논문 2018-13-27

복구 성능 향상을 위한 플래시 메모리 관리 기법

(A Flash Memory Management Method for Enhancing the Recovery Performance)

박 송 화*, 이 정 훈, 조 성 우, 김 상 현
(Song-Hwa Park, Jung-Hoon Lee, Sung-Woo Cho, Sang-Hyun Kim)

Abstract : NAND flash memory has been widely used for embedded systems as storage device and the flash memory file systems such as JFFS2, YAFFS/YAFFS2 have been adopted by these embedded systems. The flash memory file systems provide the high performance and overcome the limitations of flash memory. However, these file systems don't solve the slow mount time problem when a sudden power failure happens. In this paper, we proposed a flash memory management method for enhancing the recovery performance. The proposed method manages the flash memory block type and stores the block type information at recovery image block. When file operations are occurred, our method stores the file information at the metadata block before and after the file operation. When mounting the flash memory, our method only scans the recovery image blocks and metadata blocks. The proposed method reduces the mount time by seeking the metadata block locations fast by using the recovery image blocks. We implemented the proposed method and evaluation results show that our method reduces the mount time 13 ~ 46 % compared with YAFFS2.

Keywords : NAND flash memory, Recovery, YAFFS2

Ⅰ. 서 론

스마트폰, 내비게이션 등 휴대용 기기의 대중화로 인하여 임베디드 환경에서 NAND 플래시 메모리의 사용이 증가하고 있다. 빠른 부팅, 저전력, 데이터의 안정성을 요구하는 임베디드 환경에서 NAND 플래시 메모리를 효율적으로 사용하기 위해서 NAND 플래시 메모리 전용 파일 시스템에 대한 연구가 활발히 진행되고 있다.

NAND 플래시 메모리는 임베디드 시스템에 요구되는 비휘발성, 저전력, 빠른 입출력 특성을 보유하고 있으나, 극복해야 할 3 가지 특성을 가지고 있다. 첫째, 플래시 메모리의 각 비트가 1에서 0으로 단방향으로만 전환되는 하드웨어적인 특성으로 인하여 제자리 덮어쓰기 (in-place-update)가 불가하다. 따라서 덮어쓰기를 위해서 지움 연산을 선행해야만 한다. 둘째, 읽기와 쓰기 연산은 페이지 단위로 수행되지만, 지움 연산은 블록 단위로 수행되며 지움 연산 속도가 느리다. 셋째, 플래시 메모리의 각 블록은 지움 횟수가 제한되어 있으며, 전체 공간을 균등하게 사용하도록 해야 한다 [1].

NAND 플래시 메모리의 제약사항을 극복하고 효율적으로 사용하기 위하여 JFFS (Journaling Flash File System)2, YAFFS (Yet Another Flash File System), YAFFS2 등과 같은 플래시 메모리 전용 파일 시스템을 사용한다. JFFS2는 LFS (Log Structured File System)에 기반한 파일 시스템으로, 갱신 처리 시 다른 자리에 쓰기 (out-of-place) 기법을 사용하여 제자리 덮어쓰기가 불가능한 단점을 해결하였다. YAFFS는 JFFS의 느린 마운팅 속도와 많은 메모리 소비와 같은 문제점들을 개선시켰으나, 마운트 시에 전체 플래시 영역을 읽어야 하고, 파일데이터의 위치정보를 저장하기 위한 자료구조를 메모리상에 구축 및 검색하는데 빠른 연산을 지원하지 못한다. YAFFS2는 체크포인트 정책을 통

*Corresponding Author (shpark25@lignex1.com)
Received: Sep. 10. 2018, Revised: Oct. 1. 2018, Accepted: Oct. 2. 2018.
S. Park, J. Lee, S. Cho, S. Kim: LIG Nex1

ⓒ IEMEK J. Embed. Sys. Appl. 2018 Oct. 13(5) 235-243
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2018.13.5.235

해서 YAFFS의 마운트 시간을 개선하였으나, 전원 결함 등의 이유로 정상적인 언마운트 작업이 되지 않으면 체크포인트를 저장하지 못하고, 메타데이터 자료구조를 구축하기 위해 플래시 메모리의 전체 영역을 스캔해야 한다 [2].

안정적인 전원 공급을 보장하지 못하는 모바일 기기와 같은 임베디드 시스템에서 비정상 종료 후 재부팅 시에 빠른 마운트 및 복구가 가능한 기법들이 연구되고 있다.

ERFFS (Efficient and Reliable Flash File System)는 최소한의 메타데이터를 이용한 빠른 마운트와 복구를 지원하는 구조를 제안하여 YAFFS에 비하여 마운트 및 복구 성능을 향상시켰으나 연산 중 비정상적인 종료 시에 이전 상태로 롤백 (roll-back)이 보장되지 않는 단점이 있다 [3].

Ju는 메모리 기반 데이터베이스인 모바일 기기에서 고장 발생 시 신속한 회복과 효율적인 로깅이 이뤄지도록 T-트리를 이용한 회복기법을 제안하였다. T-트리를 이용한 회복기법은 로그 버퍼를 위한 T-트리를 사용하여 입출력 횟수를 감소시키고 쓰기 작업을 마지막 위치에 이뤄지게 함으로써 성능을 향상시켰으나, T-트리 사용에 따른 추가적인 연산 및 메모리 사용이 발생하는 단점이 있다 [4].

본 논문에서는 YAFFS2를 기반으로 하여 복구 성능을 향상시킬 수 있는 기법을 제안한다. 제안 기법은 플래시 메모리의 블록을 6 가지 타입으로 나누어 관리하고 파일 연산에 대한 형상 관리를 수행한다. 블록 타입 정보와 파일에 대한 형상 정보를 각각 복구 이미지 블록과 메타데이터 블록에 저장하고, 마운트 시에는 복구 이미지 블록과 메타데이터 블록만을 스캔하여 복구 및 마운트가 가능하도록 하였다.

본 논문의 구성은 다음과 같다. Ⅱ 장에서는 플래시 메모리와 YAFFS2에 대해 기술하고, Ⅲ 장에서는 본 논문에서 제안하는 플래시 메모리 관리 기법에 대해 설명한다. Ⅳ장에서는 제안하는 기법의 성능을 실험을 통해 평가하고, Ⅴ장에서는 결론 및 향후 과제에 대해 제시한다.

Ⅱ. 관련 연구

1. 플래시 메모리 [5]

플래시 메모리는 비휘발성 컴퓨터 기억장치로, 부피가 작고 가벼우며 소비 전력이 낮고 입출력 속도가 빠르다. 플래시 메모리는 게이트 타입 (gate type)과 셀 (cell)을 구성하는 구조에 따라 NOR 플래시 메모리와 NAND 플래시 메모리로 구분하며, NOR 플래시 메모리는 바이트 단위 접근이 가능하며 집적도가 낮고 비싸기 때문에 코드 저장 및 실행 용도로 주로 사용되며, NAND 플래시 메모리는 대용량이며 NOR 보다 저렴하기 때문에 데이터 저장용으로 주로 사용된다.

NAND 플래시 메모리는 동일한 크기의 페이지의 집합으로 구성된 블록의 집합으로 구성되며, 읽기 및 쓰기 연산은 페이지 단위로 처리하고, 지움 연산은 블록 단위로 처리한다. NAND 플래시 메모리는 각 비트가 1에서 0으로 단방향으로만 전환되는 하드웨어적인 특성으로 인하여 제자리 덮어쓰기 (in-place-update)가 불가하다. 따라서 데이터 갱신을 위해서는 지움 연산을 통하여 초기화된 메모리 공간의 확보가 필요하다 (erase-before-write). 플래시 메모리의 각 블록은 지움 횟수에 제약이 있으므로 특정 블록들에 지움 연산이 집중되지 않고, 전체 공간을 균등하게 사용하도록 해야 한다.

이러한 NAND 플래시 메모리의 장점을 극대화하고 단점을 극복하여 다양한 시스템의 저장 장치로 사용하기 위해서 NAND 플래시 메모리 파일 시스템에 대한 연구가 꾸준히 진행되고 있다.

2. YAFFS2 (Yet Another Flash File System2)

YAFFS는 NAND 플래시 전용 파일 시스템으로, 외부 갱신 기법 (out-place-update)을 사용하여 제자리 덮어쓰기 문제를 극복하였다. YAFFS2는 YAFFS의 다음 버전으로 기본적인 특징은 YAFFS와 동일하며 속도가 개선되었다. YAFFS2는 그림 1과 같이 chunk와 태그 (tag)로 구성된 구조의 배열 형태로 구성되며, chunk는 데이터 chunk와 헤더 chunk로 구분된다. 데이터 chunk에는 파일 데이터가 저장되며, 헤더 chunk에는 표 1과 같은 파일 및 디렉터리의 속성이 저장된다. YAFFS는 마운트 시에 전체 플래시 메모리 영역을 스캔하여 파일 시스템을 구축한다 [6, 7].

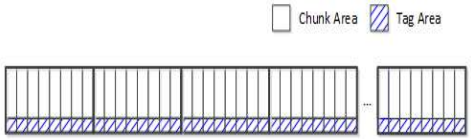

그림 1. YAFFS/YAFFS2 구조
Fig. 1 Structure of a YAFFS/YAFFS2

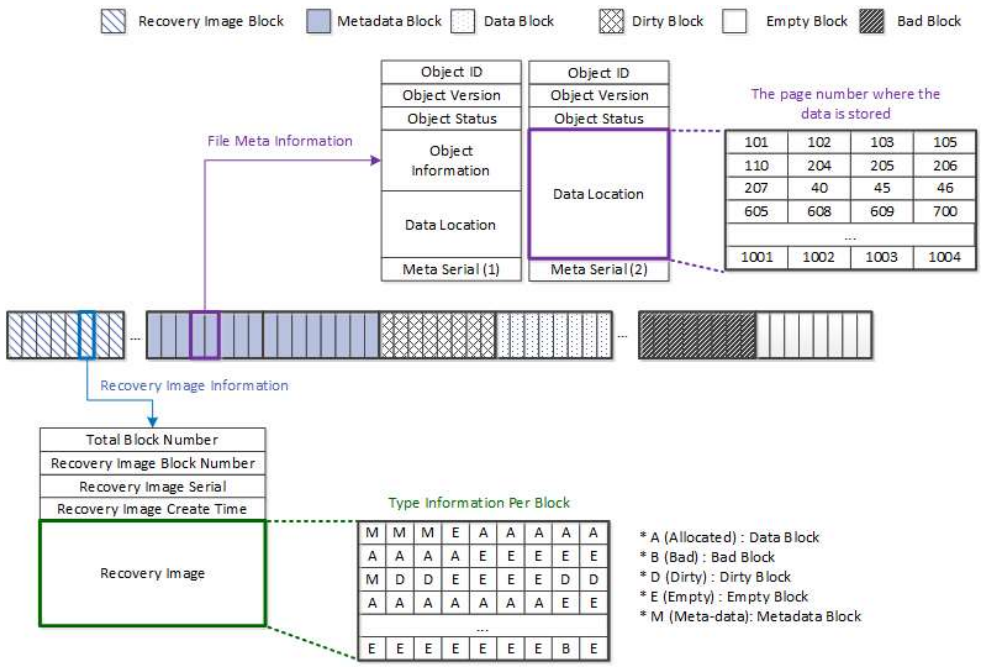

그림 2. 제안 기법의 플래시 메모리 구조

Fig. 2 A flash memory structure of the proposed method

표 1. YAFFS2의 헤더 Chunk 정보

Table 1. Header chunk information of the YAFFS2

Item	Description
Object Type	• Object type information : Unknown, General File, Symbolic Link, Directory, Hard Link, Special File
Parent Object ID	• Object ID of the parent
File/Directory Name Checksum	• If normal, OxFFFF
Mode	• Mode information
UID	• Same as Linux UID
GID	• Same as Linux GID
Access Time	• UNIX time
Modify Time	• UNIX time
Metadata Modify Time	• UNIX time
File Size	• The total size of the file

YAFFS2는 YAFFS의 긴 마운트 시간을 개선하기 위하여 체크포인트 정책을 사용한다. 언마운트 시점에 메인 메모리의 메타데이터 자료 구조를 NAND 플래시 메모리에 저장하고, 마운트 시에 체크포인트를 읽어 메타데이터를 구축한다. 그러나 전원 결함 등의 이유로 정상적으로 언마운트가 수행되지 않으면 체크 포인트를 저장하지 못한다. 체크포인트가 저장되지 않는 경우에는 메타데이터를 구축하기 위해 플래시 메모리 전 영역을 스캔해야 하므로, YAFFS와 동일한 마운트 시간이 소요된다 [2].

Ⅲ. 복구 성능 향상을 위한 플래시 메모리 관리 기법

본 절에서는 복구 성능 향상을 위한 플래시 메모리 관리 기법에 대하여 기술한다.

1. 시스템 구조

제안하는 시스템의 플래시 메모리 구조는 그림 2와 같다. 전체 플래시 메모리 영역 중 고정된 일부 영역이 플래시 메모리 이미지 저장을 위하여 사용되며 그 외 영역은 파일 데이터와 파일 데이터에 대한 메타 정보를 저장하기 위하여 사용된다.

제안하는 시스템은 그림 3과 같이 YAFFS2를 기반으로 하며 플래시 메모리 관리를 위한 블록 상태 관리자, 복구 관리자, 파일 관리자로 구성된다.

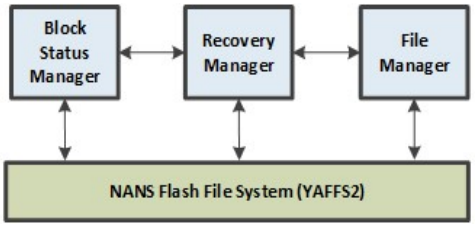

그림 3. 플래시 메모리 관리 구조
Fig. 3 Flash memory management structure

표 2. 블록 타입
Table 2. Block type

Block Type	Description
Recovery Image Block	• Stores block status information • Stored in fixed area of flash memory
Metadata Block	• Stores meta information about the file
Data Block	• Stores file data
Dirty Block	• Blocks in which all pages in the block are invalidated • Target block to perform erase operation
Empty Block	• Blocks in which data can be stored • Blocks that can be allocated for write operations
Bad Block	• Unavailable block

1.1 블록 상태 관리자 (Block Status Manager)

제안 기법에서는 플래시 메모리의 블록을 표 2와 같이 6 가지 타입으로 분류하여 관리하며, 타입에 따라 특정 용도로 사용한다.

블록 상태 관리자는 플래시 메모리 블록 할당, 지움 등에 따라 블록 상태가 변경되는 것을 모니터링하고 해당 정보를 관리한다. 블록 상태 정보 변경 시 블록 상태 관리자는 복구 관리자에게 최신 블록 상태 정보를 알려준다.

1.2 복구 관리자 (Recovery Manager)

복구 관리자는 블록 상태 관리자로부터 블록 상태 정보를 수신하고 복구 이미지를 생성하여 저장하는 역할을 수행한다. 대용량 파일의 생성, 가비지 컬렉션 (Garbage Collection) 등의 경우에는 블록의 상태 정보가 빈번하게 변경되며 복구 이미지 생성으로 인한 부하가 발생된다. 부하를 최소화하기 위하여 복구 관리자는 복구 이미지 저장 후 일정 시간 이내에는 새 복구 이미지를 생성하지 않는다. 복구 관리자는 복구 이미지의 저장 시간을 관리하고 블록 상태가 변경되더라도 지정된 복구 이미지 생성 시간 간격 후에 새 복구 이미지를 생성하여 저장한다.

표 3. 파일 메타 정보
Table 3. Meta information of a file

Item	Description
Object ID	• The identifier of the object
Object Version	• 1 at object creation • Increment by 1 for each complete operation on the object
Object Status	• Completion of operation : Operation start, Operation finish
Object Type	• Same as header chunk information
Parent Object ID	
File/Directory Name Checksum	
Mode	
UID	
GID	
Access Time	
Modify Time	
Metadata Modify Time	
File Size	
Data Location	• Location information per page
Meta Serial	• When meta information is stored in multiple pages, the order of meta information

1.3 파일 관리자 (File Manager)

제안 기법에서는 파일 연산 수행 중 비정상적인 종료 시에도 파일 복구가 가능하도록 파일에 대하여 표 3과 같이 메타 정보를 생성하여 관리한다.

파일 관리자는 파일 연산 발생을 모니터링하여 파일의 형상을 관리한다. 그림 4와 같이 파일 관리자는 파일 연산 시작 전 현재 파일 정보를 백업하고 파일 연산 완료 후 갱신된 파일 정보를 기록한다. 파일 관리자가 저장한 메타 정보는 파일 시스템 마운트 시에 사용된다.

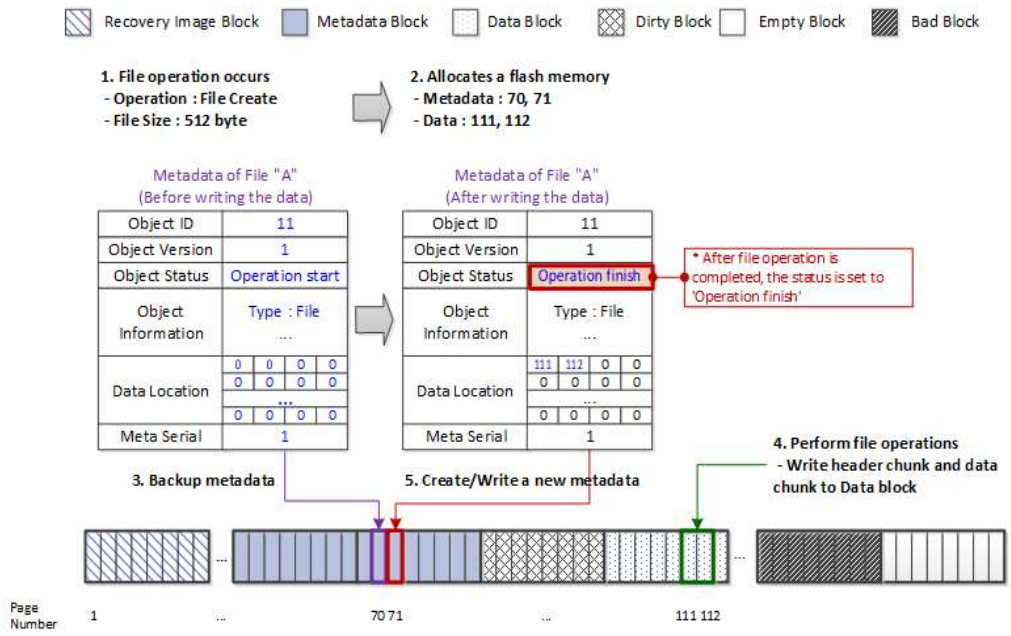

그림 4. 제안 기법의 파일 연산 수행 과정의 예

Fig. 4 Example of a file operation process of the proposed method

2. 복구 성능 향상 기법

제안 기법에서는 복구 성능을 향상시키기 위하여 복구 이미지 블록과 메타데이터 블록을 사용하여 마운트를 수행한다. 마운트 시에 메타데이터 블록 정보를 사용하여 파일 구조를 구축하는데, 이를 위하여 파일 연산 전·후에 메타데이터를 기록한다.

제안 기법에서의 파일 연산 수행 과정은 다음과 같다. 파일 연산이 발생하면 파일 연산 수행을 위해 필요한 플래시 메모리 영역을 할당한다. 파일 연산의 종류가 쓰기 (수정) 또는 삭제인 경우, 메타데이터 및 파일 데이터 쓰기를 위하여 각각 메타데이터 블록과 데이터 블록에서 빈 페이지가 할당된다. 이때, 메타데이터 쓰기용 공간은 파일 전·후로 2 번 저장되므로 최소 2 개의 페이지가 할당된다. 플래시 메모리 쓰기 영역 할당 후에는 메타데이터의 백업을 수행한다. 파일 연산 수행 전의 현재 상태 정보를 포함한 메타데이터를 메타데이터 블록에 기록한다. 이때 파일 연산 수행 전이므로 메타데이터의 오브젝트 상태는 '연산 시작'으로 설정된다. 메타데이터의 백업 후에는 파일 연산 종류에 따라 실제 파일 데이터를 기록하거나 기존 데이터 영역을 무효화 (invalid) 처리한다. 파일 연산이 완료되면 신규 메타데이터를 생성하여 메타데이터 블록에 기록하며, 이때 신규 메타데이터의 오브젝트 상태는 '연산 종료'로 설정된다. 따라서 오브젝트 별 메타데이터의 버전이 가장 높고 오브젝트 상태가 '연산 종료'인 메타데이터가 파일의 최신 정보를 포함하고 있는 메타데이터이다. 만약 파일 연산 중 비정상 종료가 발생했을 경우에는 파일에 대한 최신 메타데이터는 오브젝트 상태가 '연산 시작'으로 설정되어 있게 된다.

그림 5는 제안 기법의 마운트 과정을 나타내고 있다. 마운트가 시작되면 복구 관리자는 복구 이미지 블록 영역을 스캔하여 최신 복구 이미지를 검색한다. 복구 이미지 블록 영역은 고정된 영역이며, 복구 이미지에는 복구 이미지 정보의 시리얼 정보와 생성 시간 정보가 포함되어 있으므로 복구 관리자는 최신 복구 이미지 정보를 검색할 수 있다. 복구 관리자는 최신 복구 이미지 검색 후, 복구 이미지에 포함된 블록 별 타입 정보를 메인 메모리 (RAM)에 적재하고 해당 정보를 사용하여 플래시 메모리 영역 중 메타데이터 블록에 대해서만 스캔한다. 복구 관리자는 메타데이터 블록에 저장된 메타데이터를 사용하여 파일 구조의 구축 및 복구를 수행한다. 최신 메타데이터의 오브젝트 상태가 '연산 종료'인 경우에는 해당 메타데이터에 저장된 정보를 사용하여 해당 파일의 구조 구축이 가능하다.

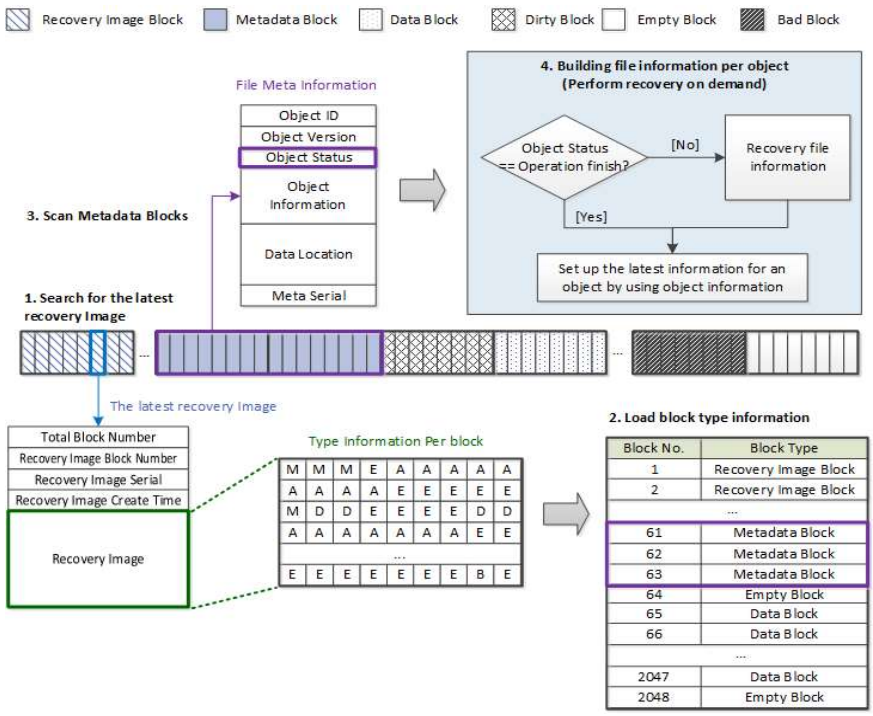

그림 5. 제안 기법의 플래시 메모리 마운트 과정
Fig. 5 A flash memory mount process of the proposed method

오브젝트의 상태가 '연산 시작'인 경우에는 파일 연산 중 비정상 종료가 발생된 것이므로 해당 메타데이터에 저장된 정보를 사용하여 연산 수행 전의 상태로 파일을 복구한다. 즉, 파일 복구를 위해서 전체 플래시 메모리 영역을 스캔할 필요가 없이 복구 이미지 블록, 메타데이터 블록만을 스캔하여 복구가 가능하다.

그림 6은 제안 기법의 플래시 메모리 마운트 과정의 예를 나타내고 있다. 그림 6의 파일 A와 같이 메타 정보의 오브젝트 상태가 '연산 종료'인 경우에는 파일 연산이 정상적으로 완료된 것이므로 파일에 대한 복구가 필요 없으며 메타 정보를 사용하여 파일 정보를 구축한다. 그림 6의 파일 B와 같이 메타 정보의 오브젝트 상태가 '연산 시작'인 경우에는 파일 연산이 비정상적으로 종료되었다고 판단한다. 복구 관리자는 파일 연산이 비정상적으로 종료되었다고 판단한 경우에는 파일 연산 전 상태로 복구한다. 그림 6에서 파일 B의 최신 헤더 chunk는 페이지 907에 저장되었으나 파일 연산이 완료되지 못하였으므로 페이지 901의 정보로 복구되어야 한다. 페이지 901에 저장된 헤더 chunk의 정보가 메타정보에 이미 포함되어 있으므로 별도의 데이터 블록 스캔 과정 없이 메타데이터 블록만 사용하여 파일 연산 수행 전 상태로 복구를 수행한다.

Ⅳ. 성능평가

1. 실험 환경

본 논문에서 제안한 기법의 성능을 평가하기 위하여 SSD Extension for DiskSim Simulation을 참고하였으며 [8, 9], 시뮬레이션 환경은 표 4와 같다. 제안 기법에서 복구 블록 저장을 위해 사용되는 블록 개수와 YAFFS2에서 체크 포인트 저장을 위해 사용되는 블록 개수는 동일하게 60 개로 설정하였다.

2. 기존 기법과의 성능 비교

제안한 기법의 성능을 평가하기 위하여 YAFFS2의 성능과 비교하였다. 복구 성능을 확인하기 위하여 파일 연산 수행 중 비정상적인 종료 후 마운트 시에 소요되는 시간을 측정하였다.

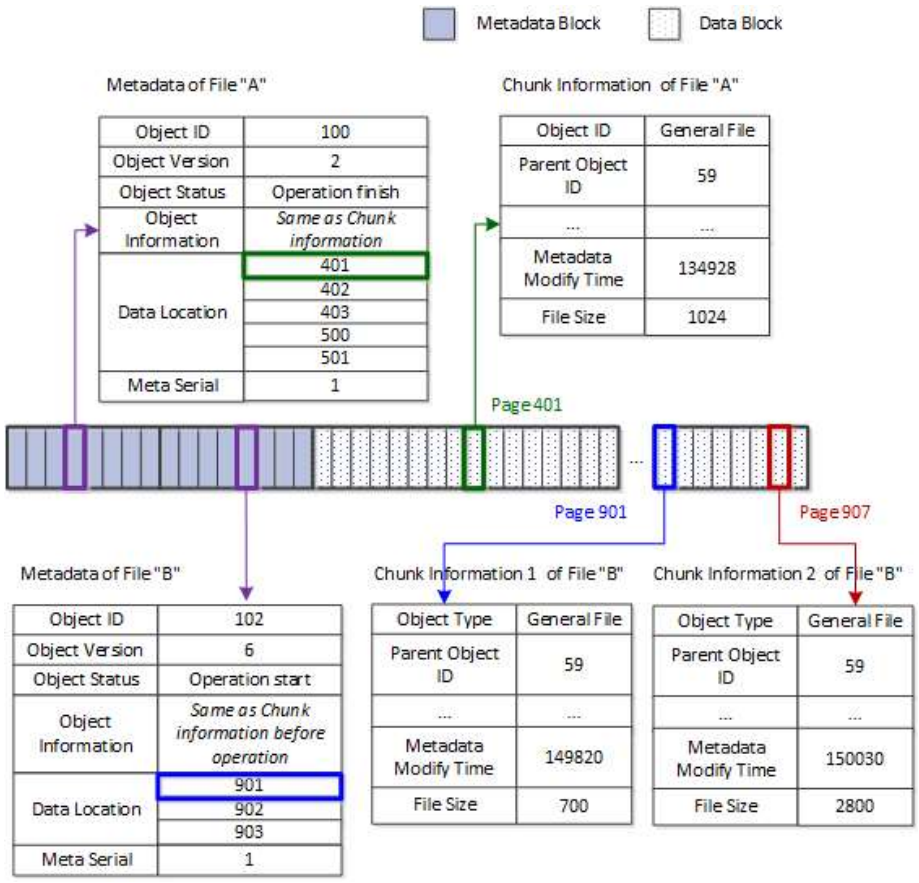

그림 6. 제안 기법의 플래시 메모리 마운트 과정 예
Fig. 6 Example of a flash memory mount process of the proposed method

표 4. 시뮬레이션 파라미터
Table 4. Parameter of simulation

Item	Value
#Blocks/Chip	2,048
#Pages/Block	64
Page Read/Write Latency	60 us / 800 us
Block Erase Latency	1.5 ms
Program-Erase (P/E) Cycle	10,000

파일 시스템 복구 및 마운트에 소요되는 시간을 측정하기 위하여 플래시 메모리의 사용률이 10 %, 40 %, 70 %인 경우에 대하여 파일 연산 수행 중 비정상적인 종료 상황을 모의하였다. 플래시 메모리에 저장된 파일의 크기 및 개수는 랜덤하게 생성하였다.

제안 기법과 YAFFS2 모두 비정상적인 종료 후 마운트 시 파일 복구를 수행하였으며, 복구가 수행될 경우에 마운트 시간은 그림 7과 같다. 제안 기법은 YAFFS2에 비해 마운트 시간이 약 13 ~ 46 % 감소되었으며, 플래시 메모리 사용량 증가에 따른 마운트 시간 증가율이 YAFFS2에 비하여 적은 것을 확인할 수 있다.

제안 기법은 파일 연산 발생 시, 파일에 대한 형상 관리를 위하여 메타데이터 블록에 파일 연산 전·후에 메타 정보를 저장한다. 따라서 YAFFS2에 비하여 메타 정보를 위한 공간을 추가로 사용하게 되며 본 시험 시에는 그림 8과 같이 약 12 % 정도 블록 사용률이 높은 것을 확인하였다. 제안한 기법은 저용량 파일의 개수가 많거나 파일 갱신 연산이 많을 경우, 메타데이터 블록을 많이 사용하게 되어 YAFFS2에 비해 추가로 사용하는 블록 개수가 많아진다.

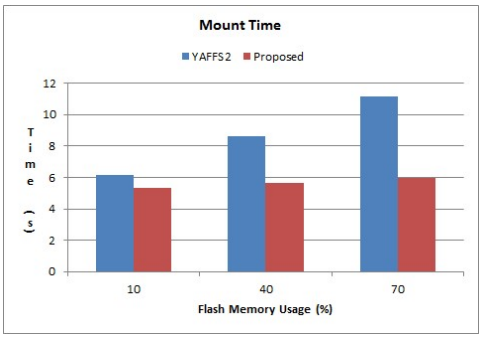

그림 7. 마운트 시간 비교
Fig. 7 Comparison of the mount time

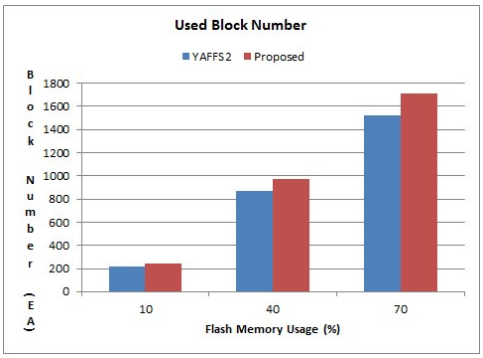

그림 8. 블록 개수 사용 비교
Fig. 8 Comparison of the used block number

V. 결 론

본 논문에서는 NAND 플래시 전용 파일 시스템인 YAFFS2를 기반으로 복구 성능을 향상시키기 위한 기법을 제안하였다.

제안한 기법은 플래시 메모리의 블록을 6 가지 타입 (복구 이미지 블록, 메타데이터 블록, 데이터 블록, 무효 블록, 빈 블록, 배드 블록)으로 구분하여 관리하고 용도에 따라 사용하도록 하였다. 파일 연산 수행 시, 메타데이터 블록에 파일 연산 전·후의 파일에 대한 형상 변경 정보를 기록하여 메타데이터 블록만을 스캔하여 마운트 및 복구가 가능하도록 하였다. 또한 복구 이미지 블록에 블록 타입 정보를 저장하여 메타데이터 블록을 찾기 위한 스캔 과정을 최소화하였다.

실험 결과, 비정상적인 종료 후 마운트 수행 시에 YAFFS2에 비하여 마운트 시간이 약 13 ~ 46 % 감소된 것을 확인하였다. 하지만 파일의 형상 관리를 위한 메타데이터 블록의 사용으로 인하여 약 12 % 정도의 메모리를 추가로 사용하는 것을 확인하였다.

향후 본 연구는 복구 성능 향상을 위해 사용되는 저장 용량을 감소시킬 수 있는 방향으로 나아가야 할 것이다.

References

[1] F. Douglis, R. Caceres, M.F. Kaashoek, P. Krishnan, K. Li, B. Marsh, J. Tauber, “Storage Alternatives for Mobile Computers,” Proceedings of the 1st USENIX Symposium on Operating System Design and Implementation, pp. 25-37, 1994.

[2] H.W. Seo, M.S. Shin, D.J. Park, “A Metadata Management Scheme for Efficient Mount on the YAFFS2 Flash File System,” Journal of KIISE, Vol. 41, No. 2, pp.122-131, 2014 (in Korean).

[3] J.W. Jin, T.H. Lee, S.H. Lee, K.D. Chung, “Implementation of Efficient Reliable Flash File System,” Journal of Korea Multimedia Society, Vol. 11, No. 5, pp.651-660, 2008 (in Korean).

[4] H.J. Ju, S.J. Cho, “A Efficient Logging and Recovery using T-tree Index for Mobile Transaction,” Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology, Vol. 7, No. 11, pp.837-844, 2017 (in Korean).

[5] S.H. Park, J.H. Lee, W.O. Lee, H.E. Kim, “A Adaptive Garbage Collection Policy for Flash-Memory Storage System in Embedded Systems,” Journal of Embedded Systems, Vol. 12, No. 3, pp.121-130, 2017 (in Korean).

[6] A Robust Flash File System Since 2002, https://yaffs.net

[7] J.W. Jin, T.H. Lee, S.H. Lee, K.D. Chung, “Implementation of Efficient and Reliable Flash File System,” Journal of Korea Multimedia Society, Vol. 11, No. 5, pp.651-660, 2008 (in Korean).

[8] V. Prabhakaran, T. Wobber, “SSD Extension

for DiskSim Simulation Environment." Microsoft Research 2009.
[9] K9GAG08U0M 2G x 8bit MLC NAND Flash Memory Data Sheet, Samsung Electronics, https://www.samsung.com, 2007.

Song-Hwa Park (박 송 화)

Park received a B.S. and M.S. degrees in Computer Engineering from Pusan National University in 2005 and 2007, respectively. She is currently a research engineer at LIG Nex1. Her research interests include flash file system, embedded system and track fusion.
Email: shpark25@lignex1.com

Sung-Woo Cho (조 성 우)

Cho received a B.S. and M.S. degrees in Information and Communication Engineering from Sejong University in 2011 and 2013, respectively. Currently, he is a research engineer at LIG Nex1. His research interests include embedded system and signal processing
Email: sungwoo.cho@lignex1.com

Jung-Hoon Lee (이 정 훈)

Lee received a B.S., M.S. and Ph.D. degrees in Electronic Engineering from Kyungpook National University in 1997, 1999 and 2004, respectively. He is now a research engineer at LIG Nex1. His research interests include SONAR system, embedded system, and signal processing.
Email: junghoon.lee04@lignex1.com

Sang-Hyun Kim (김 상 현)

Kim is received B.S. degree in electronic engineering from Chonbuk national University in 2014 and M.S degree from Hanyang University in 2016. Currently, he is a research engineer at LIG Nex1. His research interests include embedded system, target tracking and pattern recognition.
Email: sanghyeon.kim@lignex1.com