플래시 메모리 기반 LFS에 그림자 페이지 기법을 적용한 회복기법

황의덕° 차재혁
한양대학교 정보통신공학과
comedu@ihanyang.ac.kr, chajh@hanyang.ac.kr

A Recovery Mechanism applying the Shadow-Paging technique to Flash Memory based LFS

Eui-deok Hwang° Jae-Hyuk Cha
Dept. of Information and Communication, Hanyang University

요 약

모바일 장치에서 많이 사용되는 플래시 메모리는 작고, 저전력을 사용하며, 내구성을 지니는 비휘발성 저장장치이다. 플래시 메모리의 읽기 속도는 램과 비슷하며, 대용량화 되어가고 있지만 쓰기 속도가 램에 비해 느리고, 블록에 대한 쓰기가 제한되어 있다. 현재의 디스크 기반의 DBMS 와는 달리 플래시 메모리용 저장장치를 설계함에 있어 트랜잭션 실패시의 회복기법이 같은 블록에 다시 쓰기가 불가능한 플래시 메모리의 특성을 고려하는 것이 중요하다. 본 연구에서는 LFS에 Shadow Paging을 응용하여 플래시 메모리의 블록에 대한 쓰기 횟수를 줄이고 플래시 메모리의 특성에 맞추어 트랜잭션 실패시 효율적인 데이타 복구를 가능하게 하는 회복기법을 제안한다.

1. 서 론

플래시 메모리는 기존의 기억장치인 하드디스크를 대체할 것으로 주목 받고 있는 차세대 기억장치로서 집적회로로 구성된 비휘발성 메모리 이다, 집적도가 높고 전력소모가 작아 PDA 나 소형 가전기기, 모바일 컴퓨터 등에 많이 사용되고 있다. 플래시 메모리는 같은 크기의 DRAM 에 비해 크기가 약 30% 작고 대용량화가 가능하다. 또한 플래시 메모리에 저장된 데이타는 주 기억 장치에 로드될 필요 없이 직접 플래시 메모리 상에서 입출력이 가능하며[1][3], 전원이 차단되어도 저장된 데이타의 보존이 가능한 저장장치이다.

본 논문에서는 플래시 메모리에 데이타를 저장하고 회복 기법을 적용함에 있어 플래시 메모리의 특성과 관련한 데이타 저장 시스템을 설계하고, 설계한 저장 시스템에 포함된 회복 기법의 성능이 전체 데이타베이스의 성능에 크게 영향을 주지 않도록 하는 Shadow Paging을 이용한 LD-Shadow Paging방법을 제안한다. 제안된 회복 기법은 플래시에 쓰기 횟수를 줄이는 것에 목적을 둔다. 제안된 회복 기법이 적용될 PDA 와 같은 모바일 기기는 네트웍을 통한 데이타 싱크나 단순 입출력이 주로 이루어진다고 가정하고, 이는 동시에 여러 개의 작업을 수행하기 보다는 한 작업이 이루어지는 경우가 많으므로, 동시성 제어보다는 단일 트랜잭션 처리에 효율적인 설계에 중점을 둔다.

2장 에서는 플래시 메모리의 특성과 제약점에 대해서 좀 더 자세히 기술하고, 3장에서는 기존의 플래시 메모리 기반의 화일 시스템과 기존에 제안된 복구기법을 서술한다. 4장에서는 제안된 회복기법을 설명하고, 5장에서는 이 논문의 결론과 향후 과제를 제시한다.

2. 플래시 메모리의 특성

플래시 메모리는 저장된 데이타의 변경이 가능한 램의 장점과 전원이 차단되어도 데이타가 보존되는 ROM의 장점을 갖춘 비휘발성 메모리이다[3]. 플래시 메모리는 읽는 속도는 일반적인 DRAM과 비슷하나 쓰는 속도는 램 보다 느리다. 그 이유는 플래시 메모리에 데이타를 저장할 때 초기화 되어 있는 블록에만 데이타를 쓸 수 있고, 블록을 초기화 하는 시간이 오래 걸리기 때문이다. 이것은 플래시 메모리가 데이타를 저장할 때 상당한 시간적 손실을 가져오게 한다. 플래시 메모리는 한 방향으로만 값을 변화시키면서 데이타를 나타내기 때문에 제자리 덮어쓰기(in-place-update)가 불가능하고, 초기화 되어있는 블록이 없을 경우에 쓰기 작업이 원활하게 이루어지지 않게 된다. 또한 초기화 횟수의 제한으로, 제한된 횟수를 초과한 블록은 쓰기 오류(write failure)가 발생한다. 초기화 횟수는 약 10만~100만 번 정도이고 기록 횟수를 평준화 하기 위해서 데이타 기록이 각 블록에 걸쳐 균등하게 이루어져야 하므로, 이를 위해 블록 초기화와 사이클을 평준화하는 정책이 필요하다[2].

플래시 메모리의 종류는 크게 NAND형 방식과 NOR형 방식으로 나누어진다. NAND형 방식은 데이타 저장에 적합하며, NOR형 방식은 프로그램 격납에 적합한 구조를 가진다. NOR와 NAND 플래시 메모리의 차이는 단위 면적당 차지하는 Cell의 크기와 배치 상태가 다르며, 이에 따라 NOR 플래시 메모리에서는 임의 접근이 가능하고 프로그램의 실행이 가능한 반면, NAND 플래시 메모리에서는 빠른 접근은 가능하나 임의접근이 아닌 블록 단위의 접근만이 가능하고 프로그램의 실행이 메모리 상에서 이루어지지 못하는 차이점이 있다[3][10]. 또한 플래시 메모리에서 프로그램을 수행하는 경우 쓰기속도의 문제로 성능 저하가 발생하게 되어 램을 이용한 shadowing 기법으로 성능을 개선하기도 한다[10]. 결국 사이클이 집중적으로 발생하지 않도록 하는 것이 플래시 메모리의 수명에 직결된다.

3. 관련 연구

플래쉬 메모리를 사용하는 화일 시스템에서는 제자리 덮어쓰기가 불가능하므로 데이타를 변경하고자 할 때 변경하고자 하는 데이타가 들어있는 블록을 무효화하고 초기화된 블록을 할당받아 수정된 데이타를 저장하게 된다. 이때 시스템에 오류가

발생하여 동일한 논리 블록이 두 개가 존재하는 상황이 생기거나 또는 변경하고자 하는 데이타가 들어있던 블록이 삭제되는 상황이 발생할 수 있다. 이러한 모순이 발생하는 것을 막기 위한 각종 화일 시스템들의 구조와 기존의 플래시 메모리용 데이타 관리자의 회복 기법을 알아본다.

3.1 Flash-Memory Based file System

Flash-Memory Based file System[4] 에서는 블록의 상태 플래그를 일정한 순서에 의해 변경시키는 방법을 사용하고 있는데, 순서는 다음과 같다.

(1) 새로 할당된 블록을 allocated 라고 표시
(2) 블록 번호를 쓰고, 할당된 블록에 데이타 저장
(3) 할당된 블록을 pre-valid 로 표시
(4) 이전 블록을 invalid 로 표시
(5) 할당된 블록을 valid 로 표시

Flash-Memory Based File System 에서는 특정 블록의 상태 정보를 allocated, pre-valid, valid 의 순서로 변경하면서 데이타를 저장하게 된다. 이 블록은 언젠가는 invalid 상태가 되어 새로운 데이타를 저장할 수 있도록 초기화 된다. 상태 정보를 이용하는 것은 시스템에 오류가 발생하여 재시작하는 경우에 상태 플래그를 이용하여 블록의 상태를 파악하고 오류가 발생한 블록을 초기화 하거나 초기화되기 직전의 블록을 이전의 상태로 되돌릴 수 있다. 이 방법의 목적은 어느 한 순간에도 동일한 논리 블록 번호를 가진 valid 상태의 블록이 두 개 이상 존재하지 않도록 하는 것이다.

3.2 eNVy(Non-Volatile) System

eNVy 시스템은 별도의 비 휘발성 SRAM을 사용하는 방식을 취한다. SRAM은 Copy-on-Write scheme 을 위해 버퍼처럼 사용된다. SRAM은 휘발성 메모리이기 때문에 SRAM에는 지속적으로 전원을 공급하는 형태를 지녀야 한다. 이 시스템의 목적은 비휘발성 메모리에 간결한 배열 형태로 데이타를 저장하면서, SRAM 만큼의 엑세스 속도를 내는 것이다[5].

Copy-on-Write scheme 은 Flash Memory 의 블록 배열의 특정 주소를 사용하고자 하는 요청이 있을때 그것에 대응되는 새로운 복사본을 만들고 수정시에는 복사본의 내용을 수정한다. 페이지 테이블은 복사본의 주소를 가리키게 되고 원래의 데이타가 있는 플래시 메모리상의 블록을 무효화 한다. 이때 SRAM은 복사본을 저장하는 버퍼의 역할을 수행하게 되고 SRAM 상에서 데이타의 수정이 이루어지기 때문에 속도가 빨라진다. SRAM에 데이타가 채워졌을 경우 eNVy 시스템은 플래시 메모리에 데이타를 쓰게 되는데 이때 빈 공간을 찾차 데이타를 저장하고 빈 공간이 부족한 경우에는 *cleaning* 정책을 작동하게 된다.

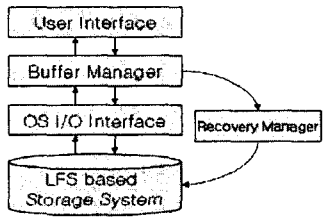
3.3 MF 로깅 기법

MF 로깅 방법은 레코드 단위의 입출력을 제공하는 플래시 메모리 기반의 레코드 데이타 관리자[6]를 위한 회복 기법으로서 메인 메모리와 플래시 메모리 상에 동시에 로그를 유지한다. 메인 메모리에 존재하는 로그를 MLog, 플래시 메모리 상에 존재하는 로그를 FLog 라고 하며, 각각의 로그는 의존관계를 유지하며, 데이타와 여러 단계의 메타 데이타에 대한 쓰기를 지연 쓰기 기법으로 수행한다[7].

4. 제안방법

기존의 회복 기법에서는 대부분 로그를 사용하여 데이타와의 의존 관계를 형성하고 트랜잭션 실패시 로그를 사용하여 복구하도록 한다. 로그는 빈번하게 저장과 수정이 이루어지므로 플래시 메모리에 적용하기에는 불합리한 요소를 포함하고 있다. 또한 시스템에 문제가 발생하여 데이타를 복구할 때 로그를 이용한 방법은 복구 시간이 상당히 오래 걸리는 단점이 있다. 본 논문에서의 데이타 저장은 플래시 메모리 상에서 이루어지고 버퍼 관리는 램 상에서 동작한다는 가정 하에 플래시 메모리 상에서 제자리 덮어쓰기가 되지 않는 문제를 개념적으로 해결한 Log-Structured File System[8](이하 LFS)과 shadow paging[9] 방법을 이용한 회복 기법을 제안한다.

4.1 기반 저장 시스템

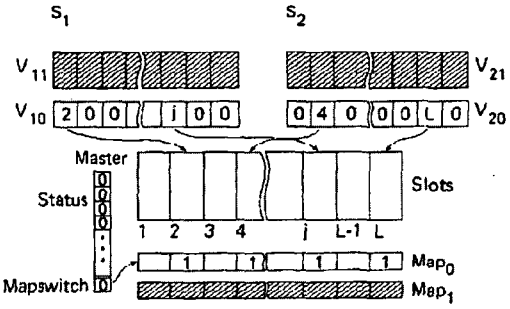


[그림 1] LFS 기반 저장 시스템

본 연구에서는 LFS 기반의 스토리지 시스템을 구현하고 구현된 저장 시스템에 제안된 회복 기법을 수행하는 회복 관리자를 추가하였다. 데이타는 플래시 메모리의 세그먼트 크기에 따른 LFS 형태로 관리하고, 단순하고 속도면에서 장점을 보이는 shadow paging 방법을 롤백(Rollback)과 트랜잭션 실패시의 회복 기법에 사용한다. 시스템에 문제가 발생했을 경우를 대비하여 주기적으로 검사점을 수행하고 시스템 재시작시 회복 기법을 수행하도록 한다. 버퍼 관리자는 단순히 Log-Structured Data List(이하 LDL)와 Data Page Flag List(이하 DFL)만을 추가하여 사용한다.

4.2 기존의 그림자 페이지 기법

기존의 shadow paging 방법에서 버퍼관리의 형태는 [그림 2]와 같다.

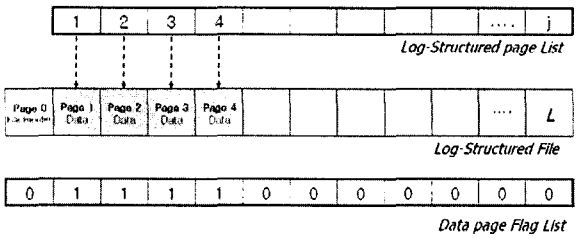


[그림 2] Shadow paging 의 버퍼형태[9]

각 슬롯(slot) 은 페이지를 의미하고, Map은 페이지의 상태 플래그를 저장한다. V는 현재 트랜잭션 S가 사용중인 페이지의 리스트를 의미한다. 제안된 방법은 버퍼 관리자의 역할에 LDL과 DFL을 추가하여 시스템에 문제 발생시 데이타 페이지를 회복하는 것이다. 또한 한 개의 트랜잭션을 처리하는데 중점을 두었기 때문에 기존의 Shadow Paging 기법에서 사용되는 여러 가지 버퍼 리스트들을 LDL과 DFL로 단순화 할 수 있고 로그를 사용하는 회복 기법에 비해 버퍼와 데이타 저장소의 수정된 데이타 저장 횟수를 줄일 수 있다.첫 번째로 LDL은 기존 방법의 페이지 리스트 V와 같은 역할을 하게 된다. 차이점은 V가 버퍼상의 페이지를 가리키게 되고 LDL은 플래시 메모리 상의 데이타 페이지를 가리키게 된다는 것이다. LDL 은 현재 사용중인 페이지 번호를 저장하는 것으로 트랜잭션의 완료나 커밋시에 LFS 데이타 저장소의 마지막에 덧붙이게 된다. 두 번째로 DFL 은 기존 방법의 Map 과 유사한 것으로 LFS 데이타 저장소의 데이타 페이지 상태를 가지고 있는 것이다.

DFL 의 플래그는 0:빈페이지, 1:현재 사용중, 2:수정후 의 3가지 상태를 가진다. 커밋 수행시 LDL 과 마찬가지로 LFS 데이타 저장소의 마지막에 덧붙이게 된다. 제안된 방법의 형태는 [**그림 3**]과 같다.

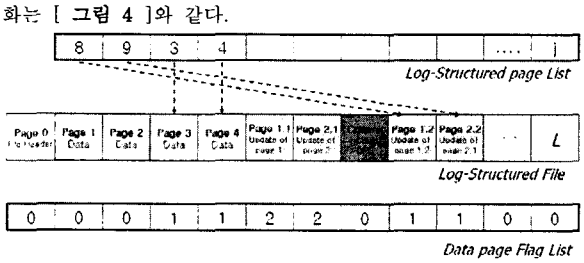
L은 LFS 데이타 저장소의 마지막 데이타 페이지이다. DFL은 데이타 페이지들의 상태를 나타내고, LDL 에는 현재 사용중인 페이지의 번호가 저장된다. 사용 중인 페이지는 LDL에 원래 데이타 페이지 번호가 저장되고, 수정된 데이타는 별도의 버퍼 관리자에 의해 관리된다. 이때 버퍼 관리자는 LDL을 이용하여야 하므로 버퍼 관리자가 LDL을 연동하여 사용할 수 있도록 LDL 과 버퍼 관리자의 페이지들 간에 연동할 수 있도록 한다. DFL 의 초기 상태는 LFS 데이타 저장소의 상태를 반영한다.



[**그림 3**] LD-Shadow Paging 의 버퍼 형태

4,3 데이타 삽입, 삭제, 수정

데이타 수정시에는 버퍼 관리자에서 데이타를 수정하고 이를 커밋 할 경우 LFS 데이타 저장소의 마지막 페이지 이후에 수정된 데이타를 추가하고 LDL과 DFL을 추가한다. 데이타 삽입은 저장된 페이지가 없는 상태에서 수정하는 것과 같이 동작한다. 저장이 끝난 후 LDL의 페이지 번호를 초기화 하고, 새로운 페이지 번호로 갱신 한다. DFL 역시 페이지의 상태 플래그를 수정하게 된다. 한 번 수정된 페이지가 다시 수정되었을 경우에도 데이타 수정은 버퍼 관리자에서 이루어지고, 수정된 페이지가 커밋 되지 않은 상태에서는 LDL은 기존의 데이타 페이지 번호를 유지한다. 수정이 완료되어 커밋된 페이지는 LFS에 데이타 페이지로 쓰여지게 된다. 이때 DFL 은 수정된 페이지의 상태 플래그를 2로 갱신하고 새로 저장된 페이지의 상태 플래그를 1로 갱신한다. 마지막으로 LDL과 DFL은 데이타 페이지의 마지막에 덧붙여지게 된다. 삭제는 수정 과정과 동일하게 이루어 지고 페이지의 데이타가 모두 삭제되었을 경우에는 LDL에 저장된 페이지 번호를 갱신하지 않고 제거한 상태로 LDL을 저장한다. 이 때 DFL에는 추후 초기화와 재사용을 위해 데이타가 모두 삭제된 페이지의 플래그를 2 로 설정한다. 데이타 수정후의 LFS 데이타 저장소와 LDL, DFL의 상태 변화는 [**그림 4**]와 같다.



[**그림 4**] 데이타 수정 후의 상태 변화

4,4 데이타 회복

수정중인 데이타를 롤백하는 경우 LDL에서 원래 페이지의 번호를 확인하고 원래 버퍼의 페이지를 데이타 페이지로 교체하면 된다. 문제가 발생하여 시스템이 재시작 되었을 경우 LFS 데이타 저장소의 마지막 페이지부터 읽기 시작하여 커밋 또는 검사점 페이지를 찾아 LDL과 DFL을 이용하여 버퍼관리자를 초기화 하게 되므로 안정된 상태를 유지하게 된다. 제안된 방법을 이용하면 데이타 저장은 커밋 또는 검사점 수행시에만 이루어지게 되므로 기존의 LFS가 수정된 데이타가 발생할 경우마다 데이타를 저장하는 것에 비해 데이타 저장 횟수를 줄일 수 있으며, 버퍼 관리자는 LDL 과 DFL 두 개의 버퍼만을 추가하여 효율적인 회복 기법을 수행 할 수 있다.

5. 결론과 향후 과제

플래시 메모리는 하드디스크를 대체할 수 있는 새로운 비휘발성 기억장치 이다. 램과 비슷한 속도로 데이타를 읽을 수 있고 내구성 등의 장점이 있다. 하지만 초기화 횟수의 제한과 쓰기 속도 문제 등의 단점도 있다. 이 때문에 플래시 메모리용 화일 시스템이나 데이타 관리자는 플래시 메모리의 특성에 따른 설계를 고려해야 한다. LFS와 shadow paging은 플래시 메모리의 쓰기 연산이 동작하는 방법과 유사한 부분이 많아 플래시 메모리용 데이타 관리자를 설계하는데 유용한 지표가 된다. 또한 제안된 회복 기법을 사용하면 기존의 LFS를 좀 더 효율적으로 사용할 수 있다. 향후에는 메모리 사용량과 사이클 발생 횟수를 좀 더 줄일 수 있는 회복 기법이 필요할 것이다.

참고문헌

[1] 김한준, 이상구, "신뢰성 있는 플래시메모리 저장시스템 구축을 위한 플래시메모리 저장 공간 관리 방법", 정보과학회 논문지(시스템 및 이론), 27(6), pp. 567-582, 2000

[2] Deborah See and Clark Thurlo, "Managing Data in an Embedded System Utilizing Flash Memory", Intel, 1995

[3] Brian Diper and Markus Levy, *Designing with Flash Memory*, Annabooks, 1993

[4] Atsuo Kawaguchi, Shingo Nishioka and Hiroshi Motoda, "A Flash-Memory Based File System", Proceedings of the 1995 USENIX Technical Conference

[5] M. Wu, and W. Zwaenepoel, "eNVy : A Non-Volatile Main Memory Storage System", Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.86-97, 1994

[6] 유아영, 김한준, 정재헌, 이상구, "플래쉬 메모리 화일 관리자의 설계 및 구현", 정보과학회 추계학술 발표회, pp. 383-386, 1997

[7] 정유나, 이상구, "MF로깅 기법 : 플래쉬 메모리 기반 데이타 관리자를 위한 새로운 복구 기법", 한국 데이타베이스 학술대회 논문집, 15(1) , pp. 82-89, 1999

[8] 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

[9] Raymond A. Lorie, Physical Integrity in a Large Segmented Database, ACM TODS, Vol2,No. 1,March 1977,pages 91-104

[10] Cost Saving with NAND Shadowing Reference Design with MotorolaTM MPC8260TM and ToshibaTM CompactFlashTM, System Solutin from Toshiba American Electronic Components, Inc. Revision 1.0, July 2002