Ext3 파일 시스템을 위한 log-ordered mode 저널링

석진선[O] 편상형[1], 노재춘[1], 김경훈[2]
세종대학교[O], 세종대학교[1], 글루시스[2]
durgatm @naver.com[O] , coolsmekr@yahoo.co.kr[1], jano@sejong.ac.kr[1], kgh@gluesys.com[2]

log-ordered mode journaling for Ext3 filesystem

Jinsun Suk[O], Sanghyung Pyun[1], Jaechun No[1], Gyeong-Hun Kim[2]
Sejong university[O], Sejong university[1], Gluesys.co.,ltd[2]

요 약

안정성은 가장 중요시 되는 컴퓨터의 특성이다. 안정성에 문제가 생긴 경우, 전체의 동작이 중단 되어 수정 중이던 데이터가 손실되거나 기존 데이터의 복구가 불가능하게 되는 상황이 초래될 수 있다. 이러한 문제점들을 극복하기 위해 DualFS[8], log-structured 파일 시스템[10] 등의 다양한 저널링 파일 시스템들이 사용된다. 특히 Ext3 파일 시스템은 일반적으로 매우 안정적으로 동작하며, 치명적인 문제도 없기 때문에 많이 사용 되고 있는 저널링 파일 시스템이다. 하지만 Ext3 파일 시스템의 기본 저널링 모드인 ordered mode는 메타 데이터에 대한 기록을 남기기 위해 여분의 디스크 공간이 필요 하고, 저널링을 위한 추가적인 작업이 수행되어야 한다. 본 논문에서는 ordered mode의 추가적인 공간과 추가적인 작업의 필요 없이 저널링을 수행하는 log-ordered mode를 제안한다.

1. 서 론

오늘날 컴퓨터는 사회 전체에서 사용되고 있으며 사용 범위도 넓어져가고 있다. 당연히 사용자가 요구하는 컴퓨터의 특성도 다양해지고 있는데 그 중에서도 특히 안정성이 중요시되고 있다. 안정성에 문제가 발생한 경우 파일 시스템 전체의 동작이 중단 되어 수정 중이던 데이터가 손실되거나 기존 데이터의 복구가 불가능하게 되는 상황이 초래될 수 있기 때문이다.

이러한 안정성의 미약함으로 인해 발생하는 문제점들을 해결하기 위해 다양한 저널링 파일 시스템이 사용된다. 특히 Ext3 파일 시스템이 매우 안정적으로 동작하며, 치명적인 문제도 없기 때문에 많이 사용된다. 하지만 Ext3 파일 시스템의 ordered mode 저널링은 로그 기록을 저장하기 위한 추가적인 디스크 공간과 로그 를 저널링 공간에 기록하기 위한 추가적인 쓰기 작업이 필요하다. 이러한 Ext3 파일 시스템의 저널링 방법은 DualFS[8]와 log-structure 파일 시스템[10]의 저널링이 여분의 공간 확보와 추가적인 쓰기 작업 없이 수행된다는 것에 비하면 비효율적인 방법이다.

본 논문에서는 Ext3 파일 시스템의 ordered mode를 추가적인 디스크 공간과 추가적인 작업 없이 효율적으로 수행하는 log-ordered mode를 제안한다.

2. 관련 연구

DualFS[8], Log-structured 파일 시스템[10] ,Ext3 파일 시스템[1][2][5][6] 등의 파일 시스템들은 시스템의 예기치 못한 재부팅 등의 재난으로 인해 발생하는 메타데이터들의 손상을 다양한 방법으로 복구한다. Ext3 파일 시스템은 메타데이터의 손상을 막기 위해 저널이라는 특정 공간에 변경되어진 메타데이터에 대한 로그 기록을 기록한 후, 저널에 쓰여 진 로그 기록을 참고로 디스크 상의 실제 메타데이터의 내용을 수정한다. 이러한 Ext3 파일 시스템의 저널링 기법은 데이터를 저장하기 위한 공간 외의 로그를 기록하기 위한 추가적인 공간이 필요하고, 변경되어진 메타데이터를 디스크에 쓰기 위한 작업외의 로그를 저널링 공간에 기록하기 위한 추가적인 작업이 필요하다. 반면에 Log-structured 파일 시스템과 DualFS는 여분의 공간 확보와 추가적인 쓰기 작업 없이 저널링을 수행한다. Log-structured 파일 시스템은 세그먼트(segment) 단위 별로 변경된 메타데이터를 저장하고, 데이터 복구 시 기존의 메타데이터를 저장하고 있는 세그먼트를 대신하도록 한다. Log-structured 파일 시스템에서 변경된 메타데이터에 의해 대체된 기존의 메타데이터는 클리너에 의해 삭제된다. DualFS는 Log-structured 파일 시스템을 변형, 보완한 것으로 주요 특징은 데이터와 메타데이터를 다른 디스크 또는 동일한 디스크 내의 서로 다른 파티션과 같은 서로 분리된 공간에 저장, 관리하고 저널링을 위한 메타데이터의 복사본을 가지지 않는 다는 것이다. 파일 시스템의 안정성에 문제가 생겨 갑자기 중단된다고 해도 메타데이터가 분리된 디스크에 저장되어 있기 때문에 시스템의 로그를 메타데이터에 첨부하는 것만으로도 안정성을 유지할 수 있다.

3. Ext3 파일 시스템

Ext3 파일 시스템은 일반적으로 매우 안정적으로 동작하며, 치명적인 문제도 없기 때문에, 많은 사람들이 사용 하고 있는 저널링 파일 시스템이다. 또한 Ext2 파일 시스템의 디스크 구조를 그대로 상속받음으로써 다른 저널링 파일 시스템에서는 사용 할 수 없는 fsck[11]를 사용하여 파일 시스템의 일관성을 점검, 보수 하는 것도 가능하다.

fsck는 링크 수와 데이터 블록들의 값을 사용하여 디스크의 이상 유무를 점검한다. 점검하는 항목은 슈퍼블록, 실린더 그룹 디스크립터, 아이노드, 파일 시스템의 데이터 블록과 디

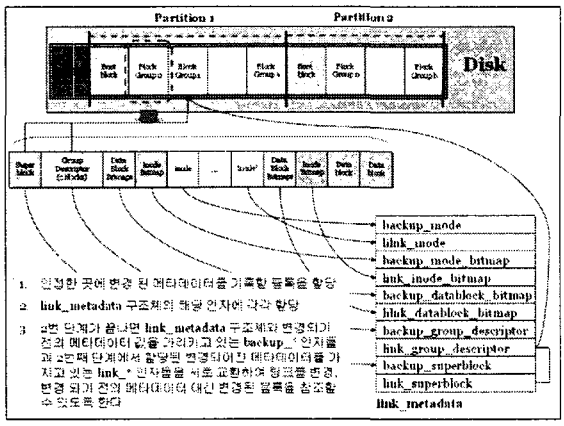
렉토리 정보이다. 이런 방법으로 수행되는 fsck의 수행 시간은 파일 시스템의 메타데이터의 양에 비례한다. 큰 파일시스템일 경우에는 fsck의 수행 시간이 매우 길어질 수 있다. 이런 소모적인 fsck를 수행하지 않기 위해 제안된 방법이 저널링 파일 시스템 이지만 저널링으로 복구할 수 없는 메타데이터의 손실이 발생한 경우에 fsck는 매우 유용하게 사용될 수 있을 것이다.[7]

3.1 Ext3의 ordered mode 저널링

ext3 파일 시스템의 저널링은 모든 데이터와 메타데이터의 변화를 저널에 기록하는 *journal mode*[1][2][7], 메타데이터에 관한 정보만을 저널에 기록하는 *Ordered mode*[1][2][7] 그리고 메타데이터의 변화만을 로그로 남기는 Write back mode 세 가지로 나뉘어 진다. 세 가지 모드 중 Ext3 파일 시스템의 기본 저널링 모드인 ordered mode 는 메타데이터에 대한 로그 기록을 데이터 블록보다 먼저 디스크의 저널에 기록한다. 로그의 기록이 끝나면 데이터 블록이 디스크에 기록되고 마지막으로 첫 번째 단계에서 저널에 기록했던 로그 기록을 참고하여 디스크에 기록되어 있는 메타데이터를 변경한다. ordered mode의 수행방법은 기존의 메타데이터를 먼저 디스크에 기록하고 데이터 블록을 기록하는 저널링 방법과는 전혀 다른 새로운 방식이다.[7] XFS, ReiserFS등의 다른 저널링 파일 시스템들의 저널링이 메타데이터를 수정하는 중 시스템이 멈추는 상황에서 파일 시스템의 일관성을 유지하지 못하는 것과는 다르게 Ext3 파일 시스템의 ordered mode 저널링은 어떠한 상황에서도 시스템의 일관성이 유지되는 강력함을 보여준다.

하지만 ordered mode로 저널링을 수행하기 위해서는 메타데이터의 로그 기록을 저장 할 여분의 디스크 공간인 저널이 필요하며 데이터 블록을 디스크에 쓰기 위한 작업과 로그 기록을 저널에 쓰기 위한 작업으로 이루어진 두 번의 쓰기 작업이 수행되어야 한다. 디스크의 처리속도는 상대적으로 느리기 때문에 디스크에 두 번 접근하는 것은 성능 저하의 원인이 될 수 있다[8].

다음 섹션에서 Ext3 파일 시스템의 ordered mode 저널링을 기반으로 설계되고 보다 효율적으로 저널링을 수행하는 log-ordered mode 의 구조에 대해 언급하겠다.



[그림1] log-ordered mode 저널링의 구조

4. log-ordered mode 의 구조

Log-ordered mode는 변경되기 전의 메타데이터를 포함하고 있는 블록을 변경된 메타데이터를 포함하고 있는 블록으로 대체함으로써 저널을 위한 추가적인 작업을 수행하지 않는 방법이다. 한 번의 쓰기 작업으로 메타데이터의 손상을 예방하면서 사용자의 쓰기 작업 요청을 처리하는 것이다. Log-ordered mode 저널링은 [그림1]과 같은 구조를 가지며 그림의 link_metadata 구조체를 필요로 한다. link_metadata 구조체는파일이 변경되었을 경우에 영향을 받는 메타데이터인 파일 아이노드, 데이터블록 비트맵, 아이노드 블록 비트맵, 그룹 디스크립터, 슈퍼블록들의 수정되기 전과 수정된 후의 블록 모두를 포함되어 있는 구조체로 log-ordered mode가 저널 영역과 추가적인 작업 없이 저널링을 수행할 수 있도록 해준다.

저널링은 다음과 같이 3단계로 수행 된다. 첫 단계로 수정되어진 메타데이터의 값을 쓸 공간을 확보하여 구조체 link_metadata의 link_inode, link_inode_bitmap, link_dtatabock_bitmap, link_superblock에 각각 할당한다. 이 작업이 끝나면 수정 된 메타데이터의 값을 첫 단계에서 구조체 link_metadata에 할당한 공간에 기록한다. 마지막으로, 수정되기 전의 메타데이터를 가지고 있는 link_metadata 구조체의 backup_inode, backup_inode_bitmap, backup_dtatabock_bitmap, backup_superblock들의 값과 link_inode, link_inode_bitmap, link_dtatabock_bitmap, link_superblock들이 가리키는 수정된 메타데이터 값이 서로 교환하여 메타데이터의 변경이 파일 시스템에 반영 되도록 한다.

log-ordered mode 저널링은 변경되기 전의 메타데이터를 포함하고 있는 블록을 그대로 두고 다른 공간에 변경된 메타데이터를 포함하고 있는 블록을 기록한다. 때문에 변경된 메타데이터를 포함하고 있는 블록을 쓸 장소를 확보해야만 한다. 다음 섹션에서 log-ordered mode 가 메타데이터의 종류 별로 어떻게 저널링을 수행하는지에 대해 자세하게 언급하겠다.

4.1 log-ordered mode 의 슈퍼 블록 (super block) 저널링

슈퍼 블록은 파티션의 정보를 담는 블록으로 파티션내의 모든 그룹이 동일한 정보를 가진다.[1][12][13] log-ordered mode 는 이런 Ext3 파일 시스템의 특성을 이용하여 슈퍼 블록을 저널링 한다. 먼저 동일 파티션 내에서 가장 인접한 슈퍼블록에 변경된 슈퍼 블록을 기록하고, link_superblock에 연결한다. 파티션내의 모든 슈퍼블록의 값은 동일해야 하므로 동일 파티션에 포함된 모든 슈퍼블록의 값을 link_superblock에 연결된 슈퍼블록의 값으로 변경 한다. 이 작업은 Ext3 파일 시스템의 동일 파티션내의 슈퍼블록은 동일한 정보를 가진다는 특성을 유지하기 위한 것으로 Ext3 파일 시스템이 원래 수행하는 작업이다. 그렇기 때문에 log-ordered mode 저널링은 변경된 슈퍼 블록의 쓰기 작업을 한번만 수행하면 저널링을 수행할 수 있다고 말할 수 있다.

4.2 log-ordered mode의 그룹 디스크립터 (group descriptor) 저널링

그룹 디스크립터는 파티션내의 모든 그룹들에 대한 정보를 담는 블록으로 동일한 파티션에 포함되는 모든 블록그룹들

은 동일한 값의 그룹 디스크립터를 가진다.[1][12][13] log-ordered mode 저널링은 이러한 Ext3 파일 시스템의 그룹 디스크립터의 특성을 이용하여 슈퍼블록을 저널링 하는 방법과 유사하게 저널링 한다. 먼저 동일한 파티션내의 인접한 그룹디스크립터에 변경된 그룹 디스크립터를 쓰고 link_group_descriptor 에 할당한다. 그리고 동일 파티션내의 모든 그룹 디스크립터가 동일한 정보를 가지게 하기 위한 Ext3 파일 시스템의 작업을 이용하여 파티션내의 모든 그룹 디스크립터의 정보를 link_group_descriptor 에 할당되어진 값으로 변경해준다.

4.3 log-ordered mode의 아이노드 (inode) 저널링

아이노드는 파일에 대한 정보를 담는 것으로 파일마다 하나의 아이노드를 가진다. 아이노드는 해당 파일과 동일한 블록 그룹에 위치하며 그룹 내에는 여러 개의 아이노드가 할당 될 수 있다. log-ordered mode 저널링은 수정된 후의 아이노드를 저장할 공간으로 그룹 내의 아이노드를 위한 공간을 사용한다. 이렇게 사용되어진 공간은 link_metadata 구조체의 link_inode 인자에 연결된다.

4.4 log-ordered mode 의 데이터 블록 비트맵과 아이노드 비트맵의 저널링

Ext3 파일 시스템의 디스크 공간에서 데이터 블록 비트맵과 아이노드 비트맵은 각각 1블록이라는 정해진 공간을 할당받는다.

log-ordered mode 저널링은 그룹 내의 데이터 블록을 데이터 블록 비트맵 과 아이노드 비트맵의 변경된 값을 저장하기 위한 공간으로 사용 한다. 변경된 데이터 블록 비트맵과 아이노드 비트맵의 값을 기록한 후 link_metadata 구조체의 link_datablock_bitmap와 link_ inode_bitmap에게 할당한다.

4.5 클리너

log-ordered mode 저널링은 log-structured 파일 시스템에서 사용되는 클리너와 동일한 개념의 클리너를 backup_inode, backup_datablock_bitmap, backup_inode_bitmap 에 할당되어진 공간을 초기화하기 위해 사용한다. 이 부분들에 기록되어진 데이터들은 수정된 후의 메타 데이터를 디스크에 쓰는 도중에 발생할 수 있는 파일 시스템의 예기치 못한 리부팅 등으로 인한 메타데이터의 손실을 대비하기 위한 것이므로 모든 메타데이터의 수정이 성공적으로 끝난 상황에선 더 이상 보존해야할 필요성이 없어진다.

5. 결론 및 향후 과제

본 논문에서 제시한 log-ordered mode 는 Ext3 파일 시스템의 ordered mode 저널링을 변형, 보완한 저널링 기법으로써 변경된 메타데이터 값을 변경된 데이터 블록보다 먼저 디스크에 기록함으로써 예기치 못한 파일 시스템의 리부팅으로 인해 파일 시스템의 일관성이 무너지는 것을 방지하고, 사용자에게 손상된 정보를 포함하는 파일을 보여주는 일이 없도록 한다.

또한 log-ordered mode 저널링은 ordered mode 의 저널링이 수행되기 위해 필요한 여분의 디스크 공간과 추가적인 작업의 필요 없이 ordered mode와 동일한 저널링을 수행한다

현재 NAS와 같은 대형 파일 시스템에서의 저널링을 수행하기 위해 구현 중에 있다.

6. 참고 문헌

[1] DANIELP.BOVET 와 MARCO CESATI, "Understanding the Linux Kernel"

[2] Linux kernel source code http://www.kernel.org/

[3] Stephen C.Tweedie, "Journaling the Linux ext2fs Filesystem" : LinuxExpo'98 ,1998

[4] Randy Appleton, " A Non-Technical Look inside the EXT2 File System" : Linux Journal, 1997

[5] Theodore Y.Ts'o 와 Stephen Tweedie, " Pl anned Extensions to the Linux Ext2/Ext3 Filesytem" : Frenenix Track: 2002 USENIX Annual Technical conference , 2002

[6] RedHat,"Red Hat's New Journaling File System : ext3" , 2001

[7] Daniel Robbins, "고급 파일 시스템 개발자 가이드, part 7 & 8 "
http://www-128.ibm.com/developerworks/kr/library/l-fs7.html
http://www-128.ibm.com/developerworks/kr/library/l-fs8.html

[8] Juan Piernas ,Toni Cortes & Jose M.Garcia , "DualFS: a New Journaling File System without Meta -Data Duplication" : ICS'02 , 2002

[9] Adam Sweeney, Doug Doucette, Wei Hu, Curtis Anderson, Mike Nishimoto & Geoff Peck, "Scalability in the XFS File System" : USENIX 1996 Annual Technical Conference, 1996

[10] Margo Seltzer, Keith Bostic, Marshall Kirk Mc Kusick & Carl Staelin, "An Implementation of a Log-Structured File System for UNIX" : USENIX. 1993

[11] T.J.Kowalski, "Fsck-The UNIX File System Check Program", 1996

[12] The Design and Implementation of the 4.4BSD Operating system

[13] 강기봉 , "시스템 관리자를 위한 리눅스 바이블 : 북스피아