(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년10월25일
(11) 등록번호 10-2316198
(24) 등록일자 2021년10월18일

(51) 국제특허분류(Int. Cl.)
G06F 16/00 (2019.01) *G06F 16/11* (2019.01)
G06F 16/13 (2019.01) *G06F 16/23* (2019.01)
G06F 3/06 (2006.01) *G06F 9/30* (2018.01)
(52) CPC특허분류
G06F 16/1873 (2019.01)
G06F 12/0253 (2013.01)
(21) 출원번호 10-2016-0029633
(22) 출원일자 2016년03월11일
심사청구일자 2021년03월11일
(65) 공개번호 10-2016-0146506
(43) 공개일자 2016년12월21일
(30) 우선권주장
62/175,073 2015년06월12일 미국(US)
14/954,885 2015년11월30일 미국(US)
(56) 선행기술조사문헌
JP2015095262 A
(뒷면에 계속)
전체 청구항 수 : 총 14 항

(73) 특허권자
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
(72) 발명자
기양석
미국 캘리포니아주 94303 팔로알토 알테어 워크 873
(74) 대리인
특허법인 고려

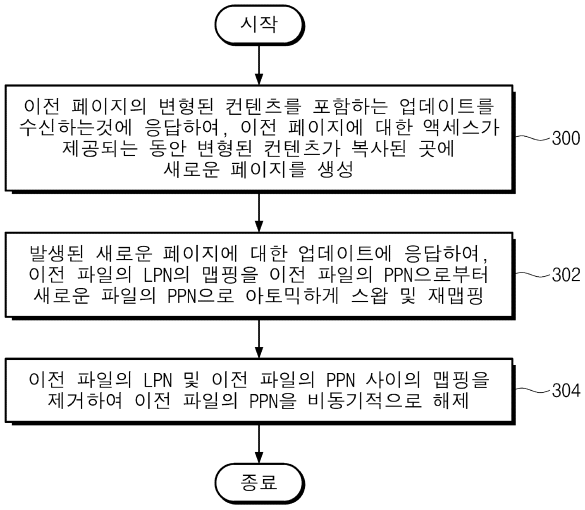
심사관 : 박미정

(54) 발명의 명칭 **파일 액세스 방법, 컴퓨터, 및 컴퓨터-읽기 가능한 매체에 실행 가능한 소프트웨어 제품**

(57) 요 약

본 발명의 실시 예는 논리적 페이지 넘버(LPN; logical page number)의 제1 리스트 및 업데이터를 위한 논리적 페이지 넘버의 제2 리스트를 수신하는 것에 응답하여, 적어도 하나의 프로세서에 의해 수행되는 업데이트된 파일에 대한 액세스를 제공한다. 논리적 페이지 넘버의 제1 리스트는 물리적 페이지 넘버의 제1 리스트와 맵핑되고, 논리적 페이지 넘버의 제2 리스트는 물리적 페이지 넘버의 제2 리스트와 맵핑된다. 방법은 논리적 페이지 넘버의 제1 리스트를 아토믹하게 재맵핑하여 논리적 페이지 넘버의 제1 리스트를 물리적 페이지 넘버의 제2 리스트와 맵핑하는 단계 및 논리적 페이지 넘버의 제1 리스트 및 물리적 페이지 넘버의 제1 리스트 사이의 맵핑을 제거하는 단계를 포함한다.

대 표 도 - 도3



(52) CPC특허분류
G06F 16/122 (2019.01)
G06F 16/13 (2019.01)
G06F 16/23 (2019.01)
G06F 3/0643 (2013.01)
G06F 3/0676 (2013.01)
G06F 9/30043 (2013.01)
G06F 2212/7201 (2013.01)

(56) 선행기술조사문헌
US20130332654 A1
US20090109788 A1
CN101458961 A
US20130238869 A1
JP2004280752 A
JP2005115561 A
JP2012203864 A
KR1020150053720 A

명 세 서

청구범위

청구항 1

외부 프로세서와 연결된 스토리지 장치를 포함하는 파일 관리 시스템에서 업데이트된 파일로의 엑세스를 제공하는 방법에 있어서,

파일 변환 계층(FTL; file translation layer) 맵은 파일 시스템에 의해 유지되는 논리 페이지 넘버들(LPN; logical page number) 및 상기 스토리지 장치에 의해 유지되는 물리 페이지 넘버들(PPN; physical page number) 사이의 FTL에 의해 유지되고,

상기 방법은:

상기 스토리지 장치에 의해, 상기 외부 프로세서로부터 제1 커맨드를 수신하고, 상기 제1 커맨드는 제1 파일로의 액세스가 유지되는 동안, 상기 제1 파일의 변형된 컨텐츠 및 상기 변형된 컨텐츠가 복사된 제2 파일의 생성을 포함하는 업데이트를 수신한 애플리케이션에 응답하여 전송되고, 상기 제1 커맨드의 파라미터들은 상기 제1 파일에 대응하는 제1 리스트의 LPN들 및 상기 제2 파일에 대응하는 제2 리스트의 LPN들을 포함하고, 상기 제1 리스트의 LPN들은 상기 스토리지 장치 상의 상기 제1 파일의 저장 위치를 나타내는 제1 리스트의 PPN들로 맵핑되고, 상기 제2 리스트의 LPN들은 상기 스토리지 상의 상기 제2 파일의 저장 위치를 나타내는 제2 리스트의 PPN들로 맵핑되는 단계;

상기 제1 리스트의 LPN들이 상기 제2 리스트의 PPN들로 맵핑되도록 상기 제1 리스트의 LPN들을 아토믹하게 재맵핑(atomically remapping)하고, 상기 제1 리스트의 LPN들의 각 LPN에 대하여, 상기 FTL 맵에서, 상기 제1 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 생성되는 단계;

상기 제1 리스트의 PPN들로의 상기 제1 리스트의 LPN들의 맵핑을 비동기적으로(asynchronously) 제거(trimming)하고, 상기 제1 리스트의 LPN들의 모든 LPN에 대하여, 상기 제1 리스트의 LPN들의 LPN 및 상기 제1 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되는 단계; 및

상기 제2 리스트의 PPN들로의 제2 리스트의 LPN들의 맵핑이 언맵(unmap)되고, 상기 제2 리스트의 LPN들의 모든 LPN들에 대하여, 상기 제2 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되는 단계를 포함하는 방법.

청구항 2

제 1 항에 있어서,

상기 제1 커맨드는 스왑 및 트림(SWAT; swap and trim) 커맨드를 포함하고, 애플리케이션 프로그래밍 인터페이스(API; application programming interface)로 발행되는 방법.

청구항 3

제 2 항에 있어서,

상기 SWAT 커맨드는 애플리케이션 또는 운영 체제로부터의 호출에 응답하여, 스토리지 장치 드라이버 API로부터 상기 스토리지 장치로 발행되는 방법.

청구항 4

제 2 항에 있어서,

상기 제1 리스트의 LPN들 및 상기 제2 리스트의 LPN들에 포함된 LPN들은 상기 제1 리스트의 LPN들 및 상기 제2 리스트의 LPN들에 의해 정해진 순서대로 아토믹하게 재맵핑되는 방법.

청구항 5

제 4 항에 있어서,

비동기적으로 제거될 페이지들이 가비지 콜렉션에 의해 리클레임되기 전까지 동시적인 프로세스들이 상기 페이지를 엑세스하는 것을 허용하는 약한 맵핑(weak mapping)을 제공하는 단계를 더 포함하는 방법.

청구항 6

제 5 항에 있어서,

가비지 콜렉션, 쓰기 커맨드, 또는 트림 커맨드에 의해 상기 약한 맵핑을 삭제(remove)하는 단계를 더 포함하는 방법.

청구항 7

시스템에 있어서,

프로세서 및 메모리를 포함하는 컴퓨터; 및

상기 프로세서 및 상기 메모리의 외부에 위치한 스토리지 장치를 포함하고,

파일 변환 계층(FTL; file translation layer) 맵은 파일 시스템에 의해 유지되는 논리 페이지 넘버들(LPN; logical page number) 및 상기 스토리지 장치에 의해 유지되는 물리 페이지 넘버들(PPN; physical page number) 사이의 FTL에 의해 유지되고,

상기 스토리지 장치는:

상기 프로세서로부터 제1 커맨드를 수신하고, 상기 제1 커맨드는 제1 파일로의 액세스가 유지되는 동안, 상기 제1 파일의 변형된 컨텐츠 및 상기 변형된 컨텐츠가 복사된 제2 파일의 생성을 포함하는 업데이트를 수신한 애플리케이션에 응답하여 전송되고, 상기 제1 커맨드의 파라미터들은 상기 제1 파일에 대응하는 제1 리스트의 LPN들 및 상기 제2 파일에 대응하는 제2 리스트의 LPN들을 포함하고, 상기 제1 리스트의 LPN들은 상기 스토리지 장치 상의 상기 제1 파일의 저장 위치를 나타내는 제1 리스트의 PPN들로 맵핑되고, 상기 제2 리스트의 LPN들은 상기 스토리지 상의 상기 제2 파일의 저장 위치를 나타내는 제2 리스트의 PPN들로 맵핑되고;

상기 제1 리스트의 LPN들이 상기 제2 리스트의 PPN들로 맵핑되도록 상기 제1 리스트의 LPN들을 아토믹하게 재맵핑(atomically remapping)하고, 상기 제1 리스트의 LPN들의 각 LPN에 대하여, 상기 FTL 맵에서, 상기 제1 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 생성되고;

상기 제1 리스트의 PPN들로의 상기 제1 리스트의 LPN들의 맵핑을 비동기적으로(asynchronously) 제거(trimming)하고, 상기 제1 리스트의 LPN들의 모든 LPN에 대하여, 상기 제1 리스트의 LPN들의 LPN 및 상기 제1 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되고;

상기 제2 리스트의 PPN들로의 제2 리스트의 LPN들의 맵핑이 언맵(unmap)되고, 상기 제2 리스트의 LPN들의 모든 LPN들에 대하여, 상기 제2 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되도록 구성된 시스템.

청구항 8

제 7 항에 있어서,

상기 제1 커맨드는 스왑 및 트림(SWAT; swap and trim) 커맨드를 포함하고, 애플리케이션 프로그래밍 인터페이스(API; application programming interface)로 발행되는 시스템.

청구항 9

제 8 항에 있어서,

상기 제1 리스트의 LPN들 및 상기 제2 리스트의 LPN들에 포함된 LPN들은 상기 제1 리스트의 LPN들 및 상기 제2 리스트의 LPN들에 의해 정해진 순서대로 아토믹하게 재맵핑되는 시스템.

청구항 10

제 8 항에 있어서,

상기 SWAT 커맨드는 애플리케이션 또는 운영 체제로부터의 호출에 응답하여, 스토리지 장치 드라이버 API로부터 상기 스토리지 장치로 발행되는 시스템.

청구항 11

제 7 항에 있어서,

비동기적으로 제거될 페이지들이 가비지 콜렉션에 의해 리클레임되기 전까지 동시적인 프로세스들이 상기 페이지를 엑세스하는 것을 허용하는 약한 맵핑(weak mapping)이 제공되는 시스템.

청구항 12

제 11 항에 있어서,

가비지 콜렉션, 쓰기 커맨드, 또는 트림 커맨드에 의해 상기 약한 맵핑이 삭제(remove)되는 시스템.

청구항 13

외부 프로세서와 연결된 스토리지 장치를 포함하는 파일 관리 시스템에서 업데이트된 파일로의 액세스를 제공하는 프로그램 명령어들을 포함하는 비-일시적 저장 매체에 있어서,

파일 변환 계층(FTL; file translation layer) 맵은 파일 시스템에 의해 유지되는 논리 페이지 넘버들(LPN; logical page number) 및 상기 스토리지 장치에 의해 유지되는 물리 페이지 넘버들(PPN; physical page number) 사이의 FTL에 의해 유지되고,

상기 프로그램 명령어들은:

상기 스토리지 장치에 의해, 상기 외부 프로세서로부터 제1 커맨드를 수신하고, 상기 제1 커맨드는 제1 파일로의 액세스가 유지되는 동안, 상기 제1 파일의 변형된 컨텐츠 및 상기 변형된 컨텐츠가 복사된 제2 파일의 생성을 포함하는 업데이트를 수신한 애플리케이션에 응답하여 전송되고, 상기 제1 커맨드의 파라미터들은 상기 제1 파일에 대응하는 제1 리스트의 LPN들 및 상기 제2 파일에 대응하는 제2 리스트의 LPN들을 포함하고, 상기 제1 리스트의 LPN들은 상기 스토리지 장치 상의 상기 제1 파일의 저장 위치를 나타내는 제1 리스트의 PPN들로 맵핑되고, 상기 제2 리스트의 LPN들은 상기 스토리지 상의 상기 제2 파일의 저장 위치를 나타내는 제2 리스트의 PPN들로 맵핑되는 것;

상기 제1 리스트의 LPN들이 상기 제2 리스트의 PPN들로 맵핑되도록 상기 제1 리스트의 LPN들을 아토믹하게 재맵핑(atomically remapping)하고, 상기 제1 리스트의 LPN들의 각 LPN에 대하여, 상기 FTL 맵에서, 상기 제1 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 생성되는 것;

상기 제1 리스트의 PPN들로의 상기 제1 리스트의 LPN들의 맵핑을 비동기적으로(asynchronously) 제거(trimming)하고, 상기 제1 리스트의 LPN들의 모든 LPN에 대하여, 상기 제1 리스트의 LPN들의 LPN 및 상기 제1 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되는 것; 및

상기 제2 리스트의 PPN들로의 제2 리스트의 LPN들의 맵핑이 언맵(unmap)되고, 상기 제2 리스트의 LPN들의 모든 LPN들에 대하여, 상기 제2 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되는 것을 포함하는 비-일시적 저장 매체.

청구항 14

적어도 하나의 프로세서를 포함하는 외부 컴퓨터와 연결된 스토리지 장치에 의해 수행되는 업데이트된 파일로의 엑세스를 제공하는 방법에 있어서,

파일 변환 계층(FTL; file translation layer) 맵은 파일 시스템에 의해 유지되는 논리 페이지 넘버들(LPN; logical page number) 및 상기 스토리지 장치에 의해 유지되는 물리 페이지 넘버들(PPN; physical page number) 사이의 FTL에 의해 유지되고,

상기 방법은:

상기 적어도 하나의 프로세서로부터 제1 커맨드를 수신하고, 상기 제1 커맨드는 제1 파일로의 액세스가 유지되는 동안, 상기 제1 파일의 변형된 컨텐츠 및 상기 변형된 컨텐츠가 복사된 제2 파일의 생성을 포함하는 업데이트를 수신한 애플리케이션에 응답하여 전송되고, 상기 제1 커맨드의 파라미터들은 상기 제1 파일에 대응하는 제

1 리스트의 LPN들 및 상기 제2 파일에 대응하는 제2 리스트의 LPN들을 포함하고, 상기 제1 리스트의 LPN들은 상기 스토리지 장치 상의 상기 제1 파일의 저장 위치를 나타내는 제1 리스트의 PPN들로 맵핑되고, 상기 제2 리스트의 LPN들은 상기 스토리지 상의 상기 제2 파일의 저장 위치를 나타내는 제2 리스트의 PPN들로 맵핑되는 단계;

상기 제1 파일에 대한 상기 제1 리스트의 LPN들이 상기 제2 파일에 대한 상기 제2 리스트의 PPN들로 맵핑되도록 상기 제1 리스트의 LPN들을 아토믹하게 재맵핑(atomically remapping)하고, 상기 제1 리스트의 LPN들의 각 LPN에 대하여, 상기 FTL 맵에서, 상기 제1 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 생성되는 단계;

상기 제1 파일에 대한 상기 제1 리스트의 PPN들의 맵핑을 비동기적으로(asynchronously) 제거(trimming)하고, 상기 제1 리스트의 LPN들의 모든 LPN에 대하여, 상기 제1 리스트의 LPN들의 LPN 및 상기 제1 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되는 단계; 및

상기 제2 리스트의 PPN들로의 제2 리스트의 LPN들의 맵핑이 언맵(unmap)되고, 상기 제2 리스트의 LPN들의 모든 LPN들에 대하여, 상기 제2 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되는 단계를 포함하는 방법.

청구항 15

삭제

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

발명의 설명

기 술 분 야

[0001] 본 발명은 파일 관리 시스템에 관한 것으로 더욱 상세하게는 파일 액세스 방법, 컴퓨터, 및 컴퓨터-읽기 가능한 매체에 실행 가능한 소프트웨어 제품에 관한 것이다.

배 경 기 술

[0002] 파일 관리 시스템들은 파일의 가장 최신 버전을 추적하기 위하여 데이터 구조를 업데이트한다. 이러한 프로세스에서, 일부 시스템들은 이전 버전(the older version)에 덮어 쓰는 대신에 파일의 다른 버전을 생성하고, 잠시 동안 파일의 이전 버전을 유지한다. 결과적으로, 시스템들은 새로운 버전이 업데이트되었음에도 불구하고 읽기 동작 동안 파일의 이전 버전을 액세스할 수 있다. 예를 들어, 사용자가 블로그 페이지를 업데이트할 경우, 사용자가 블로그 페이지를 업데이트하는 동안 다른 유저들은 블로그 페이지의 이전 버전을 읽을 수 있다. 이 때, 시스템들은 새로운 파일을 액세스하고 이전 파일을 교체하기 위하여 시스템 정보를 업데이트하는 것을 요구한다. 일반적으로, 시스템의 데이터 구조에서 연속적인 업데이트들이 연속적으로 발생되고, 이는 스토리지로 많은 쓰기를 필요로 한다.

[0003] 따라서, 시스템의 데이터 구조들에서 연속적인 갱신을 수행하지 않고 갱신된 파일로의 접근을 제공하는 시스템 및 향상된 방법이 제공될 것이다.

발명의 내용

해결하려는 과제

[0004] 본 발명의 목적은 향상된 수명, 향상된 신뢰성, 및 향상된 성능을 갖는 파일 액세스 방법, 컴퓨터, 및 컴퓨터-읽기 가능한 매체에 실행 가능한 소프트웨어 제품이 제공하는 것이다.

과제의 해결 수단

[0005] 본 발명의 실시 예는 논리적 페이지 넘버(LPN; logical page number)의 제1 리스트 및 업데이터를 위한 논리적 페이지 넘버의 제2 리스트를 수신하는 것에 응답하여, 적어도 하나의 프로세서에 의해 수행되는 업데이트된 파일에 대한 액세스를 제공한다. 논리적 페이지 넘버의 제1 리스트는 물리적 페이지 넘버의 제1 리스트와 맵핑되고, 논리적 페이지 넘버의 제2 리스트는 물리적 페이지 넘버의 제2 리스트와 맵핑된다. 방법은 논리적 페이지 넘버의 제1 리스트를 아토믹하게 재맵핑하여 논리적 페이지 넘버의 제1 리스트를 물리적 페이지 넘버의 제2 리스트와 맵핑하는 단계 및 논리적 페이지 넘버의 제1 리스트 및 물리적 페이지 넘버의 제1 리스트 사이의 맵핑을 제거하는 단계를 포함한다.

발명의 효과

[0006] 본 발명에 따르면, 향상된 수명, 향상된 신뢰성, 및 향상된 성능을 갖는 파일 액세스 방법, 컴퓨터, 및 컴퓨터-읽기 가능한 매체에 실행 가능한 소프트웨어 제품이 제공된다.

도면의 간단한 설명

[0007] 도 1은 본 발명의 실시 예에 따른 파일 관리 시스템을 보여주는 블록도이다.

도 2는 파일을 업데이트하는 일반적인 파일 관리 시스템을 보여주는 도면이다.

도 3은 본 발명의 실시 예에 따라 업데이트 동안 파일에 대한 액세스를 제공하기 위한 파일 관리 시스템에 의해 수행되는 절차를 보여주는 순서도이다.

도 4는 도 3의 절차를 도식적으로 보여주는 도면이다.

도 5 내지 도 12는 포워드 맵핑 테이블(FMT), 물리적 페이지 넘버의 리스트, 및 리버스 맵핑 테이블(RMT)의 다양한 상태를 보여주는 도면들이다.

도 13은 사용되지 않는 페이지들에 대한 예시적인 SWAT 커맨드를 보여준다.

도 14는 맵핑이 없는 기본 논리적 페이지 넘버에 대한 예시적인 SWAT 커맨드를 보여준다.

도 15는 반복적인 SWAT 커맨드들의 예를 보여준다.

발명을 실시하기 위한 구체적인 내용

[0008] 첨부된 도면들에 도시된 예들 및 본 발명의 실시 예들이 상세하게 설명될 것이다. 유사한 참조번호들은 유사한 구성 요소들을 칭한다. 이하에서, 본 발명의 기술적 사상을 설명하기 위하여 도면들을 참조하여 실시 예들이 설명된다.

[0009] 본 발명의 이점들 및 특징들 및 이를 달성하기 위한 방법들은 이하의 실시 예들의 상세한 설명 및 첨부된 도면들을 참조함으로써 쉽게 이해될 수 있다. 그러나, 본 발명의 기술적 사상은 다양한 다른 형태들로 구현될 수 있고, 본문에 개시된 실시 예들에 한정되어 구성되지 않을 수 있다. 이러한 실시 예들은 본문이 완벽하게 되고, 당업자에게 본 발명의 기술적 사상이 전달되도록 실시 예들이 제공될 수 있다. 본 발명의 기술적 사상은 첨부된 특허청구범위에 의해 정의될 것이다. 도면들에서 계층들(layers) 및 영역들(regions)의 두께는 명확성을 위하여 과장될 수 있다.

[0010] 본문(특히 이하의 특허청구범위)에서의 단수 용어 및 유사한 용어들의 사용은 본문에서 다르게 지칭되거나 또는 명확하게 반박되지 않는 한 단수 및 복수를 모두를 포함하도록 이해된다. 다르게 언급되지 않는 한, "포함하다"(comprising, having, including, 및 containing)의 용어들은 개방형 의미로서 이해된다.

[0011] 본문에서 사용되는 구성(component) 또는 모듈(module)의 용어는 특정 동작을 수행하는 FPGA (field programmable gate array)또는 ASIC(application specific integrated circuit)과 같은 소프트웨어 또는 하드웨어 구성을 의미한다. 그러나, 이에 한정되지는 않는다. 구성 또는 모듈은 유리하게 주소지정 스토리지 매체 내에 위치하도록 구성될 수 있고, 하나 또는 그 이상의 프로세스들을 수행하도록 구성될 수 있다. 즉, 예를 들어, 구성 또는 모듈은 소프트웨어 구성들, 객체 지향 소프트웨어 구성들(object-oriented software

components) 클래스 구성들(class components) 및 작업 구성들(task components), 프로세스들(processes), 기능들(functions), 속성들(attributes), 절차들(procedures), 서브 루틴들(subroutines), 프로그램 코드의 세그먼트들(segments of program code), 드라이버들(drivers), 펌웨어(firmware), 마이크로코드(microcode), 회로(circuitry), 데이터(data), 데이터베이스(databases), 데이터 구조들(data structures), 테이블들(tables), 어레이들(arrays) 등과 같은 구성들을 포함할 수 있다. 구성들 또는 모듈들을 위해 제공되는 기능은 몇몇 구성들 또는 모듈들로 조합될 수 있다.

[0012] 다르게 정의되지 않는 한, 본문에서 사용되는 모든 기술적 과학적 용어들은 본 발명이 속하는 기술 분야에서 통상의 기술자에 의해 공통적으로 이해되는 동일한 의미를 갖는다. 본문에서 제공되는 예시적인 용어들 또는 모든 예시들의 사용은 단지 본 발명을 설명하기 위해 사용되며, 본 발명의 범위가 다르게 한정되는 것은 아니다. 더욱이, 다르게 정의되지 않는 한, 일반적으로 사용되는 사전에서 정의된 모든 용어들은 다르게 정의되지 않는다.

[0013] 도 1은 본 발명의 실시 예에 따른 파일 관리 시스템을 보여주는 블록도이다. 파일 관리 시스템(10)은 전자 문서들 또는 파일들을 추적하고, 저장하는 컴퓨터(12)를 포함할 수 있다.일 실시 예에서, 컴퓨터(12)는 데스크탑, 랩탑, 또는 워크 스테이션을 포함할 수 있다. 다른 실시 예에서, 컴퓨터(12)는 네트워크(미도시)를 통해 클라이언트 컴퓨터와 통신하는 서버를 포함할 수 있다. 컴퓨터(12)는 고속 스토리지(18, 예를 들어, 솔리드 스테이트 드라이브(SSD))와 같은 스토리지 디바이스, 프로세서(14), 및 메모리(16)를 포함하는 일반적인 컴퓨터 구성들을 포함할 수 있다.

[0014] 프로세서(14)는 하나 또는 그 이상의 코어들을 갖는 싱글 프로세서 또는 멀티 프로세서를 포함할 수 있다. 프로세서(14)는 메모리(16)로부터의 프로그램 명령어를 실행한다. 소프트웨어의 예시적인 형태들은 애플리케이션(20), 운영 체제(22), 파일 시스템(24), 및 고속 스토리지(18)를 위한 장치 드라이버 애플리케이션 프로그래밍 인터페이스(26, device driver API; application programming interface)를 포함할 수 있다.

[0015] 본 발명이 속하는 기술 분야에서 잘 알려진 바와 같이 애플리케이션(20)은 프로세서(14)에 의해 실행되어 컴퓨터(12)가 컴퓨터(12)의 구동 중에 작업들을 수행하도록 하는 컴퓨터 소프트웨어를 가리킬 수 있다. 예를 들어, 애플리케이션(20)은 웹브라우저, 워드 프로세서, 및 데이터베이스 프로그램을 포함할 수 있다. 일반적으로, 애플리케이션(20)은 파일(28)을 생성하고, 변형 또는 갱신한다. 운영 체제(22)는 컴퓨터(12)의 동작을 제어하고 관리하는 시스템 소프트웨어이다. 운영 체제(22)는 마이크로소프트 윈도우, 맥OS X, 및 리눅스를 포함할 수 있다.

[0016] 파일 시스템(24)은 파일(28)과 같은 정보를 고속 스토리지(18)와 같은 데이터 저장 장치들에 어떻게 저장하고, 검색하고, 갱신하는지를 제어하는 소프트웨어이다. 일부 애플리케이션/파일 형태들은 파일들(28)에 페이지들과 같은 데이터가 고속 스토리지(18) 또는 다른 컴퓨터 스토리지 자이들에서 어디에 저장되는지 특정하기 위하여 논리적 페이지 넘버링(logical page numbering)을 사용할 수 있다. 좀 더 상세하게는, 논리적 페이지 넘버링(logical page numbering)은 논리적 페이지 넘버들(LPNs)(30)을 고속 스토리지(18)의 특정 저장 위치들로 사상된 파일들(28)의 페이지들로 할당하는 개념이다.

[0017] 장치 드라이버 애플리케이션 프로그램 인터페이스(API; application programming interface)(26)는 애플리케이션(20), 운영체제(22), 및 파일 시스템(24)이 고속 스토리지(18)와 통신할 수 있도록 한다. 장치 드라이버 API(26)는 고속 스토리지(18)로부터 데이터를 수신하거나 또는 저장하기 위한 커맨드들을 제공한다.

[0018] 일 실시 예에서, 고속 스토리지(18)는 고속 스토리지(18)에 저장된 페이지들의 어드레스들을 제공하기 위하여 물리적 페이지 넘버링(physical page numbering)을 사용한다. 어드레스들의 이러한 형태들은 특정 저장 위치들과 사상될 수 있는 물리적 페이지 넘버(PPNs)(32)라 불린다. 일 실시 예에서, 고속 스토리지(18)는 SSD(solid-state drive)(또는, solid-state disk)를 포함할 수 있다. SSD는 메모리와 같은 집적 회로를 사용하여 파일(28)과 같은 데이터를 지속적으로 저장하는 데이터 저장 장치이다. 일 실시 예에서, SSD는 낸드 플리시 메모리 또는 랜덤 액세스 메모리(RAM)를 사용할 수 있다.

[0019] 일 실시 예에서, 고속 스토리지(18)는 컴퓨터(12)로 논리 섹터 인터페이스를 제공하면서 고속 스토리지(18)를 관리하는 파일 변환 계층(34, FTL; file translation layer) 또는 동등한 구성을 더 포함할 수 있다. 이와 같이, FTL(34)은 파일 시스템(24)에 의해 유지되는 LPNs(30) 및 고속 스토리지(18)에 의해 유지되는 PPNs(32)의 매핑 관계을 제어한다. 다른 실시 예에서, 예시적인 실시 예들은 SSD와 다른 형태의 스토리지 장치들을 사용할 수 있다.

[0020] 도 2는 파일을 업데이트하는 일반적인 파일 관리 시스템을 보여주는 도면이다. 도시된 실시 예는 애플리케이션

이 사용자의 멀티 페이지 블로그(multipage blog)를 가리키는 파일(200)을 유지하는 것으로 가정한다. 사용자가 블로그를 업데이트할 때마다, 애플리케이션은 파일(200)을 메모리로부터 읽고, 이에 따라 컨텐츠를 변형한다. 업데이트가 수행되는 동안, 시스템은 변형된 파일이 복사된 새로운 파일(204)을 생성한다. 그 동안, 사용자가 블로그를 업데이트하는 반면에, 다른 사용자들은 여전히 이전 파일(202)(old file)을 액세스하고 볼 수 있다. 사용자가 블로그 작성을 마무리하고, 페이지들을 제출할 경우, 애플리케이션은 새로운 파일(204)을 스위치하고, 이전 파일(202)을 삭제한다.

[0021] 실제 파일들이 저장된 스토리지 시스템이 물리적 페이지 넘버들(PPN)(208)을 유지하는 반면에,파일 시스템은 이전 파일(202) 및 새로운 파일(204)의 페이지들을 위한 논리적 페이지 넘버(LPN)(206)를 유지한다. 이러한 실시예에서, 스토리지 장치가 제4 및 제5 논리적 페이지 넘버들(LPN4, LPN5)를 제0 및 제1 물리적 페이지 넘버들(PPN0, PPN1)에 각각 맵핑하는 반면에, 파일 시스템은 이전 파일(202)의 페이지들을 제4 및 제5 논리적 페이지 넘버들(LPN4, LPN5)로 맵핑한다. 유사하게, 새로운 파일(204)의 페이지들은 제24 및 제25 논리적 페이지 넘버들(LPN24, LPN25)로 맵핑되고, 제24 및 제25 논리적 페이지 넘버들(LPN24, LPN25)은 제11 및 제12 물리적 페이지 넘버들(PPN11, PPN12)로 차례대로 각각 맵핑된다.

[0022] 이와 같은 종래의 시스템의 문제점은 이전 파일(202)이 교체된 새로운 파일(204)을 액세스하기 위하여 시스템들이 시스템 정보에 대한 업데이트를 요구하는 것이다. 일반적으로, 시스템의 데이터 구조에서 연속적인 업데이트를 발생시키고, 스토리지로의 많은 쓰기를 유발한다.

[0023] 시스템 정보는 각 파일 또는 문서에 대한 메타 데이터를 포함할 수 있다. 예를 들어, 메타 데이터는 문서가 저장된 날짜, 파일을 저장한 사용자의 식별자를 포함할 수 있다. 메타 데이터는 일반적으로 데이터 구조에 저장된다. 이러한 데이터 구조의 일 실시 예는 저장된 데이터를 유지하고 로그 시간으로 검색, 연속적인 액세스, 삽입, 및 삭제가 가능하고, 저장된 데이터를 유지하는 트리 데이터 구조인 B-트리이다. 각 파일(200)을 위한 메타 데이터 또는 논리적 페이지 넘버(206)의 리스트는 B-트리의 리프 노드(leaf node)에 의해 유지된다. 일반적으로, 파일(200)당 하나의 리프 노드가 존재한다. 파일(200)의 이름이 B-트리의 리프 레벨 근처에 저장된 경우, 노드로부터 루트 노드까지의 경로 상의 모든 노드들은 노드의 변화들을 반영하여 갱신하는 것이 필요할 수 있다. 따라서 연속적인 업데이트 및 스토리지로의 쓰기가 발생한다.

[0024] 예를 들어, 상술된 바와 같이 사용자가 블로그를 업데이트할 때마다, 애플리케이션은 적어도 하나의 디스크 쓰기를 유발하는 블로그를 포함하는 파일 정보를 갱신하는 것을 요구한다. 더욱 많은 유저들인 경우, 더욱 많은 디스크 쓰기가 유발된다.

[0025] 예시적인 실시 예들은 시스템 데이터 구조들을 갱신하지 않고, 새로운 업데이트 파일에 대한 액세스를 제공하기 위한 향상된 방법 및 시스템과 관련된다. 이로 인하여, 연속적인 업데이트들 및 시스템에서 과도한 디스크 쓰기들이 제거되거나 또는 최소화된다.

[0026] 예시적인 실시 예들은 이러한 문제점을 해결하기 위한 새로운 스토리지(예를 들어, SSD) 커맨드 및 애플리케이션 프로그래밍 인터페이스(API; Application Programming Interface)를 제공한다. 즉, 파일의 새로운 버전은 시스템 데이터 구조를 업데이트하지 않고 액세스될 수 있는 경우, 스토리지 쓰기들은 최신 데이터 시스템 정보를 반영하기 위한 연속적인 업데이트들의 제거로 인하여 상당히 감소되거나 또는 피할 수 있다. 단순화된 실시예 이전에, B-트리, 문서 로깅, 쉐도우 페이징, 더블 버퍼 쓰기들, 및 다른 것들과 같은 다양한 애플리케이션들은 예시적인 실시 예가 적용될 수 있는 이러한 특징들을 포함할 수 있다.

[0027] 예시적인 실시 예들은 본문에서 SWAT(SWap And Trim) 커맨드라 칭하는 커맨드 및 관련된 API를 제안한다. 논리적 페이지 넘버의 리스트들이 주어진 경우, SWAT 커맨드는 아토믹하게(atomically) 리스트의 논리적 페이지 넘버(LPN)의 맵핑을 순서대로 스왑 또는 재맵핑하고, 사용되지 않는 논리적 페이지 넘버(unused LPN)는 제거(trim)한다. 동시적인 프로세스들이 페이지가 가비지 콜렉션에 의해 리클레임되기 전까지 비동기적으로 제거되는 페이지들을 액세스할 수 있는 약한 맵핑 컨셉(weak mapping concept)이 제공된다.

[0028] 도 3은 본 발명의 일 실시 예에 따라 업데이트 동안 파일에 대한 액세스를 제공하기 위한 파일 관리 시스템에 의해 수행되는 프로세스를 보여주는 순서도이다. 도 4는 프로세스를 도식화한 도면이다.

[0029] 도 1, 도 3, 및 도 4를 참조하면, 블록(300)에서, 프로세스는 이전 파일(old file)로의 액세스가 유지되는 동안, 목표 페이지의 변형된 컨텐츠가 복사된 새로운 파일(new file)을 생성하기 위하여 이전 파일의 변형된 기본 페이지(modified base page)를 포함하는 업데이트를 수신하는 것에 응답하여 시작될 수 있다.

[0030] 이는 업데이트되는 파일(400)에 응답하여 파일의 이전 버전(즉, 이전 파일(old file))(402)로의 액세스가 임시

적으로 유지되고, 변형된 목표 페이지들을 포함시켜 파일의 새로운 버전(즉, 새로운 파일(new file))(402)이 생성되는 것을 보여주는 도 4에 도시된다. 파일(400)은 애플리케이션(20)의 하나 또는 운영 체제(22)를 통해 업데이트될 수 있다.

[0031] SWAT 동작 이전에, 파일 시스템(24)은 이전 파일의 논리적 페이지 넘버들(LPN)(406)의 기본 리스트(예를 들어, LPN4, LPN5)를 사용하여 이전 파일(402)에 논리적 저장 위치를 표현하고, 고속 스토리지(18)는 이전 파일의 물리적 페이지 넘버들(PPN)(408)의 리스트(예를 들어, PPN0, PPN1)를 사용하여 물리적 저장 위치를 나타낸다. 유사하게, 새로운 파일(404)을 위한 페이지들의 논리적 저장 위치는 새로운 파일의 논리적 페이지 넘버(410)의 리스트(예를 들어, LPN24, LPN25)를 사용하여 표현되고, 고속 스토리지(18)의 물리적 저장 위치는 새로운 파일의 물리적 페이지 넘버(412)의 리스트(예를 들어, PPN11, PPN12)를 사용하여 표현된다.

[0032] 도 3 및 도 4를 다시 참조하면, 블록(302)에서, 발생된 새로운 파일(404)의 업데이트에 응답하여, 이전 파일의 논리적 페이지 넘버(406)의 맵핑은 이전 파일의 물리적 페이지 넘버(408)로부터 새로운 파일의 물리적 페이지 넘버(412)로 아토믹하게 스왑 또는 재맵핑된다. 도 4에 도시된 바와 같이, 이전 파일의 논리적 페이지 넘버(406)의 리스트(즉, LPN4, LPN5)의 물리적 맵핑은 새로운 파일(404)의 물리적 페이지 넘버(412)의 리스트(즉, PPN11, PPN12)로 리맵핑된다. 좀 더 상세하게, SWAT 커맨드가 발생될 때, LPN4는 PPN11로 맵핑되고, LPN5는 PPN12로 맵핑된다. 결과적으로, SWAT API는 파일 정보를 갱신하는 요구를 제거한다.

[0033] 더욱이, 블록(304)에서, 이전 파일의 논리적 페이지 넘버(406) 및 이전 파일의 물리적 페이지 넘버(408)의 맵핑은 이전 파일의 물리적 페이지 넘버(408)를 해제(releasing)하여 비동기적으로(asynchronously) 제거된다. 도 4에 도시된 바와 같이, 이전 파일의 물리적 페이지 넘버(408)(PPN0, PPN1)를 제거한 이후에, 참조 번호 "416"의 "X"표시와 같이 해제된다.

[0034] SWAT 커맨드가 실행될 때, 새로운 파일(404)은 이전 파일의 물리적 페이지 넘버(408)(실제로 더 이상 새로운 파일이 아니다.)로 맵핑된다. 사용자들이 새로운 물리적 페이지들이 가비지 콜렉션에 의해 재활용될 때까지 새로운 물리적 페이지들(즉, 이전 파일(402)의 물리적 페이지들)을 읽게 하여 새로운 파일(404)은 휘발성 읽기 전용이 된다.

[0035] 이하의 설명 및 도면들은 SWAT 동작들이 SSD에 적용될 때, 맵핑 테이블이 어떻게 업데이트되는지를 보여준다. 이러한 실시 예들은 포워드 맵핑 테이블(FMT; Forward Mapping Table) 및 리버스 맵핑 테이블(RMT; Reverse Mapping Table)과 같은 2개의 맵핑 테이블을 사용한다. 맵핑 형태는 S(강함; Strong) 또는 W(약함; Weak)로써 표시된다. 포워드 맵핑 테이블들 및 리버스 맵핑 테이블들의 어레이 표현은 단순히 설명을 위한 것이다. 이러한 것들은 타겟 성능, 자원 활용성 등을 기반으로 어레이들, 리스트들, 트리들, 해쉬 맵들 등과 같은 다른 데이터 구조들로써 구현될 수 있다.

[0036] SWAT 커맨드는 타겟을 위한 맵핑들을 약함(WEAK)으로 하여 포워드 맵핑 테이블(FMT) 및 리버스 맵핑 테이블(RMT) 모두의 논리적 페이지 넘버(LPN)의 두 개의 리스트들의 맵핑들을 아토믹한 방식으로 스왑한다. 약한 맵핑들은 적어도 가비지 콜렉션이 발생할 때 제거된다. 예시적인 실시 예가 이하의 도면에서 설명된다.

[0037] 도 5 내지 도 12는 리버스 맵핑 테이블(RMT), 물리적 페이지 넘버의 리스트, 및 포워드 맵핑 테이블(FMT)의 다양한 상태들을 보여주는 도면들이다. 도 5에 도시된 바와 같이, FMT(500)는 논리적 페이지 넘버들이 정렬된 논리적 페이지 넘버(LPN) 행, 물리적 페이지 넘버들이 정렬된 물리적 페이지 넘버(PPN) 행, 및 동일한 열의 논리적 페이지 넘버(LPN) 및 (물리적 페이지 넘버(PPN) 사이의 강함(S) 또는 약함(W) 맵핑들에 대한 값을 포함하는 맵핑 형태 행을 포함한다. 도 5의 실시 예는 FMT(500)가 LPN_A 및 PPN_X가 강한 맵핑을 포함하는 것으로 도시된 엔트리들을 포함하는 것을 가정한다. 실시 예는 SWAT 커맨드가 LPN_A 및 LPN_B에 대하여 이슈되고, 그 다음에 LPN_B 및 LPN_C에 대한 SWAT 커맨드가 이슈되는 것으로 가정한다. 제1 SWAT 커맨드는 FMT(500)에서 LPN_A의 포워드 맵핑 정보를 PPN_Y로 갱신하고, LPN_B를 PPN_X로 갱신한다. SWAT 커맨드는 RMP(502)의 PPN_X의 리버스 맵핑 정보를 LPN_B 및 LPN_A로 갱신한다. LPN_B 및 PPN_X 사이의 결과적인 맵핑은 볼드체(bold)인 RMT(502)의 LPN_B를 위한 엔트리에 의해 지칭되는 바와 같이 강함(strong)이고, LPN_A 및 PPN_X 사이의 결과적인 맵핑은 RMT(502)의 LPN_A를 위한 볼드체가 아닌(non-bold) 엔트리에 의해 지칭되는 바와 같이 약함(weak)이다. 약한 맵핑들은 즉시 제거되거나 또는 가비지 콜렉션 시간에 제거될 수 있다.

[0038] 이하의 경우들 중 하나는 약한 맵핑을 제거할 수 있다. 1) 가비지 콜렉션, 2) 쓰기 커맨드, 3) 트림 커맨드, 또는 3) 다른 SWAT 커맨드.

[0039] 약한 맵핑을 갖는 LPN에 대한 쓰기는 새로운 PPN에 대한 강한 맵핑을 생성할 수 있고, 약한 맵핑을 제거할 수

있다. 예를 들어, 도 6에 도시된 바와 같이, 쓰기 커맨드(WRITE_A)를 통한 LPN_A로의 쓰기는 새로운 PPN_Y를 할당하고, FMT(500)의 LPN_A의 포워드 맵핑을 PPN_Y로 갱신할 수 있다. 병렬적으로, RMT(502)는 PPN_X의 리버스 맵 리스트로부터 LPN_A를 위한 약한 맵핑을 제거하도록 업데이트되고, PPN_Y의 리버스 맵 리스트로의 LPN_A를 위한 새로운 리버스 맵핑을 추가한다.

[0040] PPN과의 강한 맵핑을 갖는 LPN으로의 쓰기는 새로운 PPN으로의 강한 맵핑을 생성할 수 있고, 이전 PPN에 대한 모든 약한 맵핑들을 제거할 수 있다. 예를 들어, 도 7에 도시된 바와 같이, 쓰기 커맨드(WRITE_B)를 통한 LPN_B로의 쓰기는 새로운 PPN_Y를 할당하고, FMT(500)의 LPN_B의 포워드 맵핑을 PPN_Y로 업데이트한다. 이는 LPN_A의 약한 맵핑을 이전 것으로 만들고, 이 맵핑은 즉시 또는 가비지 콜렉션 시간에 제거될 수 있다. 이와 병행하여, RMT(502)는 PPN_X의 리버스 맵 리스트로부터 LPN_B를 위한 엔트리를 제거하기 위하여 업데이트되고, PPN_Y의 리버스 맵 리스트에 LPN_B를 위한 새로운 엔트리를 추가하고, LPN_A를 위한 엔트리는 PPN_X의 리스트로부터 즉시 또는 가비지 콜렉션 시간에 제거될 수 있다.

[0041] 약한 맵핑을 갖는 LPN에 대한 TRIM은 약한 맵핑을 즉시 제거할 수 있다. 예를 들어, 도 8에 도시된 바와 같이, TRIM 커맨드(Ttrm_A)를 통한 LPN_A에 대한 TRIM은 PPN_X로의 LPN_A의 포워드 맵핑을 제거한다. 이와 병행하여, RMT(502)는 PPN_X의 리버스 맵 리스트로부터 LPN_A를 위한 엔터를 제거하기 위하여 업데이트된다.

[0042] PPN으로의 강한 맵핑을 갖는 LPN에 대한 TRIM은 PPN으로의 모든 맵핑들을 제거할 수 있다. 예를 들어, 도 9에 도시된 바와 같이, LPN_B로의 TRIM은 LPN_B의 포워드 맵핑을 제거한다. 이는 LPN_A의 약한 맵핑을 이전의 것으로 만들고, 이러한 맵핑은 즉시 또는 가비지 콜렉션 시간에 제거될 수 있다. 이와 병행하여, RMT(502)는 PPN_X의 리버스 맵 리스트로부터의 LPN_B를 위한 엔트리를 제거하기 위하여 갱신되고, LPN_A를 위한 엔트리는 PPN_X의 리버스 맵 리스트로부터 즉시 또는 가비지 콜렉션 시간에 제거된다.

[0043] 약한 맵핑을 포함하는 물리적 페이지 넘버로의 강한 맵핑을 포함하는, 기본 논리적 페이지 넘버(base LPN)로써의 LPN에 대한 SWAT는 PPN에 대한 모든 약한 맵핑들을 제거할 수 있다. 예를 들어, 도 10에 도시된 바와 같이, 기본으로써 LPN_B 및 타겟으로써 LPN_C를 갖는 SWAT는 PPN_X로의 LPN_C의 포워드 맵핑 엔트리를 약하게 만든다. PPN_X가 오직 약한 맵핑들을 포함하므로, 약한 맵핑들은 결과적으로 즉시 또는 가비지 콜렉션 시간에 제거된다. 이와 함께, LPN_C를 위한 엔트리가 RMT(502)의 PPN_X의 리버스 맵 리스트에 추가된다. RMT(502)에 포함된 PPN_X의 모든 약한 맵핑들은 즉시 또는 가비지 콜렉션 시간에 제거될 수 있다.

[0044] 다른 LPN으로의 강한 맵핑을 포함하는 PPN으로의 약한 맵핑을 포함하는 기본 LPN으로써의 LPN에 대한 SWAT는 LPN의 약한 맵핑을 제거할 수 있다. 약한 맵핑이 오래되지 않았으므로, 이러한 약한 맵핑은 강한 맵핑처럼 간주될 수 있다. 예를 들어, 도 11에 도시된 바와 같이, 기본으로써 LPN_A 및 타겟으로써 LPN_C를 포함하는 SWAT는 FMT(500)로부터 LPN_A의 약한 맵핑을 제거하고 PPN_Z로의 강한 맵핑을 생성한다. 이는 PPN_X로의 LPN_C의 맵핑을 약하게 만들고, 결과적으로, PPN_X는 강한 맵핑 및 약한 맵핑을 모두 갖는다. 이와 함께, RMT(502)는 LPN_C를 위한 약한 맵핑 엔트리를 PPN_X의 리버스 맵 리스트에 추가하고, LPN_A를 위한 강한 맵핑 엔트리를 PPN_Z의 리버스 맵 리스트에 추가하기 위하여 업데이트된다.

[0045] 다른 LPN으로의 약한 맵핑을 포함하는 PPN으로의 약한 맵핑을 포함하는 기본 LPN으로써 LPN에 대한 SWAT은 LPN의 약한 맵핑을 제거할 수 있다. 약한 맵핑이 오래되지 않았으므로, 이러한 약한 맵핑은 맵핑이 아닌 것(no-mapping)처럼 간주도리 수 있다. 예를 들어, 도 12에 도시된 바와 같이, 기본으로써 LPN_A 및 타겟으로써 LPN_B를 포함하는 SWAT은 FMT(500)로부터 LPN_A의 약한 맵핑을 제거하고, LPN_A로부터 PPN_Z로의 강한 맵핑을 생성한다. 이는 LPN_A 및 LPN_B 모두가 PPN_Z를 공유하도록 하고, 결과적으로, PPN_Z는 강한 맵핑 및 약한 맵핑을 모두 포함한다. 이와 함께, RMT(502)는 LPN_A를 위한 강한 맵핑 엔트리 및 LPN_B를 위한 약한 맵핑 엔트리가 PPN_Z의 리버스 맵 리스트에 추가되도록 업데이트된다.

[0046] 가비지 콜렉션 또는 트림 커맨드가 약한 맵핑을 제거하는 경우, LPN은 무효화(invalid)되고, 논리적 페이지를 위한 읽기는 0xFF와 같이 미리 정해진 값을 반환한다. 약한 맵핑을 포함하는 논리적 페이지에 대한 쓰기 동작은 새로운 물리적 페이지를 할당하고, 강한 PPN 맵핑을 생성한다. 기본 페이지가 현재 사용되지 않는 경우, (즉, 물리적 페이지 맵핑이 없는 경우) 타겟 페이지는 그것의 현재 맵핑을 유지하나, 맵핑은 약한 맵핑이 된다. SWAT 커맨드는 강한 맵핑, 약한 맵핑, 또는 맵핑이 없는 논리적 페이지와 함께 동작한다. 타겟을 위한 LPN이 강한 맵핑을 포함하는 반면에, 기본을 위한 LPN은 이러한 3가지 경우 중 어떤 것일 수 있다. 기본 LPN이 약한 맵핑을 포함하는 경우, 맵핑이 없는 것으로 간주된다. 물리적 페이지가 최대 N 약한 맵핑들(기본적으로, N은 1)을 포함할 수 있다. 이하에서, 좀 더 상세하게 설명된다.

[0047] 예를 들어, 도 4에서, SWAT 커맨드는 LPN의 기본 리스트들(예를 들어, LPN4, LPN5) 및 LPN의 타겟 리스트들(예를 들어, LPN25, LPN25)와 같은 두 개의 리스트들에 대하여 동작한다. 본 실시 예에서, 각 리스트들은 두 개의 페이지들을 포함한다. 이러한 실시 예에서, LPN4 및 LPN5는 기본 리스트에 포함되고, 원래(originally) PPN0 및 PPN1 각각으로의 강한 맵핑을 포함한다. LPN24 및 LPN25는 타겟 리스트에 포함되고, 원래 PPN11 및 PPN12 각각으로의 강한 맵핑을 포함한다. 이러한 두 개의 리스트들에 대한 SWAT 커맨드를 실행하는 것은 LPN4 및 PPN11 사이의 강한 맵핑 및 LPN5 및 PPN12 사이의 다른 강한 맵핑을 생성하고, LPN24 및 PPN0 사이의 약한 맵핑 및 LPN25 및 PPN1 사이의 다른 약한 맵핑을 생성한다. (도 4의 점선으로 도시됨.)

[0048] 물리적 페이지가 어떤 LPN으로의 강한 맵핑을 포함하지 않는 경우, 물리적 페이지는 가비지 콜렉션의 자격이 있다. SWAT 커맨드 동작이 완료되는 것은 타겟 리스트에서 LPN을 위한 약한 맵핑을 생성한다. 결과적으로, 가비지 콜렉션은 약한 LPN24 및 LPN25 맵핑들을 제거하여 PPN0 및 PPN1 모두를 반환한다. 논리적 페이지들(예를 들어, LPN24 및 LPN25)은 가지비 콜렉션이 발생되기 전에 읽어질 수 있다. 논리적 페이지가 기입된 경우, 약한 맵핑은 제거되고, 새로운 물리적 페이지로의 강한 맵핑이 생성된다. 맵핑은 변환은 아토믹하게 수행된다.

[0049] 도 13은 사용되지 않는 페이지에 대한 SWAT 커맨드의 예를 보여준다. 도 13에서, 기본 리스트의 LPN4 및 LPN5는 사용되지 않는(unuesed 또는 empty) 논리적 페이지들이고, 타겟 리스트의 LPN24 및 LPN25는 PPN11 및 PPN12로의 강한 맵핑들을 포함한다. 상술된 실시 예와 같이, 이러한 두 개의 리스트들에 대한 SWAT 커맨드를 완료하는 것은 LPN4 및 LPN11 사이의 강한 맵핑 및 LPN5 및 PPN12 사이의 다른 강한 맵핑을 생성한다. LPN4 및 LPN5가 사용되지 않기 때문에, LPN24 및 LPN25 그들의 원리 맵핑을 유지하나, 맵핑 강도는 강함에서 약함으로 변하고, 니는 SAWT 동작의 결과인 것을 가리킨다.

[0050] 맵핑이 없는 기본 LPN으로써 LPN에 대한 SWAT는 LPN을 위한 새로운 유효한 약함 맵핑들을 생성한다. 예를 들어, 도 14에 도시된 바와 같이, 기본으로써 LPN_A 및 타겟으로써 LPN_B에 대한 SWAT는 PPN_X로의 LPN_A의 새로운 약한 맵핑을 FMT(500)에 생성한다. 이는 LPN_A 및 LPN_B가 PPN_X를 공유하도록 하고, 결과적으로, PPN_X는 강한 맵핑 및 약한 맵핑 모두를 포함한다. 이와 함께, RMT(502)는 LPN_B를 위한 강한 맵핑 엔트리 및 LPN_A에 대한 약한 맵핑 엔트리가 PPN_X의 리버스 맵 리스트에 추가되도록 갱신된다.

[0051] 물리적 페이지가 강한 LPN 맵핑을 포함하지 않는 경우, 물리적 페이지는 가비지 콜렉션의 대상이 될 수 있다. 그러므로, 강한 맵핑들이 PPN11 및 PPN12를 위하여 존재하므로, 공유된 물리적 페이지들은 가비지 콜렉션에서 남는다. 결과적으로, 강한 LPN4 및 LPN5 맵핑들이 제거되거나 또는 가비지 콜렉셩이 페이지들을 재활용하지 않는 한, LPN24 및 LPN25는 PPN11 및 PPN12를 액세스하기 위하여 각각 사용될 수 있다.

[0052] 가비지 콜렉션이 공유된 물리적 페이지를 재배치한 경우, 이에 따라 강한 맵핑 및 약한 맵핑들 모두는 이동해야 된다.

[0053] 도 15는 반복되는 SWAT 커맨드들의 예를 보여준다. 반복된 SWAT 동작들이 도시된 바와 발생하는 경우, 물리적 페이지(예를 들어, PPN11)는 복수의 다른 약한 LPN 맵핑들(예를 들어, LPN4 및 LPN24)을 포함할 수 있다. 기본적으로, 물리적 페이지는 하나의 강한 맵핑(예를 들어, LPN3 및 PPN11 사이의 맵핑) 및 하나의 선택적인 약한 맵핑을 포함할 수 있다. 그러나 하나의 물리적 페이지가 지원할 수 있는 미리 정해진 파라미터에 의해 정의된 최대 값은 구현에 의존(implementation-dependent)될 수 있다.

[0054] 예시적인 실시 예들에 따르면, SWAT 커맨드 및 API는 현존하는 기술들에 대하여 다양한 이점들을 제공한다. SWAT 커맨드는 OS의 어떠한 변형도 요구하지 않고, 일부 펌웨어 변형을 통해 SSD 장치들의 모든 형태들에 적용될 수 있다. SWAT 커맨드는 애플리케이션으로부터의 디스크 쓰기 횟수를 감소시킴으로써 SSD의 내구성을 현저하게 향상시킬 수 있다. SWAT 커맨드는 사용되지 않는 공간을 미리 반환함으로써 시스템의 성능을 향상시킬 수 있다. 더욱이, SWAT 커맨드는 애플리케이션들, 특히, 다중 버전 동시 제어에 대한 애플리케이션의 성능상의 현저한 이점을 제공할 수 있다.

[0055] 이하에서, SWAT 커맨드의 일부 실시 예들이 더욱 상세하게 설명된다. 파일이 애플리케이션(20)을 통해 발생될 때, 운영 체제(22) 또는 파일 시스템(24) 중 어느 하나는 장치 드라이버 API(26)를 호출한다. 장치 드라이버 API(26)는 고속 스토리지(18)로 SWAT 커맨드를 발행할 수 있다.

[0056] 일 실시 예에서, 커맨드는 새로운 파일 LPN(410)의 타겟 리스트 및 이전 파일 LPN(406)의 기본 리스트인 LPN 리스트들의 쌍을 정의할 수 있다. 고속 스토리지(18)의 플래시 변환 계층(34)(FTL; Flash Translation Layer)은 이전 파일 LPN(406)의 리스트 및 새로운 파일 LPN(410)의 리스트를 수신하고, 정해진 순서에 따라 LPN의 쌍의 LPN의 맵핑을 아토믹하게 재맵핑할 수 있다. 상술된 바와 같이, SWAT 커맨드는 아토믹한 동작을 수행할 수

있다.

[0057] SWAT 커맨드는 SATA, SAS, PCIe, eMMC, UFS 및 제조사 커맨드(vendor specific command)와 같은 스토리지 프로토콜의 어떤 형태로든지 구현될 수 있다. SWAT 커맨드들의 일시 예들 및 사상은 본 발명이 특정 실시 예들의 용어를 설명하는 이하의 API 의사 코드(pseudo-code)를 참조하여 좀 더 잘 이해될 것이다. 의사 코드는 특정 코드 언어를 나타 내거나 또는 그것에 따르거나 또는 단순히, 시스템의 동작을 좀 더 형식적인 용어로 나타내는 것을 의미하지 않는다. 이는 명확성을 위하여 제공되는 것이며, 본 발명이 이에 한정되는 것은 아니다. 본 발명 및 본 발명의 기술적 사상은 방법들, 장치들, 본문에 특정하게 기재되지 않은 코드들의 다양한 형태의 애플리케이션을 포함할 수 있다.

[0058] [SWAP API]

[0059] #define SI_LBA_PER_PAGE 16 // 8KB page

[0060] #define SI_MAX_SWAT_PAGE 64 // 64 pages

[0061] #define SI_MAX_WEAK_MAPPING 1 // maximum number of weak mappings

[0062] #define SI_MAP_STRONG 1

[0063] #define SI_MAP_WEAK 2

[0064] /*

[0065] * DESCRIPTION

[0066] * See SWAT command for details

[0067] * PARAMETER

[0068] * base_page: the start LPN of base page(s); base_page(s) are strongly mapped to the PPN(s) of target page(s), if LPN is not within the range of device, SI_ERROR_INVALID_PAGE is returned

[0069] * target_page: the start LPN number of target page(s);

[0070] *target_page(s) are weakly mapped to the PPN(s) of base_page(s), if LPN is not within the range of device, SI_ERROR_INVALID_PAGE is returned

[0071] *target_page must have strong mapping(s), otherwise return SI_ERROR_VOLATILE_PAGE

[0072] * page_cnt: # of continuous LPNs to be SWATed; the maximum number of LPNs is defined as SI_MAX_SWAT_PAGE, if this is larger than SI_MAX_SWAT_PAGE, operation fails and SI_ERROR_MAX_SWAT_PAGE is returned

[0073] *

[0074] * RETURN

[0075] * SI_SUCCESS: SWAT operation is done successfully

[0076] * SI_ERROR_INVALID_PAGE: LPN value is not valid (out of range of device)

[0077] * SI_ERROR_VOLATILE_PAGE: a logical page that has only a weak mapping cannot be used

[0078] * SI_ERROR_OVERLAP_PAGE: the LPN range of base_page and target_page is overlapped

[0079] * SI_ERROR_MAX_SWAT_PAGE: the number of logical pages exceeds SI_MAX_SWAT_PAGE

[0080] * SI_ERROR_ATOMIC: an error occurs during the operation so all changes are rolled back

[0081] * SI_ERROR_MAX_WEAK_MAPPING: the number of weak mappings for a physical page exceeds the SI_MAX_WEAK_MAPPING

[0082] SI_ERROR si_swat (SI_PAGE base_page, SI_PAGE target_page, SI_PAGE_COUNT page_cnt)

[0083] {

[0084] // the number of pages must be smaller than SI_MAX_SWAT_PAGE

[0085] if page_cnt > SI_MAX_SWAT_PAGE

[0086] return SI_ERROR_MAX_SWAT_PAGE;

[0087] // the page range of base and target must be valid

[0088] if !isValid(base_page, page_cnt) or !isValid(target_page, page_cnt)

[0089] return SI_ERROR_INVALID_PAGE;

[0090] // two ranges must not be overlapped

[0091] if isOverlapped(base_page, target_page, page_cnt)

[0092] return SI_ERROR_OVERLAP_PAGE;

[0093] // target range must not have any weak mappings

[0094] // any weak mappings in the base range are trimmed, thogh if hasWeakMapping(target_page, page_cnt)

[0095] return SI_ERROR_VOLATILE_PAGE;

[0096] for i = 0 to page_cnt - 1

[0097] // swap reverse mapping table entries

[0098] if forward_map_table[base_page+i] is not NIL

[0099] removeRMT(reverse_map_table, forward_map_table[base_page+i], base_page+i);

[0100] removeRMT(reverse_map_table, forward_map_table[target_page+i], target_page+i);

[0101] addRMT(reverse_map_table, forward_map_table[target_page+i], base_page+i, SI_MAP_STRONG);

[0102] if forward_map_table[base_page+i] has a strong mapping or

[0103] reverse_map_table[forward_map_table[base_page+i] has a strong mapping

[0104] addRMT(reverse_map_table, forward_map_table[base_page+i], target_page+i, SI_MAP_WEAK);

[0105] else

[0106] addRMT(reverse_map_table, forward_map_table[target_page_i], target_page+i, SI_MAP_WEAK);

[0107] // swap forward mapping table entries

[0108] if forward_map_table[base_page+i] is not NIL

[0109] tmp = forward_map_table[base_page+i];

[0110] else

[0111] tmp = forward_map_table[target_page+i];

[0112] setFMT(forward_map_table, base_page+i, forward_map_table[target_page+i], SI_MAP_STRONG);

[0113] setFMT(forward_map_table, target_page+i, tmp, SI_MAP_WEAK);

[0114] return SI_SUCCESS;

[0115] }

[0116] RemoveRMT(reverse_map_table, ppn, lpn)

[0117] {

[0118] Remove the entry for lpn from the reverse list for ppn in the reverse_map_table

[0119] }

[0120] addRMT(reverse_map_table, ppn, lpn, mapping_type)

[0121] {

[0122] Add lpn to the list for ppn in the reverse_map_table &

[0123] set the type of the mapping

[0124] }

[0125] setFMT(forward_map_table, lpn, ppn, type)

[0126] {

[0127] Map lpn to ppn in the forward_map_table &

[0128] set the type of the mapping

[0129] }

[0130] Example 1)

[0131] // swat a page starting from 0 with one starting from 100,

[0132] // the page of LPN 0 is mapped to the physical page of LPN 100

[0133] If (si_swat(0, 100, 1) != SI_SUCCESS) {

[0134] // error

[0135] }

[0136] Example 2)

[0137] // swat an extent starting from 0 with one starting from 100,

[0138] // the extent of LPN 0 is mapped to the physical pages of the extent of LPN 1000

[0139] If (si_swat(0, 1000, SI_MAX_SWAT_PAGE) != SI_SUCCESS) {

[0140] // error;

[0141] }

[0142] typedef struct _si_extent {

[0143] SI_PAGE lpn;

[0144] SI_PAGE_COUNT count;

[0145] } si_extent;

[0146] /*

[0147] * PARAMETER

[0148] * base_page: an array of (LPN, page_count) pairs, base_extent(s) are strongly mapped to the physical pages for target_extent(s); if any LPN is not within the range of device, SI_ERROR_INVALID_LBA is returned

[0149] * base_cnt: # of extents, the maximum number of pages is defined as SI_MAX_SWAT_PAGE, if the total number of pages in the list is larger than SI_MAX_SWAT_PAGE, SI_ERROR_MAX_SWAT_PAGE is returned

[0150] * subsitute_page: an array of (LPN, page_count) pairs, target_page(s) are weakly mapped to the

physical pages for base_extent(s), if any LPN is not within the range of device, SI_ERROR_INVALID_LBA is returned

[0151] * target_cnt: # of extents, the maximum number of pages is defined as SI_MAX_SWAT_PAGE, if the total number of pages in the list is larger than SI_MAX_SWAT_PAGE, SI_ERROR_MAX_SWAT_PAGE is returned

[0152] * RETURN

[0153] * SI_SUCCESS: SWAT operation is done successfully

[0154] * SI_ERROR_EXTENT_MISMATCH: the number of page count in extent is not matched

[0155] * SI_ERROR_INVALID_PAGE: LPN value is not valid (out of range of device)

[0156] * SI_ERROR_VOLATILE_PAGE: a logical page that has only a weak mapping cannot be used

[0157] * SI_ERROR_OVERLAP_PAGE: the LPN range of base_page and target_page is overlapped

[0158] * SI_ERROR_MAX_SWAT_PAGE: the total number of logical pages in the list exceeds SI_MAX_SWAT_PAGE

[0159] * SI_ERROR_ATOMIC: an error occurs during the swap operation so all changes are rolled bak

[0160] * SI_ERROR_MAX_WEAK_MAPPING: the number of weak mappings for a physical page exceeds the SI_MAX_WEAK_MAPPING

[0161] */

[0162] SI_ERROR si_swat_extent (si_extent **base_extents,

[0163] SI_EXTENT_COUNT

[0164] base_cnt, si_extent **target_extents,

[0165] SI_EXTENT_COUNT target_cnt);

[0166] Example 3)

[0167] si_extent base_extents[3] = {

[0168] {0, 1},

[0169] {2, 2},

[0170] {4, 7}

[0171] };

[0172] si_extent target_extents[2] = {

[0173] {100, 2},

[0174] {110, 8}

[0175] };

[0176] // total number of pages to be swat is equal

[0177] if (si_swat_extent(&base_extents, 3, &target_extents, 2) != SI_SUCCESS) {

[0178] // error;

[0179] }

[0180] 본 발명은 앞서 설명된 실시 예들에 따라 설명되었으며, 본 발명의 기술적 사상 범위 내에서 실시 예들은 변형될 수 있다. 예를 들어, 예시적인 실시 예는 하드웨어, 소프트웨어, 프로그램 명령어를 포함하는 컴퓨터 판독가능 매체, 또는 그것들의 조합을 사용하여 구현될 수 있다. 본 발명에 따라 기입된 소프트웨어는 메모리, 하드 디스크, 또는 CD/DVD-ROM과 같은 컴퓨터 판독 가능 매체에 저장되고, 프로세스에 의해 실행된다. 따라서, 다양한 변형들은 첨부된 특허청구범위의 사상과 범위 내에서 당업자에 의해 구현될 수 있다.

부호의 설명

[0181] 10: 시스템

12: 컴퓨터

14: 프로세서

16: 메모리

18: 고속 스토리지

20: 애플리케이션

22: 운영체제

24: 파일 시스템

26: 장치 드라이버 API

28: 파일들

30 논리적 페이지 넘버

32: 물리적 페이지 넘버

34: 플래시 변환 계층

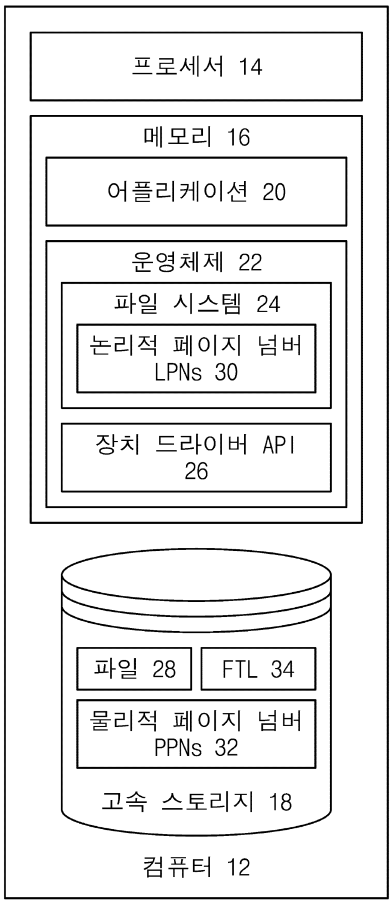
500: 포워드 맵핑 테이블(FMT)

502: 리버스 맵핑 테이블(RMT)

도면

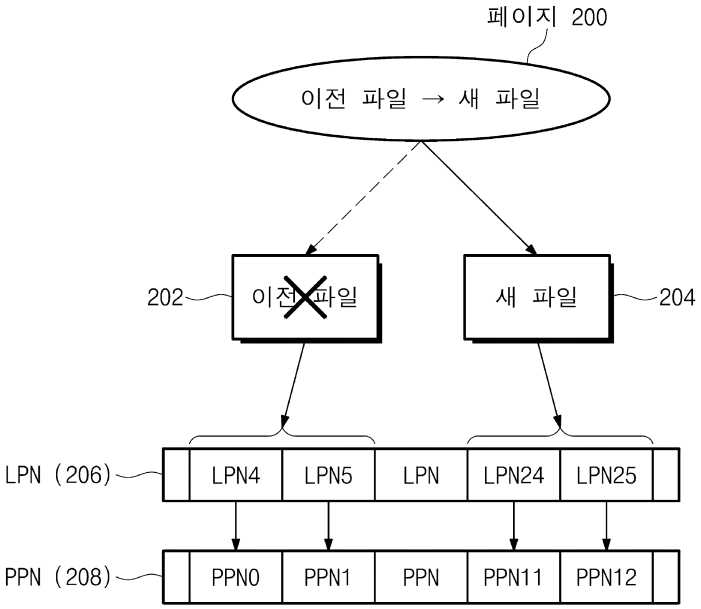
도면1

10

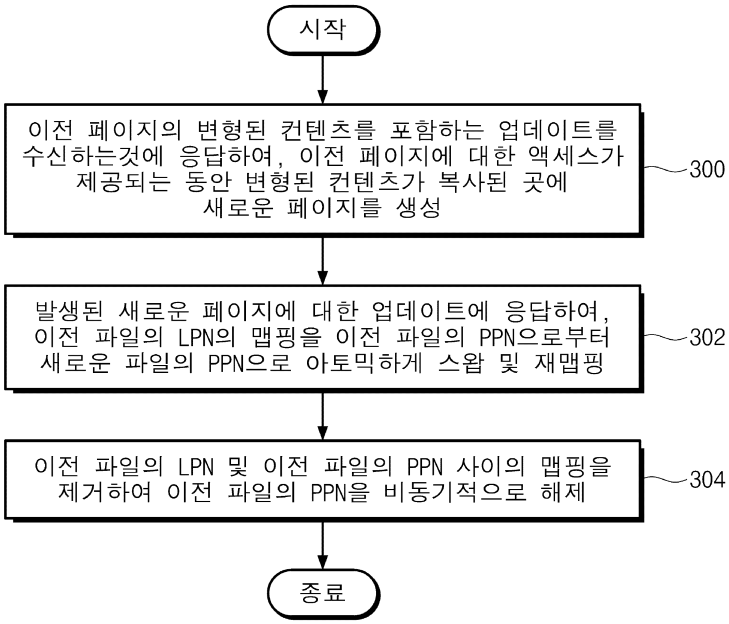

프로세서 14
메모리 16
어플리케이션 20
운영체제 22
파일 시스템 24
논리적 페이지 넘버
LPNs 30
장치 드라이버 API
26
파일 28
FTL 34
물리적 페이지 넘버
PPNs 32
고속 스토리지 18
컴퓨터 12

도면2

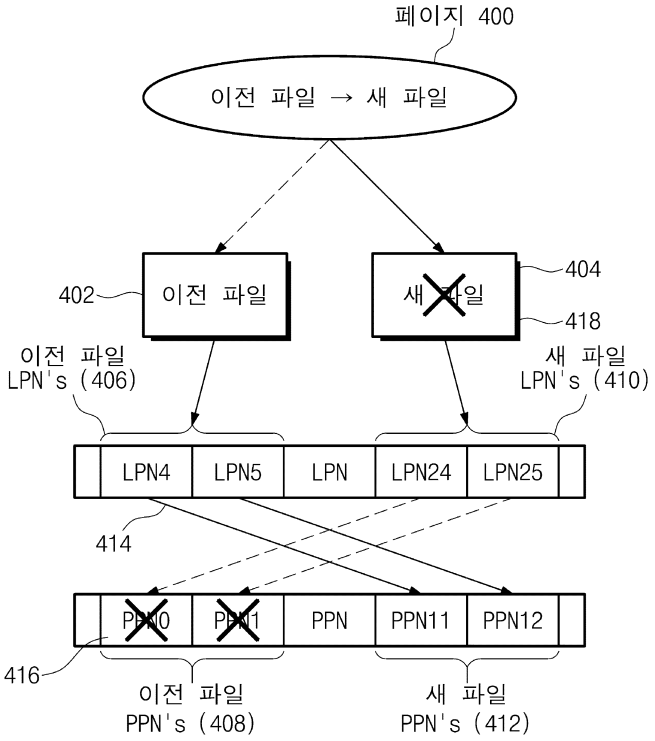
(종래 기술)


페이지 200
이전 파일 → 새 파일
202
이전 파일
새 파일
204
LPN (206)
LPN4
LPN5
LPN
LPN24
LPN25
PPN (208)
PPN0
PPN1
PPN
PPN11
PPN12

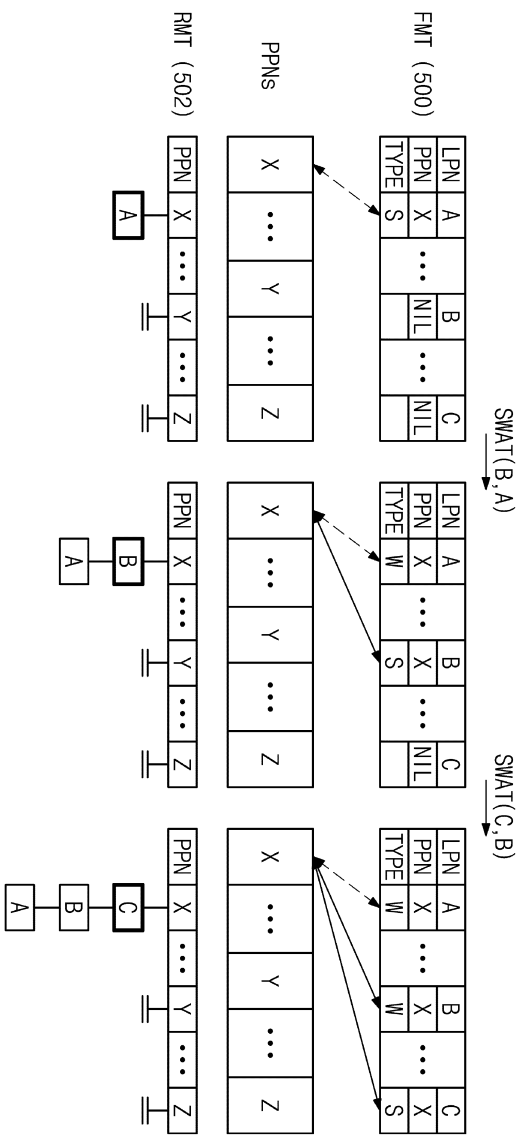
도면3


시작
이전 페이지의 변형된 컨텐츠를 포함하는 업데이트를 수신하는것에 응답하여, 이전 페이지에 대한 액세스가 제공되는 동안 변형된 컨텐츠가 복사된 곳에 새로운 페이지를 생성
300
발생된 새로운 페이지에 대한 업데이트에 응답하여, 이전 파일의 LPN의 맵핑을 이전 파일의 PPN으로부터 새로운 파일의 PPN으로 아토믹하게 스왑 및 재맵핑
302
이전 파일의 LPN 및 이전 파일의 PPN 사이의 맵핑을 제거하여 이전 파일의 PPN을 비동기적으로 해제
304
종료

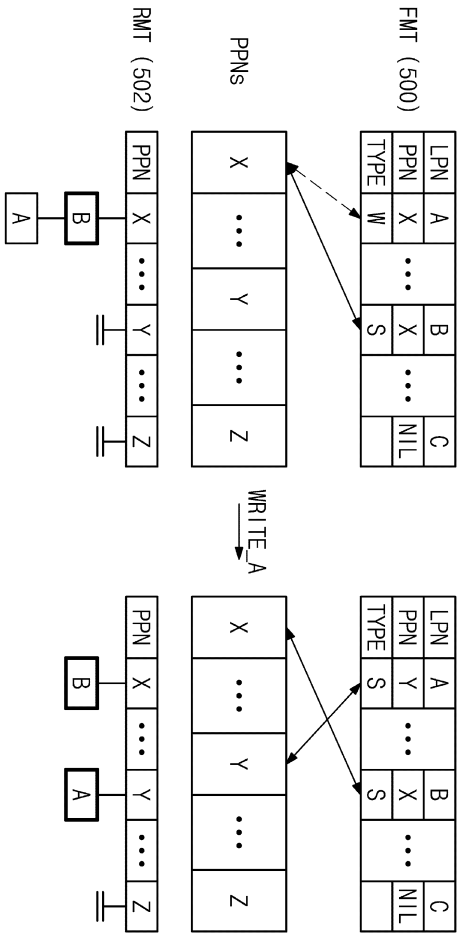
도면4


페이지 400
이전 파일 → 새 파일
402
이전 파일
404
새 파일
418
이전 파일
LPN's (406)
새 파일
LPN's (410)
LPN4
LPN5
LPN
LPN24
LPN25
414
PPN0
PPN1
PPN
PPN11
PPN12
416
이전 파일
PPN's (408)
새 파일
PPN's (412)

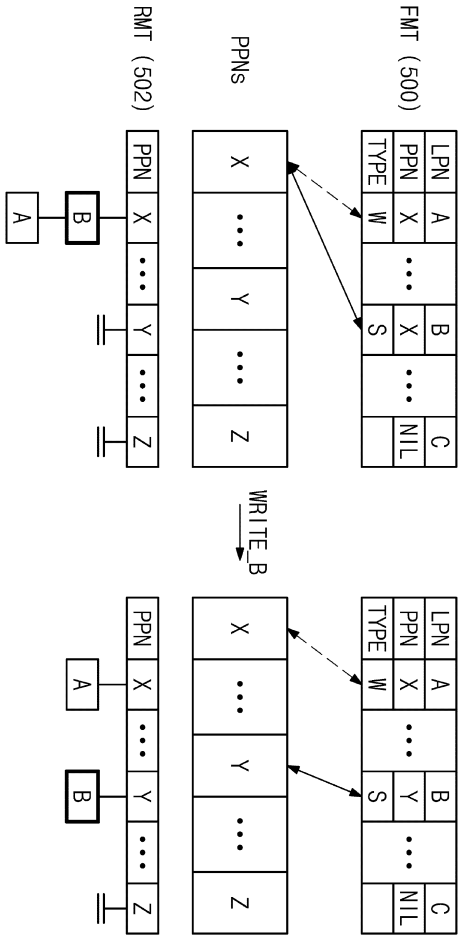
도면5



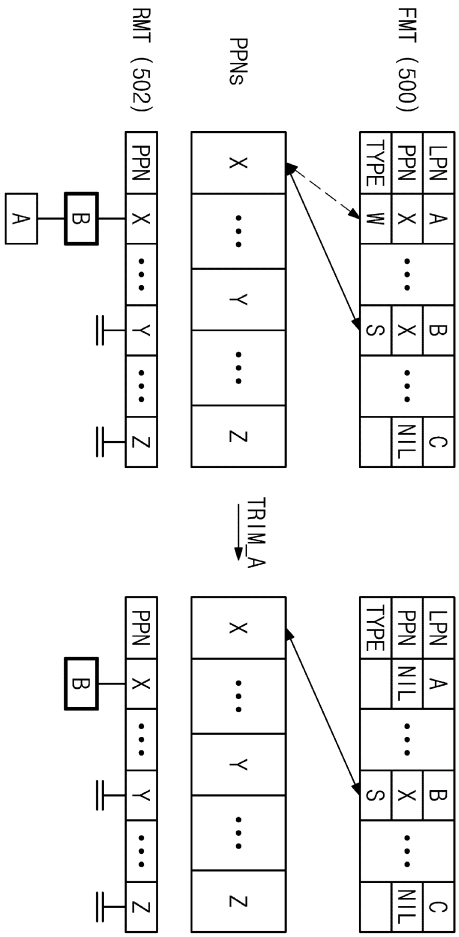
도면6



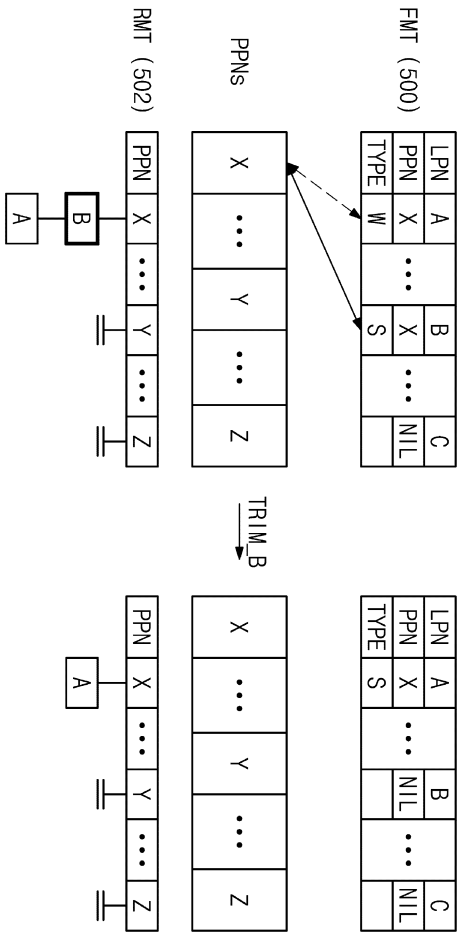
도면7



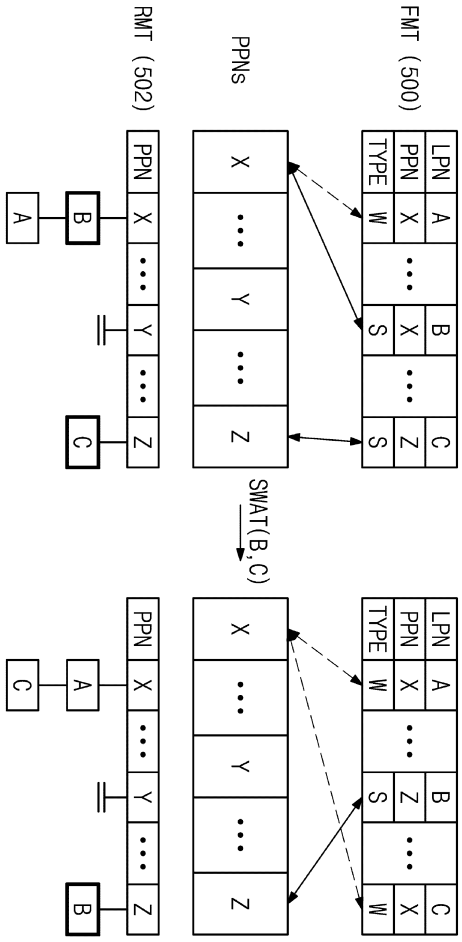
도면8



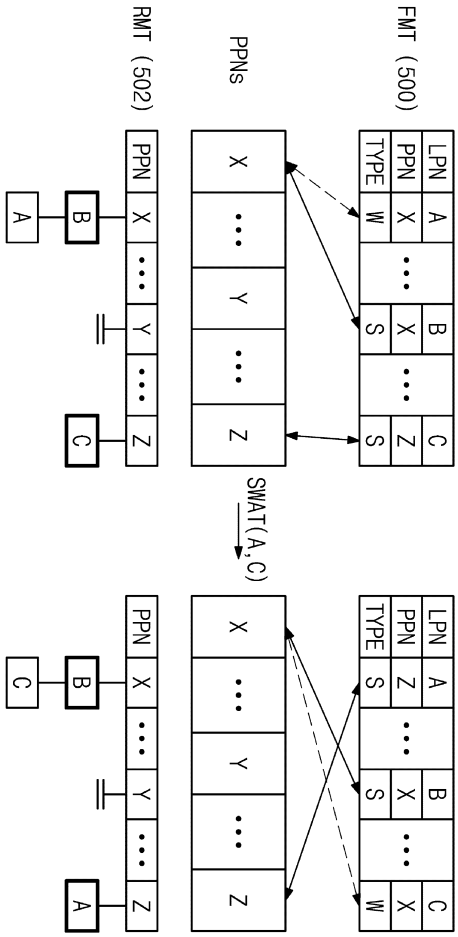
도면9



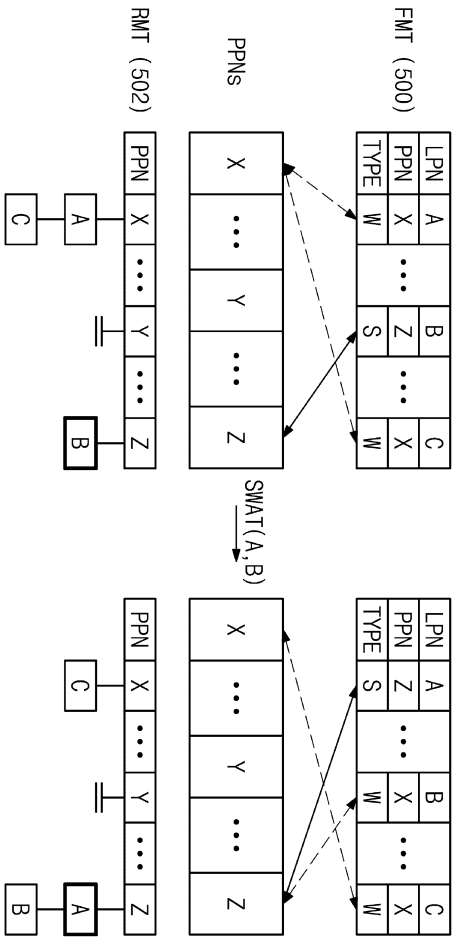
도면10



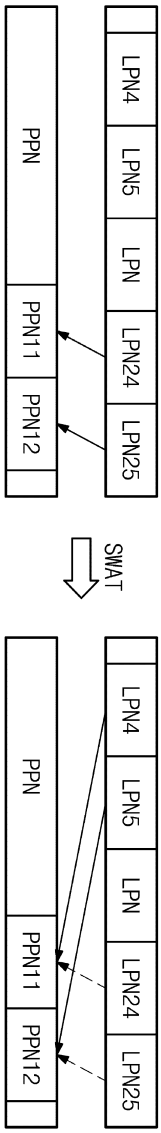
도면11



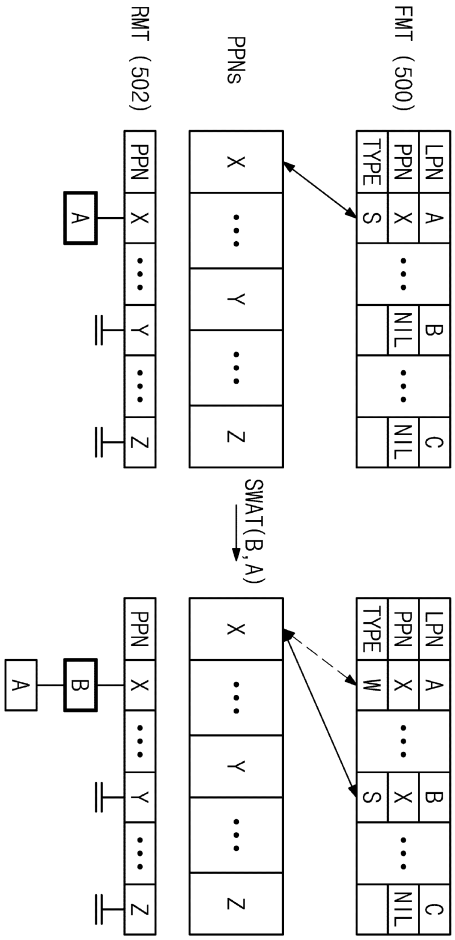
도면12



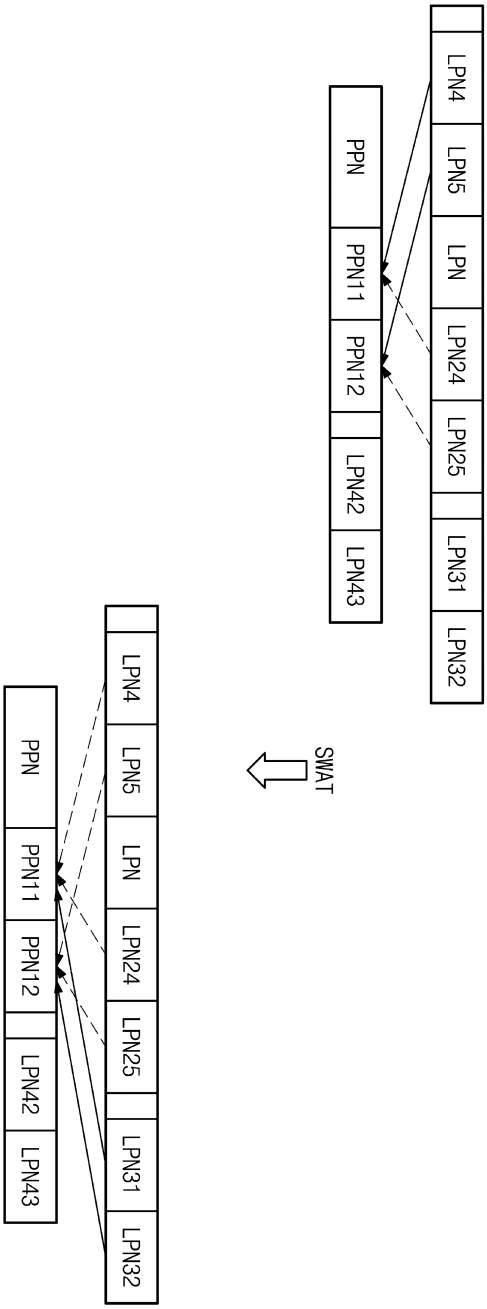
도면13


PPN
PPN11
PPN12
LPN4
LPN5
LPN
LPN24
LPN25
SWAT
PPN
PPN11
PPN12
LPN4
LPN5
LPN
LPN24
LPN25

도면14



도면15



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 청구범위

【보정세부항목】 청구항 7

【변경전】

시스템에 있어서,

프로세서 및 메모리를 포함하는 컴퓨터; 및

상기 프로세서 및 상기 메모리의 외부에 위치한 스토리지 장치를 포함하고,

파일 변환 계층(FTL; file translation layer) 맵은 파일 시스템에 의해 유지되는 논리 페이지 넘버들(LPN; logical page number) 및 상기 스토리지 장치에 의해 유지되는 물리 페이지 넘버들(PPN; physical page

number) 사이의 FTL에 의해 유지되고,

상기 스토리지 장치는:

상기 외부 프로세서로부터 제1 커맨드를 수신하고, 상기 제1 커맨드는 제1 파일로의 액세스가 유지되는 동안, 상기 제1 파일의 변형된 컨텐츠 및 상기 변형된 컨텐츠가 복사된 제2 파일의 생성을 포함하는 업데이트를 수신한 애플리케이션에 응답하여 전송되고, 상기 제1 커맨드의 파라미터들은 상기 제1 파일에 대응하는 제1 리스트의 LPN들 및 상기 제2 파일에 대응하는 제2 리스트의 LPN들을 포함하고, 상기 제1 리스트의 LPN들은 상기 스토리지 장치 상의 상기 제1 파일의 저장 위치를 나타내는 제1 리스트의 PPN들로 맵핑되고, 상기 제2 리스트의 LPN들은 상기 스토리지 상의 상기 제2 파일의 저장 위치를 나타내는 제2 리스트의 PPN들로 맵핑되고;

상기 제1 리스트의 LPN들이 상기 제2 리스트의 PPN들로 맵핑되도록 상기 제1 리스트의 LPN들을 아토믹하게 재맵핑(atomically remapping)하고, 상기 제1 리스트의 LPN들의 각 LPN에 대하여, 상기 FTL 맵에서, 상기 제1 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 생성되고;

상기 제1 리스트의 PPN들로의 상기 제1 리스트의 LPN들의 맵핑을 비동기적으로(asynchronously) 제거(trimming)하고, 상기 제1 리스트의 LPN들의 모든 LPN에 대하여, 상기 제1 리스트의 LPN들의 LPN 및 상기 제1 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되고;

상기 제2 리스트의 PPN들로의 제2 리스트의 LPN들의 맵핑이 언맵(unmap)되고, 상기 제2 리스트의 LPN들의 모든 LPN들에 대하여, 상기 제2 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되도록 구성된 시스템.

【변경후】

시스템에 있어서,

프로세서 및 메모리를 포함하는 컴퓨터; 및

상기 프로세서 및 상기 메모리의 외부에 위치한 스토리지 장치를 포함하고,

파일 변환 계층(FTL; file translation layer) 맵은 파일 시스템에 의해 유지되는 논리 페이지 넘버들(LPN; logical page number) 및 상기 스토리지 장치에 의해 유지되는 물리 페이지 넘버들(PPN; physical page number) 사이의 FTL에 의해 유지되고,

상기 스토리지 장치는:

상기 프로세서로부터 제1 커맨드를 수신하고, 상기 제1 커맨드는 제1 파일로의 액세스가 유지되는 동안, 상기 제1 파일의 변형된 컨텐츠 및 상기 변형된 컨텐츠가 복사된 제2 파일의 생성을 포함하는 업데이트를 수신한 애플리케이션에 응답하여 전송되고, 상기 제1 커맨드의 파라미터들은 상기 제1 파일에 대응하는 제1 리스트의 LPN들 및 상기 제2 파일에 대응하는 제2 리스트의 LPN들을 포함하고, 상기 제1 리스트의 LPN들은 상기 스토리지 장치 상의 상기 제1 파일의 저장 위치를 나타내는 제1 리스트의 PPN들로 맵핑되고, 상기 제2 리스트의 LPN들은 상기 스토리지 상의 상기 제2 파일의 저장 위치를 나타내는 제2 리스트의 PPN들로 맵핑되고;

상기 제1 리스트의 LPN들이 상기 제2 리스트의 PPN들로 맵핑되도록 상기 제1 리스트의 LPN들을 아토믹하게 재맵핑(atomically remapping)하고, 상기 제1 리스트의 LPN들의 각 LPN에 대하여, 상기 FTL 맵에서, 상기 제1 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 생성되고;

상기 제1 리스트의 PPN들로의 상기 제1 리스트의 LPN들의 맵핑을 비동기적으로(asynchronously) 제거(trimming)하고, 상기 제1 리스트의 LPN들의 모든 LPN에 대하여, 상기 제1 리스트의 LPN들의 LPN 및 상기 제1 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되고;

상기 제2 리스트의 PPN들로의 제2 리스트의 LPN들의 맵핑이 언맵(unmap)되고, 상기 제2 리스트의 LPN들의 모든 LPN들에 대하여, 상기 제2 리스트의 LPN들의 LPN 및 상기 제2 리스트의 PPN들의 대응하는 PPN 사이의 맵핑이 상기 FTL 맵에서 삭제(remove)되도록 구성된 시스템.