(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년06월09일
(11) 등록번호 10-1744685
(24) 등록일자 2017년06월01일

(51) 국제특허분류(Int. Cl.)
G06F 21/62 (2013.01) *G06F 17/30* (2006.01)
(52) CPC특허분류
G06F 21/6227 (2013.01)
G06F 17/3012 (2013.01)
(21) 출원번호 10-2015-0191670
(22) 출원일자 2015년12월31일
심사청구일자 2015년12월31일
(56) 선행기술조사문헌
KR1020080058834 A*
KR100906454 B1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
한양대학교 산학협력단
서울특별시 성동구 왕십리로 222(행당동, 한양대학교내)
울산과학기술원
울산광역시 울주군 언양읍 유니스트길 50
(72) 발명자
원유집
경기도 성남시 분당구 백현로 227, 509동 805호 (수내동, 푸른마을쌍용아파트)
남범석
울산광역시 울주군 범서읍 구영로 75-25, 107동 405호 (울산구영동문굿모닝힐)
(74) 대리인
심경식, 홍성욱, 특허법인 무한

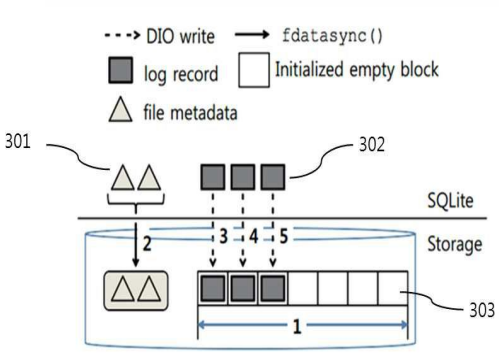
전체 청구항 수 : 총 10 항 심사관 : 구대성

(54) 발명의 명칭 **파일의 메타데이터에 대한 보호 방법 및 보호 장치**

(57) 요 약

파일의 메타데이터를 위한 보호 방법 및 보호 장치가 개시된다. 보호 방법은 저널 파일을 위한 일정량의 초기화된 블록들을 저장 장치에 미리 할당(preallocation)하는 단계; 및 데이터 동기화 함수를 호출함으로써, 상기 저널 파일에 대한 업데이트된 메타데이터를 저장 장치에 저널링하는 단계를 포함할 수 있다.

대 표 도 - 도3



(52) CPC특허분류
G06F 17/30144 (2013.01)
G06F 17/30174 (2013.01)
G06F 17/30368 (2013.01)
G06F 21/6209 (2013.01)

공지예외적용 : 있음

명 세 서

청구범위

청구항 1

저널 파일을 위한 일정량의 초기화된 블록들을 저장 장치에 미리 할당(preallocation)하는 단계;

데이터 동기화 함수를 호출함으로써, 상기 저널 파일에 대한 업데이트된 메타데이터를 저장 장치에 저널링하는 단계; 및

상기 저널링하는 단계 이후에, 데이터베이스의 복구 정보인 로그를 상기 블록에 커밋하는 단계

를 포함하고,

상기 커밋하는 단계는,

상기 로그를 데이터베이스로부터 저장 장치에 미리 할당된 블록에 직접 입출력 (DIO) 방식 또는 버퍼링된 입출력(Buffered IO) 방식으로 기록하는 보호 방법.

청구항 2

삭제

청구항 3

삭제

청구항 4

제1항에 있어서,

상기 저널링하는 단계는,

상기 저널 파일에 대한 메타데이터를 데이터 동기화 함수를 통해 동기화함으로써 저장 장치에 저장하는 보호 방법.

청구항 5

제1항에 있어서,

상기 블록은,

상기 저장 장치에 할당된 블록을 제로로 채우거나, 상기 저장 장치에 폐기 명령(discard command)을 적용함으로써 초기화되는 보호 방법.

청구항 6

제1항에 있어서,

로그를 커밋함에 따라 미리 할당된 블록 전부가 차면(full), 미리 설정된 양의 블록을 재할당하는 단계

를 더 포함하는 보호 방법.

청구항 7

제6항에 있어서,

상기 블록이 재할당됨에 따라 저널 파일의 파일 크기가 변경되는 경우, 저널 파일에 대한 메타데이터를 동기화함으로써 저널링하는 단계

를 더 포함하는 보호 방법.

청구항 8

파일의 메타데이터를 위한 보호 방법을 수행하는 보호 장치는,

프로세서를 포함하고,

상기 프로세서는,

저널 파일을 위한 일정량의 초기화된 블록들을 저장 장치에 미리 할당(preallocation)하는 단계;

데이터 동기화 함수를 호출함으로써, 상기 저널 파일에 대한 업데이트된 메타데이터를 저장 장치에 저널링하는 단계; 및

상기 저널링하는 단계 이후에, 데이터베이스의 복구 정보인 로그를 상기 블록에 커밋하는 단계

를 수행하고,

상기 프로세서는,

상기 로그를 데이터베이스로부터 저장 장치에 미리 할당된 블록에 직접 입출력 (DIO) 방식 또는 버퍼링된 입출력(Buffered IO) 방식으로 기록하는 보호 장치.

청구항 9

삭제

청구항 10

삭제

청구항 11

제8항에 있어서,

상기 프로세서는,

상기 저널 파일에 대한 메타데이터를 데이터 동기화 함수를 통해 동기화함으로써 저장 장치에 저장하는 저널링을 수행하는 보호 장치.

청구항 12

제8항에 있어서,

상기 블록은,

상기 저장 장치에 할당된 블록을 제로로 채우거나, 상기 저장 장치에 폐기 명령(discard command)을 적용함으로써 초기화되는 보호 장치.

청구항 13

제8항에 있어서,

상기 프로세서는,

로그를 커밋함에 따라 미리 할당된 블록 전부가 차면(full), 미리 설정된 양의 블록을 재할당하는 단계

를 더 수행하는 보호 장치.

청구항 14

제13항에 있어서,

상기 프로세서는,

상기 블록이 재할당됨에 따라 저널 파일의 파일 크기가 변경되는 경우, 저널 파일에 대한 메타데이터를 동기화함으로써 저널링하는 단계

를 더 수행하는 보호 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 데이터를 많이 저장하는 분야에 사용될 수 있으며, 특히, 버퍼기반 입출력이나 직접 입출력 또는 버퍼를 이용한 입출력으로 데이터베이스 복구정보를 기록하는 방법을 사용하는 분야에서는 유용하게 사용될 수 있다.

배 경 기 술

[0002] 종래의 경우, 사용자 버퍼에 저장된 데이터를 페이지 캐시(page cache)에 임시적으로 저장하고, fsync() 또는 fdatasync()라는 함수를 통해 페이지 캐시에 저장된 데이터를 저장 장치에 기록하였다. 또한, 직접 입출력(DIO)를 통해 데이터를 페이지 캐시를 거치지 않고 바로 저장 장치에 기록하였다.

[0003] fsync()에서, 파일 시스템은 각각의 파일에 대해 업데이트된 메타데이터를 저널링할 수 있다. fdatasync()에서, 파일 블록들이 할당되거나 또는 할당되지 않을 때, 파일 시스템은 업데이트된 메타데이터를 저널링할 수 있다. DIO write는 그 자체로 어떠한 파일 시스템 저널링을 수반하지 않는다.

[0004] 기록 동작(write operation)은 2가지 타입들(write를 할당하는 과정, write를 할당하지 않는 과정)로 분류될 수 있다. Write를 할당하는 과정은 블록 비트맵, 아이노드 테이블, 중간 노드 블록 등과 같은 다양한 메타데이터를 업데이트할 수 있다. Write를 할당하지 않는 과정은, 파일 블록들을 할당하는 것을 수반하지 않는다. 그것은 필드와 관련된 액세스 시간과 메타데이터에서 초기화된 플래그를 업데이트할 뿐이다.

[0005] write를 할당하는 것과 write를 할당하지 않는 경우 모두에서, fsync()는 업데이트된 메타데이터를 저널링할 수 있다. Write를 할당하는 경우, fdatasync()는 fsync()와 동일한 행동을 나타낸다. 그리고, write를 할당하지 않는 경우, fdatasync()는 어떠한 메타데이터를 저널링하지 않는다. write를 할당하는 것과 write를 할당하지 않는 경우 모두에서, DIO(Direct IO)는 파일 시스템의 저널링을 수반하지 않는다. DIO에서, 업데이트된 메타데이터는 손실되기 쉽다.

[0006] 종래의 경우, DIO를 사용하지 않으면 메타데이터를 페이지 캐시에 저장하고, 메타데이터를 fsync()나 fdatasync()를 통해 저장 장치에 동기화하는 파일 시스템 저널링 과정이 발생하는데, 이와 같은 불필요한 파일 시스템 저널링으로 인해서 데이터베이스에 대한 입출력 양이 증가하고, 저장 장치의 수명이 단축되는 문제가 있었다.

발명의 내용

해결하려는 과제

[0007] 본 발명은 파일에 기록한 내용을 저장 장치에 동기화할 때 실질적으로 발생하는 입출력의 양을 대폭 감소시킴으로써 저장 장치의 성능을 향상시키고 수명이 연장될 수 있는 보호 방법 및 보호 장치를 제공한다.

과제의 해결 수단

[0008] 본 발명의 일실시예에 따른 보호 방법은 저널 파일을 위한 일정량의 초기화된 블록들을 저장 장치에 미리 할당(preallocation)하는 단계; 및 데이터 동기화 함수를 호출함으로써, 상기 저널 파일에 대한 업데이트된 메타데이터를 저장 장치에 저널링하는 단계를 포함할 수 있다.

[0009] 상기 보호 방법은 상기 저널링하는 단계 이후에, 데이터베이스의 복구 정보인 로그를 상기 블록에 커밋하는 단계를 더 포함할 수 있다.

[0010] 상기 커밋하는 단계는, 상기 로그를 데이터베이스로부터 저장 장치에 미리 할당된 블록에 직접 입출력 (DIO) 방식 또는 버퍼링된 입출력(Buffered IO) 방식으로 기록할 수 있다.

[0011] 상기 저널링하는 단계는, 상기 저널 파일에 대한 메타데이터를 데이터 동기화 함수를 통해 동기화함으로써 저장 장치에 저장할 수 있다.

[0012] 상기 블록은, 상기 저장 장치에 할당된 블록을 제로로 채우거나, 상기 저장 장치에 폐기 명령(discard comman

d)을 적용함으로써 초기화될 수 있다.

[0013] 상기 보호 방법은 상기 로그를 커밋함에 따라 미리 할당된 블록 전부가 차면(full), 미리 설정된 양의 블록을 재할당하는 단계를 더 포함할 수 있다.

[0014] 상기 보호 방법은 상기 블록이 재할당됨에 따라 저널 파일의 파일 크기가 변경되는 경우, 저널 파일에 대한 메타데이터를 동기화함으로써 저널링하는 단계를 더 포함할 수 있다.

[0015] 본 발명의 일실시예에 따른 보호 장치는 프로세서를 포함하고, 상기 프로세서는, 저널 파일을 위한 일정량의 초기화된 블록들을 저장 장치에 미리 할당(preallocation)하는 단계; 및 데이터 동기화 함수를 호출함으로써, 상기 저널 파일에 대한 업데이트된 메타데이터를 저장 장치에 저널링하는 단계를 수행할 수 있다.

[0016] 상기 프로세서는, 상기 저널링하는 단계 이후에, 데이터베이스의 복구 정보인 로그를 상기 블록에 커밋하는 단계를 더 수행할 수 있다.

[0017] 상기 프로세서는, 상기 로그를 데이터베이스로부터 저장 장치에 미리 할당된 블록에 직접 입출력 (DIO) 방식 또는 버퍼링된 입출력(Buffered IO) 방식으로 기록할 수 있다.

[0018] 상기 프로세서는, 상기 저널 파일에 대한 메타데이터를 데이터 동기화 함수를 통해 동기화함으로써 저장 장치에 저장하는 저널링을 수행할 수 있다.

[0019] 상기 블록은, 상기 저장 장치에 할당된 블록을 제로로 채우거나, 상기 저장 장치에 폐기 명령(discard command)을 적용함으로써 초기화될 수 있다.

[0020] 상기 프로세서는, 상기 로그를 커밋함에 따라 미리 할당된 블록 전부가 차면(full), 미리 설정된 양의 블록을 재할당하는 단계를 더 수행할 수 있다.

[0021] 상기 프로세서는, 상기 블록이 재할당됨에 따라 저널 파일의 파일 크기가 변경되는 경우, 저널 파일에 대한 메타데이터를 동기화함으로써 저널링하는 단계를 더 수행할 수 있다.

발명의 효과

[0022] 본 발명의 일실시예에 따르면, 파일에 기록한 내용을 저장 장치에 동기화할 때 실질적으로 발생하는 입출력의 양을 대폭 감소시킴으로써 저장 장치의 성능을 향상시키고 수명이 연장될 수 있다.

도면의 간단한 설명

[0023] 도 1은 본 발명의 일실시예에 따른 전체 구성을 도시한 도면이다.

도 2는 본 발명의 일실시예에 따른 파일의 메타데이터에 대한 보호 방법을 도시한 플로우차트이다.

도 3은 본 발명의 일실시예에 따른 파일의 메타데이터에 대한 보호 방법에 대한 예시를 도시한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0024] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0025] 도 1은 본 발명의 일실시예에 따른 전체 구성을 도시한 도면이다.

[0026] 본 발명의 일실시예에 따르면, 데이터베이스(101), 저장 장치(102), 및 보호 장치(103)로 구성될 수 있다. 보호 장치(103)는 본 발명의 일실시예에 따른 파일의 메타데이터를 위한 보호 방법을 수행하며, 특정 단말에 포함된 파일 시스템을 지원하는 프로세서가 될 수 있다.

[0027] 본 발명의 일실시예에 따르면, 보호 장치(103)는 파일을 위해 미리 설정된 정도의 초기화된 블록들을 미리 할당(preallocation)할 수 있다. 그리고, 보호 장치(103)는 fdatasync()를 호출함으로써, 생성된 파일에 대한 메타데이터를 저널링할 수 있다.

[0028] 구체적으로, 보호 장치(103)는 저널 파일을 위한 초기화된 블록들은 저장 장치(102)에의 미리 할당될 수 있다. 그리고, 보호 장치(103)는 저널 파일의 메타데이터를 fdatasync()를 통해 저장 장치(102)에 동기화함으로써, 저널 파일의 메타데이터를 저널링할 수 있다. 이 후, 로그들은 직접 입출력(DIO: Direct IO) 또는 버퍼링된 입출력(Buffered IO)를 통해 저널 파일에 미리 할당된 블록에 커밋될 수 있다. 저널 파일의 메타데이터는, 저널 파일의 사이즈와 같은 속성을 의미한다.

[0029] 여기서, 버퍼링된 입출력은 로그들이 페이지 캐쉬에 write()를 통해 임시적으로 기록된 후, fdatasync()나 fsync()와 같은 함수를 통해 저장 장치(102)에 기록하는 방식을 의미한다. 그리고, 직접 입출력(Direct IO)은 사용자가 데이터를 저장 장치(102)(storage device)로부터 데이터를 읽거나 또는 저장 장치(102)로 데이터를 기록하는 파일 시스템의 기능 중 하나이다. 직접 입출력에 의하면, 데이터는 페이지 캐시에 기록되지 않고 저장 장치(102)에 바로 기록될 수 있다. 즉, 데이터는 페이지 캐시를 바이패스하여 페이지 캐시에 임시적으로 저장되지 않고, 바로 저장 장치(102)에 기록될 수 있다.

[0030] 본 발명은 데이터베이스에서 지원하는 저널 모드(DELETE, TRUNCATE, PERSIST, WAL, MEMORY 및 OFF 등)에 적용될 수 있다. 데이터베이스에 대해 예상하지 못한 시스템 실패(failure)를 보호하기 위한 방법이 요구된다. 이 때, 크래시로부터 회복하기 위해 데이터베이스는 로그를 유지하는 저널 파일을 사용한다. 트랜잭션의 보장을 위해, 데이터베이스는 로그를 저널 파일에 커밋(commit)하거나 또는 데이터베이스(101)를 업데이트한 후, fdatasync()를 이용하여 로그 파일과 데이터베이스 파일을 동기화한다. 커밋은 트랜잭션의 수행 결과가 데이터베이스(101)에 반영되어 영속적으로 남아 있는 것을 의미한다.

[0031] 본 발명에서 언급하는 로그를 커밋하는 것은, 데이터베이스(101)에서 데이터의 변경 사항을 나타내는 데이터베이스(101)의 복구 정보인 로그를 저널 파일을 위해 미리 할당된 블록에 저장하는 과정을 의미한다.

[0032] 본 발명의 일실시예에 따른 보호 장치(103)는 초기화된 블록을 미리 할당(preallocating) 과정과 저널링 과정을 수행함으로써, 예상되지 않은 시스템 크래시에 의해 파일에 대한 메타데이터를 적절히 보호하기 위해 파일 시스템 저널링을 제거할 수 있다. 여기서, 저널링은 저장 장치(102)에 데이터의 변경 이력을 저장하기 전에 저널이라는 로그 안에 데이터의 변경 이력을 저장하는 기법이다. 저널링은 데이터의 변경 이력을 저장하는 도중에 시스템 장애가 발생하는 경우 데이터가 비정상적으로 손상되는 것을 회복하기 위해 사용된다.

[0033] 도 2는 본 발명의 일실시예에 따른 파일의 메타데이터에 대한 보호 방법을 도시한 플로우차트이다.

[0034] 단계(201)에서, 보호 장치는 저널 파일을 위한 초기화된 블록들을 미리 할당할 수 있다. 이 때, 미리 할당한다는 의미는, 데이터베이스의 복구 정보인 로그를 기록하기 위한 저널 파일을 생성할 때 또는 생성하기 전에 블록을 미리 할당한다는 것을 의미한다. 초기화된 블록들은 저장 장치에 미리 할당될 수 있다.

[0035] 단계(202)에서, 보호 장치는 저널 파일에 대해 업데이트된 메타데이터를 저널링할 수 있다. 이 때, 보호 장치는 fdatasync()와 같은 데이터 동기화 함수를 통해 데이터베이스에 존재하는 저널 파일의 메타데이터를 저장 장치에 동기화함으로써, 메타데이터를 저널링할 수 있다.

[0036] 단계(203)에서, 보호 장치는 직접 입출력(DIO) 또는 버퍼링된 입출력(Buffered IO)를 이용하여 데이터베이스의 복구 정보인 로그를 데이터베이스에서 저장 장치로 커밋할 수 있다. 그러면, 로그는 저널 파일을 위해 미리 할당된 블록에 저장될 수 있다.

[0037] 단계(203)을 통해 로그가 커밋됨으로써 미리 할당된 블록 전부에 로그가 기록되면, 보호 장치는 다시 미리 설정된 크기의 새로운 블록들을 할당할 수 있다. 이를 통해 저널 파일의 파일 크기가 변경되면, 저널 파일의 메타데이터가 변경될 수 있다. 그러면, 보호 장치는 단계(202)와 같이 변경된 메타데이터를 저널링을 통해 저장 장치에 동기화할 수 있다.

[0038] 도 3은 본 발명의 일실시예에 따른 파일의 메타데이터에 대한 보호 방법에 대한 예시를 도시한 도면이다.

[0039] 본 발명의 일실시예에 따르면, 로그(302)들을 저널 파일에 커밋하기 위해 DIO(Direct IO) 기반의 쓰기 연산 또는 Buffered IO 기반의 쓰기 연산을 제안한다. 본 발명의 일실시예에 따르면, 저장 장치에 로그(302)들을 DIO에 따라 직접적으로 기록하거나 또는 Buffered IO에 따라 로그를 페이지 캐시에 임시로 저장한 후에 저장 장치에 기록할 수 있다. 로그(302)를 커밋하는 행위가 데이터 블록(303)이나 메타데이터(301)에 대해 어떠한 업데이트들이 페이지 캐시 엔트리들에서 수반되지 않도록 한다. 이러한 접근을 위해, fdatasync()와 같은 데이터베이스의 동기화 과정은 IO와 관련된 어떠한 파일 시스템 저널링을 트리거링하지 않는다.

[0040] 본 발명은 로그(302)를 커밋하는 동작에서 저널링의 간섭을 제거하고, 저널 파일의 메타데이터(301)를 보호할 수 있다. 본 발명과 같이, 명백한 저널링을 수반하는 미리 할당 과정(preallocation with explicitly journaling)은 (i) 저널 파일을 위해 일정 량의 초기화된 특정 블록(303)들을 미리 할당하고, (ii) fdatasync()를 호출함으로써, 생성된 저널 파일을 위한 메타데이터(301)를 저널링할 수 있다. 종래의 경우, 데이터베이스 저널 파일의 메타데이터(301)를 보호하기 위해, 파일 시스템 저널링에 의존할 필요가 있다. 그러나, 이는 파일시스템 저널링을 수반하기 위한 모든 로그(302) 커밋 동작을 억압할 수 있는 문제가 있으나,

본 발명은 이러한 파일 시스템 저널링을 제거함으로써 이러한 억압에서 해소될 수 있다.

[0041] 도 3에 의하면, (i) 저널 파일을 위한 초기화된 블록(303)들이 미리 할당될 수 있다. 이러한 블록들은 저장 장치에 미리 할당된다. (ii) 저널 파일의 메타데이터(301)는 fdatasync()를 통해 동기화됨으로써 저널링된다. 즉, 저널 파일의 메타데이터(301)는 저장 장치에 저장된다. 그리고, (iii) 로그(302)들은 DIO를 통해 저널 파일에 커밋된다. 즉, 데이터베이스의 복구 정보인 로그(302)들은 DIO를 통해 페이지 캐시를 거치지 않고 바로 저장 장치에 미리 할당된 블록에 기록될 수 있다.

[0042] 파일 시스템은 각각의 블록(303)에 대해 초기화된 플래그를 유지할 수 있다. 플래그가 설정될 때, 블록(303)은 초기화될 수 있다. 초기화되지 않은 블록(303)을 읽기 위한 어떠한 시도들은 모드 0을 리턴할 수 있다. 이러한 매커니즘에 대한 가장 우선적인 이유는 오래된 데이터(stale data)를 노출하는 것을 피하기 위함이다.

[0043] 저널 파일을 위한 블록(303)들을 미리 할당할 때, 로그(302)의 커밋 동작 기반의 Direct IO는 write를 할당하지 않을 수 있다. 그러면, 페이지 캐쉬 엔트리들과 저널 파일의 메타데이터(301)는 온전하게 유지된다. 본 발명의 일실시예에 따른 저널 모드에서 로그(302)의 커밋 동작은 간섭을 일으키는 파일 시스템 저널링 모듈을 위한 어떠한 공간도 남기지 않는다. 본 발명의 일실시예에 따른 저널 모드는 파일 시스템 저널링으로부터 데이터베이스를 보호한다. 저널 파일이 생성되거나 또는 확장될 때, 본 발명의 일실시예에 따른 저널 모드는 파일에 대한 메타데이터(301)를 동기화하기 위해 fdatasync()를 호출할 수 있다. 명백한 저널링이 수반되는 경우, 저널 파일은 시스템 오류에 강건할 수 있다.

[0044] 블록들을 미리 할당할 때, 할당된 블록(303)들을 초기화하기 위해 특별한 보호가 요구된다. 이와는 달리, 예상되지 않은 시스템 실패 이후에, 본 발명의 일실시예에 따른 저널 모드에서 기록된 로그(302)들은 읽을 수 없다. 파일 시스템의 fallocate() 시스템 호출은 초기화되지 않은 블록(303)들을 리턴한다. 처음에 블록(303)들이 기록될 때, 초기화된다. 블록(303)이 DIO로 기록될 때, 블록(303)들이 아직 초기화되지 않은 경우, 파일 시스템은 초기화된 플래그를 설정할 수 있다. 그러나, DIO 기록은 파일 시스템을 저널링하는 것을 수반하지 않기 때문에, 업데이트된 플래그는 예상되지 않은 시스템 실패에 손실되기 쉽다.

[0045] 할당된 블록(303)들을 초기화하는 것은 3가지 방법 있다. 첫번째 방법으로, 할당된 블록(303)들에 대해 제로(0)를 채우는 것이다. 두번째 방법과 세번째 방법은, 노출로부터 오래된 컨텐츠를 보호하기 위해 저장 장치인 eMMC(embedded MultiMediaCard)에 폐기 (또는 트림) 명령을 사용하는 것이다. eMMC는 낸드 플래시 메모리와 플래시 메모리 컨트롤러가 패키지로 통합된 저장장치다. 빠른 입/출력 속도를 지원하기 때문에 모바일 기기에 많이 사용되지만 낸드 플래시 저장장치의 특성상 쓰기/지우기 횟수에 제한이 있다. 폐기 명령은 입력으로서 논리적인 블록(303) 주소들의 리스트를 취할 수 있고, eMMC 저장 장치가 각각의 논리적인 블록(303)을 위한 맵핑 테이블 엔트리들을 제거하도록 요청할 수 있다.

[0046] 두 번째 방법은 파일 시스템에 폐기 옵션(discard option)을 마운트하고, 초기화된 플래그 셋트를 가지는 블록(303)들을 할당하는 fallocate()를 수정할 수 있다. 파일 시스템이 폐기 마운트 옵션을 이용할 때, 파일 블록(303)들이 할당 해제될 때 폐기 명령이 발행될 수 있다. 초기화된 플래그 셋트를 가지는 블록(303)들을 리턴하는 fallocate()를 강제하기 위해, 특정 스마트폰을 위해 리눅스 소스에 NO HIDE STALE patch를 포트하는 방안이 제안되었다.

[0047] 세번째 방법은 초기화된 플래그 셋트를 가지는 블록(303)들을 할당하고, 할당된 블록(303)들을 폐기하는 fallocate()를 수정하는 방안이 제안되었다. 본 발명의 일실시예에 따르면, 폐기 명령은 두번째 방법을 위해 개발된 NO HIDE STALE patch에 임베드되고, fallocate()를 위한 NO HIDE STALE DISCARD의 신규 플래그가 제안되었다.

[0048] 두번째 방법과 세번째 방법 간의 가장 큰 차이는 블록(303)이 언맵핑되는 시간이다. 두번째 방법과 세번째 방법에 의하면, 파일 블록(303)들이 할당 해제되거나, 할당될 때, 파일 블록(303)들은 언매핑(unmapped)될 수 있다. 두번째 방법에서, 파일 시스템은 모든 할당해제된 블록(303)들을 위해 폐기 명령을 발행할 수 있다. 하지만, 세번째 방법에 의하면, 파일 시스템은 저널 파일에 할당되는 파일 블록(303)들을 폐기할 수 있다. 그러면, 세번째 방법은 두번째 방법보다 좀더 작은 오버헤드를 나타낼 수 있다.

[0049] 첫번째 방법에 따라 블록(303)에 제로를 채우는 zero-fill 과정은 IO 오버헤드를 수반한다. 폐기 마운트 옵션을 이용하는 것은, 파일 시스템의 속도를 늦출 수 있다. 최근 스마트폰들은 파일 시스템에 폐기 옵션을 마운트한다. 폐기 옵션은 가비지 컬렉션을 좀더 효율적으로 만들도록 디자인될 수 있다. 하지만, 폐기 옵션은 오래된 컨텐츠를 숨기도록 디자인되지 않는다.

[0050] eMMC 표준은 폐기된 블록(303)들에 접근할 때 무엇을 읽을 필요가 있는지에 대해 정의하지 않는다. 어떤 eMMC 저장 장치는 폐기된 블록(303)들에 접근할 때 모두 0을 리턴할 수 있다. 오래된 컨텐츠를 숨기기 위한 폐기 명령을 사용할 때, 주어진 eMMC 저장 장치가 오래된 컨텐츠를 유출하지 않는 것을 보장하는 것이 필요하다. 이러한 경우, 폐기된 블록(303)들에 접근할 때 모두 0을 리턴하거나 또는 모두 1을 리턴하는 것이 보장될 수 있다.

[0051] 반면에, 본 발명의 일실시예에 따르면, 좀더 많은 오버헤드가 발생될 수 있다고 하더라도 트림 명령(trim command)를 이용하는 것을 의존할 필요가 있다. 트림된 블록(303)들에 접근할 때, 트림 명령은 모두 0을 리턴하거나 또는 모두 1을 리턴할 수 있다. 뿐만 아니라, 주어진 eMMC 저장 장치를 보장하기 위해 eMMC 저장 장치가 백그라운드에서 가비지 컬랙션을 수행하더라도, 특정 환경에서 호스트로부터의 폐기 명령을 무시할 수 없다.

[0052] 본 발명의 실시 예에 따른 방법들은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 본 발명을 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다.

[0053] 이상과 같이 본 발명은 비록 한정된 실시예와 도면에 의해 설명되었으나, 본 발명은 상기의 실시예에 한정되는 것은 아니며, 본 발명이 속하는 분야에서 통상의 지식을 가진 자라면 이러한 기재로부터 다양한 수정 및 변형이 가능하다.

[0054] 그러므로, 본 발명의 범위는 설명된 실시예에 국한되어 정해져서는 아니 되며, 후술하는 특허청구범위뿐 아니라 이 특허청구범위와 균등한 것들에 의해 정해져야 한다.

부호의 설명

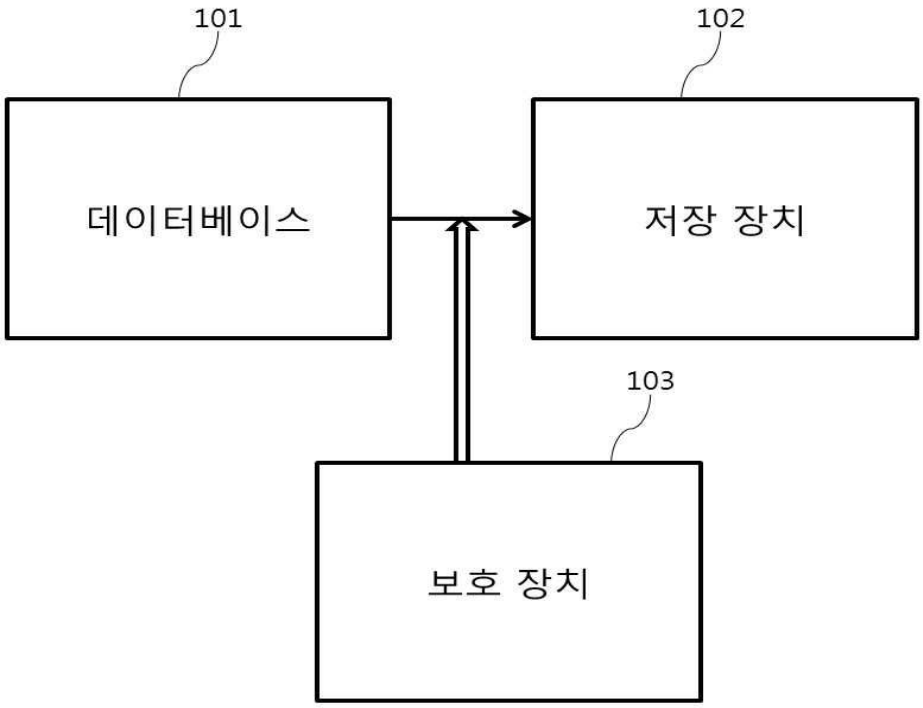
[0055] 301: 파일 메타데이터

302: 로그

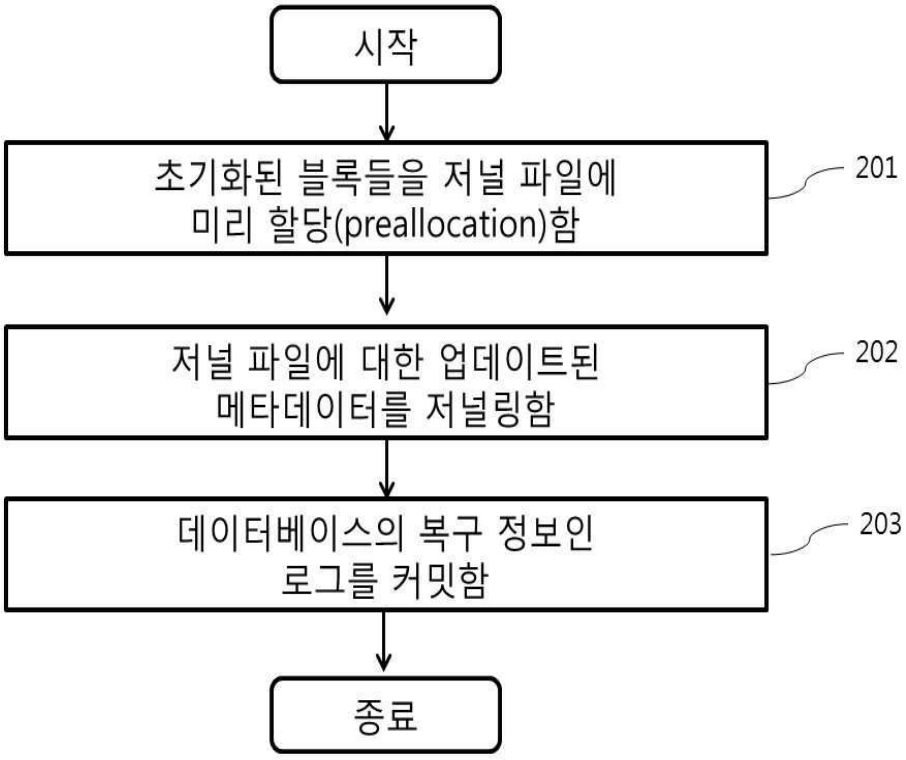
303: 블록

도면

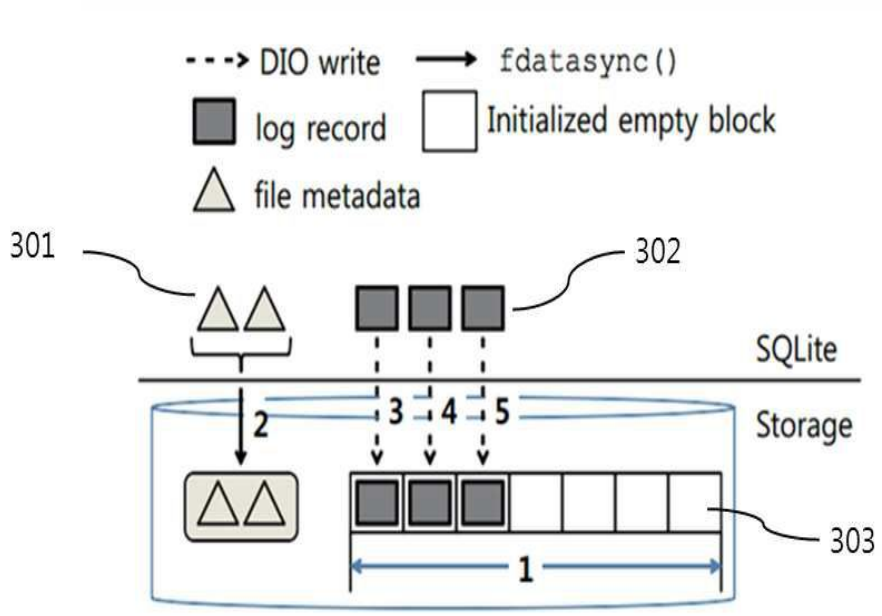
도면1



도면2


시작
초기화된 블록들을 저널 파일에 미리 할당(preallocation)함
201
저널 파일에 대한 업데이트된 메타데이터를 저널링함
202
데이터베이스의 복구 정보인 로그를 커밋함
203
종료

도면3


DIO write
fdatasync()
log record
Initialized empty block
file metadata
301
302
SQLite
Storage
2
3
4
5
1
303