# EXT3 저널링 파일 시스템 기능 개선 연구

장승주*, 이성헌*
*동의대학교 컴퓨터공학과
e-mail : sjjang@deu.ac.kr, kkulee@deu.ac.kr

# A Study of EXT3 Journaling File System

Seung-Ju Jang*, Seong-Heon Lee*
*Dept. of Computer Engineering, Dong-Eui University

**요 약**

컴퓨터에서 안정성은 가장 중요시된다. 파일 시스템의 안정성이 문제가 되어 시스템이 다운되거나 여러 가지 문제가 발생할 경우, 데이터가 손상 되거나 데이터 복구가 불가능하게 될 수 있다. EXT3 파일 시스템은 이러한 문제점을 보안하기 위해 기존의 EXT2 파일 시스템에 저널링 기능이 추가된 파일 시스템이다. EXT3 파일 시스템은 매우 안정적이고, 기존의 EXT2 파일 시스템에서의 변경 과정이 쉽고 간단하여 일반적으로 많이 사용 되고 있는 저널링 파일 시스템이다. EXT3 파일 시스템은 기본적으로 ordered mode 를 사용하는데 메타데이터가 저널에 기록되는 추가적인 과정이 이루어져야 하기 때문에 성능의 저하가 발생한다. 본 논문에서는 ordered mode 에 압축 저장 기법을 적용하여 효율적인 공간 관리와 쓰기 속도가 향상된 ordered mode 저널링 기능을 제안하여 파일 시스템 기능을 개선하고자 한다.

## 1. 서론

컴퓨터 사용량이 증가함으로써 파일 시스템의 안정성도 중요시 되고 있다. 파일 시스템이란, 파일의 내용과 그 파일과 관련된 데이터 즉, 메타데이터들을 유지하고 관리하는 체계이다. 리눅스는 표준 파일 시스템으로 EXT 파일 시스템, EXT2 파일 시스템으로 변천해왔다. EXT2 파일 시스템은 파일의 데이터와 메타데이터를 동시에 저장하지 않는 비동기식 파일 시스템이다. 비동기식 파일 시스템은 메타데이터를 파일의 내용을 저장할 때 저장하는 것이 아니라, 메모리에 두었다가 일정한 시간 간격을 두고 저장한다. 비동기식 파일시스템은 성능상의 장점이 있기는 하지만 반대로 단점도 가지고 있다. 예상치 못했던 비정상적 시스템 종료나 여러 가지 문제가 발생하여 시스템이 다운될 경우, 재부팅 하는 과정에서 파일 시스템의 일관성을 검사하기 위해 상당한 시간을 소비한다. EXT3 파일 시스템은 이런 문제점을 해결하기 위해 데이터 베이스에서 쓰이는 저널링 기술을 적용하였다.

저널링 파일 시스템은, 사용자가 데이터를 입력 또는 수정하면 그 데이터를 저장장치에 기록하기 전에 메타 데이터를 로그에 기록한다. 만약, 기록 중 문제가 발생하여 비정상적인 종료를 하게 되면 재부팅 할 때 로그에 기록된 메타 데이터를 참고로 하여 파일 시스템을 재 작성 하거나 복구 할 수 있다. EXT3 파일 시스템은 매우 안정적이고 치명적인 문제점도 없으며, EXT2 파일 시스템에서 EXT3 파일 시스템으로의 변경도 쉽고 간단하여 일반적으로 많이 사용 되고 있는 저널링 파일 시스템이다. EXT3 파일 시스템은 기본적으로 ordered mode 를 사용한다. Ordered mode 는 메타데이터를 수정하기 전, 저널링 공간에 수정하고자 하는 메타데이터를 기록하고 저장장치에 데이터를 쓴 다음 저널에 있는 메타데이터에 대한 정보를 저장장치에 옮긴다. 저장장치에 메타데이터보다 데이터를 먼저 써서 데이터를 저널링 하지 않고도 데이터의 손상을 막을 수 있다. 하지만 한번의 쓰기 동작을 위해 실제로 두 번의 쓰기 동작이 필요하고, 추가적인 디스크 공간이 필요하다.

본 논문에서는 압축 저장 기법을 적용하여 효율적인 저장장치 공간 관리와 쓰기 속도를 향상시키는 EXT3 파일 시스템의 ordered mode 의 개선을 제안한다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 논문과 관련된 연구를 설명하고, 3 장 에서는 EXT3 파일 시스템의 구성에 대해 기술하였다. 4 장에서는 개선된 저널링 파일 시스템의 설계에 대해 설명하고 5 장에서는 본 연구의 결론으로 구성한다.

## 2. 관련 연구

저널링 파일 시스템에서 저널은 파일 시스템에 반영될 수정 사항을 버퍼에 로그로 남겨두었다가 주기적으로 파일 시스템에 커밋된다. 비정상적 종료가 발생하면, 저널은 저장되지 않은 데이터를 복구하기 위한 검사 지점으로 사용되고 파일시스템의 메타 데이터의 손상을 막아준다[7].

EXT3 파일 시스템에서 저널링은 세 가지로 나뉘어진다. writeback mode 는 메타데이터만 저널에 기록되며, 데이터는 저장장치 해당 위치에 직접 기록된다. 이렇게 하면 파일 시스템 구조를 보존해 손상을 방지

하지만, 메타데이터를 저장하고 나서 데이터를 기록하기 전에 시스템이 비정상적인 종료를 일으킬 때 데이터가 손상될 수 있다. 이런 문제를 해결 하기 위해 ordered mode 를 사용할 수 있다. ordered mode 는 데이터를 기록한 다음에야 메타데이터를 저널에 기록한다. 이런 방식으로 복구 이후에 데이터와 파일 시스템 일관성을 보장한다. 마지막으로 journal mode 는 메타데이터와 데이터가 모두 저널에 기록된다. journal mode 는 파일 시스템 손상과 데이터 손실에 대해 리스크를 최소화 할 수 있지만 성능이 저하 된다. XFS, ReiserFS 등 다른 저널링 파일 시스템은 메타데이터를 수정하는 중 시스템에 문제가 발생하여 비정상적인 종료를 하게 되면 일관성을 유지하지 못하는 것과 다르게 EXT3 파일 시스템의 ordered mode 는 파일시스템의 일관성을 유지시키는 안정성을 보여준다[4].

## 3. EXT3 파일 시스템 구조

EXT3 파일 시스템은 매우 안정적이며 많은 리눅스 커널 기반 운영 체제에서 주 파일 시스템으로 쓰이고 있다. EXT3 는 EXT2 코드 기반으로 EXT2 와 EXT3 의 디스크 포맷이 동일하다. 그리고, EXT2 에서 자료 삭제 및 손실 없이 EXT3 파일 시스템으로 변경할 수 있고, XFS, ReiserFS 등 다른 저널링 파일 시스템에서는 사용 할 수 없는 fsck 를 사용하여 파일 시스템 일관성을 점검할 수 있다. EXT3 는 저널링 기능을 사용하여 불필요한 fsck 를 피하는 것이다.

fsck 란, 예상치 못한 시스템 종료 시 일어나는 파일 시스템의 일관성을 체크하고, 수정하는 프로그램을 말한다. fsck 는 링크 수 나 데이터 블록의 값들을 사용해서 디스크의 슈퍼블록, 아이노드, 디렉토리 정보 등의 이상유무를 점검 한다. fsck 는 파일 시스템의 크기가 커짐에 따라 수행 시간이 길어 진다. 메타데이터의 손실이 저널링 기능으로 복구할 수 없는 경우 fsck 는 유용하게 사용될 수 있다.

EXT3 에서 저널링은 Journaling Block Device layer(JBD)라고 하는 API 를 사용하여 관리한다. JBD 는 모든 종류의 블록 디바이스상의 저널을 구현하도록 설계되었다[5, 7, 8].

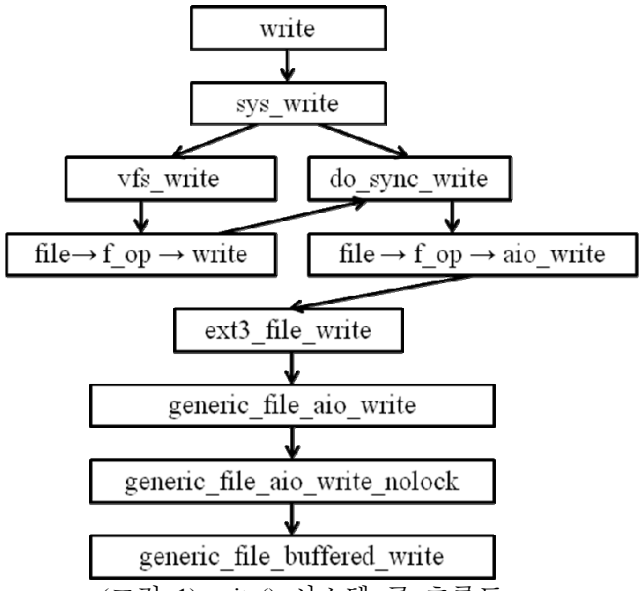
### 3.1 EXT3 의 ordered mode 저널링

EXT3 파일 시스템은 기본적으로 ordered mode 로 마운트 된다. ordered mode 에서는 메타데이터만 저널에 기록된다. 데이터는 기록되지 않지만 만일 관련된 메타데이터가 저널에 기록되면 데이터는 저장장치에 반드시 기록된다. 메타데이터에 대한 로그를 데이터 블록보다 먼저 디스크의 저널에 기록하고 로그의 기록이 끝나면 데이터 블록이 디스크에 기록된다. 저널에 기록했던 로그 기록을 참고하여 디스크에 기록되어있는 메타 데이터를 수정한다.

파일을 읽거나 수정 중 비정상적인 종료나 커널 패닉 상태가 되면, 저널은 새로운 파일을 가리키게 되거나 추가된 데이터가 넘겨지지 않으며 삭제 처리가 된다. 이러한 방식으로 메타데이터만 저널에 기록되더라도 EXT3 는 데이터와 메타데이터의 일관성을 유지할 수 있다[7, 8].

### 3.2 EXT3 의 write() 시스템 콜

리눅스 시스템에서 파일에 쓰는 작업은 write() 시스템 콜을 통해 이루어 진다. EXT3 파일 시스템도 역시 데이터를 저장장치에 저장하기 위해서 write() 시스템 콜을 호출한다.



(그림 1) write() 시스템 콜 흐름도

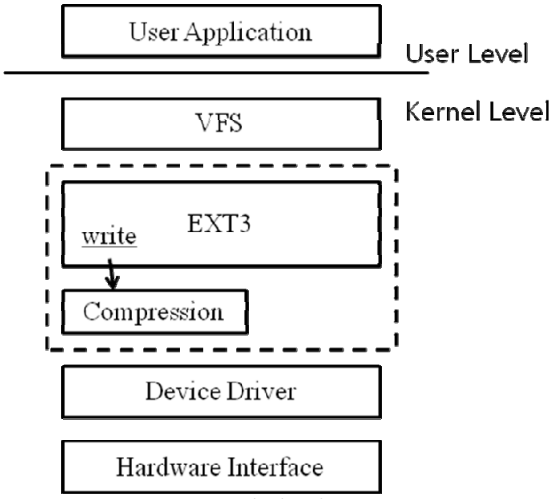
(그림 1)은 EXT3 파일 시스템에서 write() 시스템 콜을 호출 하였을 때 커널에서의 처리를 보여준다. write()는 sys_write()함수를 호출하게 되고, sys_write()는 vfs_write()또는 do_sync_write()를 호출하게 된다. vfs_write()와 do_sync_write()함수는 EXT3 의 file operation 에 정의 되어있는 write 메소드 또는 aio_write 메소드를 호출하게 되고 두 메소드는 결국 ext3_file_write()함수를 호출하게 된다.

ext3_file_write()함수에서 generic_file_aio_write()함수와 generic_file_aio_write_nolock()과정을 거쳐 실제로 generic_file_buffered_write()에서 저장 장치에 write 작업을 수행하게 된다. generic_file_buffered_write()함수에 압축 저장 기법을 적용한다. 저장장치에 데이터가 write 되기 전에 압축 과정을 거쳐 저장장치 공간을 효율적으로 사용할 수 있게 한다[1, 3].

## 4. 저널링 파일 시스템 설계

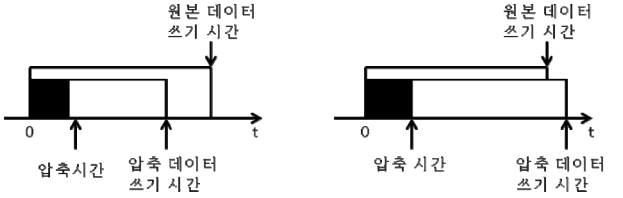
EXT3 파일 시스템에 저장장치의 효율적인 공간 활용과 쓰기 속도를 향상시키기 위해 파일에 압축 저장 기법을 적용한다. 파일의 압축 시 사용자 수준의 압축 파일과 비압축 파일을 식별한 후 비압축 파일만 압축 저장 기법을 적용한다. 여기서 사용자 수준의 압축 파일이란, mp3, mpeg, jpeg, pdf 파일등과 같은 멀티미디어 파일과 같이 사용자 수준에서 압축 기법이 적용된 파일을 말한다.

(그림 2)는 논문에서 제시하는 전체 시스템 구조이다.




(그림 2) 전체 시스템 구조

높은 압축률을 보이는 파일의 경우, 데이터 압축 시 소모되는 시간과 압축된 데이터를 디스크에 기록하는 시간이 원본 파일을 디스크에 기록하는 시간보다 수행시간이 단축 될 것이다. 반면, 사용자 수준의 파일과 같이 낮은 압축률을 보이는 파일은 원본 크기와 비슷한 크기의 압축 파일이 생성 되므로 불필요한 파일 압축에 따른 자원이 낭비가 되고 디스크에 기록하는 시간 또한 늘어날 것이다. 따라서 압축률이 낮은 파일과 압축률이 높은 파일을 식별하여 선택적으로 압축하는 기법이 요구된다.




(그림 3) 높은 압축률을 보이는 파일과 낮은 압축률을 보이는 파일의 쓰기 수행 시간 비교

(그림 3)은 파일의 압축률에 따른 파일 쓰기 수행 시간의 비교를 보여준다. 선택적 압축을 하지 않고, 사용자 수준의 압축파일을 압축 저장할 경우 중복 압축에 따른 불필요한 자원낭비와 쓰기 속도를 저하시킬 것이다.

파일 압축 시 사용자 수준의 압축 파일을 판단하기 위하여 파일의 일정 부분을 압축한 뒤 압축률을 검사한다. 압축률의 임계값을 설정하여 파일이 임계값보다 낮은 압축률을 보일 경우 사용자 수준의 압축 파일로 판단하고 파일의 나머지 부분은 압축하지 않는다. 임계값보다 압축률이 높을 경우 파일의 나머지 부분도 압축을 수행하고 압축된 파일을 디스크에 쓰기 한다.

파일 압축은 gzib 포멧의 zlib 라이브러리를 사용하여 압축한다. zlib 압축 라이브러리는 거의 아무런 제한 없이 자유로이 사용할 수 있는 라이선스이며 메모리상에서의 압축 및 압축해제 함수들과 압축 해제된 데이터의 무결성 검사기능을 제공한다. 그리고 zlib 는 안정성이 입증되어 압축한 다음 정상적인 압축 풀기를 보장하고, 같은 방식으로 다시 압축을 시도했을 때 더 이상 압축되지 않는다[6].

본 논문에서는 리눅스 기반의 커널 2.6.33 버전을 사용 하였다. 커널 2.6.33 버전은 Fedora core 9 에서 제공하고 EXT3 파일 시스템을 지원한다.

커널의 /fs/ext3/inode.c 파일의 ext3_journal_start()부분에 printk()함수를 적용하여 커널을 리빌딩 한 뒤 val/log 의 메시지파일을 확인해 보면 ext3_write_begin() 함수에서 저널링 기능이 시작됨을 알 수 있다. 사용자 프로그램이 write() 시스템 콜을 호출하면/mm/filemap.c 파일의 generic_file_buffered_write() 함수에 의해 쓰기 연산의 수행이 시작된다. Zlib 라이브러리를 이용하여 generic_file_buffered_write()함수에 압축 기법을 적용한다.

## 5. 결론 및 향후과제

본 논문에서 EXT3 파일 시스템 기능 개선 연구는 기존의 EXT3 파일 시스템의 ordered mode 에 압축 저장 기법 적용을 제안한다. 기존 EXT3 의 파일 쓰기 시 저장장치의 효율적인 공간 활용과 쓰기속도를 향상 시키기 위해 선택적으로 압축 저장 기법을 적용시킨다.

파일 시스템의 안정성에 문제가 생기면 파일 시스템의 동작이 중단되어 수정 중이던 데이터가 손상되거나 복구가 불가능하게 될 수도 있다. 본 논문에서 제안하는 EXT3 파일 시스템 기능 개선 연구는 기존 EXT3 파일 시스템의 ordered mode 저널링을 수행하기 때문에 안정성을 그대로 유지할 수 있다.

본 논문에서 제안하는 개선된 EXT3 파일 시스템에서 압축률이 높은 파일은 추가적인 파일 압축 과정을 수행하지만 파일 쓰기속도 향상과 디스크 공간을 효율적으로 사용할 수 있을 것이다. 하지만 사용자 수준의 압축 파일등과 같은 압축률이 낮은 파일의 경우도 파일의 일정 부분이 압축 작업이 수행되기 때문에 성능상 단점이 발생 할 것이다.

## 참고문헌

[1] The Linux Cross Reference
http://lxr.linux.no/
[2] Korean Linux Documentation Project
http://kldp.org/
[3] Linux kernel source code
http://www.kernel.org/
[4] Daniel Robbins, “고급 파일 시스템 개발자 가이드, part7 & 8”
http://www.ibm.com/developerworks/kr/library/l-fs7.html
http://www.ibm.com/developerworks/kr/library/l-fs8.html
[5] RedHat, “Red Hat’s New Journaling File System : ext3”, 2001
[6] http://www.zlib.net/manual.html
[7] TAKAHASI HIROKAZU, “리눅스 커널 2.6 구조와 원리”, 한빛미디어, 2007
[8] Daniel P.Bovet, Marco Cesati, “Understanding the LINUX KERNEL”, O’Relly, January 2001.