(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년09월30일
(11) 등록번호 10-2449585
(24) 등록일자 2022년09월27일

(51) 국제특허분류(Int. Cl.)
G06F 12/02 (2018.01) *G06F 3/06* (2006.01)
(52) CPC특허분류
G06F 12/0253 (2013.01)
G06F 3/0616 (2013.01)
(21) 출원번호 10-2016-0180318
(22) 출원일자 2016년12월27일
심사청구일자 2021년12월21일
(65) 공개번호 10-2017-0085951
(43) 공개일자 2017년07월25일
(30) 우선권주장
62/279,655 2016년01월15일 미국(US)
15/086,020 2016년03월30일 미국(US)
(56) 선행기술조사문헌
US20100241790 A1
US20140059279 A1
US20150199138 A1
US20130318196 A1

(73) 특허권자
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
(72) 발명자
레자에이, 아라시
미국 노스캘리포니아주 27612 롤리 베이마르 드라이브 4504 207호
수리, 타메쉬
미국 캘리포니아주 95136 산호세 마블 아치 애비뉴 361
브레넌, 로버트
미국 캘리포니아주 95054 산타 클라라 밀 크릭 레인 586 201호
(74) 대리인
특허법인 고려

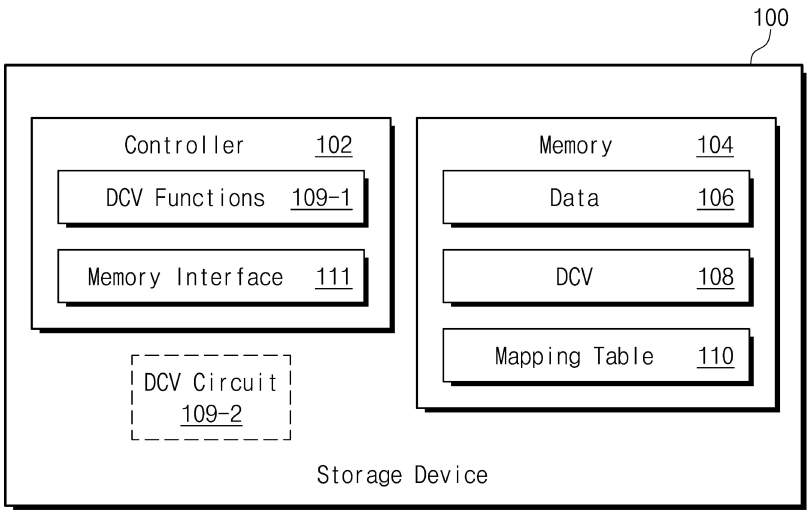
전체 청구항 수 : 총 20 항 심사관 : 남윤권

(54) 발명의 명칭 **버저닝 저장 장치 및 방법**

(57) 요 약

본 발명의 실시 예에 따른 저장 장치는 메모리와 컨트롤러를 포함한다. 상기 컨트롤러는 상기 메모리에 연결되는 메모리 인터페이스를 포함한다. 상기 컨트롤러는, 상기 메모리에 저장된 제 1 데이터와 연관된 어드레스에 쓸 쓰기 데이터와, 상기 메모리에 저장된 제 1 차등 압축된 값을 입력받고, 상기 쓰기 데이터와 상기 제 1 데이터에 근거하여 제 2 차등 압축된 값을 계산하고, 상기 메모리에 상기 제 2 차등 압축된 값을 저장하고, 상기 제 1 차등 압축된 값을 대신하여 상기 제 2 차등 압축된 값을 참조하기 위하여, 상기 어드레스의 연관성을 변경할 수 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 3/0659 (2013.01)
G06F 3/0679 (2013.01)
G06F 2212/1024 (2013.01)

명 세 서

청구범위

청구항 1

메모리; 및

상기 메모리에 연결되는 메모리 인터페이스를 포함하는 컨트롤러를 포함하되,

상기 컨트롤러는:

상기 메모리에 저장된 제 1 데이터와 연관된 어드레스에 쓸 쓰기 데이터와, 상기 메모리에 저장된 제 1 차등 압축된 값(DCV; differentially compressed value)을 입력받되, 상기 제 1 차등 압축된 값은 상기 제 1 데이터의 해시 함수(hash function) 및 압축 함수(compression function)에 기초하여 결정되고,

상기 쓰기 데이터와 상기 제 1 데이터에 기초하여 제 2 차등 압축된 값을 계산하되, 상기 제 2 차등 압축된 값은 상기 제 1 데이터 및 상기 쓰기 데이터의 해시 함수 및 압축 함수에 기초하여 결정되고,

상기 메모리에 상기 제 2 차등 압축된 값을 저장하고, 그리고

상기 제 1 차등 압축된 값을 대신하여 상기 제 2 차등 압축된 값을 참조하도록 상기 어드레스의 연관성을 변경하는 저장 장치.

청구항 2

제 1 항에 있어서,

상기 컨트롤러는:

상기 어드레스와 연관된 읽기 요청을 입력받고,

상기 제 1 데이터를 읽고,

상기 제 2 차등 압축된 값을 읽고,

상기 제 1 데이터 및 제 2 차등 압축된 값을 결합하여 제 2 데이터를 생성하고, 그리고

상기 제 2 데이터를 가지고 상기 읽기 요청에 응답하도록 더 구성되는 저장 장치.

청구항 3

제 1 항에 있어서,

상기 컨트롤러는:

상기 어드레스와 연관된 읽기 요청을 입력받고,

상기 제 1 데이터를 읽고, 그리고

상기 제 1 데이터를 가지고 상기 읽기 요청에 응답하도록 더 구성되는 저장 장치.

청구항 4

제 1 항에 있어서,

상기 컨트롤러는:

상기 어드레스와 상기 제 1 차등 압축된 값의 연관성을 유지하고,

상기 어드레스와 연관된 읽기 요청을 입력받고,

상기 제 1 데이터를 읽고,

상기 제 1 차등 압축된 값을 읽고,

상기 제 1 데이터와 상기 제 1 차등 압축된 값을 결합하여 제3 데이터를 생성하고, 그리고

상기 제 3 데이터를 가지고 상기 읽기 요청에 응답하도록 더 구성되는 저장 장치.

청구항 5

제 1 항에 있어서,

상기 어드레스는 논리적 어드레스로서 참조되고,

상기 컨트롤러는:

상기 제 1 데이터를 읽고,

상기 제 2 차등 압축된 값을 읽고,

상기 제 1 데이터와 상기 제 2 차등 압축된 데이터를 결합하여 제2 데이터를 생성하고,

상기 제 2 데이터를 상기 메모리 내의 물리적 어드레스에 저장하고, 그리고

상기 제 2 데이터를 저장하는 상기 물리적 어드레스를 참조하도록 상기 논리적 어드레스의 연관성을 업데이트하도록 더 구성되는 저장 장치.

청구항 6

제 1 항에 있어서,

상기 컨트롤러는 복수의 엔트리를 포함하는 상기 메모리 내의 맵핑 테이블을 유지하고, 각각의 엔트리는 논리적 어드레스, 물리적 어드레스, 그리고 차등 압축된 값의 표시를 포함하도록 더 구성되는 저장 장치.

청구항 7

제 1 항에 있어서,

상기 메모리는 불휘발성 메모리와 휘발성 메모리를 포함하고;

상기 컨트롤러는 상기 제 1 데이터를 상기 불휘발성 메모리에 저장하고 상기 제 2 차등 압축된 값을 상기 휘발성 메모리에 저장하도록 더 구성되는 저장 장치.

청구항 8

제 7 항에 있어서,

상기 컨트롤러는 상기 제 2 차등 압축된 값을 상기 휘발성 메모리로부터 상기 불휘발성 메모리로 전달하도록 더 구성되는 저장 장치.

청구항 9

제 1 항에 있어서,

상기 제 1 데이터는 순차 쓰기(in-place writes)를 수행할 수 없는 메모리의 적어도 한 부분에 저장되는 저장 장치.

청구항 10

제 1 항에 있어서,

상기 컨트롤러는,

상기 메모리에 저장된 데이터와 연관되지 않은 새로운 어드레스로 쓰기 요청을 입력 받고, 그리고

상기 쓰기 요청의 데이터를 상기 메모리에 쓰고, 상기 메모리에 쓰여진 데이터와 상기 새로운 어드레스의 연관성을 생성하는 저장 장치.

청구항 11

메모리에 저장된 제 1 데이터와 연관된 어드레스에 쓸 쓰기 데이터와, 상기 메모리에 저장된 제 1 차등 압축된 값을 입력 받는 단계;

상기 쓰기 데이터와 상기 제 1 데이터에 기초하여 제 2 차등 압축된 값을 계산하는 단계;

상기 메모리에 상기 제 2 차등 압축된 값을 저장하는 단계; 및

상기 제 1 차등 압축된 값을 대신하여 상기 제 2 차등 압축된 값을 참조하도록, 상기 어드레스의 연관성을 변경하는 단계를 포함하고,

상기 제 1 차등 압축된 값은 상기 제 1 데이터의 해시 함수 및 압축 함수에 기초하여 결정되고,

상기 제 2 차등 압축된 값은 상기 제 1 데이터 및 상기 쓰기 데이터의 해시 함수 및 압축 함수에 기초하여 결정되는 저장 장치의 동작 방법.

청구항 12

제 11 항에 있어서,

상기 어드레스와 연관된 읽기 요청을 입력 받는 단계;

상기 제 1 데이터를 읽는 단계;

상기 제 2 차등 압축된 값을 읽는 단계;

상기 제 1 데이터 및 제 2 차등 압축된 값을 결합하여 제 2 데이터를 생성하는 단계; 및

상기 제 2 데이터를 가지고 상기 읽기 요청에 응답하는 단계를 더 포함하는 저장 장치의 동작 방법.

청구항 13

제 11 항에 있어서,

상기 어드레스와 연관된 읽기 요청을 입력 받는 단계;

상기 제 1 데이터를 읽는 단계; 및

상기 제 1 데이터를 가지고 상기 읽기 요청에 응답하는 단계를 더 포함하는 저장 장치의 동작 방법.

청구항 14

제 11 항에 있어서,

상기 어드레스는 논리적 어드레스로서 참조되고;

상기 저장 장치의 동작 방법은,

상기 제 1 데이터를 읽는 단계;

상기 제 2 차등 압축된 값을 읽는 단계;

상기 제 1 데이터와 상기 제 2 차등 압축된 값을 결합하여 제 2 데이터를 생성하는 단계;

상기 제 2 데이터를 상기 메모리 내의 물리적 어드레스에 저장하는 단계; 및

상기 제 2 데이터를 저장하는 상기 물리적 어드레스를 참조하도록 상기 논리적 어드레스의 연관성을 업데이트하는 단계를 더 포함하는 저장 장치의 동작 방법.

청구항 15

제 11 항에 있어서,

복수의 엔트리를 포함하는 상기 메모리 내의 맵핑 테이블을 유지하고, 각각의 엔트리는 논리적 어드레스, 물리적 어드레스, 그리고 차등 압축된 값의 표시를 포함하는 단계를 더 포함하는 저장 장치의 동작 방법.

청구항 16

제 11 항에 있어서,

상기 제 1 데이터를 불휘발성 메모리에 저장하고 상기 제 2 차등 압축된 값을 휘발성 메모리에 저장하는 단계를 더 포함하는 저장 장치의 동작 방법.

청구항 17

제 16 항에 있어서,

상기 제 2 차등 압축된 값을 상기 휘발성 메모리로부터 상기 불휘발성 메모리로 전달하는 단계를 더 포함하는 저장 장치의 동작 방법.

청구항 18

통신 인터페이스; 및

상기 통신 인터페이스를 통해 메모리에 연결되는 프로세서를 포함하되,

상기 프로세서는:

상기 프로세서에 연결되는 저장 장치에 저장된 제 1 데이터와 연관된 어드레스에 쓸 쓰기 데이터와, 제 1 차등 압축된 값을 입력받되, 상기 제 1 차등 압축된 값은 상기 제 1 데이터의 해시 함수 및 압축 함수에 기초하여 결정되고,

상기 쓰기 데이터와 상기 제 1 데이터에 기초하여 제 2 차등 압축된 값을 계산하되, 상기 제 2 차등 압축된 값은 상기 제 1 데이터 및 상기 쓰기 데이터의 해시 함수 및 압축 함수에 기초하여 결정되고, 그리고

상기 제 1 차등 압축된 값을 대신하여 상기 제 2 차등 압축된 값을 참조하도록 상기 어드레스의 연관성을 변경하는 시스템.

청구항 19

제 18 항에 있어서,

상기 프로세서에 연결된 메모리를 더 포함하고;

상기 프로세서는 상기 메모리에 상기 제 2 차등 압축된 값을 저장하도록 구성되는 시스템.

청구항 20

제 18 항에 있어서,

상기 프로세서는 상기 저장 장치에 상기 제 2 차등 압축된 값을 저장하도록 구성되는 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 저장 장치에 관한 것으로, 더욱 상세하게는 버저닝 저장 장치들(versioning storage devices) 및 그 방법들에 관한 것이다.

배 경 기 술

[0002] 저장 장치들은 레이턴시에 영향을 주는 여러 방법으로 동작할 수 있다. 예를 들면, 데이터는 SSD에 페이지 단위로 쓰일 수 있다. 블록은 다수의 페이지로부터 이루어질 수 있다. 플래시 메모리는 블록 단위로 소거 동작이 수행될 수 있다. 만약 하나의 블록에 있는 여러 페이지가 더 이상 필요하지 않다면, 그 블록에 있는 다른 유효한 페이지들을 읽어서 다른 블록에 쓰고, 그 블록을 프리 업(free up) 상태로 만들 수 있다. 그런 다음에, 그 상태 블록을 소거할 수 있다. 이러한 프로세스를 가비지 컬렉션이라고 한다.

[0003] 가비지 컬렉션은 저장 장치의 레이턴시를 증가시킬 수 있다. 특히, SSD는 가비지 컬렉션을 수행하는 동안에 읽기 및/또는 쓰기 요청을 처리할 수 없다. 그 결과로, 들어오는 읽기/쓰기 요청들이, 가비지 컬렉션이 끝날 때까지, 지연될 수 있다.

[0004] 어떤 하드 디스크는 싱글드 마그네틱 레코딩(SMR; shingled magnetic recording)을 사용한다. SMR로 인해, 저장 매체의 트랙들이 오버랩될 수 있다. 트래픽에 저장된 데이터가 변경되고 그 트리픽이 다시 쓰일 때, 오버랩핑 트래픽들 또한 읽혀지고 다시 쓰여야 한다. 이 추가적인 동작들은 그것들이 수행되는 동안에 레이턴시를 일으킬 수 있다.

발명의 내용

해결하려는 과제

[0005] 본 발명은 상술한 기술적 과제를 해결하기 위해 제안된 것으로, 본 발명의 목적은 레이턴시 성능을 향상시키고 메모리 셀의 마모를 개선하여 수명을 증가하는 버저닝 저장 장치 및 방법을 제공하는 데 있다.

과제의 해결 수단

[0006] 본 발명의 실시 예에 따른 저장 장치는 메모리와 컨트롤러를 포함한다. 상기 컨트롤러는 상기 메모리에 연결되는 메모리 인터페이스를 포함한다. 상기 컨트롤러는, 상기 메모리에 저장된 제 1 데이터와 연관된 어드레스에 쓸 쓰기 데이터와, 상기 메모리에 저장된 제 1 차등 압축된 값을 입력받고, 상기 쓰기 데이터와 상기 제 1 데이터에 근거하여 제 2 차등 압축된 값을 계산하고, 상기 메모리에 상기 제 2 차등 압축된 값을 저장하고, 상기 제 1 차등 압축된 값을 대신하여 상기 제 2 차등 압축된 값을 참조하기 위하여, 상기 어드레스의 연관성을 변경할 수 있다.

[0007] 본 발명의 실시 예에 따른 저장 장치의 동작 방법은, 메모리에 저장된 제 1 데이터와 연관된 어드레스에 쓸 쓰기 데이터와, 상기 메모리에 저장된 제 1 차등 압축된 값을 입력받는 단계, 상기 쓰기 데이터와 상기 제 1 데이터에 근거하여 제 2 차등 압축된 값을 계산하는 단계, 상기 메모리에 상기 제 2 차등 압축된 값을 저장하는 단계, 및 상기 제 1 차등 압축된 값을 대신하여 상기 제 2 차등 압축된 값을 참조하기 위하여, 상기 어드레스의 연관성을 변경하는 단계를 포함한다.

[0008] 본 발명의 실시 예에 따른 시스템은 통신 인터페이스와 프로세서를 포함한다. 상기 프로세서는 상기 통신 인터페이스를 통해 메모리에 연결된다. 상기 프로세서는, 상기 프로세서에 연결되는 저장 장치에 저장된 제 1 데이터와 연관된 어드레스에 쓸 쓰기 데이터와, 제 1 차등 압축된 값을 입력받고, 상기 쓰기 데이터와 상기 제 1 데이터에 근거하여 제 2 차등 압축된 값을 계산하고, 상기 제 1 차등 압축된 값을 대신하여 상기 제 2 차등 압축된 값을 참조하기 위하여, 상기 어드레스의 연관성을 변경할 수 있다.

발명의 효과

[0009] 본 발명에 의하면, 저장 장치의 레이턴시 성능이 향상될 수 있다. 또한, 본 발명에 의하면, 저장 장치 내의 메모리 셀의 마모를 개선하여 수명을 증가시킬 수 있다.

도면의 간단한 설명

[0010] 도 1은 본 발명의 실시 예에 따른 저장 장치를 보여주는 블록도이다.

도 2a 내지 도 2e는 본 발명의 실시 예에 따른 저장 장치에 대한 쓰기를 보여주는 블록도이다.

도 3a 내지 도 3e는 본 발명의 실시에 따른 저장 장치로부터 읽기를 보여주는 블록도이다.

도 4a 및 도 4b는 본 발명의 실시 예에 따른 저장 장치들을 보여주는 블록도이다.

도 5는 본 발명의 실시 예에 따른 저장 장치 내의 페이지 사이즈들을 보여주는 블록도이다.

도 6은 본 발명의 실시 예에 따른 SSD를 보여주는 블록도이다.

도 7은 본 발명의 실시 예에 따른 버저닝 저장 시스템을 보여주는 블록도이다.

도 8은 본 발명의 실시 예에 따른 서버를 보여주는 블록도이다.

도 9는 본 발명의 실시 예에 따른 서버 시스템을 보여주는 블록도이다.

도 10은 본 발명의 실시 예에 따른 데이터 센터를 보여주는 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0011] 본 발명은 버저닝 저장 장치들 및 방법들에 관한 것이다. 후술되는 설명은 이 분야에 통상적인 지식을 가진 자(이하, 당업자)가 실시 예들을 실시할 수 있도록 제공된다. 실시 예들 및 일반적인 원칙들 및 특성들에 대한 다양한 변경들이 가능함은 명백하다. 실시 예들은 주로 구체적인 구현들에서 제공되는 구체적인 방법들 및 시스템들의 형태로 설명된다.

[0012] 그러나 방법들, 장치들, 및 시스템들은 다른 구현들에서 유효하게 동작할 수 있다. "실시 예", "일 실시 예", "다른 실시 예"와 같은 용어들은 다중의 실시 예들뿐 아니라 동일한 실시 예들을 참조할 수 있다. 실시 예들은 어떤 구성 요소들을 구비한 시스템들 그리고/또는 장치들을 참조하여 설명된다. 그러나 시스템들 및/또는 장치들은 도시된 것보다 많거나 또는 그보다 적은 구성 요소들을 포함할 수 있다. 그리고 본 발명의 기술적 사상을 벗어나지 않는 범위 내에서 구성 요소들의 다양한 배치 및 타입의 변경이 다양하게 실시될 수 있다. 실시 예들은 특정한 단계들을 구비한 구체적인 방법들을 참조하여 설명된다. 그러나 방법 및 시스템은 상이한 그리고/또는 추가적인 단계들 및 실시 예들과 일치하기 않는 상이한 순서들을 갖는 단계들을 구비한 다른 방법들에 따라 동작할 수 있다. 따라서, 실시 예들은 도시된 구체적인 실시 예들에 한정되지 않으며, 설명된 원리들 및 특성들과 일치하는 가장 넓은 범위에 따른다.

[0013] 실시 예들은 특정한 구성 요소들을 구비한 구체적인 시스템들을 참조하여 설명된다. 본 발명의 실시 예들이 다른 및/또는 추가적인 구성 요소들 및/또는 다른 특성들을 구비한 시스템들이나 장치들의 사용과 일관될 수 있다는 것은 당업자에게 자명하다. 또한, 당업자는 다른 방법들 및 시스템들이 다른 구조들과 일치함을 이해할 것이다. 또한, 당업자는 방법들 및 시스템들이 다중 원소들을 구비한 메모리 시스템 구조의 사용에 적용될 수 있음을 이해할 것이다.

[0014] 여기에서 사용되는 용어들, 특히 첨부된 청구범위에 사용되는 용어들은 열린 의미로 해석됨이 이해될 것이다. 예를 들어, "포함한다" 또는 "포함하는"의 용어는 "포함하지만 한정되지 않는"으로 해석되어야 한다. "구비한" 또는 "갖는"의 용어는 "적어도 구비한" 또는 "적어도 갖는"의 의미로 해석되어야 한다.

[0015] 도 1은 본 발명의 실시 예에 따른 저장 장치를 보여주는 블록도이다. 실시 예로서, 저장 장치(100)는 컨트롤러(102)와 메모리(104)를 포함한다. 컨트롤러(102)는 저장 장치(100)의 동작을 관리하도록 구성되는 회로이다. 그리고 컨트롤러(102)는 범용 목적 프로세서(general purpose processor), 디지털 신호 프로세서(DSP; digital signal processor), ASIC(application specific integrated circuit), 마이크로컨트롤러(microcontroller), 프로그램 가능한 로직 장치(programmable logic device), 이산 회로(discrete circuits), 그러한 장치의 조합 등과 같은 요소들(components)을 포함한다. 컨트롤러(102)는 레지스터들, 캐시 메모리, 프로세싱 코어들 등과 같은 내부 부분들(internal portions)을 포함할 수 있다. 그리고 컨트롤러(102)는 또한, 어드레스 및 데이터 버스 인터페이스들, 인터럽트 인터페이스들 등과 같은 외부 인터페이스들을 포함할 수 있다. 비록 단 하나의 컨트롤러(102)가 저장 장치(100) 내에 도시되어 있지만, 복수의 컨트롤러가 존재할 수 있다. 또한, 버퍼들, 메모리 인터페이스 회로들, 통신 인터페이스들 등과 같은 다른 인터페이스 장치들이 컨트롤러(102)를 내부 및 외부 요소들과 연결하는 저장 장치(100)의 일부분이 될 수 있다.

[0016] 다른 실시 예로서, 컨트롤러(102)는 저장 장치(100)가 통신할 수 있도록 하는 회로를 포함하는 통신 인터페이스를 포함할 수 있다. 예를 들면, 통신 인터페이스에는 universal serial bus (USB), small computer system interface (SCSI), peripheral component interconnect express (PCIe), serial attached SCSI (SAS), parallel ATA (PATA), serial ATA (SATA), NVM Express (NVMe), universal flash storage (UFS), Fiber channel, Ethernet, remote direct memory access (RDMA), Infiniband, or other interfaces 등이 포함될 수 있다. 그러한 통신 인터페이스들을 이용하여, 저장 장치(100)는 관련된 미디엄을 통해서 외부 장치들 및 시스템들과 통신하도록 구성될 수 있다. 다른 실시 예로서, 컨트롤러(102)는 통신 인터페이스를 거친 읽기 및 쓰기 요청들을 입력받도록 구성될 수 있다.

[0017] 메모리(104)는 데이터를 저장할 수 있는 어떤 장치이다. 여기에서, 하나의 메모리(104)가 저장 장치(100)를 위해 도시되어 있지만, 복수의 메모리들이 저장 장치(100) 내에 포함될 수 있다. 저장 장치(100)는 복수의 다른 메모리 타입들을 포함할 수 있다. 예로서, 메모리(104)에는, a dynamic random access memory (DRAM), DDR, DDR2, DDR3, DDR4과 같은 규격에 따른 a double data rate synchronous dynamic random access memory (DDR

SDRAM), static random access memory (SRAM), flash memory, spin-transfer torque magentoresistive random access memory (STT-MRAM), Phase-Change RAM, nanofloating gate memory (NFGM), or polymer random access memory (PoRAM), magnetic or optical media과 같은 non-volatile memory 등이 포함될 수 있다.

[0018] 메모리(104)는 데이터(106), DCVs(differentially compressed values, 108), 그리고 맵핑 테이블(110)을 저장하도록 구성된다. 아래에서 좀 더 자세하게 설명되겠지만, 메모리(104)는 다수의 메모리 장치들을 포함할 수 있다. 메모리(104)에 저장된 데이터는 다양한 방법으로 그러한 장치들 사이에 분포될 수 있다. 그러나 설명의 편의를 위해, 여기에서는 데이터는 하나의 메모리(104)에 저장되는 것으로 설명될 것이다.

[0019] 컨트롤러(102)는 메모리(104)에 연결되는 메모리 인터페이스(111)를 포함할 수 있다. 컨트롤러(102)는 메모리 인터페이스(111)를 통해 메모리(104)를 액세스하도록 구성될 수 있다. 메모리 인터페이스(111)는 명령, 어드레스, 및/또는 데이터 버스들을 위한 인터페이스들을 포함할 수 있다. 인터페이스들을 통해, 컨트롤러(102)와 메모리(104)는 통신할 수 있다. 메모리(104)가 컨트롤러(102)와 별도로 구성되는 것으로 설명되었지만, 다른 실시예에서는 캐시 메모리나 SRAM 등과 같은 메모리(104)의 일부분들이 컨트롤러(102)에 포함될 수도 있다. 컨트롤러(102)는 내부 통신 버스들을 포함할 수 있다. 프로세싱 코어들, 외부 통신 인터페이스, 캐시 메모리 등과 같은 내부 요소들은 내부 통신 버스들을 통해 통신할 수 있다.

[0020] 데이터(106)는 저장 장치(100)에 저장된 데이터를 나타낸다. 아래에서 좀 더 자세하게 설명되겠지만, DCV(108)는, 관련된 데이터(106)와 결합할 때, 저장 장치(100)에 저장된 현재 데이터를 나타내는 데이터를 의미한다. 다른 실시 예로서, DCV(108)는 대응하는 데이터(106)의 사이즈보다 적은 사이즈를 갖는 값이다. 예를 들면, 데이터(106)와 DCV(108)는 각각 다른 사이즈들을 갖는 페이지들로 저장될 수 있다. 예로서, 데이터(106)의 페이지는 8K 바이트의 사이즈를 가질 수 있다. 이와 대조적으로, 대응하는 DCV(108)를 위한 페이지의 사이즈는 4K 바이트일 수 있다. 특정 사이즈들이 예로서 사용될 수 있지만, 다른 실시 예에서는 사이즈들이 다를 수도 있다.

[0021] 컨트롤러(102)는 메모리(104)에 저장된 데이터(106)와 관련된 어드레스에 쓸 쓰기 데이터와 쓰기 요청을 입력받도록 구성될 수 있다. 맵핑 테이블(110)은 논리적 어드레스들, 물리적 어드레스들, DCV 어드레스들/값들 등과 같은 정보를 갖는 엔트리들을 포함하고, 어드레스들/값들 또는 다른 정보 사이의 연관성을 만들 수 있다. 맵핑 테이블(110)은 페이지, 블록, 또는 하이브리드 맵핑 정책들을 사용할 수 있다. 그러나 여기에서의 예들은 설명을 위해 블록 맵핑을 사용할 것이다.

[0022] 컨트롤러(102)는 맵핑 테이블(110)에 저장된 데이터를 사용해서, 물리적 어드레스, DCV(108), 또는 논리적 어드레스와 관련된 것들을 식별할 수 있다. 예를 들면, 통신 인터페이스를 통해 논리적 어드레스와 함께 읽기 또는 쓰기 요청을 입력받은 후에, 컨트롤러(102)는 맵핑 테이블(110)을 액세스해서 논리적 어드레스와 관련된 엔트리를 읽도록 구성될 수 있다. 실시 예로서, 컨트롤러(102)는 맵핑 테이블(110)을 저장하는 내부 캐시 메모리를 액세스하도록 구성될 수 있다. 그러나 다른 예로서, 컨트롤러(102)는 DRAM과 같은 외부 메모리를 액세스하도록 구성될 수도 있다.

[0023] 컨트롤러(102)는 논리적 어드레스와 연관된 물리적 어드레스에서 메모리(104)에 저장된 데이터(106)를 읽도록 구성될 수 있다. 컨트롤러(102)는 쓰기 요청에 포함된 쓰기 데이터 및 물리적 어드레스로부터 읽은 데이터에 근거하여 DCV(108)를 계산하도록 구성될 수 있다. 이 계산은 물리적 어드레스에 저장된 데이터(106) 및 들어오는 쓰기 데이터 사이의 차이에 기초하여 DCV(108)를 생성할 수 있다. 또한, 이 계산은 물리적 어드레스로부터 읽은 데이터 및/또는 쓰기 데이터보다 작은 사이즈를 갖는 DCV(108)를 생성할 수 있다.

[0024] DCV는 다양한 방법으로 계산될 수 있다. 실시 예로서, 컨트롤러(102)는 소프트웨어 또는 내부 회로 내에서 DCV를 계산하도록 구성될 수 있다. DCV 기능들(109-1)은 컨트롤러(102)의 이러한 동작을 나타낸다. 즉, 컨트롤러(102)가 쓰기 데이터 및 물리적 어드레스로부터 읽은 데이터(106)를 입력받은 후에, 컨트롤러(102)는 쓰기 데이터 및 메모리(104)로부터의 읽기 데이터를 사용하여 수학적 계산을 수행함으로, DCV를 발생하도록 구성될 수 있다. 다른 예로서, 컨트롤러(102)는 외부 회로를 사용하여 DCV를 계산할 수 있다. 예를 들면, 컨트롤러(102)는 쓰기 데이터 및 읽기 데이터를 DCV 회로(109-2)에 직접 연결되도록 할 수 있다. 그리고 컨트롤러(102)는 응답으로, DCV를 생성할 수 있다. DCV회로(109-2)는 연산 유닛들(arithmetic units), 룩업 테이블들(lookup tables), 입출력 버퍼들(input/output buffers) 등을 포함함으로, DCV를 계산하고 컨트롤러(102) 및/또는 메모리(104)와 인터페이스할 수 있다. DCV 회로(109-2)는 컨트롤러(102)의 DCV 기능들(109-1)에 대한 대체(alternative)나 결합(conjuction)으로 사용될 수 있다는 것을 가리키도록 점선으로 도시되어 있다.

[0025] DCV 회로(109-2)를 통해 직접 또는 간접적으로 컨트롤러(102)에 의해 수행되는 다양한 기능들은 DCV를 생성하도

록 사용될 수 있다. 실시 예로서, 기능은 간단한 뺄셈 동작(subtraction operation)일 수 있다. 다른 예로서, 그 기능은 보다 복잡한 해쉬 함수일 수 있다. 다른 예로서, 기록들은 비트들이 플립하는 것을 가리키도록 생성될 수 있다(records can be created indicating which bits have flipped). 실시 예로서, 그 기능은 "diff" 함수로서 언급될 수 있다. 특별한 실시 예로서, 그 기능은 DCV의 사이즈를 줄이도록 최적화될 수 있다. 실시 예로서, 그 기능은 차이의 사이즈를 줄일 수 있는 압축 함수(compression function)를 포함할 수 있다.

[0026] 컨트롤러(102)는 또한 DCV 회로(109-2)를 통해 직접 또는 간접적으로 DCV 함수의 역(inverse)인 함수를 수행하도록 구성될 수 있다. 역 DCV 함수는 소스 데이터 및 다른 데이터로부터 생성된 소스 데이터 및 DCV를 입력들로 사용해서, 다른 데이터를 재생산하는 함수일 수 있다. 따라서 데이터(106) 및 DCV(108)를 유지함으로서, 다른 데이터는 역 DCV 함수를 통해 이용할 수 있다.

[0027] 컨트롤러(102)는 메모리(104)에 계산된 DCV를 저장하도록 구성될 수 있다. 예를 들면, 컨트롤러(102)는 DCVs(108)와 함께 계산된 DCV를 메모리(104)에 저장할 수 있다. 그러나 아래에서 좀 더 자세하게 설명되겠지만, 다른 실시 예로서, DCV는 DCVs(108)와 함께 저장되기 전에, 메모리(104)의 하나 또는 그 이상의 다른 부분에 캐시될 수도 있다.

[0028] 컨트롤러(102)는 어드레스의 결합을 변화시켜서, 어드레스와 관련된 이전 DCV 대신에 계산된 DCV를 참조할 수 있다. 아래에서 좀 더 자세하게 설명되겠지만, 어드레스와 관련된 데이터(106)가 읽혀질 때, 새로운 DCV는 이전 DCV 대신에 액세스될 수 있다. 다른 예로서, 단지 하나의 DCV가 어드레스와 관련된 데이터를 나타내도록 유지될 수 있다.

[0029] 실시 예로서, 메모리(104)의 적어도 한 부분은 쓰기들의 관점에서 비대칭 성능(asymmetric performance)을 가질 수 있다. 예를 들면, 플래시 메모리 기반 저장 장치들은 in-place 쓰기들을 허용하지 않는다. 새로운 블록은 쓰기를 위해 할당되어야 한다. 그리고 다음 쓰기들(future writes)을 준비하기 위해, 이전 블록은 삭제되어야 한다. SMR(shingled magnetic recording)을 가진 저장 장치들에서, 다른 트랙과 겹치는 트랙에 쓰는 것은 겹치는 트랙(overlapping track)을 다시 쓰는 것을 포함할 수 있다. 아래에서 좀 더 자세하게 설명되겠지만, DCVs을 사용함으로, 비대칭 성능의 영향은 줄어들 수 있다.

[0030] 도 2a 내지 도 2e는 본 발명의 실시 예에 따른 저장 장치에 대한 쓰기를 설명하기 위한 개념도이다. 도 1의 저장 장치(100)가 예로서 사용될 것이다. 도 1 및 도 2a를 참조하면, 실시 예로서, 맵핑 테이블(210)은 메모리(104)에 저장된 맵핑 테이블(110)에 대응한다. 맵핑 테이블(210)은 다수의 엔트리들(211)을 포함한다. 각각의 엔트리(211)는 논리적 블록 어드레스(LBA), 물리적 블록 어드레스(PBA), 그리고 DCV의 표시(indication)를 위한 필드를 포함한다. 비록 특정 필드들이 예로서 사용되었지만, 다른 실시 예로서, 다른 필드들이 존재할 수 있고, 그 필드들이 다른 형태를 취할 수도 있다. 예를 들면, 여기에서, 논리적 및 물리적 어드레스들은 블록들과 관련될 수 있지만, 다른 실시 예에서는, 논리적 및 물리적 어드레스들이 블록 내의 페이지들 또는 메모리(104)의 다른 구조들과 관련될 수도 있다. 다른 예로서, 물리적 어드레스와 같은 싱글 어드레스(single address)는 현재 유일한 어드레스(the only address present)일 수 있다.

[0031] 맵핑 테이블(210)에서, 두 엔트리들(211-1, 211-2)은 이전부터 존재하던 엔트리들이다. 새로운 논리적 블록 어드레스에 쓰기 요청(202-1)이 입력될 때, 새로운 엔트리(211-3)가 컨트롤러(102)에 의해 생성된다. 이 예에서, 새로운 엔트리(211-3)는 논리적 블록 어드레스 1을 위한 것이다. 컨트롤러(102)는 논리적 블록 어드레스 1과 물리적 블록 어드레스(23)를 관련짓는다. 그러나 물리적 블록 어드레스와 관련된 유일한 데이터는 입력되는 데이터 D이다. 예를 들면, 데이터 D는 새로운 파일과 관련되고, DCV는 계산되지 않는다. 즉, 어떤 유효 데이터도 데이터 메모리(204-1) 내의 물리적 어드레스(23)에 존재하지 않는다. DCV 필드에 있는 'x' 표기는 DCV가 존재하지 않거나 유효하지 않다는 표시를 나타낸다. 실시 예로서, 플래그(flag)는 DCV가 존재하는지 혹인 유효한지를 나타낼 수 있다. 다른 예로서, 특정 어드레스/필드 값은 DCV가 존재하지 않거나 유효하지 않다는 표시로서 정의될 수 있다. 또 다른 예로서, DCV 필드는 엔트리(211-3)의 일부가 아닐 수도 있다. DCV가 존재하지 않거나 유효하지 않다는 표시는 다른 형태를 취할 수도 있다.

[0032] 쓰기 요청(202-1)이 새로운 엔트리(211-3)와 관련되기 때문에, 컨트롤러(102)는 데이터 D를 데이터 메모리(204-1)의 물리적 블록 어드레스(23)에 저장하도록 구성된다. 데이터 메모리(204-1)는 데이터(106)가 저장되는 메모리(104)의 부분을 나타낸다. 따라서, 새로운 유효 엔트리(211-3)는 데이터 메모리(204-1)에 저장된 데이터를 참조하여 생성된다. 데이터 D가 데이터 메모리(204-1)에 쓰여질 것으로 설명되었지만, 다른 실시 예로서, 다양한 버퍼링 또는 캐싱 등이 데이터 D를 데이터 메모리(204-1)에 전달하기 전에 쓰는 파트(part)로 동작될 수 있다.

[0033] 도 1 및 도 2b를 참조하면, 실시 예로서, 새로운 쓰기 요청(202-2)이 입력될 수 있다. 다시, 쓰기 요청(202-2)이 논리적 블록 어드레스 1에 직접적으로 향한다(directed towards). 그러나 엔트리(211-3)가 존재함으로, 데이터는 이미 데이터 메모리(204-1)의 관련된 물리적 블록 어드레스에 저장되어 있다. 즉, 도 2a의 데이터 D가 저장되어 있다. 따라서, 이 쓰기 요청(202-2)은 논리적 블록 어드레스 1에 저장된 데이터를 새로운 데이터 D'로 업데이트될 것이다.

[0034] 데이터 D의 본래 쓰기와 대조적으로, 도 2a와 관련하여 위에서 설명한 바와 같이, 데이터 D'는 데이터 메모리(204-1)의 물리적 블록 어드레스(23)에 쓰이지 않는다. 컨트롤러(102)는 엔트리(211)가 맵핑 테이블(210)에 존재하는지를 결정하도록 구성된다. 실시 예로서, 컨트롤러(102)는 쓰기 요청의 논리적 블록 어드레스와 존재하는 엔트리들(211)의 논리적 어드레스들을 비교한다. 만약 매치(match)가 발견되면, 이전 쓰기가 발생하고 데이터가 데이터 메모리(204-1)에 존재한다. 컨트롤러(102)는 관련되는 물리적 블록 어드레스에 저장된 데이터 D를 읽도록 구성된다. 여기에서, 컨트롤러(102)는 물리적 블록 어드레스(23)로부터 읽는다. 도 2a에 도시된 바와 같이 이전에 쓰인 데이터 D는 데이터 메모리(204-1)로부터 읽혀진다.

[0035] 쓰기 요청(202-2)의 새로운 데이터 D' 및 데이터 메모리(204-1)로부터의 존재하는 데이터 D는 DCV 기능(208-1)의 입력들로 사용된다. DCV 기능(208-1)은, 위에서 설명한 바와 같이, DCV를 계산하기 위하여, 컨트롤러(102)에 의해 수행되는 동작을 나타낸다. DCV 기능(208-1)은 새로운 데이터 D' 및 존재하는 데이터 D에 근거하여 DCV'를 생성하도록 구성된다. 여기에서, DCV'는 데이터 D와 결합되고 데이터 D'를 생성할 수 있다는 것을 나타내기 위해 어포스트로피가 붙어있다.

[0036] 컨트롤러(102)는 DCV 메모리(204-2) 내에 새로운 DCV'를 저장하도록 구성될 수 있다. 실시 예로서, 컨트롤러(102)는 DCV 어드레스를 맵핑 테이블(210)에 있는 엔트리(211-3)의 DCV 필드에 저장하도록 구성될 수 있다. 여기에서, 0x12의 값은 그 값이 새롭거나 변했다는 것을 가리키도록 밑줄로 되어 있다. 다른 예로서, DCV 메모리(204-2)는 사용되지 않을 수 있고, DCV'는 엔트리(211-3)의 DCV 필드에 저장될 수 있다. 예를 들면, DCVs는 상대적으로 작을 수 있고, 그래서 엔트리들(211)에 DCVs를 저장할 수 있는 메모리의 추가적인 양은 상대적으로 작을 수 있다. 그럼에도 불구하고, 엔트리(211-3)는 현재 논리적 블록 어드레스, 물리적 블록 어드레스, 그리고 DCV를 결합(association)할 수 있다.

[0037] 도 1 및 도 2c를 참조하면, 실시 예로서, 다른 쓰기 요청(202-3)이 컨트롤러(102)에 의해 입력된다. 이 쓰기 요청(202-3)은 논리적 블록 어드레스 1에 새로운 데이터 D''를 쓰기 위한 요청이다. 다시, 컨트롤러(102)는 맵핑 테이블(210)에서 매치(match)를 찾으려고 시도하도록 구성될 수 있다. 여기에서, 엔트리(211-3)이 존재한다. 도 2b와 마찬가지로, 컨트롤러(102)는 엔트리(211-3)에 표시된 물리적 블록 어드레스(23)로부터 데이터 D를 읽도록 구성된다. 데이터 D는 데이터 메모리(204-1)로부터 출력된다.

[0038] 데이터 메모리(204-1)로부터 출력된, 쓰기 요청(202-3)의 새로운 데이터 D''와 존재하는 데이터 D는 DCV 기능(208-1)에 대한 입력들로 사용될 수 있다. DCV 기능(208-1)은 새로운 데이터 D''와 존재하는 데이터 D에 근거하여 새로운 DCV''를 생성하도록 구성된다. 특히, DCV''의 생성은 도 2b에서 쓰인 데이터 D'의 중간 상태(intermediate state)에 관여하지는 않는다. 데이터 D가 물리적 블록 어드레스(23)에서 데이터 메모리(204-1)에 저장된 데이터이기 때문에, 관심의 차이(difference of interest)는 데이터 D와 쓰일 데이터 D'' 사이의 차이이다. 컨트롤러(102)는 결과인 DCV''를 DCV 메모리(204-2)에 저장하도록 구성될 수 있다. 컨트롤러(102)는 엔트리(211-3)의 DCV 필드에 있는 DCV 어드레스를 사용하여, 미리 존재하는 DCV'를 새로운 DCV''로 겹쳐 쓰도록 구성될 수 있다. 따라서 현재 엔트리(211-3)에 있는 DCV 필드는 새로운 DCV''를 참조로 한다(references).

[0039] 도 1 및 도 2d를 참조하면, 동작들은 도 2c에서 설명된 바와 유사할 수 있다. 그러나 실시 예로서, DCV''를 DCV 메모리(204-2)에 쓰면 새로운 DCV 어드레스가 만들어 질 수 있다. 예를 들면, 만약 DCV 메모리(204-2)가 플래시 메모리로 구현되면, 새로운 DCV''는 DCV'를 저장하는 페이지와 다른, DCV 메모리(204-2)에 있는 페이지에 쓰여질 수 있다. 새로운 DCV 어드레스는 DCV 메모리(204-2)에 있는 어드레스이고, DCV''는 거기에 저장된다. 엔트리(211-3)에서, 새로운 어드레스는 0x21로 표시되고, DCV 필드에서 변화를 나타내도록 밑줄로 되어 있다.

[0040] 비록 DCV 메모리(204-2)에 있는 동일 어드레스에서 메모리 위치(memory location)를 업데이트 하거나 엔트리(211-3)를 새로운 어드레스로 업데이트 하는 것이, 논리적 블록 어드레스와 DCV의 결합(association)을 바꾸는 방법의 예로서 사용하여 새로운 DCV를 참조함에도 불구하고, 그 결합에 있어서의 변화는 다를 수 있다. 예를 들면, 실시 예로서, 컨트롤러(102)는 DCV 필드와 같은, 맵핑 테이블에 DCV''를 저장하도록 구성될 수 있다. 엔트리(211-3)에 저장된 DCV'는 DCV''로 대체될 수 있다.

[0041] 도 1 및 도 2e를 참조하면, 실시 예로서, 컨트롤러(102)는 DCV 플러시 액세스(202-4)를 입력받도록 구성될 수 있다. 여기에서, DCV 플러시 액세스(202-4)는 위에서 설명한 도면들과 마찬가지로, 논리적 블록 어드레스 1을 가리킨다. 맵핑 테이블(210)은 도 2d의 맵핑 테이블(210)과 마찬가지로, 초기 상태(210-1)에 있다. 다시, 컨트롤러(102)는 엔트리(211-3)를 액세스하고 데이터 메모리(204-1)에 있는 관련된 물리적 블록 어드레스(23)으로부터 데이터 D를 읽도록 구성될 수 있다. 그러나 컨트롤러(102)는 DCV 필드를 사용하여, DCV 메모리(204-2)에 저장된 DCV''를 액세스하도록 구성될 수 있다. 여기에서, 컨트롤러(102)는 DCV 어드레스를 사용하여 DCV 메모리(204-2)를 액세스하도록 구성될 수 있다.

[0042] 컨트롤러(102)는 DCV'' 및 데이터 D를 역 DCV 기능(208-2)로 제공하도록 구성될 수 있다. 역 DCV 기능(208-2)은 위에서 설명한 바와 같이, 데이터와 DCV를 조합해서 데이터 업데이트된 버전을 만들어내기 위한 함수를 나타낸다. 여기에서, 역 DCV 기능(208-2)은 데이터 D와 DCV''를 사용해서 데이터 D''를 다시 생성할 수 있다. 컨트롤러(102)는 데이터 D를 대신해서 데이터 메모리(204-1)에 데이터 D''를 저장하도록 구성될 수 있다. 맵핑 테이블(210)은 상태 210-2로 업데이트 될 수 있다. 여기에서, 엔트리(211-3)는 유효 DCV가 존재하지 않는다는 것을 나타내도록 업데이트될 수 있다. 따라서 논리적 블록 어드레스 1에 연속된 쓰기가, 도 2b에서 설명한 바와 같이, 컨트롤러(102)에 의해 다루어질 수 있다.

[0043] 위에서 설명한 동작들의 결과로서, 빈번한 변경된 데이터는 비대칭 성능을 갖는 메모리들(104)을 갖는 저장 장치들의 성능에 영향을 줄여줄 수 있다. 예를 들면, 지속적으로 변경된 데이터는 전체 데이터 셋의 5% 미만을 포함할 수 있다. 예로서 200GB를 사용한다면, 단지 1% 또는 2GB가 지속적으로 업데이트 될 수 있다. DCVs의 보다 작은 사이즈는 쓰여질 데이터의 양을 줄여준다. 실시 예로서, 대부분의 DCV는 전체 데이터 블록의 사이즈의 20%일 수 있다(a majority of DCV may be on the order of 20% of a size of an entire block of data).

[0044] 나머지의 대부분은 여전히 데이터의 사이즈의 50%보다 적을 수 있다. 따라서 200GB의 예에서, DCVs의 400MB 내지는 1GB가 쓰여질 수 있다. 줄어든 사이즈는 공간을 효율적으로 만들고 낭비를 줄일 수 있다. 특히, 2GB의 새롭게 삭제된 블록들을 사용하여 업데이트 하는 데 현재는 400MB를 사용할 수 있다. 저장 장치의 주어진 용량에 대하여, 새롭게 삭제된 블록들에 대한 요구를 줄이는 것은 가비지 컬럭션의 수행 빈도를 줄여주고 매체에 대한 마모도를 줄일 수 있다.

[0045] 도 3a 내지 도 3e는 본 발명의 실시 예에 따른 저장 장치로부터 읽는 동작을 설명하기 위한 개념도이다. 도 1의 저장 장치(100)가 예로서 사용될 것이다. 도 1 및 도 3a를 참조하면, 도 2a 내지 도 2e의 구성 요소들과 유사한 구성 요소들의 설명은 간략화를 위해 생략될 것이다. 도 1 및 도 3a를 참조하면, 맵핑 테이블(310)은, 도 2a에서 설명한 바와 같이, 데이터가 쓰여진 후의 상태를 나타낸다. 즉, 데이터 D는 데이터 메모리(304-1) 내에 저장되고, 엔트리 (311-3)는 맵핑 테이블(310)에 추가된다. 그러나 엔트리(311-3)는 DCV가 존재하지 않거나 유효하지 않다는 표시를 포함할 수 있다.

[0046] 컨트롤러(102)는 읽기 요청(302)을 입력받도록 구성될 수 있다. 여기에서, 읽기 요청(302)은 논리적 블록 어드레스 1을 액세스하도록 구성될 수 있다. 응답으로, 컨트롤러(102)는 맵핑 테이블(310)을 액세스하거나 물리적 블록 어드레스(23)를 읽도록 구성될 수 있다. 물리적 블록 어드레스를 사용하여, 컨트롤러(102)는 데이터 메모리(304-1)로부터 데이터 D를 읽도록 구성될 수 있다. 컨트롤러(102)는 데이터 D와 함께 읽기 요청(302)에 응답하도록 구성될 수 있다. 특히, 엔트리(311-3)는 DCV가 존재하지 않거나 유효하지 않다는 표시를 포함하기 때문에, 컨트롤러(102)는 변경(modification)없이 데이터 D에 반응할 수 있다.

[0047] 도 1 및 도 3b를 참조하면, 실시 예로서, 맵핑 테이블(310)이 도 2b에서 설명된 바와 같이 데이터가 쓰여진 후의 상태이기 때문에, 읽기 요청(302)은 컨트롤러(102)에 의해 입력받을 수 있다. 즉, 데이터 D는 초기에 데이터 메모리(304-1)에 쓰여지고, DCV 메모리(304-2)에 저장되어 있기 때문에 업데이트된 데이터 D'가 쓰여질 수 있다.

[0048] 따라서 컨트롤러(102)는 다시 읽기 요청(302)을 입력받도록 구성되고, 이에 응답하여, 물리적 블록 어드레스(23)를 액세스함으로 데이터 메모리(304-1)로부터 데이터 D를 읽도록 구성될 수 있다. 그러나 유효 DCV 필드가 엔트리(311-3)에 존재하기 때문에, 컨트롤러(102)는 DCV 메모리(304-2)를 액세스해서 DCV'를 읽도록 구성될 수 있다. 컨트롤러(102)는 역 DCV 기능(308-2)에 대한 입력들로서 데이터 D와 DCV'를 사용하고, 데이터 D와 DCV'를 조합해서 데이터 D'를 생성하도록 구성될 수 있다. 컨트롤러9102)는 데이터 D'와 함께 읽기 요청(302)에 응답하도록 구성될 수 있다.

[0049] 증가한 읽기 양이 이 기술에 관여되는 동안에, 그 증가한 읽기 양은, 무시할 수는 없지만, 적은 양을 가질 수

있다. 예를 들면, 실시 예로서, 메모리(104)의 내부 읽기 폭(internal read bandwidths)은 저장 장치(100)의 외부 인터페이스 폭보다 높을 수 있다. 비록 읽기 성능에 무시할 수 없는 영향을 가진다 하더라도, 읽기는 삭제된 블록의 사용, 인접한 트랙들의 재쓰기 등을 초래할 수 없다. 이와 같이, 읽기 성능을 줄일 수 있는 동작들은 레이턴시를 줄이거나 레이턴시 일관성을 향상하거나, 그 밖의 것들을 하는 관련된 동작들보다 덜 영향을 받을 것이다.

[0050] 도 1 및 도 3c를 참조하면, 실시 예로서, 읽기 요청(302)은, 맵핑 테이블(310)이 도 2c에 도시된 바와 같이 데이터가 쓰여진 후의 상태에 있을 때, 컨트롤러(102)에 의해 입력받을 수 있다. 즉, 데이터 D는 초기에 데이터 메모리(304-1)에 쓰여진다. 업데이트된 데이터 D'는 DCV 메모리(304-2)에 저장되어 있는 DCV'의 결과로 쓰여진다. 그리고 추가로 업데이트된 데이터 D''는 DCV 메모리(304-2)에 저장되어 있는 DCV''의 결과로 쓰여질 수 있다.

[0051] 컨트롤러(102)는 다시 한번 물리적 블록 어드레스(23)를 위한 엔트리(311-3)를 액세스하고, 물리적 블록 어드레스(23)를 사용해서 데이터 메모리(304-1)를 액세스해서 데이터 D를 액세스하고, DCV 어드레스를 사용해서 DCV 메모리(304-2)를 액세스하도록 구성될 수 있다. 그러나 데이터 D''가 논리적 블록 어드레스 1에 가장 최근에 쓰여진 데이터이기 때문에, DCV''는 이용가능하고 DCV 메모리(304-2)에서 액세스되는 DCV이다.

[0052] 컨트롤러(102)는 DCV''와 데이터 D를 역 DCV 기능(308-2)에 대한 입력들로 사용해서 데이터 D''를 생성하도록 구성될 수 있다. 컨트롤러(102)는 데이터 D''와 읽기 요청(302)에 응답하도록 구성될 수 있다. 따라서, 비록 초기 데이터 D가 액세스되었지만, 가장 최근 데이터 D''는 다시 생성될 수 있다. 특히, 데이터 D'와 그것과 관련된 DCV'는 데이터 D''를 생성하는 데 사용되지 않는다.

[0053] 도 3a 내지 도 3c에서 설명한 바와 같이, 컨트롤러(102)는 DCV 필드가 DCV가 존재하거나 또는 유효한지를 지시하는지의 여부에 근거하여, 다르게 동작하도록 구성될 수 있다. 그러나 다른 실시 예로서, 컨트롤러(102)는, 비록 초기 데이터가 데이터 메모리(304-1)에 저장되어 있을 때라도, 도 3b 및 도 3c와 마찬가지로 동작하도록 구성될 수 있다. 특히 엔트리(311)의 DCV 필드는, DCV 필드 자치에 있거나 그렇지 않으면 DCV 메모리(304-2) 내에 있는 식별 DCV(identity DCV)를 지시하는 것으로 초기화될 수 있다. 식별 DCV는, 초기 데이터 D와 함께 역 DCV 기능(308-2)에 대한 입력으로 사용될 때, 데이터 D를 발생할 수 있다. 결과로서, 실제 데이터를 비교함으로부터 얻어지는 DCV가 존재하는지 아니면 유효한지 여부에 상관없이, 실질적으로 동일한 동작들이 컨트롤러(102)에 의해 수행될 수 있다.

[0054] 도 1 및 도 3d를 참조하면, 실시 예로서, 데이터의 더 이른 초기 버전이 액세스될 수 있다. 특히, 컨트롤러(102)는 소스 읽기 요청(302-1)을 입력받도록 구성될 수 있다. 여기에서, 소스 읽기 요청(302-1)은 논리적 블록 어드레스 1을 참조한다. 응답으로, 컨트롤러(102)는 도 3a에서 설명한 액세스와 유사하게, 데이터 메모리(304-1)의 물리적 블록 어드레스(23)에 있는 데이터 D를 액세스하도록 구성된다. 그러나 엔트리(311-3)의 DCV 필드는, 도 3b 및 도 3c와 유사하게, 유효하다. 즉, 논리적 블록 어드레스 1에는, 데이터의 업데이트 버전이 존재한다. 도 3b와 도 3c와는 대조적으로, 초기 데이터 D는 다시 생성된 현재 데이터 D' 또는 D''로 복귀된다(returned). 따라서, 소스 읽기 요청(302-1)에 유사한 요청 또는 요청들은 저장 장치(100)에 저장된 데이터의 더 이른 버전을 읽는데 사용될 수 있다.

[0055] 엔트리(311-3)의 DCV 필드로부터의 DCV 어드레스를 읽는 동작이 예로서 사용되는 반면에, 다른 실시 예로서, DCV 값은 엔트리(311-3)의 DCV 필드로부터 읽혀질 수도 있다. 예를 들면, 도 3b 및 도 3c의 DCV' 및 DCV''는 각각 엔트리(311-3)로부터 읽혀질 수 있다.

[0056] 도 1 및 도 3e를 참조하면, 실시 예로서, 도 2c에서 설명한 바와 같이 맵핑 테이블(310)이 데이터가 쓰여진 후의 상태에 있을 때, 읽기 요청(302-2)이 컨트롤러(102)에 의해 입력될 수 있다. 즉, 데이터 D는 초기에 데이터 메모리(304-1)에 쓰여진다. 업데이트된 데이터 D'는 DCV 메모리(304-2)에 저장되어 있는 DCV'의 결과로 쓰여진다. 그리고 추가로 업데이트된 데이터 D''는 DCV 메모리(304-2)에 저장되어 있는 DCV''의 결과로 쓰여질 수 있다. 그러나 이 실시 예에서, 하나 또는 그 이상의 중간 DCVs(intermediate DCVs)가 유지된다. 이 예에서, DCV''는 현재 DCV이다. 그러나 DCV'는 또한 유지된다. 엔트리(311-4)에 있는 0x55의 추가적인 파라미터는 DCV'의 표시를 나타낸다(예를 들면, 그것의 값이나 어드레스).

[0057] 이 예에서, 읽기 요청(302-2)은 LBA 1'를 위한 요청이다. LBA 1'는 데이터 D'로서의 데이터의 상태를 나타낸다. 따라서, 컨트롤러(102)는 DCV' 어드레스, 즉 DCV'가 DCV 메모리(304-2)에 저장된 위치를 나타내는 어드레스를 액세스하도록 구성될 수 있다. 결과로서, DCV'는 액세스되고 역 DCV 기능(308-2)에 있는 데이터 D와 결합되어

데이터 D'를 생성할 수 있다.

[0058] 비록 단지 하나의 중간 DCV (예를 들면, DCV')가 예로서 사용되지만, 다른 실시 예로서 어떤 수의 중간 DCV들도 저장될 수 있다. 예를 들면, DCVs, DCV', DCV'', DCV''', 그리고 DCV''''이 DCV 메모리(304-2)에 모두 저장될 수 있다이들 DCVs 각각은 데이터 D와 결합되어 데이터 D', D'', D''', 그리고 D''''와 같은 데이터의 나중 버전을 각각 생성할 수 있다.

[0059] 도 4a 및 도 4b는 본 발명의 실시 예에 따른 저장 장치들을 보여주는 블록도이다. 도 4a를 참조하면, 실시 예로서, 저장 장치(400)는 도 1에 도시된 저장 장치(100)의 컨트롤러(102)와 유사한 컨트롤러(402)를 포함한다. 그러나 저장 장치(400)는 불휘발성 메모리(404-1)와 휘발성 메모리(404-2)를 포함한다. 불휘발성 메모리(404-1)에는 예로서, flash memory, STT-MRAM, Phase-Change RAM, NFGM, or PoRAM, magnetic or optical media 등이 포함될 수 있다. 휘발성 메모리(404-2)에는 예로서, DRAM, DDR, DDR2, DDR3, DDR4, SRAM 등과 같은 다양한 표준에 따른 DDR SDRAM 등이 포함될 수 있다.

[0060] 컨트롤러(402)는 불휘발성 메모리(404-1)에 데이터(406)을 저장하고, 휘발성 메모리(404-2)에 DCVs(408)을 저장하도록 구성될 수 있다. 컨트롤러(402)는 또한 휘발성 메모리(404-2)에 맵핑 테이블(410)을 저장하도록 구성될 수 있다.

[0061] 실시 예로서, 저장 장치(400)의 사용은 일관성(consistency) 보다는 일관된 레이턴시(consistent latency)에 우선 순위를 둘 수 있다. 따라서 데이터의 일관성은 차순위로 될 수 있다(consistency of the data may be relaxed). 예를 들면, 여러 인터넷-스케일 애플리케이션들(several internet-scale applications)은 차순위 범위 내에서(within relaxed bounds) 일관성을 고려해서 액세스될 수 있다. 그러한 애플리케이션들은 트위트(tweets) 및 포토 태깅(photo tagging)을 포함할 수 있다. 그러나 그러한 애플리케이션들을 위해, 레이턴스 스파이크들(latency spikes)은 받아들여질 수 없다. DCVs(408)로 대표되는 것처럼, 이러한 범위 내의 데이터는 휘발성 메모리(404-2)에 저장될 수 있다. 컨트롤러(402)는 DCV(408)의 오버플루(overflows)를 불휘발성 메모리(404-1)로 내보내도록(flush) 구성될 수 있다. 실시 예로서, 컨트롤러(402)는 DCVs(408)를 불휘발성 메모리(404-1)의 데이터(406)로 내보내도록 구성될 수 있다. 휘발성 메모리(404-2)는 배터리, 슈퍼 캐패시터, 또는 NVRAM에 의해 회복될 수 있다(may be backed). 그러나 실시 예로서, 그러한 백업은 DCVs(408)의 손실을 일으키는 실패가 여전히 받아들여 질 수 있는 범위 내에 있기 때문에 꼭 필요한 것은 아니다. 백업을 위해 사용되는 배터리, 슈퍼 캐패시터, 또는 NVRAM을 생략하면, 저장 장치(400)의 비용(cost)를 줄일 수 있다.

[0062] 실시 예로서, 캐시(412)는 데이터를 불휘발성 메모리(404-1)로 전달하기 전에, 데이터를 캐시하는 데 사용될 수 있다. 예를 들면, 도 2a 내지 도 2d에서 설명한 읽기 요청들로부터 나온 데이터는 캐시(412)에 저장될 수 있다. 이 스토리지는 도 2a 내지 도 2d에서 설명한 다양한 기술들 없이도 동작될 수 있다. 그러나 데이터가 캐시(412)로부터 나올 때 또는 그와 유사한 경우에, 도 2a 내지 도 2d에서 설명한 기술들이 사용될 수 있다. 특별한 예로서, 캐시(412)에 저장된 데이터 블록은 다수의 읽기 요청들에 의해 업데이트될 수 있다. 이러한 업데이트들은 DCV의 계산을 관계시키지 않는다. 데이터 블록이 데이터를 되찾거나 내줄 때, 데이터의 상태에 근거하여 도 2a 내지 도 2d에서 설명한 바와 같이 DCV를 생성할 수 있다.

[0063] 도 4b를 참조하면, 실시 예로서, 저장 장치(401)는 도 4a의 저장 장치(400)와 유사하다. 그러나 저장 장치(401)에서 컨트롤러(402)는 cached DCV(408-1)에 의해 대표되는, 휘발성 메모리(404-2) 내 DCVs를 캐시하도록 구성될 수 있다. 특히, 컨트롤러(402)는 빈번히 액세스되는 DCVs를 휘발성 메모리(404-2)에 유지하도록 구성될 수 있다. 컨트롤러(402)는 캐싱 알고리즘이나 발견적 방법(heuristics) 등을 사용하여, 캐시에서 어떤 DCVs(408-1)를 유지할지를 결정하도록(determine which DCVs 408-1 to maintain in the cache) 구성될 수 있다. The controller 402 is configured to transfer other DCVs 408-1 to the DCVs 408-2 stored in the nonvolatile memory 404-1 and vice versa. 컨트롤러(402)는 다른 DCVs(408-1)를 불휘발성 메모리(404-1)에 저장된 DCVs(408-2)로 전달하도록 구성되고, 그 역도 성립될 수 있다. 결과적으로, 실시 예로서, 빈번히 액세스되거나 무겁게 액세스되는 데이터는 캐시될 수 있다. 게다가, DCVs(408-1)는 대응하는 데이터(406)보다 사이즈가 작기 때문에, 실제 데이터가 캐시 된다면 더 많은 업데이트들이 휘발성 메모리(404-2)에서 유지될 수 있다.

[0064] 도 5는 본 발명의 실시 예에 따른 저장 장치 내의 페이지 사이즈들을 보여주는 블록도이다. 실시 예로서, 데이터 페이지들(502)과 DCV 페이지들(504)은 같은 메모리(500)에 저장될 수 있다. 여기에서, 데이터 페이지들(502-1, 502-2)과 DCV 페이지들(504-1 내지 504-4)은 메모리(500)에 저장된 데이터 페이지들(502) 및 DCV 페이지들(504)의 예들이다. 여기에서, DCV 페이지들(504)은 소스 데이터와 비교하여 상대적으로 더 작은 사이즈의 DCVs를 나타내기 위하여 더 작게 도시되어 있다. 이 예에서, DCV 페이지들(504)는 데이터 페이지들(502)의 절반 사

이즈이다. 그러나 다른 실시 예에서, DCV 페이지들(504)의 사이즈들은 사용된 특별한 DCV 기능에 따라 달라질 수 있다.

[0065] 실시 예로서, 데이터 페이지들(502) 및 DCV 페이지들(504)은 둘 다 레이턴스를 증가시킬 수 있는 순차 쓰기(in-place writes)의 부족과 같은, 동일한 제한들을 받을 수 있다. 위에서 설명한 바와 같이, 데이터 페이지(502)는 일반적으로 DCV가 변하는 동안에는 유지될 수 있다. DCV 페이지들에 대한 변화는 레이턴시 영향(latency impact)을 초래할 수 있다. 그러나 DCV 페이지들(504)의 사이즈가 점점 작아짐에 따라, 레이턴시 영향은 줄어들 수 있다. 예를 들면, DCV 페이지들(504)이 데이터 페이지들(502)과 다른 블록에 저장될 수 있다. 따라서, 더 많은 DCVs가 DCV 페이지들(504)에 축적됨에 따라, 가비지 컬렉션이 그 블록들에서 수행되서 자유 블록들이 회복될 수 있다(recover). DCV 페이지들(504)이 더 작기 때문에, 가비지 컬렉션은 시간이 덜 들고 덜 빈번하게 수행될 수 있다. 이것에 더해서, 데이터 페이지들(502)은 DCV 페이지들(504)보다 더 오랫동안 유효한 상태로 남을 수 있다. 결과적으로, 데이터 페이지들(502)을 저장하는 블록들은 DCV 페이지들(504)을 저장하는 블록들보다 더 적게 가비지 컬렉션을 겪게 될 수 있다. 또한, DCV 페이지들(504)이 빠르게 무효화될 수 있기 때문에, DCV 페이지들(504)을 저장하는 하나의 블록에 더 많은 페이지들이 무효일 수 있다. 그래서 유효한 페이지들이 덜 카피되기 때문에, 가비지 컬렉션 동작을 수행하기 위한 시간이 줄어들 수 있다.

[0066] 도 6은 본 발명의 실시 예에 따른 SSD를 보여주는 블록도이다. 실시 예로서, SSD(600)는 도 1에 도시된 저장 장치(100)의 컨트롤러(102)와 유사하다. 그러나 SSD(600)는 플래시 메모리(604-1), DRAM(604-2), 그리고 SRAM(604-3)을 포함할 수 있다. 컨트롤러(602)는 플래시 메모리(604-1)에 데이터(606)와 DCVs(608-1)를 저장하도록 구성될 수 있다. 비록 메모리들의 구성이 특정하여 도시되어 있지만, 다른 실시 예로서, SSD(600)는 데이터(606) 및 DCVs(608-1, 608-2)의 구성들과 분포가 다른 메모리들을 포함할 수 있고, 맵핑 테이블(600)도 다를 수 있다. 예를 들면, SSD(600)는 도 1, 도 4a, 및 도 4b와 유사한 구성들을 가질 수 있다.

[0067] 플래시 메모리(604-1)에 저장된 데이터(606)와 DCV(608-1)는 가비지 컬렉션를 자주 일으킬 수 있다(susceptible to garbage collection). 데이터(606)와 DCV(608-1)을 저장하는 페이지들이 삭제되고, 잠재적으로 추가적인 레이턴시를 일으킬 수 있다. 그러나 앞에서 설명한 바와 같이, DCV(608-1)의 사이즈는 대응하는 데이터(606)보다 작다. 즉, 데이터(606)를 저장하기 위해 사용되는 플래시 메모리(604-1) 내의 페이지들은 DCV(608-1)를 저장하는 데 사용되는 페이지들보다 사이즈가 크다.

[0068] DCVs를 사용하면, SSD(600) 내에서 쓰기 변경(write modifications) 즉, 비순차 업데이트들(out-of-place updates)이 줄어들 수 있다. 이것은 리클레임(reclaim)될 무효 블록들을 작게 해준다. 이 인턴(in-turn)은 가비지 컬렉션의 횟수(freuency)를 줄여준다. 특히, SSD(600)는 빅데이터와 클라우드 애플리케인션들과 업데이트-헤비 I/O 트래픽을 다룰 수 있다. 예를 들면, SSD(600)는 업데이터와 클라우드 애플리케이션을 가지고, 가비지 컬렉션의 낮은 회수와 그것으로 인한 높은 레이턴시의 낮은 기회, 특히 레이턴시 스파이크들을 가지고 업데이트-헤비 I/O 트래픽을 다룰 수 있다. 특히, 데이터(606)에 대한 쓰기 업데이트들을 할 때, 데이터(606)의 제 1 카피는 플래시 메모리(604-1)에 남는다. 플래시 메모리(604-1)에 있는 DCVs(608-1)과 DRAM(604-2)에 캐시된 cached DCV(608-2)는, 쓰기 업데이트가 헤비 업무 로드라고 하더라도, 플래시 메모리(604-1)에 있는 페이지들의 수명을 증가시킨다. 이것에 더하여 앞에서 설명한 바와 같이, 데이터(606)의 이전 버전도 이용할 수 있다. 레이턴시를 향상시킬 뿐만 아니라, 아키텍처가 플래시 셀 마모(wearing)를 개선할 수도 있다.

[0069] 특히, 이전 페이지를 무효화하고 업데이트 데이터를 위한 새로운 페이지를 요청하는 대신에, 컨트롤러(602)는 이전 페이지와 업데이트 사이의 차이를 나타내는 DCV와 함께 플래시 메모리(604-1)에 유효/액티브로서 이전 페이지를 유지하도록 구성될 수 있다. 뒤이은 읽기 동작 시에, 컨트롤러(602)는 이전 페이지와 DCV를 모두 읽고 결합해서 데이터 페이지까지 대부분을 제공할 수 있다. 위에서 설명한 바와 같이, DCVs를 어디에 저장하는지를 위한 다수의 구성들이 있다. 예를 들면, 플래시 메모리(604-1), DRAM(604-2), SRAM(604-3), 또는 그러한 메모리들의 조합에 저장할 수 있다. 이에 덧붙여서, 액티브 또는 핫 페이지 DCVs를 DRAM(604-2)에 캐싱하고, 플래시 메모리(604-1) 상의 지속적인 카피를 DCV(608-1)에서 유지하는 것에 의해, DCVs는 캐시될 수 있다.

[0070] 실시 예로서, 읽기들과 쓰기들이 비대칭(asymmetric)이지 않거나, 동일 오버헤드를 가지거나, 그와 같은 것일 때일 지라도, 본 명세서에서 설명한 바와 같이, 저장 장치는 성능 이익을 가질 수 있다. 특히, 만약 쓰기들의 사이즈와 DCV의 사이즈들을 갖는 쓰기 시간 스케일(write times scale)이 대응하는 데이터 블록들의 사이즈들보다 작다면, 쓰기 시간은 줄어들 수 있다.

[0071] 실시 예로서, DRAM(604-2)은 쓰기 요청들 및/또는 쓰기 데이터를 캐시하는 데 사용될 수 있다. 쓰기 요청을 처리하는 것이 데이터(606)에 존재하는 데이터를 읽는 것에 관여될 때, 대응하는 데이터(606)는 캐시된 데이터

(606-1)로서 DRAM(604-2)에 저장될 수 있다. 컨트롤러(602)는 위에서 설명한 DCV 함수들에 대한 입력으로서, DRAM(604-2)에 저장된 캐시된 데이터(606-1)를 사용하도록 구성될 수 있다. 즉, 플래시 메모리(604-1)에 저장된 데이터(606)로부터 데이터를 읽는 대신에, 컨트롤러(602)는 DRAM(604-2)에 저장된 캐시된 데이터(606-1)로부터 데이터를 읽도록 구성될 수 있다.

[0072] 실시 예로서, 플래시 메모리(604-1)는 패널들과 채널들로 구분될 수 있다. SSD(600)는 페이지 단위로 읽고 쓰고, 블록 단위로 소거한다. 블록은 복수의 페이지들을 포함할 수 있다. 맵핑 전략은 번역 단위(granularity of translation)를 정의한다. 예를들면, 페이지 레벨 맵핑은 보다 큰 면적(larger footprint)을 필요로 하지만, 더 높은 유동성(higher degree of flexibility)를 제공할 수 있다. 블록 레벨 맵핑은 보다 작은 면적(smaller footprint)을 사용하지만, 배치(placement)에 제한적일 수 있다. 몇 개의 하이브리드 정책들의 변형들이 페이지 및 블록에 근거한 맵핑들의 조합을 이용하도록 제안되고 있다. 맵핑 테이블(610)에 있는 맵핑은 그러한 맵핑 기술들 중 어느 것이나 사용할 수 있다.

[0073] SSD(600)가 예로서 설명되었지만, 다른 실시 예로서, shingled magnetic drives와 같은 다른 형태의 저장 매체도 사용될 수 있다. 특히, shingled disk 상에서, 쓰기들은 이전에 쓰여진 마그네틱 트랙들의 일부분을 오버랩할 수 있다. 이것은 쓰기 성능을 낮출 수 있고, 또한 이웃하는 트랙들을 겹쳐 쓰기를 할 수 있다. shingled magnetic drives는 펌웨어서 이것을 다룸으로써 이러한 복잡성(complexity)을 감출 수 있다. 더 낮은 쓰기 성능(이웃하는 트랙들에 쓰기를 할 때)은 불일치 레이턴시(inconsistent latency)를 초래할 수 있고, 대부분의 클라우드 애플리케이션에서 해결해야 할 문제이다. 본 명세서에서 설명한 바와 같이, DCVs를 사용하면 겹쳐 쓰기가 줄어들고, 업데이트-민감 워크로드(update-intensive workloads)에 대한 일치된 트랜색션 레이턴시(consistent transaction latency)를 제공할 수 있다.

[0074] 도 7은 본 발명의 실시 예에 따른 버저닝 저장 시스템을 보여주는 블록도이다. 실시 예로서, 시스템(700)은 통신 링크(706)를 통해 저장 장치(704)에 연결되는 호스트(702)를 포함한다. 호스트(702)는 저장 장치의 데이터 저장 능력을 사용하는 시스템으로, general purpose processor, a digital signal processor (DSP), an application specific integrated circuit, a microcontroller, a programmable logic device, discrete circuits, 그리고 이러한 장치들의 조합이나 이와 유사한 것들을 포함한다. 실시 예로서, 호스트(702)는 컴퓨터, 서버, 워크스테이션, 또는 유사한 것들일 수 있다. 호스트(702)는 O/S(operating system) 및 애플리케이션들과 같은 소프트웨어를 수행하도록 구성될 수 있다. 호스트(702)는 메모리(708)에 연결될 수 있다. 메모리(708)는 동작적 메모리(operational memory) 및/또는 호스트(702)에 사용되는 캐시 메모리를 포함한다. 캐시 메모리의 예로는, DRAM이나, DDR, DDR2, DDR3, DDR4, SRAM 등과 같은 다양한 표준에 따른 DDR SDRAM 등이 포함될 수 있다. 메모리(708)가 호스트(708)과 분리되어 설명되었지만, 실시 예로서, 메모리(708)는 호스트(702)의 일부분일 수도 있다.

[0075] 통신 링크(706)는 호스트(702)와 저장 장치(704)가 통신하도록 구성되는 매체를 나타낼 수 있다. 예를 들면, 통신 링크(706)는 USB, SCSI, PCIe, SAS, PATA, SATA, NVMe, UFS, Fiber channel, Ethernet, RDMA, Infiniband, 또는 다른 유사한 링크들과 같은 링크일 수 있다. 호스트(702)와 저장 장치(704) 각각은 그러한 통신 링크들과 통신하기 위한 인터페이스들을 갖도록 구성될 수 있다.

[0076] 실시 예로서, 저장 장치(704)는 앞에서 설명한 저장 장치들(100, 400, 401, 600)과 유사한 저장 장치일 수 있고, 도 2a 내지 도 3d에서 설명한 방식대로 동작하도록 구성될 수 있다. 호스트(702)는 저장 장치(704)로부터 데이터를 읽거나 저장 장치(704)에 데이터를 쓰도록 구성될 수 있다. 저장 장치(704)는 앞에서 설명한 바와 같이 시스템(700)의 동작들을 향상하도록 DCVs를 사용하도록 구성될 수 있다.

[0077] 그러나 다른 실시 예로서, 호스트(702)는 DCVs 관점에서 앞에서 설명한 방법들과 유사한 동작들을 수행하도록 구성될 수 있다. 예로서, 호스트(702)는 저장 장치 드라이버(710)를 포함할 수 있다. 저장 장치 드라이버(710)는 저장 장치(704)와 동작하는 호스트(702) 상에서 수행될 수 있는 소프트웨어를 나타낸다.

[0078] 특히, 저장 장치 드라이버(710)는 DCVs 관점에서 앞에서 설명한 방법들과 유사한 동작들을 수행하도록 구성될 수 있다. 즉, 컨트롤러(102, 402, 602 등)에 의해 수행된 앞에서 설명한 동작들은 저장 장치 드라이버(710)에 의해 수행될 수 있다. 이에 더하여, 메모리(708)는 적어도 메모리들(104, 404-2, 604-2, 604-3)의 일부분에 유사하게 사용될 수 있다. 즉, 메모리(708)는 예를들면, 맵핑 테이블(110, 410, 610), DCVs(108, 408), 및/또는 캐시된 DCVs(408-1, 608-2)를 저장하는 데 사용될 수 있다.

[0079] 실시 예로서, 도 2a 내지 도 3d에서 설명한 논리적 블록 어드레스 및 물리적 블록 어드레스의 관련성은 매핑 테

이블에 나타낼 필요는 없다. 저장 장치 드라이버(710)는 논리적 어드레스와 DCVs의 표시들(indications of DCVs)의 관련성을 유지하도록 구성될 수 있다. 즉, 저장 장치 드라이버(710)는 초기 데이터 D가 저장된 곳에 대한 표시로서, 물리적 블록 어드레스에 유사하게 논리적 블록 어드레스를 사용하도록 구성될 수 있다. 실시 예로서, 저장 장치 드라이버(710)는 도 2a 내지 도 3d에서 설명한 데이터 메모리(204-1, 304-1)와 같은 저장 장치(704)를 사용하도록 구성될 수 있다.

[0080] 실시 예로서, 논리적 블록 어드레스에 대해, 저장 장치 드라이버(710)는 초기 데이터를 저장 장치(704) 상의 논리적 블록 어드레스에 쓰도록 구성될 수 있다. 저장 장치 드라이버(710)는 초기 데이터를 메모리(708)에 캐시하도록 구성될 수 있다. 데이터가 연이어 변할 때, 저장 장치 드라이버(710)는 캐시된 초기 데이터를 사용하여 DCV를 계산하고 DCV를 저장 장치에 쓰도록 구성될 수 있다. 다른 실시 예로서, 저장 장치 드라이버(710)는 DCVs를 메모리(708)에 유지하도록 구성될 수 있다. 앞에서 설명한 DCV 관리 기술들은 저장 장치 드라이버(710)에 사용될 수 있다.

[0081] 도 8은 본 발명의 실시 예에 따른 서버를 보여주는 블록도이다. 실시 예로서, 서버(800)는 a stand-alone server, a rack-mounted server, a blade server 등을 포함할 수 있다. 서버(800)는 저장 장치(802)와 프로세서(804)를 포함할 수 있다. 프로세서(804)는 저장 장치(802)에 연결된다. 비록 하나의 저장 장치(802)가 도시되어 있지만, 어떤 수의 저장 장치들(802)도 존재할 수 있다. 저장 장치(802)는 앞에서 설명된 저장 장치들 중 어느 것일 수 있다. 따라서, 서버(800)의 성능은 향상될 수 있다.

[0082] 도 9는 본 발명의 실시 예에 따른 서버 시스템을 보여주는 블록도이다. 실시 예로서, 서버 시스템(900)은 다수의 서버들(902-1 내지 902-N)을 포함한다. 서버들(902)은 각각 관리자(904)에 연결된다. 하나 또는 그 이상의 서버들(902)은 앞에서 설명한 서버(800)와 유사할 수 있다.

[0083] 관리자(904)는 서버들(902) 및 서버 시스템(900)의 다른 구성들을 관리할 수 있다. 실시 예로서, 관리자(904)는 서버들(902)의 성능을 모니터하도록 구성될 수 있다. 예를 들면, 서버들(902) 각각은 앞에서 설명한 저장 장치를 포함할 수 있다.

[0084] 도 10은 본 발명의 실시 예에 따른 데이터 센터를 보여주는 블록도이다. 실시 예로서, 데이터 센터(1000)는 다수의 서버 시스템들(1002-1 내지 1002-N)을 포함한다. 서버 시스템들(1002)은 도 9에 도시된 서버 시스템(900)과 유사할 수 있다. 서버 시스템 들(1002)은 인터넷과 같은 네트워크(1004)에 연결된다. 따라서 서버 시스템들(1002)은 네트워크(1004)를 통해 다양한 노드들(1006-1 내지 1006-M)과 통신할 수 있다. 예를 들면, 노드들(1006)은 클라이언트 컴퓨터들, 다른 서버들, 리모트 데이터 센터들, 저장 시스템들, 또는 이와 유사한 것들일 수 있다.

[0085] 실시 예들은 클라우드 및 large-scale latency-critical services과 같이, 일관된 성능이 중요한 요소(important factor)인 곳에서 사용될 수 있다. 그러한 서비스들의 예들은 데이터 분석(data analytics), 기계학습(machine learning), 오디오 및 비디오 스트리밍(audio and video streaming)을 포함할 수 있다. 일관된 레이턴시는 높은 우선 순위일 수 있다. 그리고 몇몇 실시 예들은 보다 더 예측가능한 성능을 지원하는 relaxed consistency 요구(requirements)를 채용하는 배포된 소프트웨어 스택들을 사용한다. 비록 몇몇 요소들이 자원 공유 및 대기(resource sharing and queuing)와 같은, 부하된 레이턴시(loaded latency)에서 불일치를 초래한다고 하더라도, SSDs 내에서 가비지 컬렉션은 상당한 이익을 차지할 수 있다.

[0086] 현재 소프트웨어 스택들은 복제(replication)를 사용할 수 있다. 그러나 이것의 해결은 문제의 근본은 다루지 않는다. 이것에 더하여, 복제는 내재적으로 더 많은 비용이 들게 하고, 네트워크 트래픽이 증가할 수 있다. 이것은 더욱 네트워크 레이턴시에 영향을 끼칠 수 있다. 또한, 복제는 소프트웨어 층에서 coordination을 사용한다.

[0087] 큰 데이터 생성률로 인해, 더 빠른 데이터 분석이 사용될 수 있다. 서치 엔진들과 쇼셜 미디어와 같은 온라인 서비스들에 있어서 중요한 디자인 목표는 예측 가능한 성능을 제공하는 것이다. 그런 맥락(context)에서, 평균 응답 시간은 성능을 대표할 수 없다. 가장 나쁜 케이스(worst case)가 더 성능의 관심사일 수 있다. 다양한 반응 시간이 서비스의 요소들에 있어서 더 높은 테일 레이턴시(higher tail latency)를 일으킬 수 있다. 결과적으로, 사용자들은 긴 응답 시간을 겪을 수 있다. 업무 부하(workload)와 SSD 펌웨어 정책들에 근거해서, 테일 레이턴시 스파이크들(tail latency spikes)은 드물게 또는 자주 발생할 수 있다. 그러나 대부분의 경우에, 현재 매우 경쟁적인 시장에서 사용자의 경험 및 서비스 제공자의 평판을 나쁘게 하기에 충분할 수 있다. 테일 레이턴시 패널티들은 아마존 AWS 및 구글 클라우드와 같은 공유 기반시설(shared infrastructures)에서 더 악화될 수

있다(exacerbated). 이러한 문제점은 몇 곳의 클라우드 벤더들에 의해 공유되고 있고, 하드웨어/소프트웨어 시스템 아키텍쳐들을 디자인하고 있는 회사들의 주요 도전들 중의 하나로서 인식되고 있다. 비록 몇몇 요소들이 자원 공유 및 대기(resource sharing and queuing)와 같은, 부하된 레이턴시(loaded latency)에서 불일치를 초래한다고 하더라도, SSDs 내에서 가비지 컬렉션은 상당한 이익을 차지할 수 있다. 순차 업데이트들(in-place updates)을 허용하지 않는 플래시 메모리의 특성으로 인해, SSD 펌웨어는 비순차적으로(out of place)으로 업데이트들을 쓰고, 이전 카피를 무효화시킨다. 리클레임 공간(reclaim space)을 위하여, 무효 공간은 그것이 다시 쓰여지기 전에 삭제될 필요가 있다. 그러나 삭제 동작들(in milliseconds) 은, 읽기 쓰기 동작들(in microseconds)에 비해 상당히 더 느리다. 그리고 삭제 동작들은 전형적으로 더 거친 단위(on a coarser granularity)로 행해질 수 있다. 이런 프로세스는 SSD들 안에서 가비지 컬렉션을 일으키고, 채널은 가비지 컬렉션이 일어나는 동안에 읽기/쓰기 요청들을 서비스할 수 없다. 이런 이유로, 가비지 컬렉션은 임계 동작들(critical operations)에 있어서 레이턴시 또는 성능에 악영향을 미칠 수 있다. 어떤 경우에는 읽기 레이턴시가 가비지 컬렉션 동안에 100배 증가할 수 있다. 이에 더하여, 가비지 컬렉션은 SSD가 사용됨에 따라 더 자주 발생할 수 있다.

[0088] 클라우드에서 유명한 몇몇 애플리케이션 타입들은 쓰기 업데이트가 버거울 수 있다. 한 예로서, 실 세계 사용 케이스(real-world use case)를 에뮬레이트하는 것은 YCSB(Yahoo Cloud Serving Benchmark)이다. 이것은 클라우드 시스템들을 평가하기 위한 밴치마킹 슈트(benchmarking suite)이다. YCSB에서 제공되는 몇 가지의 업데이트-과부하(update-heavy workloads)에서, 액세스 비율은 50% 읽기, 0% 삽입, 그리고 50% 업데이트이다. E-상업 애플리케이션들(E-commerce applications)은, 사용자 세션에 있는 recording recent actions, e-commerce user의 전형적인 액션들과 같은 동작들을 포함하는, 애플리케이션들의 예이다. 애플리케이션들의 이런 카테고리에서, SSD들 내의 가비지 컬렉션에 의한 테일 레이턴시의 영향은 사용자 반응 신에 더 높은 레이턴시를 일으킬 수 있다. 다른 작업부하 예로서, 액세스 비율은 95% 읽기, 0% 삽입, 5% 업데이트일 수 있다. 한 예는 소셜 미디어이고, 비록 작은 업데이트율이라 하더라도 가비지 컬렉션을 트리거하고 서비스 레벨 목표들을 어길 수 있다. 예를 들면, 포토 태깅을 가지고, 태그를 더하는 것은 업데이트이지만, 대부분의 동작들은 태그들을 읽는 것이다.

[0089] 대부분의 빅데이터와 클라우드 애플리케이션들은, 더 많은 전통적 관점들, 예를 들면 거래의(transactional, 예를 들면, atomicity, consistency, isolation, durability, 또는 ACID) 특성들보다, 확장성(scalability)과 일관된 성능(consistent performance)을 우선 순위로 한다. 강한 일관성은 제대로 스케일하지 않는다(strong consistency does not scale well). 대부분의 클라우드 애플리케이션들은 일관된 성능을 목표로 하는 좀 더 relaxed transactional consistencies를 선호한다. 이런 이유로, 약한 일관성 모델들은 대부분의 대중적인 클라우드 스케일 분포된 소프트웨어 스택들을 통해 광범위하게 사용될 수 있다. 이것은 과부하되고 통용성이 높은 시스템들에서, I/O 성능 면에서 충분한 향상들을 제공한다.

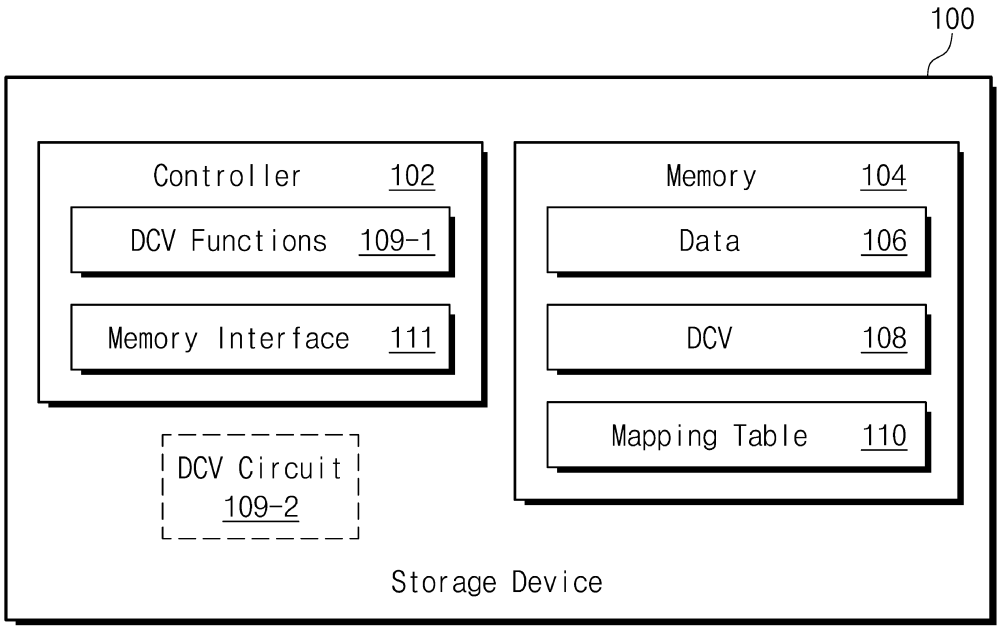
[0090] 모든 클라우드 애플리케이션들이 많은 양의 데이터를 저장하고, 증가하는 데이터 자취(footprint)에 따라 스케일되도록 디자인 되고 있지만, 단지 서브셋(subset)은 다른 것들보다 더 빈번하게 액세스되고 있다. 이러한 불균등한 배포는 데이터 센터들에서 핫 데이터 액세스 패턴들을 대표한다.

[0091] 여기에서 설명된 실시 예들은 다양한 애플리케이션들로 사용될 수 있고, 감소되고 더 일관된 레이턴시의 이익을 제공할 수 있다. 앞에서 설명한 클라우드 애플리케이션들은, 그러한 애플리케이션들의 단순한 예에 불과하다.

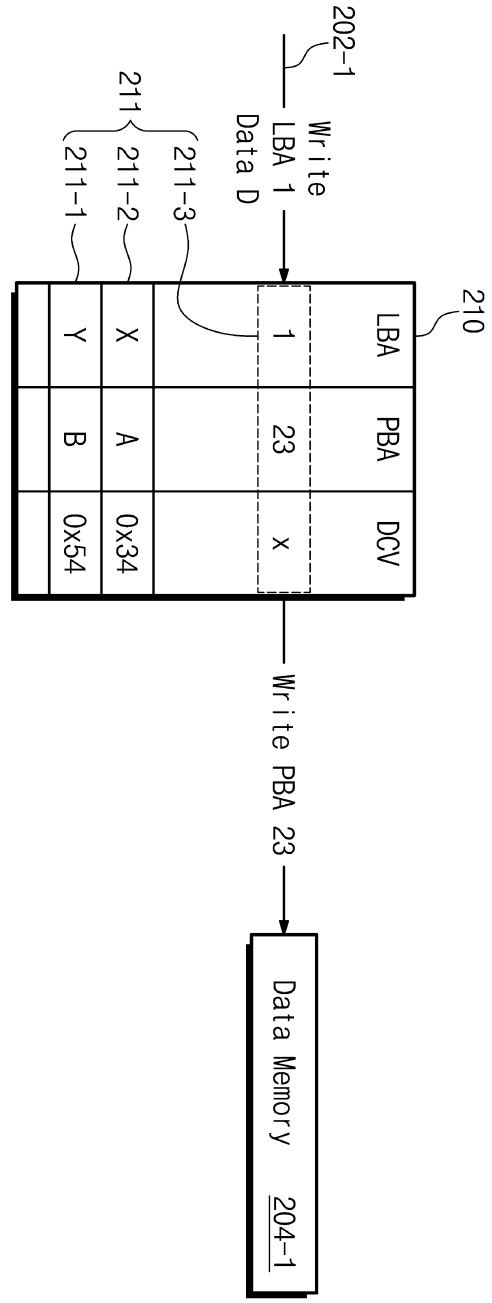
[0092] 비록 구조들, 장치들, 방법들, 그리고 시스템들이 특정 실시 예에 따라 설명되었지만, 본 발명의 기술 분야에서 통상의 지식을 가진 자는 다양한 변경이 가능하다는 것을 쉽게 인식할 수 있다. 따라서 많은 변형들이 추가된 청구항의 범위 및 기술적 사상을 벗어나지 않는 한 다양하게 만들어질 수 있다.

도면

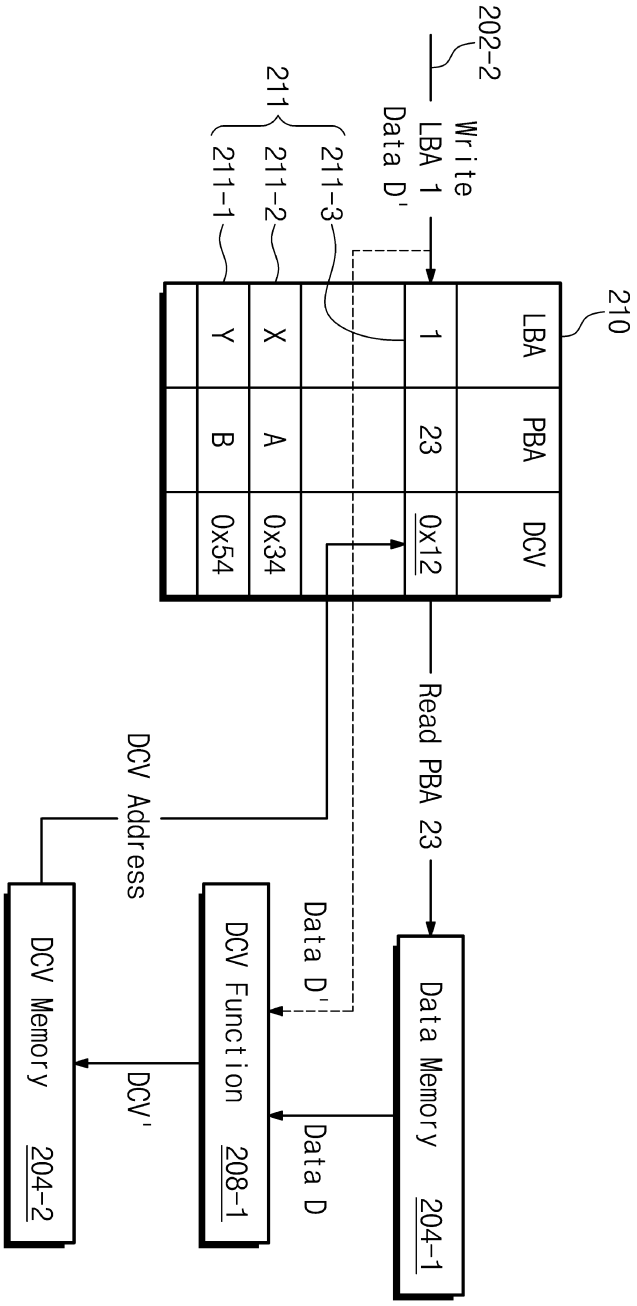
도면1


100
Controller 102
DCV Functions 109-1
Memory Interface 111
DCV Circuit 109-2
Memory 104
Data 106
DCV 108
Mapping Table 110
Storage Device

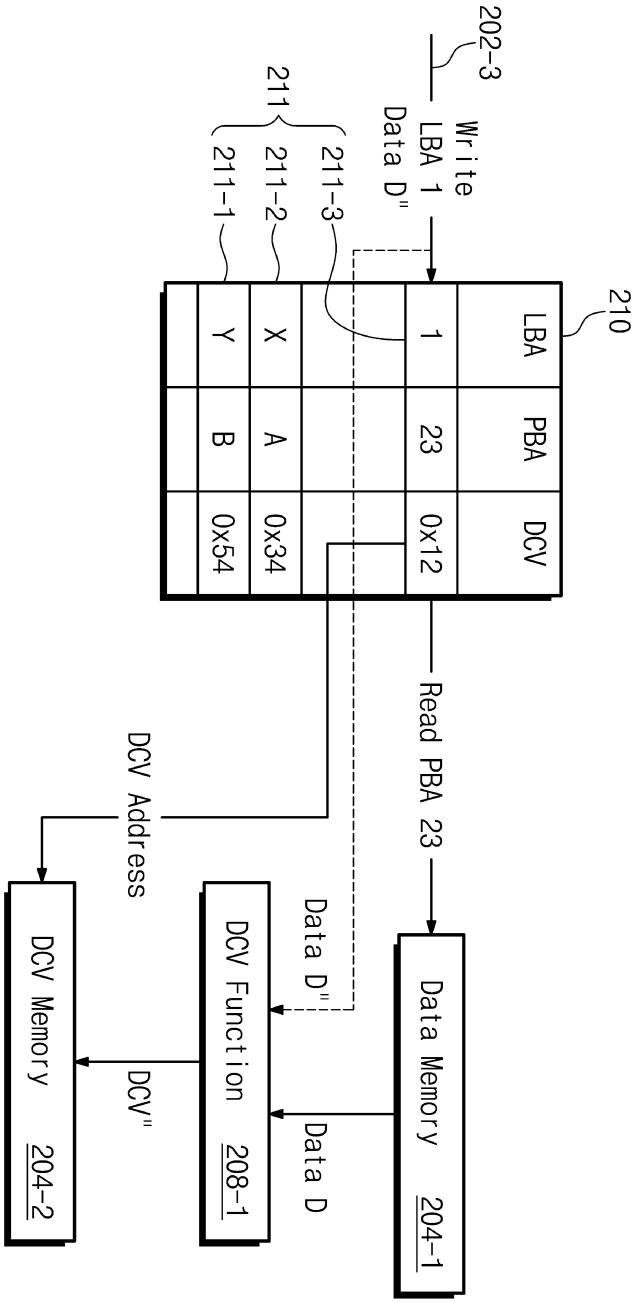
도면2a


202-1
Write
LBA 1
Data D
211
211-1
211-2
211-3
210
LBA
PBA
DCV
1
23
x
X
A
0x34
Y
B
0x54
Write PBA 23
Data Memory 204-1

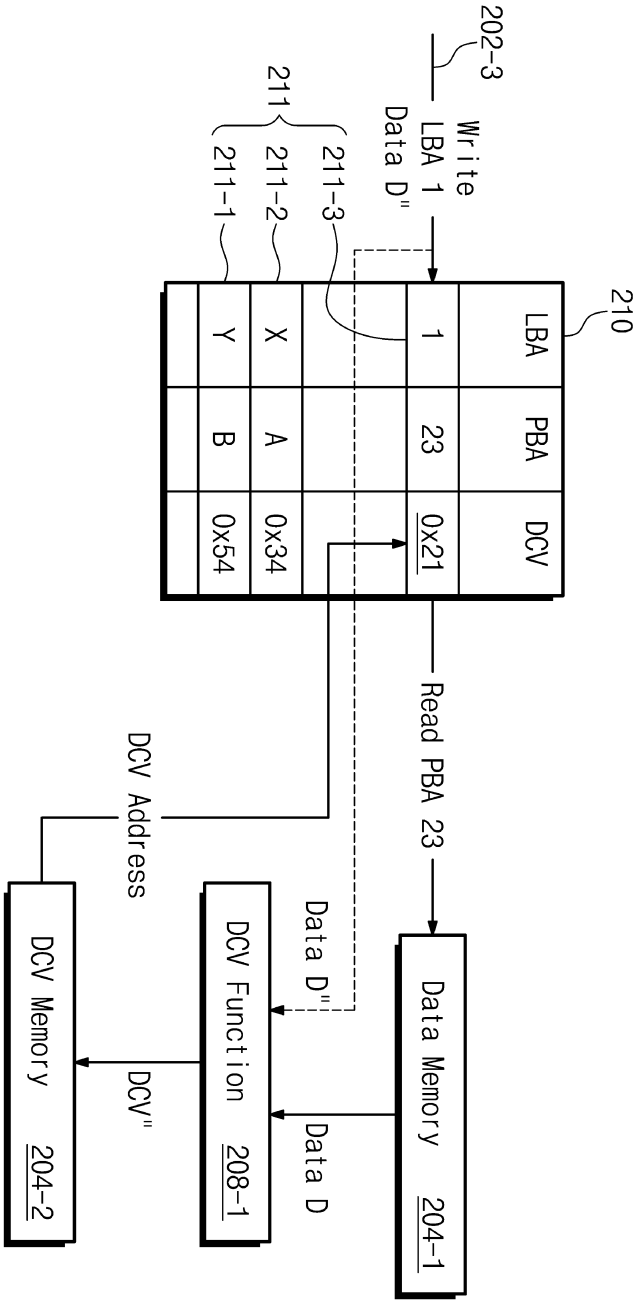
도면2b


202-2
Write
LBA 1
Data D'
211
211-1
211-2
211-3
210
LBA
PBA
DCV
1
23
0x12
X
A
0x34
Y
B
0x54
Read PBA 23
Data Memory 204-1
Data D
DCV Function 208-1
Data D'
DCV'
DCV Address
DCV Memory 204-2

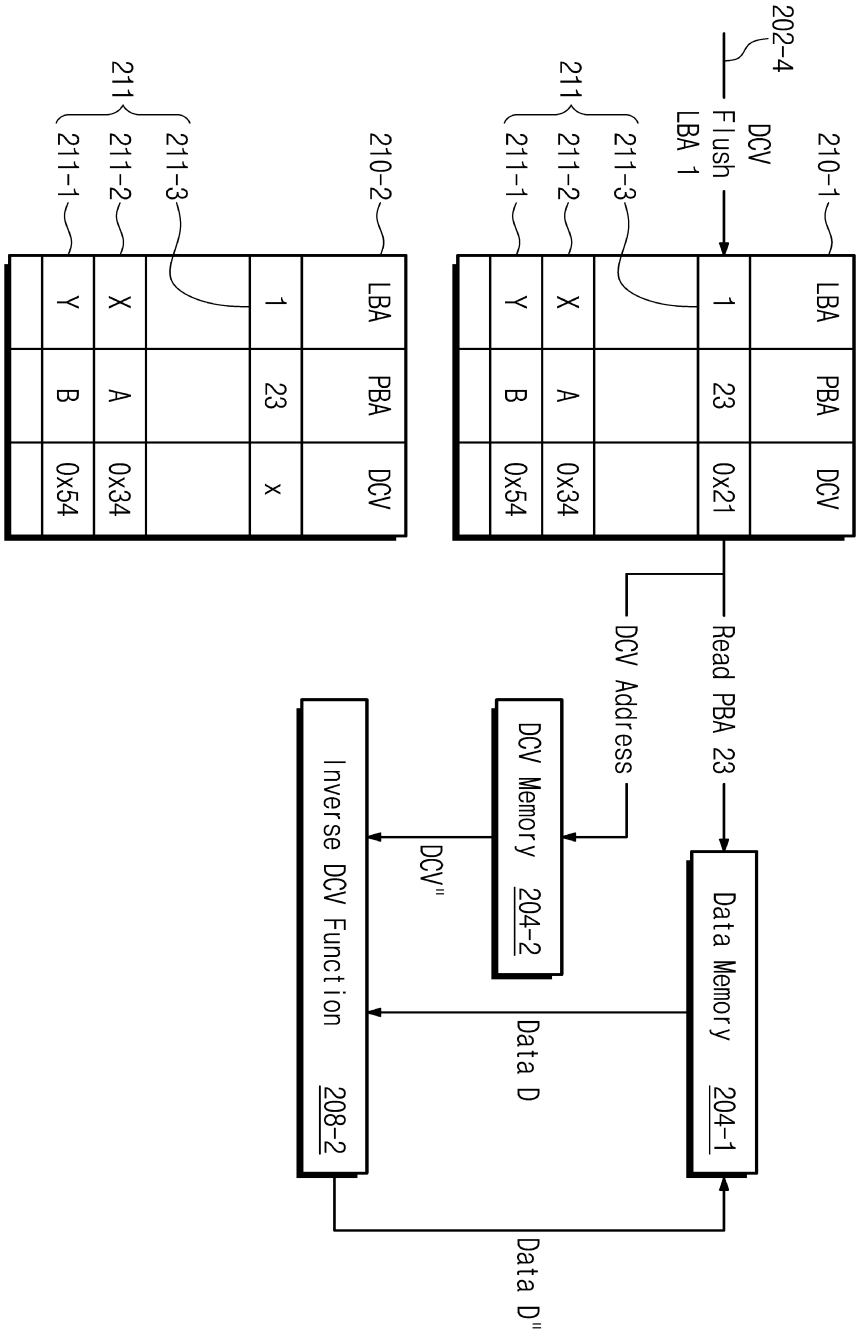
도면2c


202-3
Write
LBA 1
Data D"
211
211-1
211-2
211-3
210
LBA
PBA
DCV
1
23
0x12
X
A
0x34
Y
B
0x54
Read PBA 23
DCV Address
Data D"
Data D
DCV"
Data Memory 204-1
DCV Function 208-1
DCV Memory 204-2

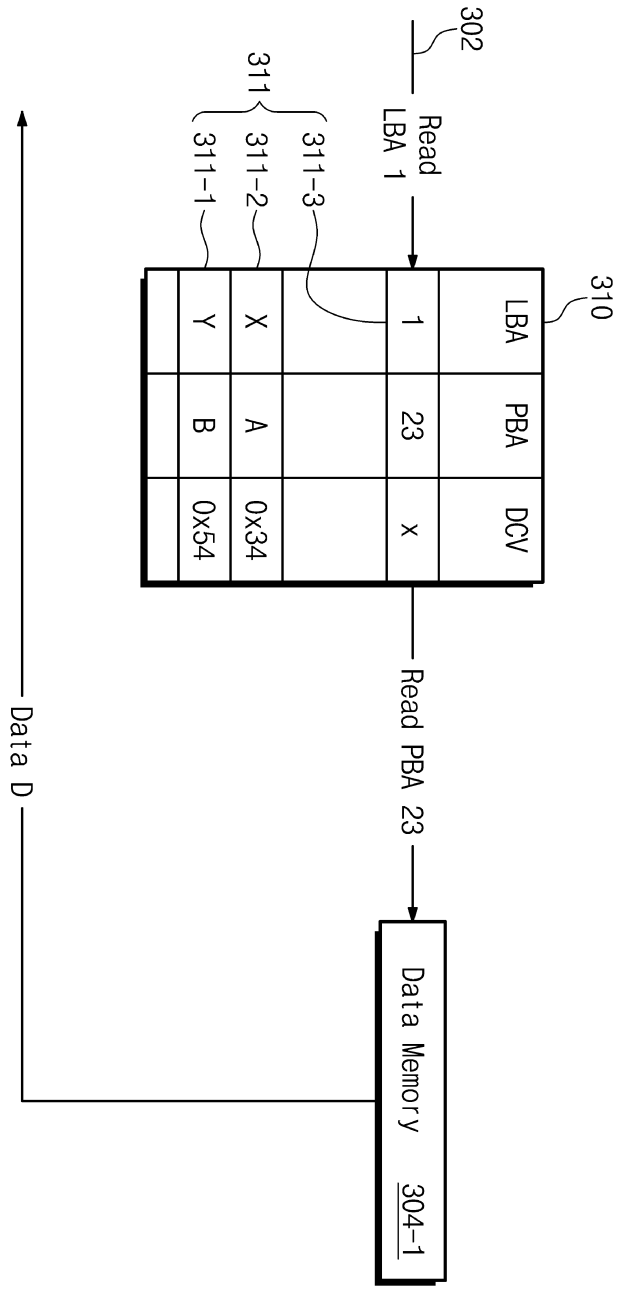
도면2d


202-3
Write
LBA 1
Data D"
211
211-1
211-2
211-3
LBA
PBA
DCV
1
23
0x21
X
A
0x34
Y
B
0x54
210
Read PBA 23
DCV Address
Data D"
Data D
DCV"
Data Memory 204-1
DCV Function 208-1
DCV Memory 204-2

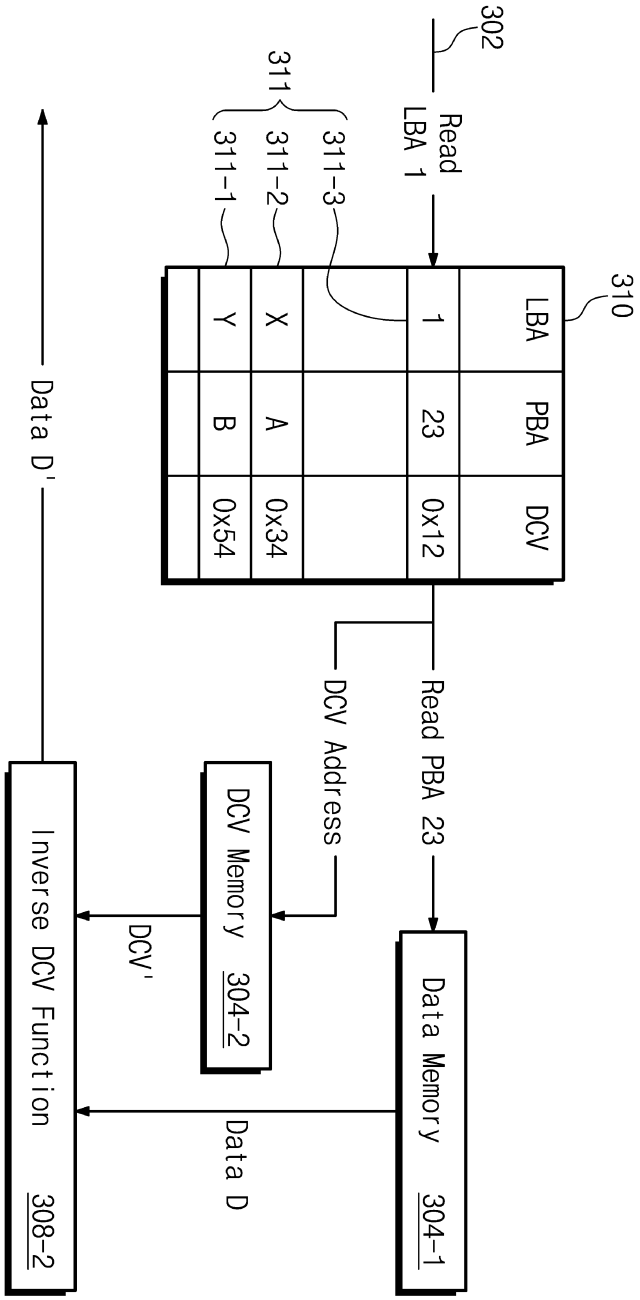
도면2e


202-4
DCV Flush LBA 1
210-1
LBA
PBA
DCV
1
23
0x21
211
211-1
211-2
211-3
Y
B
0x54
X
A
0x34
210-2
x
DCV Address
Read PBA 23
DCV Memory 204-2
DCV''
Data Memory 204-1
Data D
Inverse DCV Function 208-2
Data D''

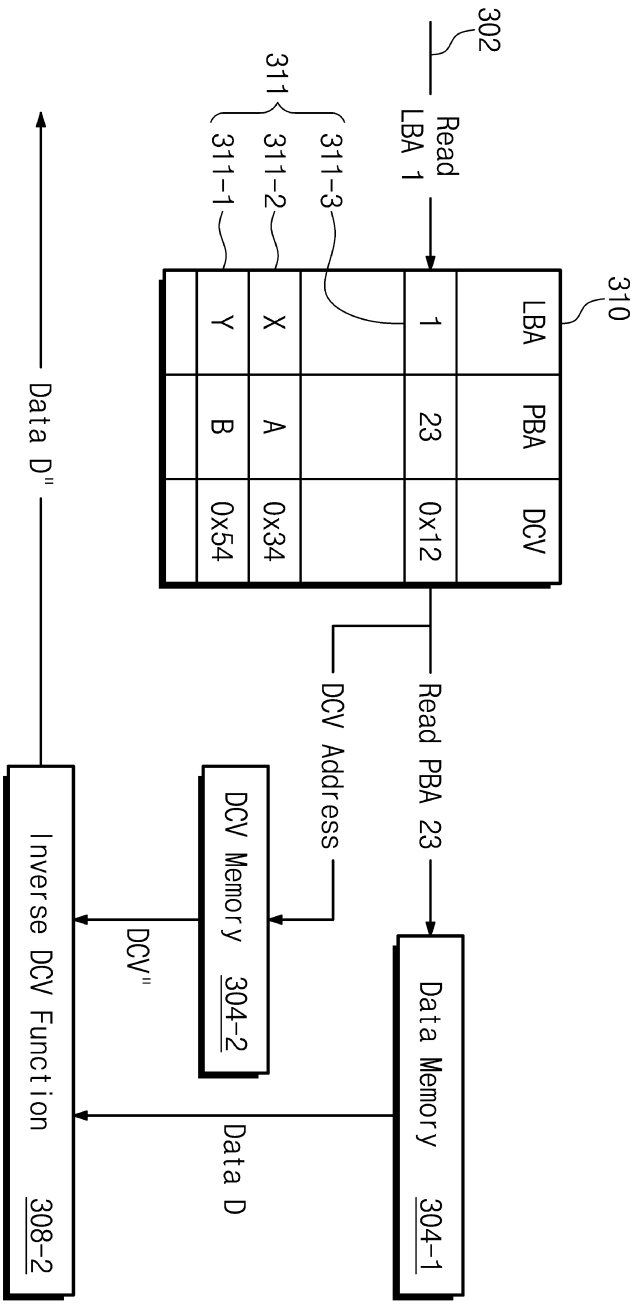
도면3a


302
Read
LBA 1
311
311-1
311-2
311-3
310
LBA
PBA
DCV
Y
X
1
B
A
23
0x54
0x34
x
Read PBA 23
Data Memory 304-1
Data D

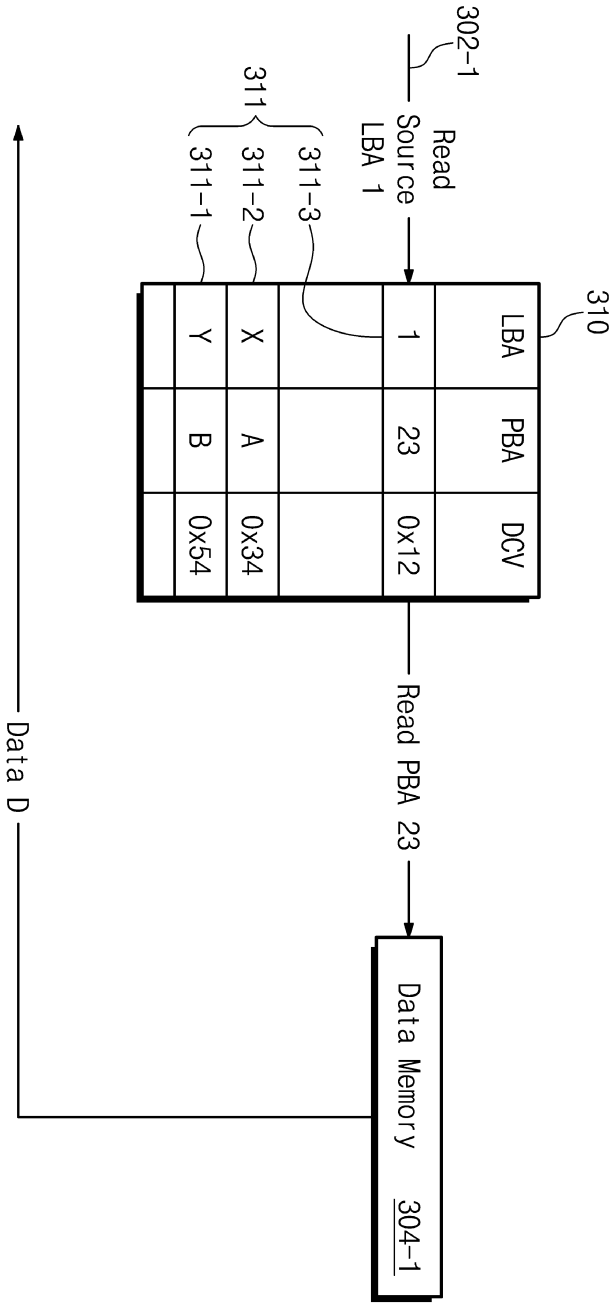
도면3b


302
Read
LBA 1
311
311-1
311-2
311-3
310
LBA
PBA
DCV
1
23
0x12
X
A
0x34
Y
B
0x54
DCV Address
Read PBA 23
DCV Memory 304-2
Data Memory 304-1
DCV'
Data D
Inverse DCV Function 308-2
Data D'

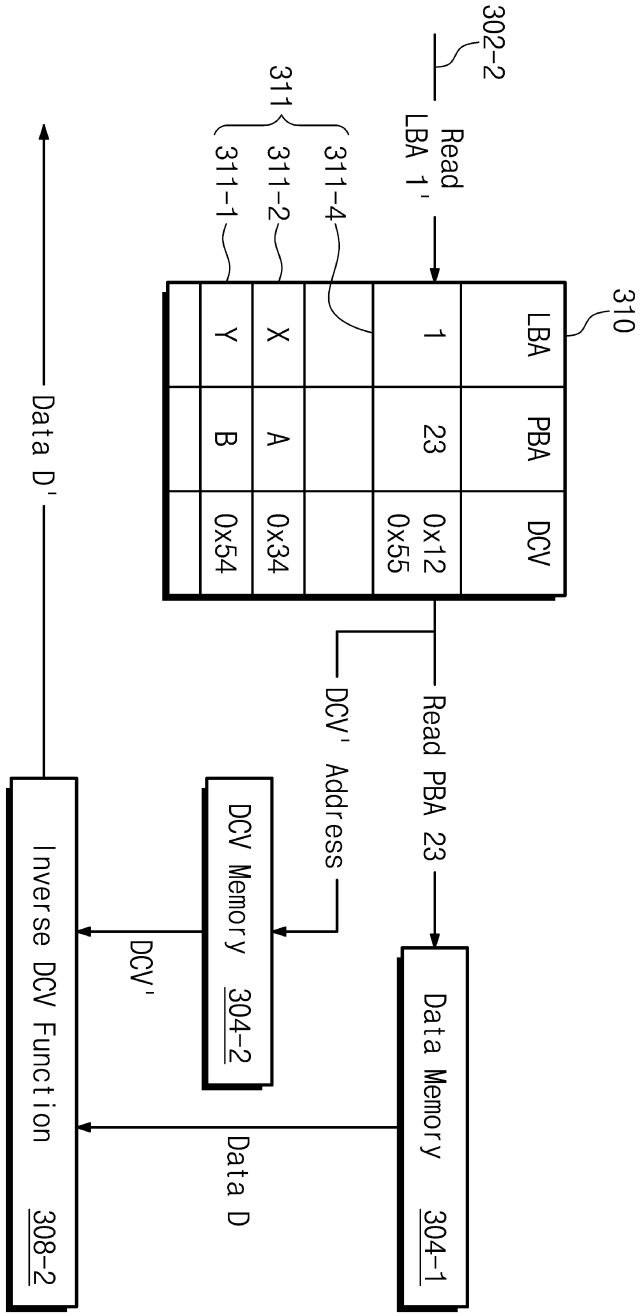
도면3c


302
Read
LBA 1
311
311-3
311-2
311-1
LBA
PBA
DCV
1
23
0x12
X
A
0x34
Y
B
0x54
310
DCV Address
Read PBA 23
Data Memory 304-1
DCV Memory 304-2
DCV"
Data D
Inverse DCV Function 308-2
Data D"

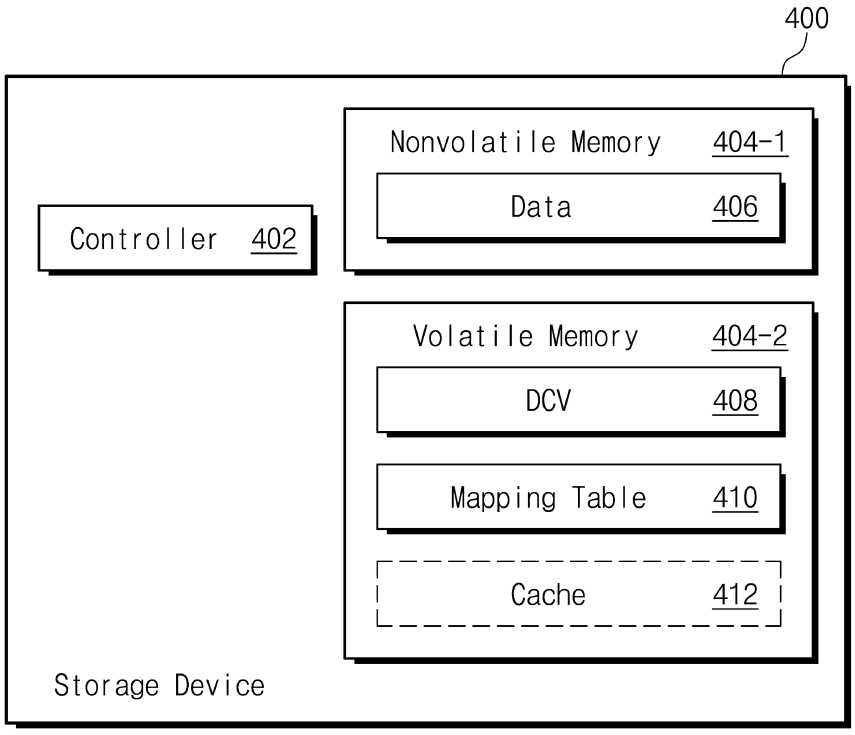
도면3d


302-1
Read
Source
LBA 1
311
311-1
311-2
311-3
310
LBA
PBA
DCV
1
23
0x12
X
A
0x34
Y
B
0x54
Read PBA 23
Data Memory 304-1
Data D

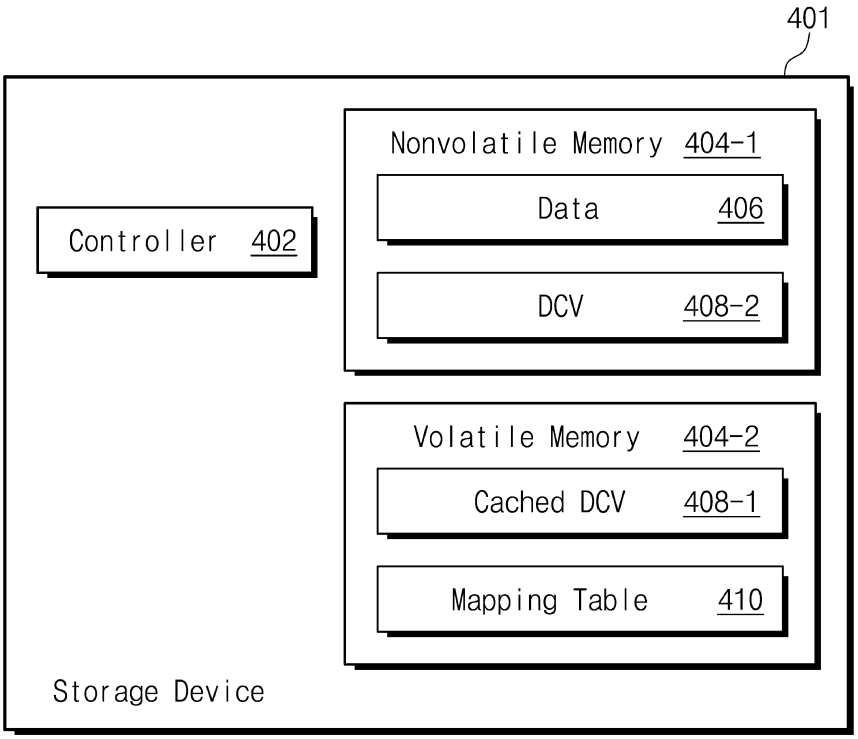
도면3e


302-2
Read
LBA 1'
311
311-1
311-2
311-4
LBA
PBA
DCV
1
23
0x12
0x55
X
A
0x34
Y
B
0x54
310
DCV' Address
Read PBA 23
Data Memory 304-1
DCV Memory 304-2
DCV'
Data D
Inverse DCV Function 308-2
Data D'

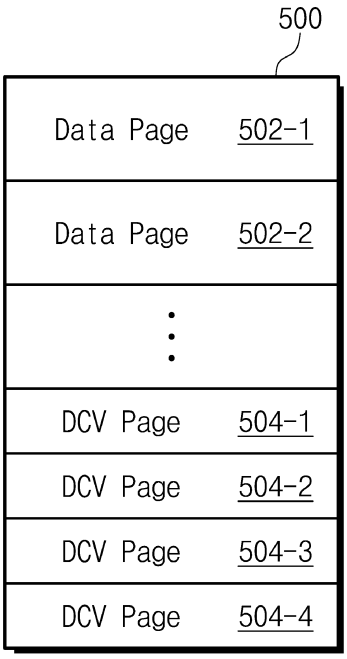
도면4a


400
Nonvolatile Memory 404-1
Data 406
Controller 402
Volatile Memory 404-2
DCV 408
Mapping Table 410
Cache 412
Storage Device

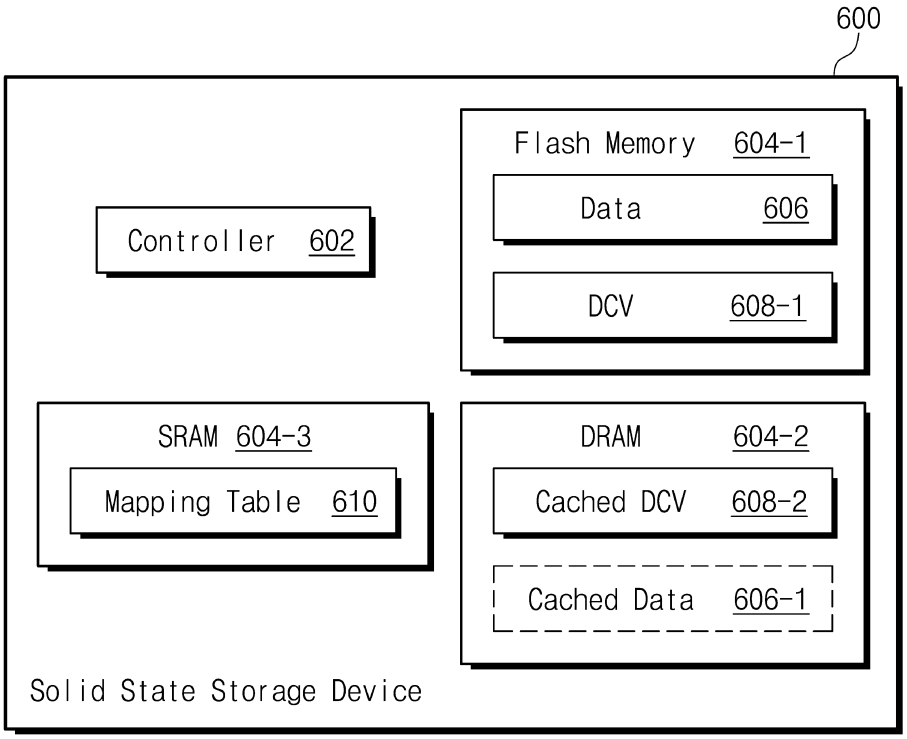
도면4b


401
Nonvolatile Memory 404-1
Data 406
Controller 402
DCV 408-2
Volatile Memory 404-2
Cached DCV 408-1
Mapping Table 410
Storage Device

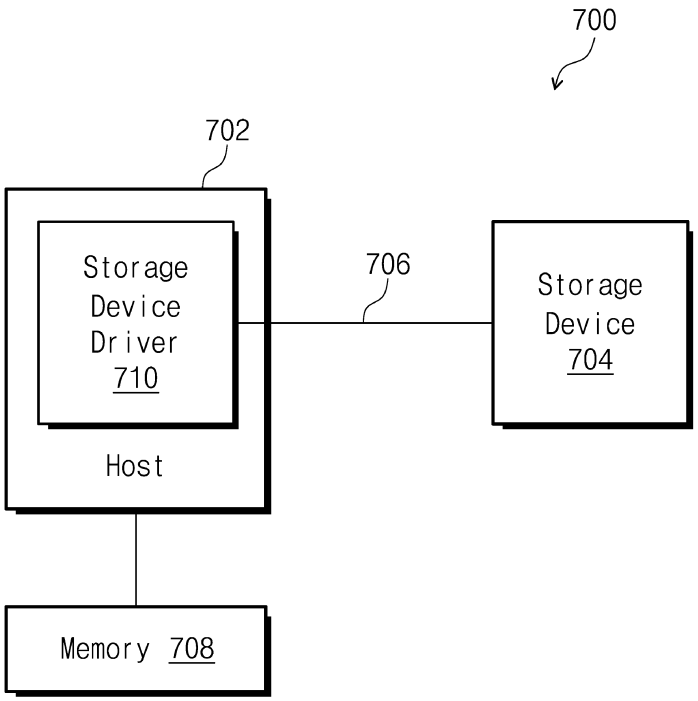
도면5


500
Data Page 502-1
Data Page 502-2
DCV Page 504-1
DCV Page 504-2
DCV Page 504-3
DCV Page 504-4

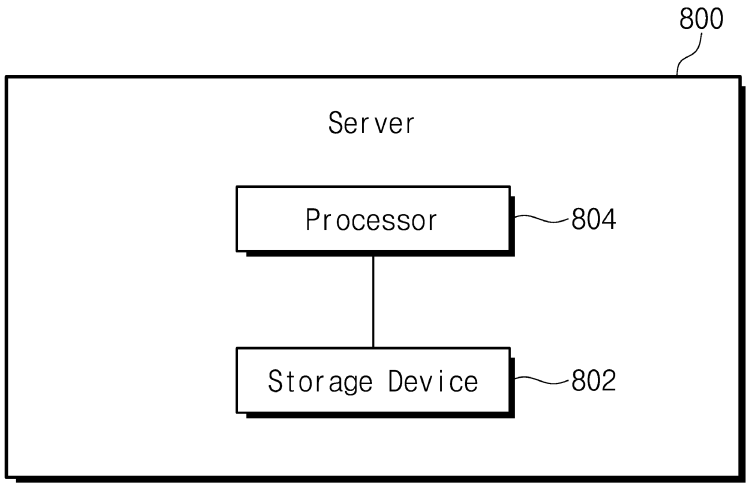
도면6


600
Flash Memory 604-1
Data 606
DCV 608-1
Controller 602
SRAM 604-3
Mapping Table 610
DRAM 604-2
Cached DCV 608-2
Cached Data 606-1
Solid State Storage Device

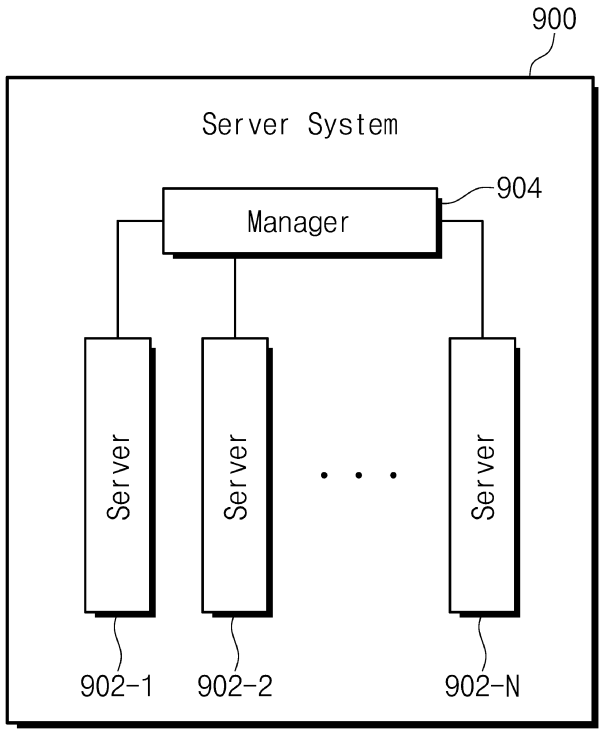
도면7


700
702
Storage Device Driver 710
706
Storage Device 704
Host
Memory 708

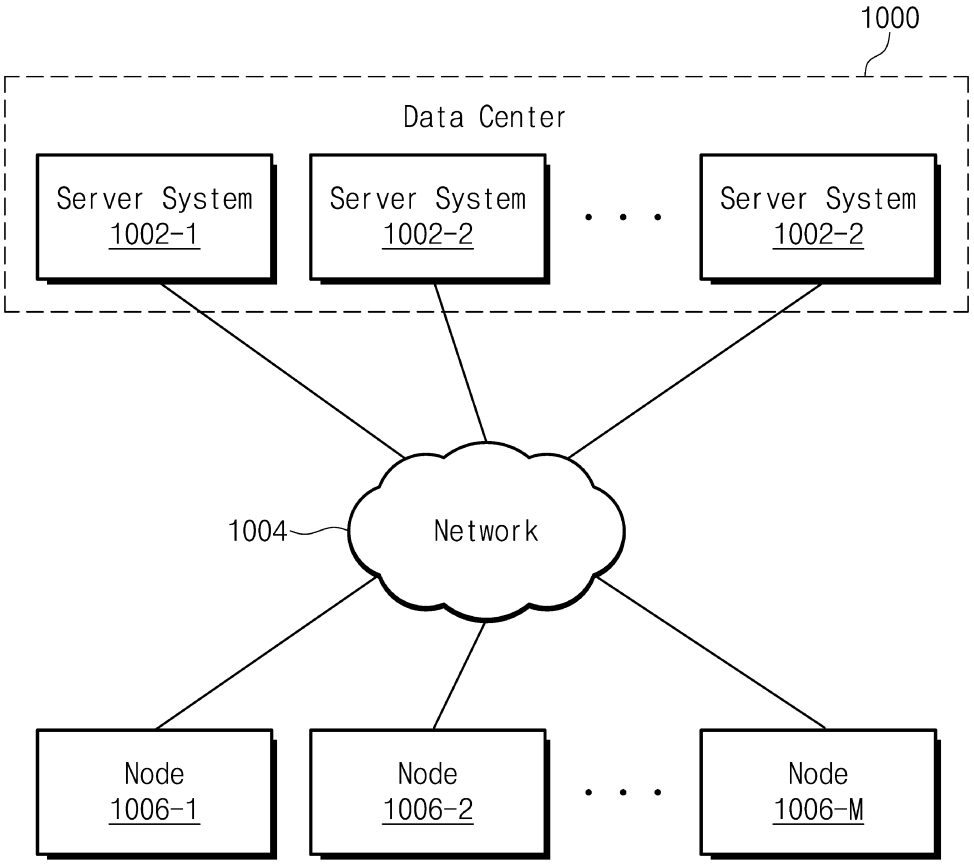
도면8


800
Server
Processor
804
Storage Device
802

도면9


900
Server System
904
Manager
Server
Server
Server
902-1
902-2
902-N

도면10


1000
Data Center
Server System
1002-1
Server System
1002-2
Server System
1002-2
1004
Network
Node
1006-1
Node
1006-2
Node
1006-M