는 2010년도 교육과학기술부의 재원으로 한국과학재단의 지원을 받아 수행하였습니다(No.2010-0016637).

† 학생회원 : 고려대학교 컴퓨터학과
sykang@os.korea.ac.kr
hcpark@os.korea.ac.kr
†† 학생회원 : 고려대학교 임베디드소프트웨어학과
keyman91@gmail.com
††† 종신회원 : 고려대학교 컴퓨터학과 교수
chuckyoo@os.korea.ac.kr
(Corresponding author임)

논문접수 : 2010년 5월 7일
심사완료 : 2010년 9월 13일

정보과학회논문지: 시스템 및 이론 제37권 제6호(2010.12)

1. 서 론

현재 낸드 플래시 메모리는 다양한 기기에 활용되면서 지속적인 성장을 하고 있다. 기술이 진보하면서 디지털 기기의 전반적인 성능이 향상되고 낸드 플래시 메모리의 용량도 황의 법칙에 따라 급격히 증가하고 있다. 하지만 시스템 성능이 향상된 것과 별개로 낸드 플래시 메모리의 용량이 늘어날수록 파일 시스템 초기화 시간이 그에 비례하여 늘어나서 사용자들에게 불편을 주고 있다.

파일 시스템의 초기화란 부팅 시 사용하고자 하는 파일 시스템에 맞게 구성된 낸드 플래시 메모리에서 특정 메타 데이터를 추출하여 메인 메모리에 적재시킴을 의미한다. 현재의 낸드 플래시 메모리 파일 시스템, 예를 들어 JFFS2나 YAFFS2는 낸드 플래시 메모리 상에서 실제 파일 입출력이 일어나는 공간에 대한 정보를 유지하지 않기 때문에 초기화시에 전체 공간을 검색해야 하므로 사용자의 대기시간이 낸드 플래시 메모리의 용량에 비례하여 늘어난다.

이러한 문제를 해결하기 위해 스냅샷[1], 로그 기반 기법(Log Based Method, 이하 LBM)[2] 등이 제시되었다. 그러나 이러한 기법들은 정상적인 종료 상황에서는 효과적일 수 있지만, 비정상적 종료 상황에서는 여전히 초기화 시간이 전체 용량에 비례하여 지연되는 문제가 있다.

이는 핸드폰, 특히 낸드 플래시 메모리의 용량이 비교적 큰 스마트폰, MP3플레이어 등 낸드 플래시 메모리가 주로 사용되는 시스템 환경에서 큰 불편을 초래한다. 배터리 방전, 시스템 오류 등 일반 데스크톱 컴퓨터에 비해 비정상적인 종료 상황이 빈번하게 일어나기 때문에 이렇게 빈번하게 일어나는 비정상적인 종료 상황에서도 시스템 초기화 속도를 일정수준으로 제한시켜주는 기법이 필요하다.

본 논문에서는 비정상적으로 파일 시스템이 종료된 상황에서 효율적으로 초기화를 수행하는 기법을 제안한다. 우리는 전체 용량을 고정 단위로 나누어 '작업영역(Working Area)'이라 표현하고 현재의 파일 입출력이 일어나는 공간에 대한 정보를 유지하는 '작업 영역 기법'을 고안했다. 이는 결함 복구 시 탐색해야 하는 공간을 전체 공간에서 마지막 작업영역으로 제한함으로써 결함 복구 수행 시간을 낸드 플래시 메모리 용량과 무관하도록 제한한다. 또한, 작업 영역에 관한 메타 데이터를 지정된 공간에 저장함으로써 초기화를 수행할 때 가장 최근에 설정된 작업 영역을 효율적으로 파악할 수 있도록 설계한다. 따라서 낸드 플래시 메모리의 용량이 황의 법칙에 따라 늘어나도 결함 복구 및 초기화 시간이 그와 비례하지 않도록 제한할 수 있다.

성능 평가 결과, 작업 영역 기법이 결함 상황에서 검색해야 할 크기를 제한시킴으로써 복구시간을 크게 단축 시켰다. 예를 들어 1GB 낸드 플래시 메모리에서 저장된 용량이 700MB인 경우 LBM은 결함복구에 49.3초가 소요된 데 비해 작업 영역 기법은 1.9초가 걸려 25배 이상 시간이 단축됨을 보였다. 작업 영역 기법을 적용한 낸드 플래시 메모리 파일 시스템의 초기화 시간이 전체 용량에 비례하지 않고 일정 시간으로 제한됨에 따라 결함 복구 수행 시간 격차는 낸드 플래시 메모리 전체 용량이 커질수록 두드러질 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 낸드 플래시 메모리의 특성을 기술하고, 3장에서는 낸드 플래시 파일 시스템에서의 초기화 기법 관련 연구를 살펴본다. 4장에서는 작업 영역 기법을 설계하고, 5장에서 초기화 및 결함복구 기법을 상세하게 기술한다. 6장에서는 본 기법의 성능을 분석하여 결함 복구 및 초기화 시간이 전체 용량과 비례하지 않음을 보이고 7장에서 결론 및 향후과제를 논의한다.

2. 배경 지식

2.1 낸드 플래시 특성

이 절에서는 효과적인 낸드 플래시 파일시스템 설계를 위한 하드웨어적인 특성을 간략하게 살펴보겠다. 먼저 낸드 플래시 메모리는 읽기 쓰기가 가능한 비휘발성 메모리로 페이지라는 단위로 구성되어있다. 이 페이지는 읽기와 쓰기 명령의 단위로 사용된다. 특히 다시 쓰기를 수행 할 때에는 일반적인 하드디스크와 달리 먼저 지움 명령을 수행해야 쓰기 명령을 수행할 수 있다. 이 때, 지움 명령은 수 페이지로 이루어진 블록(Erase Block, EB) 단위로 수행된다. 따라서 한 페이지를 다시 쓰기 위해서는 먼저 수정이 필요한 페이지를 포함한 블록을 읽어서 다른 블록에 저장해놓고 블록 전체를 지운 다음 새로운 페이지와 함께 블록 전체를 다시 써야 한다. 이러한 현상을 쓰기 증폭 효과(Write Amplification Effect)[3]라 한다. 이는 낸드 플래시 메모리의 지움 가능 횟수가 제한되어 있기 때문에 낸드 플래시 메모리의 수명에 큰 영향을 미친다. 따라서 저장 공간 전체의 수명을 고르게 유지시키기 위한 Wear Leveling이 낸드 플래시 메모리 파일 시스템의 필수요건으로 인식되고 이를 위한 방법으로 로그 구조(Log-Structured File System)[4] 방식을 적용한 JFFS2와 YAFFS2가 대표적인 낸드 플래시 파일 시스템이다.

3. 낸드 플래시 파일 시스템의 초기화 기법

낸드 플래시 파일시스템의 초기화 기법 관련 연구는 슈퍼 블록이라고 하는 파일 시스템 전체의 메타데이터

정보를 구성하는 파일시스템과 구성하지 않는 파일시스템으로 구분된다.

3.1 슈퍼블록을 이용하지 않는 초기화 기법

로그 구조 기반으로 설계된 JFFS2, YAFFS2 등과 같은 낸드 플래시 파일 시스템들은 슈퍼블록과 같은 약속된 위치에 메타 데이터가 기록되어있지 않고 메모리 전체에 분포되어 있다. 이 때문에 파일 시스템이 마운트되는 시점에 메타 데이터의 위치를 파악하고 메인 메모리 내에 해당 정보를 정리해서 저장할 필요가 있다.

JFFS2는 메타 데이터를 파일 데이터와 합쳐서 페이지 단위로 저장을 한다. 따라서 초기화 시 메타 데이터의 정보를 얻어내기 위해서는 모든 페이지의 모든 내용을 읽을 필요가 있다. 따라서 용량이 커질수록 이에 비례하여 초기화 시간이 지연된다.

YAFFS2는 JFFS2의 단점을 보완하기 위하여 메타 데이터를 모아서 한 페이지에 저장한다. 이때 데이터 페이지들의 인덱스는 생략한다. 대신 데이터가 각각의 페이지에 저장될 때 해당 페이지의 여분 공간(Spare area)에 이 데이터가 어느 파일에 소속되어있는지 기록한다. 따라서 메타 데이터 페이지와 데이터 페이지의 여분 공간을 모두 읽어야만 파일의 메타 데이터가 완성된다. 이러한 저장 방식을 역매핑[5]기법이라 한다. 그림 1은 JFFS2와 YAFFS2의 초기화 알고리즘을 직관적으로 보여준다.

전체 여분 공간을 읽어야 하는 위 알고리즘들은 낸드 플래시 메모리 용량이 증가함에 따라 초기화 수행 시간도 선형적으로 증가할 것을 쉽게 예측할 수 있다. 여러 실험 결과[2,6]에서도 이러한 점을 파악할 수 있으며 낸드 플래시 메모리의 종류에 따라 성능 차이는 있지만 1 GB 용량의 메모리에서 수십 초 가량의 수행 시간을 보인다. 파일 시스템의 초기화 수행 시간은 낸드 플래시 메모리를 사용하는 시스템의 부팅 시간에 포함되며, 초기화 수행 시간이 늘어날수록 사용자들이 겪는 불편이 증대된다. 따라서 기가바이트 단위의 낸드 플래시 메모리를 사용하는 경우, JFFS2나 YAFFS2를 파일 시스템으로 활용하기는 현실적으로 어렵다고 할 수 있다.

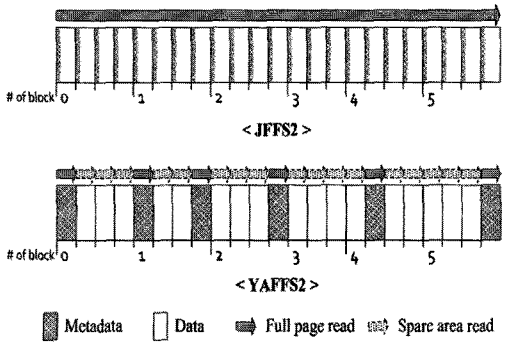

그림 1 JFFS2와 YAFFS2의 초기화 알고리즘

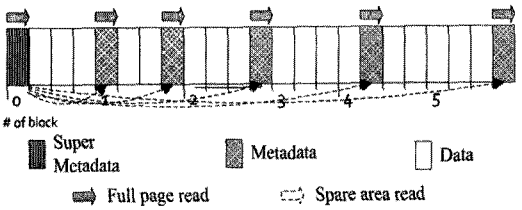

그림 2 슈퍼블록을 이용한 초기화 알고리즘

3.2 슈퍼블록을 이용한 초기화 기법

JFFS2와 YAFFS2의 초기화 성능이 저하되는 가장 큰 이유는 기존 파일 시스템과는 달리 슈퍼블록이 없어서 메타데이터의 위치나 정보를 알 수 있는 방법이 없기 때문이다. 그림 2와 같이 슈퍼블록은 정상적인 종료 이후 마운트 시에 검색 작업을 간략화시켜 초기화 속도 향상을 가져온다. 이 절에서는 슈퍼블록을 이용한 스냅샷(Snapshot) 기법, LBM에 대해 알아본다.

3.2.1 스냅샷 기법(Snapshot)

스냅샷 기법[6]은 메인 메모리 내에 저장되어 있는 파일 시스템 메타 데이터 구조를 언마운트 시점에 낸드 플래시 메모리의 미리 예약된 구역에 저장하고 마운트 시점에 해당 구역만을 메모리에 적재함으로써 초기화를 수행하는 기법이다. 이 기법은 빠른 초기화 속도를 보이지만, 언마운트가 정확하게 이루어지지 않을 경우 초기화에 수행되는 시간이 명확하게 제시되어 있지 않다. 고속 결함 복구(fast crash recovery) 기법을 제시하며 기존 낸드 플래시 메모리 파일 시스템에서의 초기화 기법을 최적화하였지만, 결함 상황에는 여전히 초기화 시간이 용량에 비례하게 증가한다.

3.2.2 로그 기반 기법(Log Based Method)

LBM[2]은 파일에 일어나는 쓰기/지움 연산을 모두 로그로 기록하고, 초기화 시점에 로그가 저장된 페이지들이 존재하는 로그 세그먼트 블록(Log Segment Block, 이하 LSB)만을 읽어 파일 시스템의 메타데이터를 재구축한다. LSB의 위치는 런타임에 할당이 되기 때문에 초기화 과정에서 즉시 해당 블록의 주소를 알 수 있는 방법이 필요하다. 따라서 약속된 위치에 수 개의 블록을 로그 세그먼트 디렉토리(Log Segment Directory, 이하 LSD)로 예약해두고 LSB의 블록 번호를 기록하는 방식을 사용한다. 그림 3은 LBM의 로그 저장 방식을 나타내고 있다. 각 로그는 페이지 단위로 묶어서 LSB에 기록되고, LSD가 LSB의 주소를 저장한다.

낸드 플래시 메모리에서 로그는 페이지 단위로 기록되어야 하기 때문에 이 기법에서는 로그 버퍼를 두어 페이지 크기만큼 로그가 생성될 때까지 메인 메모리에

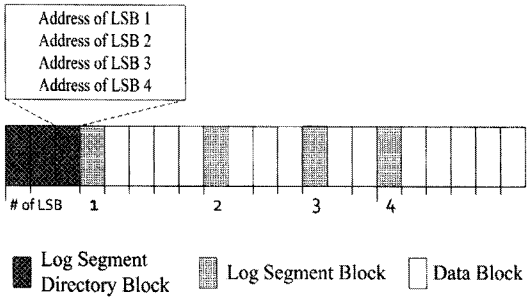

그림 3 LBM의 로그저장방식

저장한다. 이러한 버퍼의 사용으로 인해 만약 파일 시스템 언마운트가 제대로 이루어지지 않으면 모아둔 정보가 유실되어 파일 시스템의 일관성이 훼손될 가능성이 있다. 이를 복구할 때, 이미 기록된 로그 정보를 활용할 수 있기 때문에 스냅샷 기법보다는 결함 복구 수행 시간이 단축될 것이다. 그러나 로그 기록이 없는 데이터를 찾아내기 위해서 낸드 플래시 메모리 내의 프리 블록을 모두 검색해야 하므로, 결국, 낸드 플래시 메모리의 용량에 따라 결함 복구 시간이 선형적으로 증가함을 알 수 있다.

이와 같이 정상 종료 후의 초기화 성능을 개선시킨 기법들이 제시되었지만 결함 복구 수행 시간까지 고려한 기법은 아직 존재하지 않는다. 이 외에도 CFFS[1]와 같이 슈퍼블록을 활용하는 플래시 파일 시스템도 존재하지만 위와 동일한 확장성 문제를 갖고 있다. 현재의 낸드 플래시 메모리 발전 속도를 고려하면 수 년 내에 수백 기가바이트의 용량을 갖는 낸드 플래시 메모리가 개발되고 실제 개인용 기기에서 널리 활용될 것이다. 이러한 경우 기존 기법들은 결함 상황에서의 긴 복구 수행 시간으로 인해 실제 기기에 활용하기 어려울 것으로 판단된다. 따라서 초기화 및 결함 복구 시간이 낸드 플래시 메모리 용량과 비례하지 않도록 하는 기법이 필요하다.

4. 작업 영역 기법 설계

우리는 낸드 플래시 파일 시스템의 결함 복구를 효율적으로 수행하기 위해 작업 영역을 설정해 사용하는 기법을 설계하였다. 본 논문에서는 기존 낸드 플래시 메모리 파일 시스템의 결함 복구 수행시간이 전체 용량에 비례하여 선형적으로 증가하는 것을 해결하고자 한다. 기존 기법들의 문제점은 낸드 플래시 메모리 상에서 실제 파일 입출력이 일어나는 공간에 대한 정보가 부족함에 따라 결함 상황에서 전체 공간을 검색해야하기 때문에 발생한다. 우리는 파일 입출력이 일어나는 공간을 전체 용량의 크기와 무관한 '작업영역(Working Area)'으로 제한 함으로써 이를 해결하고자 한다. 전체 용량을 고정크기의 작업영역으로 나누고 현재 파일 입출력이 일어나는 작업영역에 대한 정보를 유지한다. 따라서 결함 복구 시 탐색해야 하는 공간을 전체 용량 크기에서 마지막 작업영역의 크기로 제한함으로써 결함 복구 수행 시간을 낸드 플래시 메모리 용량과 무관하도록 제한한다.

본 기법의 주안점들은 다음과 같다. 첫째, 낸드 플래시 메모리에 대해 이루어지는 변경을 작업 영역안으로 제한하는 구조 설계, 둘째, 작업 영역의 할당 및 변이에 관한 정보를 유지하는 기법, 셋째, 이러한 구조에서의 효율적인 결함 복구 기법 제시 등이다. 우리는 기 출판된 [7]보다 개념을 발전시키고 구현 및 평가사항을 구체적으로 기술한다.

4.1 블록의 분류

플래시 메모리 내의 각 블록은 슈퍼블록, 저널 블록, 데이터 블록, 프리 블록의 네 가지로 구분된다.

슈퍼블록은 플래시 메모리의 첫 블록부터 두 개 이상의 블록으로 구성되며, 파일 시스템 설정 값과 저널 블록의 메타데이터가 저장된다. 파일 시스템 설정 값은 각 블록의 첫 페이지에 기록되며 파일 시스템이 생성될 때 결정되는 값들이 저장된다. 예를 들어 현재 파티션의 크기나 첫 번째 블록의 번호 등이다. 이것은 파일 시스템의 설정에 따라 바뀔 수 있다. 다만 일반적인 파일 시스템 정보들 외에 한 작업 영역에 속하는 블록의 개수, 그리고 슈퍼 블록의 크기 등이 반드시 포함되어야 한다. 저널 블록의 메타데이터는 저널 블록 테이블(Journal Block Table, 이하 JBT)에 저장된다. JBT는 저널 블록들의 번호가 저장된 공간이며 1개 이상의 페이지들로 이루어진다. 각 페이지의 여분 공간에는 현재의 페이지에 포함된 저널 블록의 개수, 언마운트 정상 수행 여부, 현재의 작업 영역의 시작 블록 번호, 다음 사용될 작업 영역의 시작 블록 번호가 기록된다. 이러한 값들을 통해 JBT의 크기와 언마운트가 정상적으로 수행되었는지, 그리고 현재와 다음 작업 영역의 위치를 알 수 있다.

저널 블록은 저널이 기록되는 공간이다. 저널은 쓰기/

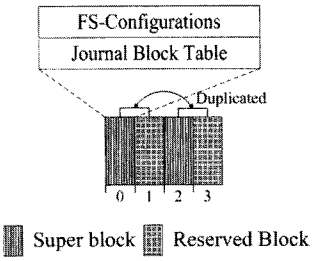

그림 4 슈퍼블록 레이아웃

지움 등의 업데이트 정보이며 메인 메모리에 캐싱된다. 이때, 메인 메모리 안에서 중복되는 저널들을 정리하는 동작을 수행한다. 메인 메모리에 캐싱된 저널들의 크기가 한 페이지에 도달하면 저널 블록에 기록한다.

데이터 블록은 데이터와 파일 메타데이터가 기록되는 블록이다. 이외의 블록들은 모두 현재 사용되지 않는 프리 블록으로 간주한다.

4.2 블록의 관리

위와 같이 분류된 블록들이 할당되는 정책은 다음과 같다. 슈퍼 블록은 파일 시스템이 생성될 때 정적으로 파티션의 가장 첫 블록부터 할당되며 그 크기는 2의 배수 개의 블록이다. 슈퍼블록에 기록된 데이터가 유실되지 않도록 하기 위해 복수개의 블록이 필요하다. 우리는 그림 4에 슈퍼 블록이 4개로 할당된 레이아웃을 간략하게 나타내었다. 0번 블록은 일반적인 슈퍼 블록이며 2번 블록은 슈퍼 블록 데이터의 유실을 막고 유실 시 복구를 위한 복제 블록이다. 그리고 1번 3번 블록은 0번 2번 블록에 대한 예비 공간이다. 플래시 메모리의 특성상 이미 데이터가 쓰여진 페이지에는 다시 데이터를 쓸 수 없고 반드시 지움 연산을 수행하여야 한다. 따라서 슈퍼 블록에 데이터를 기록하는 동작이 지움 연산으로 인해 지연되는 것을 예비 공간을 둠으로써 방지한다. 예를 들어 0번 블록에 모든 데이터가 쓰여졌다면 슈퍼 블록에 대한 쓰기 연산은 1번 블록에 수행된다. 그리고 파일 시스템의 유휴 시간에 0번 블록에 대한 지움 연산을 수행하고 이후에 0번 블록이 예비 공간이 된다.

저널 블록은 한 번에 N개 블록이 할당된다. 블록은 임의의 위치에 있을 수 있고 할당이 이루어지면 슈퍼 블록 내의 JBT를 업데이트한다. N개를 할당하는 이유는 슈퍼 블록에 너무 잦은 연산이 이루어지도록 하지 않기 위해서이며 또 결함 상황에서 다음 저널 블록을 쉽게 찾을 수 있도록 하기 위해서이다.

데이터 블록은 현재 작업 영역 내에서 최대한 순차적으로 할당된다. 만약 작업 영역 내에 충분한 블록이 없다면 해당 시점까지의 저널 정보를 기록하고 다음 작업 영역으로 이동한다.

모든 블록 내 페이지의 여분 공간에는 공통적으로 에러 교정 코드(Error Correction Code, 이하 ECC)와 버전이 기록된다. ECC는 플래시 메모리에서 발생하는 1-비트 에러를 교정하기 위하여 모든 플래시 기반 저장 시스템에서 공통적으로 사용되는 정보이다. 버전은 페이지 상호간의 기록된 순서를 알기 위해 1씩 증가시키며 기록하는 전역 정보이다. 이를 통해 어느 페이지가 가장 최신 정보를 기록하고 있는지 알 수 있다.

그림 5는 블록들이 할당된 모습을 간략하게 나타낸 것이다. 현재 작업 영역의 크기는 6개 블록이며 슈퍼블록은 JBT를 통해 저널 블록들을 가리키고 있다. 데이터 블록은 각 작업 영역 내에서 순차적으로 할당되고 있음을 알 수 있다. 현재 작업 영역 3까지는 사용되지 않았지만 미리 저널 블록을 할당해둔 상태이다.

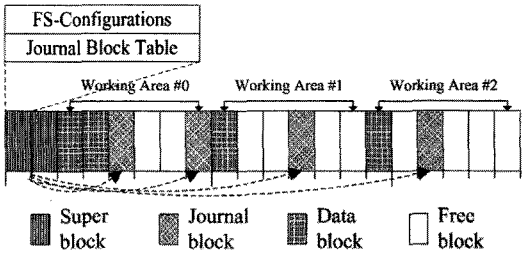

그림 5 블록 할당 예

4.3 작업 영역의 관리

앞서 언급한 바와 같이 효율적인 결함 복구를 하기 위해서는 마지막 작업영역의 위치를 빠르게 파악하는 것이 중요하다. 따라서 우리는 다음 관리 방법을 통해 작업 영역을 효율적으로 관리한다.

먼저 작업 영역은 파일 시스템이 생성될 때 그 크기가 정해지며 순차적으로 이동한다. 작업 영역의 크기는 블록의 개수로 지정되며 작업 영역의 크기가 클수록 결함 복구 소요 시간이 길어지므로 플래시 메모리의 사양에 따라 값을 설정해줄 필요가 있다.

작업 영역의 이동은 순차적으로 이루어진다. 예를 들어 현재 작업 영역이 128번 블록부터 1024개 블록이라면 다음 작업 영역은 1152(=128+1024)번 블록부터 시작한다. 이렇게 연속적으로 작업 영역이 이동하게 함으로써 가장 최근의 작업 영역 정보가 유실된다 하더라도 이를 검색하는 수행 시간을 최소화 시킬 수 있다. 작업 영역은 저널 페이지가 기록될 때, 작업 영역 크기의 일정 비율만큼을 기록하고 다음 작업 영역으로 이동한다. 이때 저널 페이지의 여분 공간에 다음 작업 영역의 시작 블록 번호를 기록한다. 따라서 저널의 여분 공간을 통해 최근의 작업 영역을 파악할 수 있다.

작업 영역의 이동은 전체 용량 대비 작업 영역의 용량에 따라 다르게 관리 한다. 본 논문에서는 1GB 낸드 플래시 메모리에 33.6MB의 작업영역을 할당했고 작업 영역 크기의 20%만큼 기록하고 다음 작업 영역으로 이동한다. 이 수치는 경험상의 수치이고 효율적인 작업 영역의 크기와 사용 비율의 연구는 향후 과제로 남겨둔다.

4.4 가비지 콜렉션(Garbage Collection) 기법

저널 블록과 데이터 블록의 해제는 블록 내에 포함된 모든 페이지가 불필요한 데이터를 기록하고 있는 경우에 가비지 콜렉션에 의해 이루어진다. 저널 블록 내의 데이터는 갱신된 정보가 다른 위치에 저장되거나 파일

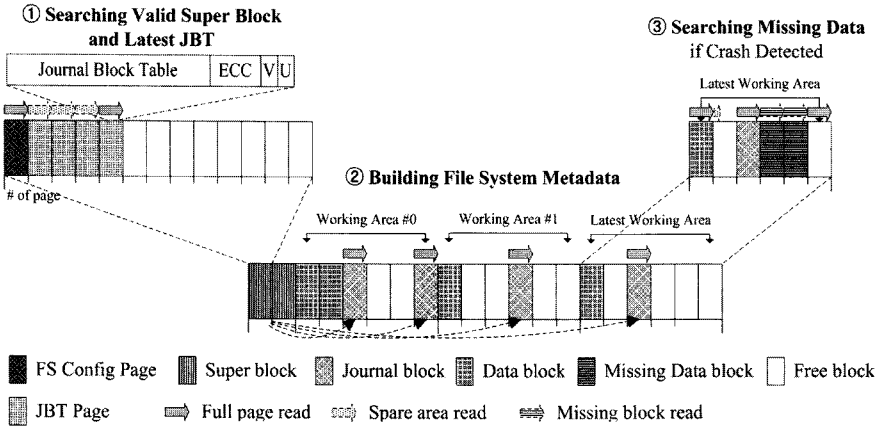

그림 6 초기화 및 결함 복구 기법

의 삭제로 인해 저널 정보가 더 이상 유효하지 않은 경우에 불필요해진다. 데이터 블록은 내부에 포함된 데이터가 삭제 혹은 수정된 경우이다.

위와 같이 불필요해진 블록들을 수집하고 실제로 지움 연산을 하는 가비지 콜렉션은 파일 시스템이 유휴 상태일 때 수행된다.

5. 초기화 및 결함 복구 기법

이 장에서는 위와 같이 설계된 작업 영역 기법을 사용한 플래시 파일 시스템에 대한 초기화 기법과 결함 복구 기법을 기술한다. 그림 6에 초기화 및 결함 복구 기법을 직관적으로 표현한 것이다.

파일 시스템의 초기화에서 가장 먼저 수행되는 것은 유효한 슈퍼블록을 찾는 것이다(그림 6의 ①참조). 슈퍼블록의 첫 번째 페이지는 파일 시스템 설정 값이므로 두 번째 페이지의 여분 공간을 참조하면 해당 슈퍼블록이 유효한지 알 수 있다. 그 다음 결함 상황 여부를 가장 최신의 JBT를 통해 파악한다. 최근의 JBT 페이지에 언마운트 수행 기록이 없다면 결함이 발생한 것이다.

5.1 정상적 상황의 초기화 기법

정상적으로 언마운트가 수행된 경우, 최근의 JBT를 이용해 전체 저널을 추적할 수 있다(그림 6의 ②참조). 전체 저널의 정보를 통합하면 전체 파일 시스템 메타데이터 정보를 구축할 수 있다. 이러한 정상적 상황의 초기화 기법은 LBM에서 정상적 상황과 결함 상황을 구분하지 못하는 단점을 보완한다.

5.2 결함 복구 기법

만약 결함이 발생하여 정상적으로 언마운트가 수행되지 않았다면 우선 최근의 JBT를 기반으로 파일 시스템 메타 데이터 정보를 구축한다. 이때 저널에 저장된 버전 정보를 이용하여 가장 마지막으로 기록된 저널을 찾고(그림 6의 ②참조), 해당 저널이 저장된 페이지의 여분 공간에 기록된 작업 영역 정보를 통해 최근 작업 영역을 파악한다.

다음은 결함 상황으로 인해 유실된 저널에 기록됐던 데이터를 찾는 것이다. 최근 작업 영역 안에서 블록들을 순차적으로 검색하면서 손실된 저널 정보를 복구한다(그림 6의 ③참조). 이때, 한 블록 내의 페이지는 언제나 순차적으로 기록되기 때문에 첫 페이지의 여분 공간을 읽음으로써 블록이 사용 중인지 여부를 알 수 있고 이를 통해 검색을 최소화할 수 있다. 위와 같이 작업 영역 내의 모든 블록 정보를 파악하면 결함 복구를 종료한다.

6. 성능 분석

이 장에서는 위와 같이 설계된 작업 영역 기법의 성능을 측정하고 그 결과를 분석한다. 이 장에서는 기존 기법들과 작업 영역 기법(Working Area, 이하 WA)을 정상 종료 상황 및 결함 상황에서 초기화 시간 비교하여 WA의 초기화 시간이 작업 영역의 크기에 따라 한정됨을 보인다. 특히 다른 슈퍼블록 초기화 기법인 LBM과 상세히 비교해 초기화 성능에 큰 차이가 있음을 보인다.

표 1 실험 환경

종류	모델
보드	EZ-PXA270[8]
CPU	520MHz PXA270
메모리	64MB SDRAM (삼성)
NAND 플래시	K9G8G08U0A[9] (삼성 1GB MLC)
NAND 페이지 읽기 시간	123us(측정 성능) 60us(명세 성능)
커널 버전	2.6.21
YAFFS2 버전	12/29/'09 다운로드 버전[10]

6.1 실험 환경 및 성능 변수

성능 측정을 위해 표 1의 내장형 시스템을 사용했다. 우리는 우선 워크로드 재생 프로그램[11] 실행 전후의 마운트/언마운트 시간측정을 통해 전반적인 성능을 비교해보고 정상적 종료 상황과 결함 상황일 때의 초기화 시간을 따로 세분하여 측정한다. 이로써 파일 시스템 및 기법별로 차이가 있음을 보이고 우리 기법의 성능 향상을 보인다.

6.1.1 워크로드 재생 툴

우리는 낸드 플래시 메모리를 실제 워크로드를 기반으로 에이징 하기위해 K.A. Smith가 [12]에서 개발한 워크로드 실험 도구를 활용하였다. 본 실험에서는 임베디드 장비의 낸드 플래시 메모리를 대상으로 수행하기 때문에 약 450MB의 용량만 사용하도록 초반 약 60000개의 연산을 재생한다. 표 2는 JFFS2와 YAFFS2, 스냅샷, LBM, WA를 워크로드 재생 전후 및 결함 상황에서의 마운트 및 언마운트 시간을 측정한 것이다. 이 표에서 주목할 점은 LBM과 WA를 제외한 나머지 기법들의 결함복구 수행시간이 적게는 15배, 많게는 50배 가까이 지연된다는 점이다.

이후의 실험은 스냅샷 기법과 LBM, WA를 정상 종료 상황과 결함 상황으로 나누어 측정했다.

표 2 워크로드 재생

용량/파일개수	0MB/0개		388.6MB/6656개			
종료여부	정상 종료				결함 발생	
마운트(ms)	M	UM	M	UM	M	UM
JFFS2	6195	4	208852	416	213770	245
YAFFS2	480	7	59375	53	59318	10
스냅샷	928	112	1011	1373	59165	1235
LBM	54	6	3536	55	4353	8
WA	45	9	3703	55	3776	8

6.2 정상 종료 상황

정상적 종료 상황에서 기법 간의 성능차이를 분석하고자 파일개수를 1000개로 고정하고 사용량을 독립 변인으로 실험했다.

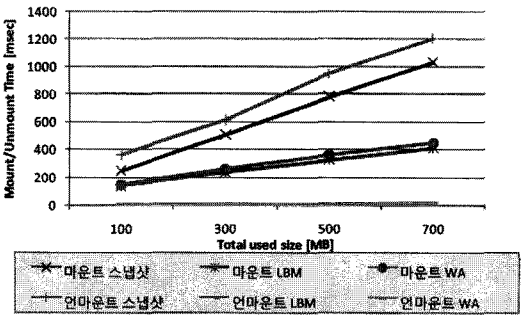

그림 7 정상 종료 상황

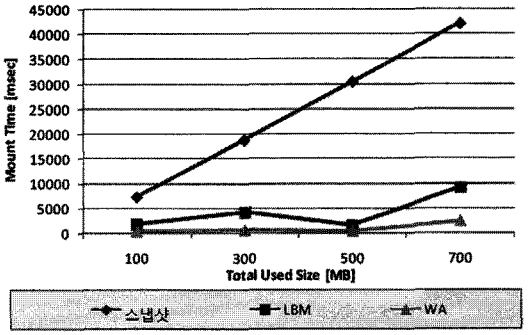

그림 8 결함 상황

그림 7에서 알 수 있듯이 사용량이 증가하면 초기화 시간이 지연된다는 것을 확인할 수 있다. 스냅샷 기법에 비하여 LBM, WA모두 좋은 성능을 확인할 수 있다. 또한, LBM, WA의 경우 거의 동일한 성능을 보여주고 있다. 이는 두기법 모두 파일 연산을 기반으로 마운팅 시 읽어야할 정보의 크기가 동일하기 때문으로 판단된다.

스냅샷기법의 경우 스냅샷을 기록하기 위한 부하로 인하여 언마운트 성능이 좋지 않음을 보여주고 있다.

결론적으로 정상적인 상황에선 LBM와 WA는 거의 비슷한 성능을 보여주고 있으며, YAFFS2에 비하여 마운트/언마운트 모두 좋은 성능을 보여주고 있다.

6.3 결함 상황

결함이 발생한 경우에는 LBM과 WA는 확연히 다른 특성을 보여준다. 그림 8은 그림 7의 동일한 구성에서 결함 복구 시간을 측정한 결과이다. 이 경우 모두 스냅샷에 비하여 좋은 성능을 보여주고 있지만 사용량 증가에 따라 마운트 시간이 비례하여 증가하던 특성을 보여주고 있지 않다.

이를 더욱더 자세히 분석하기 위하여 그림 9, 10은 마운트 시간을 결함 복구에 사용된 시간과 순수 마운팅 시간으로 구분하여 LBM과 WA를 각각 표현하였다. 또한 결함 복구를 위하여 'NAND에서 읽어야 할 크기(Size To Scan, 이하 STS)'를 함께 표시하였다. 모든 경우에 순수하게 마운팅하기 위해 소요된 시간은 사용량 증가에 비례하고 있음을 보여준다.

하지만, 결함 복구에 소요된 시간의 경우 사용량 증가에 비례하는 것이 아닌 STS에 비례하여 커지는 특성을 보여주고 있다. 즉, STS가 크면 클수록 결함 복구에 소요되는 시간은 길어지며, 이에 따라 초기화에 소요되는 시간 또한 길어지게 됨을 확인할 수 있다.

6.3.1 전체 성능 분석

표 3은 무작위 파일 연산 테스트에서 LBM과 WA에서 초기화 시간을 분석한 결과이다. WA의 경우 STS가

표 3 LBM과 작업 영역 기법의 비교

Method	Initialization Time[msec]			Crash recovery Time[msec]			Size To Scan[Mbytes]		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
LBM	673	49830	8756.25	481	49307	8393.45	0.25	839.50	136.11
WA	128	2050	854.85	32	1917	541.35	0.25	32.50	8.85

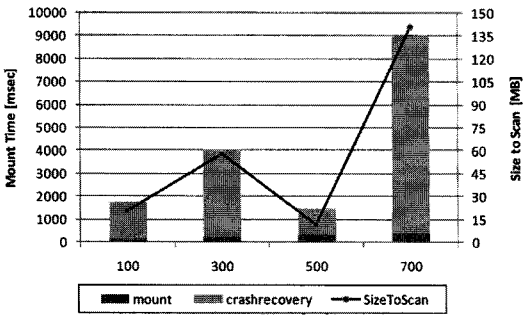

그림 9 결함 상황 분석(LBM)

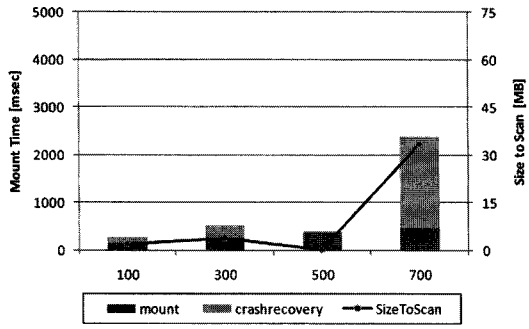

그림 10 결함 상황 분석(WA)

실험에 사용된 작업 영역 크기인 33.75MB로 제한됨을 보여주고 있으며 이와 반대로 LBM의 경우 STS가 낸드 플래시 메모리 전체 크기에 근접하는 839MB에 이르는 경우도 발생함을 보여준다. 이로 인해 마운트 시간이 WA는 최대 2초인데 반해 LBM에서는 무려 49.83초에 이르는 경우도 발생함을 확인할 수 있다.

LBM의 경우 낸드 플래시 메모리 전체 용량이 증가하면 결함 발생시 STS가 늘어나며 이로 인해 초기화 시간이 증가되지만, WA의 경우 전체 용량 증가에 상관없이 STS가 고정된다. LBM의 경우 이론적으로 STS가 최대 낸드 플래시 메모리 전체 크기만큼 읽어야 하지만, WA의 경우 STS는 최대 작업 영역 크기로 제한된다. 따라서 낸드 플래시 메모리의 용량이 증가하면 증가할수록 LBM의 경우 결함 복구 시간이 증가하지만, WA의 경우 전체 용량 증가에 관계없이 작업 영역 크기로 제한되는 장점을 갖는다.

7. 결론 및 향후과제

본 논문에서 우리는 낸드 플래시 파일 시스템의 결함 복구 수행 시간이 낸드 플래시 메모리의 용량에 비례하여 증가되는 문제를 해결하기 위해 작업 영역 기법을 제안하였고 설계한 내용을 상세히 기술하였다. 작업 영역 기법은 특정 시점에서 지정된 영역에 대해서만 파일 시스템의 변경이 이루어지도록 하는 기법으로, 결함상황 시 결함 복구를 위해 검색할 영역을 고정 크기로 제한하는 효과가 있다.

제안된 기법의 성능을 분석을 위해 우리는 작업 영역 기법을 구현하고 성능을 측정하여 기존의 기법들과 비교하였다. 그 결과, 다른 기법과 달리 작업 영역 기법의 결함 복구 수행 시간이 낸드 플래시 메모리 용량에 비례하지 않고 작업 영역의 크기에 의해 제한됨을 알 수 있었다.

향후 과제로 전체 용량 대비 작업 영역의 크기와 작업 영역 전이에 필요한 사용 비율에 대한 연구가 필요하다. 낸드 플래시 메모리의 용량이 증가함에 따라 효율적인 마운트와 결함 복구를 위한 작업 영역 비율이 있을 것으로 예상되고 이를 위한 연구가 필요하다. 특히 이러한 수치 및 작업 영역의 이동에 대한 정책은 낸드 플래시 메모리의 마모도를 균등하게 유지하는데 영향을 줄 수 있으므로, 이러한 점을 고려하여 연구를 수행하여야 한다.

본 논문에서 우리는 작업영역 제한 기법을 통해 낸드 플래시 메모리 용량이 급격하게 증가하더라도 파일 시스템의 초기화 시간 및 결함 복구 시간은 낸드 플래시 메모리 용량에 무관하도록 제한한다. 이로써 결함복구 지연으로 인한 장치의 초기화 시간 지연을 예방하고 사용자의 대기시간을 최소화 한다.

참 고 문 헌

[1] S.H. Lim and K.H. Park, "An Efficient NAND Flash File System for Flash Memory Storage," *IEEE TRANSACTIONS ON COMPUTERS*, pp. 906-912, 2006.

[2] Wu, C.H., T.W. Kuo, and L.P. Chang, The Design of efficient initialization and crash recovery for log-based file systems over flash memory. *ACM Transactions on Storage (TOS)*, vol.2, no.4, pp.449-467, 2006.

[3] Dumitru, Douglas (2007-08-16). "Understanding Flash SSD Performance," EasyCo LLC.
[4] Rosenblum, M. and J.K. Ousterhout, The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems (TOCS)*, vol.10, no.1, pp.26-52, 1992.
[5] E. Gal and S. Toledo, "Algorithms and data structures for flash memories," *ACM Computing Surveys (CSUR)*, vol.37, no.2, pp.138-163, 2005.
[6] K.S. Yim, et al., "A fast start-up technique for flash memory based computing systems," *Symposium on Applied Computing: Proceedings of the 2005 ACM Symposium on Applied computing*, vol.13, no.17, pp.843-849, 2005.
[7] Hyunchan, Park, Chuck yoo, "A Design of Efficient Crash Recovery Technique for NAND Flash File System," *Proc. of the 35th KIISE Fall Conference*, vol.35, no.2(B), pp.470-475, 2008. (in Korean)
[8] http://www.falinux.com
[9] http://193.219.66.80/datasheets/NAND%20Flash/K9G8G08.pdf
[10] http://www.aleph1.co.uk/gitweb?p=yaffs2/.git;a=summary
[11] The source code for the aging tool and benchmarks. Available : http://www.eecs.harvard.edu/~keith/usenix96/
[12] Keith A. Smith and Margo I. Seltzer, "File System Aging - Increasing the Relevance of File System Benchmarks," In *Proceedings of the 1997 ACM SIGMETRICS Conference*, pp.203-213, June, 1997.
[13] Bovet, D.P. and M. Cesati, "Understanding the Linux Kernel," 2003, O'Reilly & Associates.
[14] Love, R., "Linux Kernel Development," 2003: Sams Pub.

강 승 엽
2009년 고려대학교 컴퓨터학과 학사. 2009년~현재 고려대학교 대학원 컴퓨터학과 석사 과정. 관심분야는 운영체제, 임베디드 소프트웨어, 파일시스템

박 현 찬
2004년 고려대학교 컴퓨터학과 학사. 2004년~현재 고려대학교 대학원 컴퓨터학과 석박 통합 과정. 관심분야는 운영체제, 임베디드 소프트웨어, 파일시스템

김 기 만
1998년 아주대학교 전자공학부 학사. 2001년~현재 LG전자 CTO 멀티미디어 연구소. 2010년 고려대학교 임베디드 소프트웨어학과 석사. 조지아텍 임베디드 소프트웨어학과 석사. 관심분야는 운영체제, 임베디드 소프트웨어, 파일시스템

유 혁
정보과학회논문지 : 시스템 및 이론 제 37 권 제 1 호 참조