EXT3 파일 시스템 안정성 지원을 위한 저널링 매카니즘 개발

석진선[O], 노재춘[O], 박성순[1]
세종대학교[O], 글루시스[1]
durgatm @naver.com[O], jano@sejong.ac.kr[O], sspark@gluesys.com[1]

The Development of Journaling Mechanism for supporting Ext3 File System Reliability

Jinsun Suk[O], Jaechun No[O], Sung Soon Park[1]
Sejong university[O], Gluesys.co.,Ltd[1]

요 약

파일 시스템의 안정성에 문제가 생긴 경우, 파일 시스템의 동작이 중단되어 수정 중이던 데이터가 손실되거나 기존 데이터의 복구가 불가능하게 되는 상황이 발생 할 수 있다. 데이터의 종류에는 파일의 데이터와 같이 데이터 자체가 중요한 내용을 담고 있는 것과 파일의 데이터를 관리하기 위한 정보를 담고 있는 데이터가 있는데 후자를 메타데이터라고 한다. 단순히 파일의 데이터가 손실된 경우에 약간의 데이터 손실이 발생할 수는 있지만, 파일 시스템은 정상적으로 동작 할 수 있다. 하지만 메타데이터가 손상된 경우에는 파일 시스템이 볼륨에 접근조차 할 수 없게 되어 볼륨 내의 모든 파일을 접근할 수 없게 된다. 이러한 문제점들을 극복하기 위해 DualFS [8], log-structured 파일 시스템 [10], XFS [9] 등의 다양한 저널링 파일 시스템들이 제안되었다. 그 중 Ext3 파일 시스템은 가장 안정적이고 치명적인 문제점이 없는 것으로 알려져 있다.[7] 하지만 Ext3 파일 시스템에서 기본적으로 사용되고 있는 ordered mode 저널링은 메타데이터의 복사가 이루어져야 하기 때문에 속력의 저하가 발생한다. 본 논문에서는 ordered mode의 메타데이터의 복사 작업이 필요 없는 개선된 ordered mode 저널링을 제안한다.

1. 서 론

오늘날 컴퓨터는 사회 전체에서 사용되고 있으며 파일 시스템이 관리하는 데이터의 양도 증가했다. 보존, 관리해야 하는 데이터의 양이 증가하면서 파일 시스템의 안정성도 더욱 중요시되었는데 안정성에 문제가 생긴 경우 보존해야 하는 데이터가 손실되거나 볼륨에 접근 할 수 없는 상황이 발생할 수 있기 때문이다.

이러한 문제점들을 해결, 방지하기 위해 DualFS [8], log-structured 파일 시스템 [10], XFS [9] 등의 다양한 저널링 파일 시스템이 사용되고 있으며 이 중 Ext3 파일 시스템이 가장 널리 사용되고 있다. [7]

Ext3 파일 시스템은 세 가지 모드의 저널링을 수행한다. 세 가지 저널링 모드 중 가장 효율적인ordered mode는 Ext3 파일시스템의 기본 저널링 모드로 사용되고 있으며 3단계로 수행된다. 디스크의 메타데이터를 수정하기 전, 저널링 공간에 수정하고자 하는 메타데이터를 기록하고 디스크에 데이터를 쓴 다음 저널에 있는 메타데이터에 대한 정보를 디스크에 옮긴다. 디스크에 메타데이터보다 데이터를 먼저 써서 데이터를 저널링하지 않고도 데이터의 손상을 막을 수 있다는 것이 ordered mode 저널링의 가장 큰 장점이라고 할 수 있다. 하지만 ordered mode 저널링은 한번의 쓰기 동작을 수행하기 위해서 실제로는 두 번의 쓰기 동작이 필요하다. 이러한 Ext3 파일 시스템의 ordered mode 저널링 방법은 효과적이긴 하지만 DualFS[8]와 log-structure 파일 시스템[10]의 추가적인 쓰기 작업 없이 수행된다는 것에 비하면 비효율적이다.

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

2. 관련 연구

DualFS [8], Log-structured 파일 시스템 [10], XFS [9], Ext3 파일 시스템 [1][2][5][6] 등의 파일 시스템들은 시스템의 예기치 못한 재난으로 인해 발생하는 메타데이터들의 손상을 복구하기 위한 다양한 방법을 제시한다. 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 변경되어진 메타데이터를 디스크에 쓰기 위한 작업과 로그를 저널링 공간에 기록하기 위한 추가적인 작업이 필요하다. 반면에 Log-structured 파일 시스템과 DualFS는 추가적인 쓰기 작업 없이 저널링을 수행한다.

Log-structured 파일 시스템은 세그먼트(segment) 단위 별

로 새로운 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. 예를 들어 A라는 파일을 수정한다면, 파일 A의 메타데이터를 담고 있는 세그먼트 a를 수정하는 것이 아니라 수정된 파일 A의 새로운 메타데이터를 담는 세그먼트 a'를 새로 할당, 파일을 참조하도록 수정하고, a'를 새로 할당하는 과정에서 파일 시스템에 문제가 생겨 a'가 손실되는 경우에는 a를 파일 A의 메타데이터로 다시 사용하여 파일 시스템의 오염을 막는다. 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 복사 본을 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.

XFS는 'write ahead transaction log'를 사용하여 저널링을 수행함 으로써 파일 시스템의 atomic한 변경을 가능하게 하였다. 그리고 전통적인 유닉스 파일 시스템들이 fsck와 같은 'scavenger progrm'들을 위해 고정적이고 동기적인 데이터의 변경방식을 사용했던 것과는 다르게 비동기적인 데이터 처리 방식을 선택하여 성능을 향상시켰다.[9]

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 디렉토리 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]

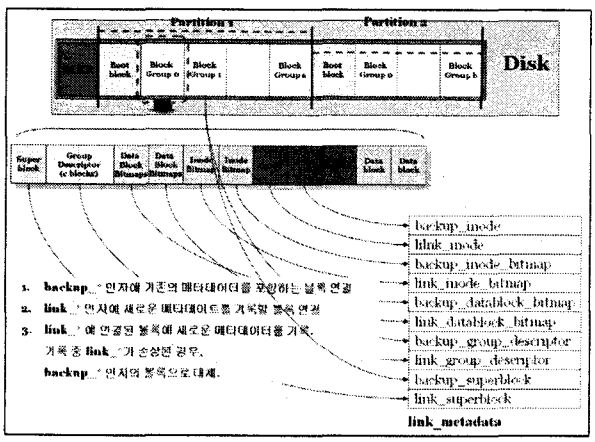
3.1 Ext3의 ordered mode 저널링

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 방식으로 사용자가 복구된 파일을 읽을 때 불필요한 정보가 포함되는 기존의 저널링 방식의 문제점을 데이터를 따로 저널링 하지 않고도 어느 정도 예방할 수 있는 유용한 방법이다.[7] 또한 XFS [8], 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. [7]

하지만 ordered mode로 저널링을 수행하기 위해서는 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 된다. [8]

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

4. log-ordered mode 의 구조



[그림1] log-ordered mode 저널링의 구조

Log-ordered mode는 변경되기 전의 메타데이터를 포함하고 있는 블록을 백업 데이터로 사용함으로써 저널을 위한 추가적인 작업을 수행하지 않는 방법이다. 한 번의 쓰기 작업으로 메타데이터의 손상에 대비하면서 사용자의 쓰기 작업 요청을 처리하는 것이다. Log-ordered mode 저널링은 [그림1]과 같은 전체 구조를 가지며 그림의 link_metadata 구조체와 같이 변경 전과 후의 메타데이터를 모두 가지는 구조체

를 필요로 한다. link_metadata 구조체는 파일이 변경되었을 경우에 영향을 받는 메타데이터인 파일 아이노드, 데이터 블록 비트맵, 아이노드 블록 비트맵, 그룹 디스크립터, 슈퍼 블록들의 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 수정되기 전의 메타데이터를 가지고 있는 블록을 link_metadata 구조체의 인자인 backup_inode, backup_inode_bitmap, backup_dtatablock_bitmap, backup_superblock들에 연결한다. 두 번째 단계로 수정되어야 하는 메타데이터를 포함하고 있는 블록을 구조체 link_metadatad의 각 인자인 link_inode, link_inode_bitmap, link_dtatabock_bitmap, link_superblock들에 연결하고 마지막으로 link_* 인자에 할당된 블록들을 수정한다. 만약 마지막 단계인 메타데이터를 수정하는 단계를 성공적으로 마무리 하지 못했다면 link_metadata의 구조체의 backup_* 인자들을 사용하여 link_* 에 연결된 블록들을 복구한다.

다음 섹션에서는 log-ordered mode 가 메타데이터 의 종류별로 어떻게 저널링을 수행하는지에 대해 자세하게 언급하겠다.

4.1 log-ordered mode 의 슈퍼 블록 저널링

슈퍼 블록은 파티션의 정보를 담는 블록으로 파티션내의 모든 블록그룹이 동일한 정보를 가진다.[1][12][13] Log-ordered mode 는 이런 Ext3 파일 시스템의 특성을 이용하여 슈퍼 블록을 따로 복사해두지 않아도 시스템을 복구 할 수 있도록 한다.

먼저 인접한 블록그룹의 슈퍼블록을 backup_superblock에 연결한다. 파티션내의 모든 슈퍼블록의 값은 동일해야 하므로 backup_superblock에 연결된 이 부분으로 슈퍼블록을 수정하는 중에 발생한 손실을 복구할 수 있다. 다음으로 수정하고자 했던 슈퍼블록을 link_superblock에 연결하고 값을 수정한다. 만약 수퍼블록의 수정 중에 데이터 손실이 발생하면 backup_superblock에 연결된 인접 블록그룹의 슈퍼블록의 값으로 link_supberlock에 연결된 블록을 대체함으로써 파일 시스템을 복구한다.

4.2 log-ordered mode의 그룹 디스크립터 저널링

그룹 디스크립터는 파티션내의 모든 그룹들에 대한 정보를 담는 블록으로 동일한 파티션에 포함되는 모든 블록그룹들은 동일한 그룹 디스크립터를 가진다.[1][12][13] log-ordered mode 저널링은 이러한 Ext3 파일 시스템의 그룹 디스크립터의 특성을 이용하여 슈퍼블록을 저널링 하는 방법과 유사하게 복구한다.

먼저 동일한 파티션내의 인접한 그룹디스크립터를 backup_group_descriptor에 연결한다. 다음으로 고치고자 했던 그룹 디스크립터를 link_group_descriptor에 연결하고 값을 수정한다. 만약 그룹 디스크립터의 수정 중에 데이터의 손실이 발생한다면 backup_group_descriptor에 연결된 인접 블록그룹의 그룹디스크립터의 값으로 link_metadata 부분을 대체함으로써 파일 시스템을 복구한다.

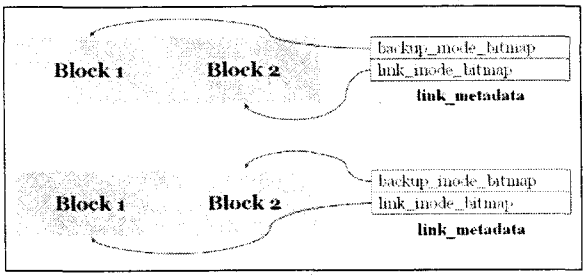
4.3 log-ordered mode의 아이노드 (inode) 저널링

아이노드는 파일에 대한 정보를 담는 것으로 파일마다 하나의 아이노드를 가진다. 아이노드는 해당 파일과 동일한 블록그룹에 위치하며 그룹 내에는 여러 개의 아이노드가 할당 될 수 있다.

log-ordered mode 저널링은 수정되기 전의 아이노드를 backup_inode에 할당하고, 수정된 후의 아이노드를 저장할 공간으로 새로운 아이노드를 할당한다. 새로 할당한 아이노드는 link_inode에 할당하고 수정한다. link_inode의 블록들 수정하던 중에 발생한 데이터의 손실을 복구하기 위해서 backup_inode를 사용한다.

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. 하지만 log-ordered mode 저널링은 데이터 블록 비트맵과 아이노드 비트맵은 각각 2블록씩 할당하여 1블록은 또 다른 블록을 위한 backup으로 사용하여, 두 블록을 backup_inode_bitmap과 link_inode_bitmap으로 교차적으로 사용한다.



[그림2] 블록 비트맵과 아이노드 비트맵의 저널링

예를 들어 데이터 블록 비트맵을 수정한다면 block2를 backup_data_bitmap에 연결하고 block1를 link_data_bitmap에 연결한 후 block1를 수정한다. block1의 수정이 성공적으로 이루어지지 못했다면 backup_data_bitmap에 연결된 block2를 참고하여 block1를 복구한다. 다음번에 다시 데이터 블록 비트맵을 수정하고자 하는 요청이 들어오면 block1를 backup_data_bimap 에 연결하고 block2를 link_data_bitmap에 연결하여 사용한다. 아이노드 블록 비트맵도 데이터 블록비트맵과 동일한 방법으로 수정, 복구한다.

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.

ordered mode 저널링을 개선시킨 log-ordered mode 저널링은 메타데이터의 복사 없이 ordered mode와 동일한 저널링을 수행하기 때문에 Ext3 파일시스템의 안정성을 그대로 유지하면서 처리 속력을 향상시킬 것으로 예상된다.
현재 NAS와 같은 대형 파일 시스템에서의 저널링을 수행하기 위해 구현 중에 있다.

6. 참고 문헌

[1] DANIELP.BOVET 와 MARCO CESATI, "Underst anding 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, "Scala bility in the XFS File System" : USENIX 1996 Ann ual 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