(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2011-0082978
(43) 공개일자 2011년07월20일

(51) Int. Cl.
G06F 9/44 (2006.01) *G06F 11/10* (2006.01)
G06F 12/00 (2006.01)
(21) 출원번호 10-2010-0002953
(22) 출원일자 2010년01월13일
심사청구일자 2010년01월13일

(71) 출원인
성균관대학교산학협력단
경기 수원시 장안구 천천동 300 성균관대학교내
(72) 발명자
임수준
경기도 용인시 기흥구 언남동 동부센트레빌아파트
신동군
경기도 과천시 원문동 래미안슈르아파트 309동 403호
(74) 대리인
김인철, 특허법인세하

전체 청구항 수 : 총 15 항

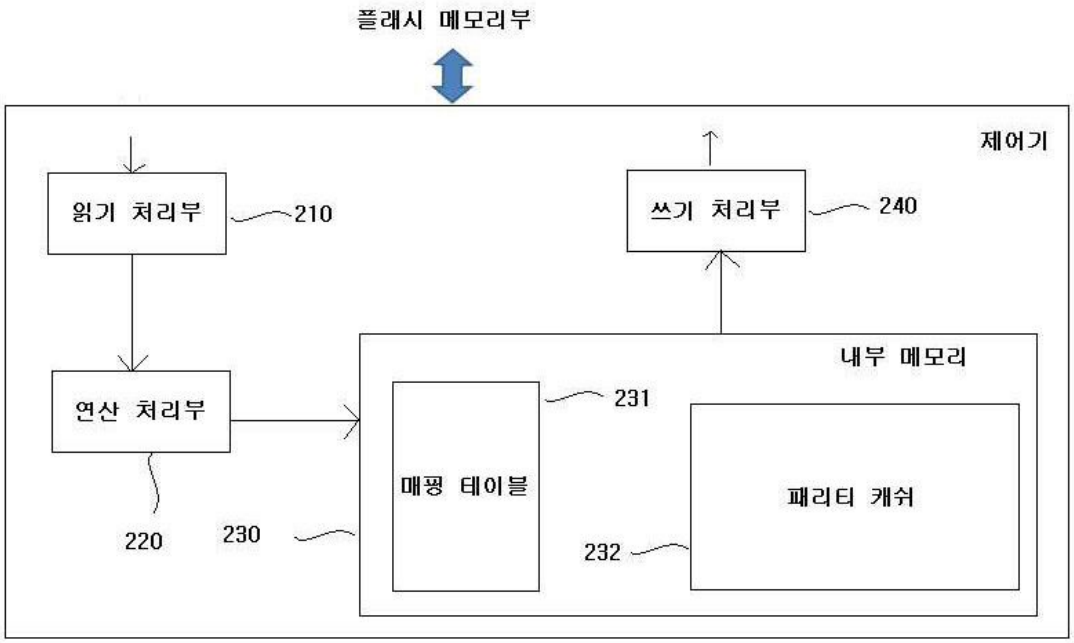
(54) 플래시 메모리를 사용하는 RAID 저장 장치에서 패리티 데이터 관리 방법 및 그 장치

(57) 요 약

본 발명은 데이터 RAID(Redundant Array of Inexpensive Disks, 이하'RAID'라 함) 저장 장치에서 데이터를 관리하기 위한 방법 및 그 장치에 관한 것으로, 특히 플래시 메모리를 사용하는 RAID 저장 장치에서 패리티 데이터를 관리하기 위한 방법 및 그 장치에 관한 것이다.

본 발명의 일 실시 예에 따른 방법은, RAID 장치에서 데이터 관리 방법으로, 저장 장치들에 저장되어 있는 데이터의 갱신이 필요한 경우 패리티 캐쉬에 갱신할 데이터의 스트라이프에 대응하는 부분 패리티 엔트리가 저장되어 있는가를 검사하는 과정과, 상기 부분 패리티 엔트리가 존재하고, 갱신할 데이터가 언커미트 데이터가 아닌 경우 새로운 부분 패리티를 계산하여 부분 패리티를 변경하는 과정과, 상기 갱신이 필요한 데이터를 상기 저장 장치들 중 해당 저장 장치에 기록하는 과정을 포함한다.

대 표 도



특허청구의 범위

청구항 1

RAID 장치에서 데이터 관리 방법에 있어서,

저장 장치들에 저장되어 있는 데이터의 갱신이 필요한 경우 패리티 캐쉬에 갱신할 데이터의 스트라이프에 대응하는 부분 패리티 엔트리가 저장되어 있는가를 검사하는 과정과,

상기 부분 패리티 엔트리가 존재하고, 갱신할 데이터가 언커미트 데이터가 아닌 경우 새로운 부분 패리티를 계산하여 부분 패리티를 변경하는 과정과,

상기 갱신이 필요한 데이터를 상기 저장 장치들 중 해당 저장 장치에 기록하는 과정을 포함하는, RAID 저장 장치에서 패리티 데이터 관리 방법.

청구항 2

제 1 항에 있어서,

상기 부분 패리티 엔트리가 존재하고, 갱신할 데이터가 언커미트 데이터인 경우 이전 데이터를 읽어와 부분 패리티를 계산하여 부분 패리티를 갱신하는 과정을 더 포함하는, RAID 저장 장치에서 패리티 데이터 관리 방법.

청구항 3

제 1 항에 있어서,

상기 부분 패리티 엔트리가 존재하지 않는 경우이고, 상기 패리티 캐쉬가 가득 찬 상태가 아닌 경우 상기 패리티 캐쉬에 새로운 공간을 할당하는 과정과,

새로이 할당된 공간에 갱신할 데이터에 대응하는 부분 패리티를 기록하는 과정을 더 포함하는, RAID 저장 장치에서 패리티 데이터 관리 방법.

청구항 4

제 3 항에 있어서,

상기 부분 패리티 엔트리가 존재하지 않는 경우이고, 상기 패리티 캐쉬가 가득 찬 상태인 경우 부분 패리티의 삭제 대상을 선정하고, 삭제 대상의 부분 패리티들을 상기 저장 장치에 기록하는 과정과,

상기 저장 장치에 기록된 부분 패리티들을 상기 패리티 캐쉬에서 삭제하는 과정을 더 포함하는, RAID 저장 장치에서 패리티 데이터 관리 방법.

청구항 5

제 4 항에 있어서, 상기 부분 패리티를 상기 패리티 캐쉬에서 삭제하는 과정은,

상기 삭제 대상 패리티가 전체 패리티인 경우 상기 삭제 대상 패리티를 상기 저장 장치에 그대로 기록하는, RAID 저장 장치에서 패리티 데이터 관리 방법.

청구항 6

제 4 항에 있어서, 상기 부분 패리티를 상기 패리티 캐쉬에서 삭제하는 과정은,

상기 삭제 대상 패리티가 전체 패리티가 아닌 경우 해당 스트라이프의 언커미티드 데이터의 수가 전체 데이터의 1/2을 초과하는 경우 언커미티드 데이터를 제외한 데이터를 상기 저장 장치로부터 읽어와 상기 저장 장치에 기록할 패리티를 계산한 후 상기 저장 장치에 계산된 패리티를 저장하는, RAID 저장 장치에서 패리티 데이터 관리 방법.

청구항 7

제 4 항에 있어서, 상기 부분 패리티를 상기 패리티 캐쉬에서 삭제하는 과정은,

상기 삭제 대상 패리티가 전체 패리티가 아닌 경우 해당 스트라이프의 언커미티드 데이터의 수가 전체 데이터의 1/2 이하인 경우 언커미티드 데이터의 이전 데이터와 이전 패리티를 상기 저장 장치로부터 읽어와 상기 저장 장치에 기록할 패리티를 계산한 후 상기 저장 장치에 계산된 패리티를 저장하는, RAID 저장 장치에서 패리티 데이터 관리 방법.

청구항 8

제 1 항 내지 제 7 항 중 어느 한 항에 있어서, 상기 부분 패리티는,

상기 데이터의 스트라이프 인덱스와, 부분 패리티 비트 맵과, 갱신될 데이터에 의한 패리티로 구성되는, RAID 저장 장치에서 패리티 데이터 관리 방법.

청구항 9

RAID 기법을 이용하여 데이터를 관리하기 위한 장치에 있어서,

각 저장 장치들로부터 데이터를 읽어오는 읽기 처리부와,

읽어온 데이터로부터 부분 패리티를 계산하기 위한 연산 처리부와,

신규 데이터 또는 갱신 데이터 또는 패리티 갱신 시에 상기 각 저장 장치로 정보를 기록하는 쓰기 처리부와,

상기 부분 패리티를 저장하며, 상기 커미트가 되지 않은 데이터의 주소와 이전 데이터의 주소를 가지는 내부 메모리를 포함하는, RAID 저장 장치에서 패리티 데이터 관리 장치.

청구항 10

제 9 항에 있어서, 상기 내부 메모리는,

논리적 주소와 커미트가 되지 않은 물리적 주소 및 이전 데이터의 물리적 주소를 매핑하여 저장하는 매핑 테이블과,

상기 부분 패리티를 저장하는 패리티 캐쉬를 포함하는, RAID 저장 장치에서 패리티 데이터 관리 장치.

청구항 11

제 10 항에 있어서, 상기 패리티 캐쉬는,

다수의 엔트리를 가지며, 상기 각 엔트리는,

갱신할 데이터의 스트라이프 인덱스와, 상기 갱신할 데이터의 부분 패리티 비트 맵과, 상기 갱신할 데이터를 이용하여 계산한 패리티로 구성되는, RAID 저장 장치에서 패리티 데이터 관리 장치.

청구항 12

제 11 항에 있어서, 상기 패리티 캐쉬의 크기는,

상기 RAID 디스크 내의 전체 스트라이프의 개수를 S라고 하고, 상기 부분 패리티 비트맵 정보를 N bits라 하며, 상기 부분 패리티 데이터를 P bits라 하면, 상기 패리티 캐쉬의 크기는 C($\log_2 S + N + P$) bits인, RAID 저장 장치에서 패리티 데이터 관리 장치.

청구항 13

제 10 항 또는 제 11 항에 있어서, 상기 패리티 캐쉬는,

상기 패리티 캐쉬가 가득 찬 상태에서 상기 데이터 갱신 시에 새로운 패리티 엔트리를 생성해야 하는 경우, 상기 패리티 캐쉬는 삭제 대상 패리티 엔트리를 결정하고, 상기 삭제 대상 패리티 엔트리가 전체 패리티인 경우 전체 패리티를 상기 쓰기 처리부를 통해 상기 저장 장치의 해당 부분에 기록한 후 해당 엔트리를 삭제하는, RAID 저장 장치에서 패리티 데이터 관리 장치.

청구항 14

제 13 항에 있어서, 상기 패리티 캐쉬는,

상기 삭제 대상 패리티 엔트리가 전체 패리티가 아닌 경우 상기 저장 장치의 해당 스트라이프의 언커미티드 데이터의 수가 전체 데이터의 1/2을 초과하는 경우 언커미티드 데이터를 제외한 데이터를 상기 저장 장치로부터 읽어와 상기 저장 장치에 기록할 패리티를 계산한 후 상기 쓰기 처리부를 통해 상기 저장 장치에 계산된 패리티를 저장하는, RAID 저장 장치에서 패리티 데이터 관리 장치.

청구항 15

제 13 항에 있어서, 상기 패리티 캐쉬는,

상기 삭제 대상 패리티가 전체 패리티가 아닌 경우 해당 스트라이프의 언커미티드 데이터의 수가 전체 데이터의 1/2 이하인 경우 언커미티드 데이터의 이전 데이터와 이전 패리티를 상기 저장 장치로부터 읽어와 상기 쓰기 처리부를 통해 상기 저장 장치에 기록할 패리티를 계산한 후 상기 저장 장치에 계산된 패리티를 저장하는, RAID 저장 장치에서 패리티 데이터 관리 장치.

명 세 서

기 술 분 야

[0001] 본 발명은 데이터 RAID(Redundant Array of Inexpensive Disks, 이하'RAID'라 함) 저장 장치에서 데이터를 관리하기 위한 방법 및 그 장치에 관한 것으로, 특히 플래시 메모리를 사용하는 RAID 저장 장치에서 패리티 데이터를 관리하기 위한 방법 및 그 장치에 관한 것이다.

배 경 기 술

[0002] 일반적으로 전자 장비에서 데이터를 저장하는 기기를 저장 매체(storage)라 한다. 이러한 저장 매체로는 하드 디스크, 광학을 이용한 컴팩트 디스크(CD), DVD, 플래시 메모리 등이 있다. 저장 매체 중 하드 디스크나 플래시 메모리의 경우는 대체로 읽기와 쓰기가 가능한 저장 매체이며, 광학을 이용하는 CD 또는 DVD 등은 대체로 한 번의 쓰기만 가능하며, 이후에는 읽기 전용으로 사용된다. 따라서 데이터의 기록, 읽기 및 갱신을 위해서는 일반적으로 하드 디스크나 플래시 메모리를 사용하게 된다.

[0003] 한편, 기술이 급속도로 발전하면서 전송을 위해 또는 보관을 위한 데이터의 용량이 증가하게 되었다. 따라서 이

러한 데이터들을 보다 안전하게 저장하기 위한 방법들이 제안되어 사용되고 있다. 데이터를 안전하게 저장하기 위해서는 하드웨어적으로 안전장치를 마련하는 방법과 리던던트 데이터(redundant data)로 저장할 데이터와 그에 해당하는 패리티 데이터를 함께 저장하는 방법이 있다.

[0004] 일반적으로 전자기기에서 하드 디스크와 플래시 메모리는 다양한 분야에 사용되고 있으며, 플래시 메모리가 일반적으로 하드 디스크보다 데이터에 대한 접근성이 용이하다. 일반적으로 하드 디스크와 플래시 메모리는 모두 여러 번의 읽기 및 쓰기가 가능한 저장 매체이므로, 데이터가 갱신되는 경우에 앞에서 언급한 바와 같이 데이터를 안전하게 저장하기 위한 기법이 필요하다. 데이터를 안전하게 이용하기 위한 기법 중 하나로 RAID 기법이 있다.

[0005] RAID 기법에서는 데이터의 안정성을 위해서 앞에서 설명한 바와 같이 리던던트 데이터(redundant data)로 패리티를 저장한다. RAID 기법은 여러 개의 저장 장치에 데이터를 나누어 저장하는 기법이다. 이러한 RAID 기법은 데이터를 나누는 방법에 따라 7개의 레벨로 분류된다. 특히, RAID 4와 RAID 5는 3개 이상의 저장장치에 데이터를 나누어 기록하는 한편, 전체 시스템의 안정성을 위해서 패리티를 각 저장장치에 분산시켜 기록한다. 이때 하나의 패리티를 공유하는 데이터들을 하나의 스트라이프(stripe)라고 부른다. 패리티를 기록함으로써 이후에 어느 한 저장장치에 에러가 발생하였을 때 패리티 정보와 나머지 저장장치의 정보를 사용하여 에러를 복구하는 것이 가능하다.

[0006] 한편, 플래시 메모리(Flash memory)는 빠른 속도 및 안정성, 가벼운 무게와 전력 소모가 적다는 등의 장점을 가지고 있어 최근 사용이 급속도로 증가하고 있는 저장 장치이다. 특히, 대용량을 필요로 하는 워크스테이션에서 기존의 하드 디스크가 소모하는 전력이 매우 컸던 것에 비해서 플래시 메모리를 사용하면 전력 사용량을 크게 절약할 수 있을 것으로 기대된다. 따라서 최근에는 플래시 메모리를 이용한 RAID 저장 장치의 사용이 확산되고 있다. 그러나 플래시 메모리는 읽기/쓰기 연산(Page 단위)과 삭제 연산(Block 단위)의 단위가 다르며 데이터를 저장하고자 할 때 미리 해당 위치를 삭제해야하는 기록 전 삭제(erase-before-write) 특성을 가지고 있다. 또한 플래시 메모리의 각 블록은 최대 가능 삭제 횟수가 있기 때문에 쓰기와 삭제를 반복함에 따라 플래시 메모리의 수명이 다하게 된다.

[0007] 위와 같은 플래시 메모리 저장 장치의 특징에 따른 쓰기 연산의 성능 저하를 방지하고자, 다양한 FTL(Flash Translation Layer)이나 플래시 파일 시스템(Flash File System)이 개발되었다. 이러한 기법에서는 플래시 메모리에 기록되어 있는 데이터에 대한 갱신이 필요할 때, 기존의 데이터가 속한 블록(block)을 삭제하는 대신에 기존의 데이터가 더 이상 유효하지 않다는 비유효(invalid)를 마킹(marking)한 후 새로운 데이터를 다른 페이지에 기록한다. 비유효 페이지(Invalid page)는 이후에 불필요 데이터 수집(garbage collection) 연산에 의해서 삭제되어 새로운 데이터 기록을 위한 저장 공간으로 재활용된다. 위와 같이 실제 데이터가 기록되는 위치가 매번 변화하기 때문에 플래시 메모리 관리 기법에서는 매핑 테이블(mapping table)이라는 자료 구조를 사용하여 호스트로부터의 읽기/쓰기 요청이 발생한 논리적 주소(logical address)를 실제 데이터가 저장된 공간인 물리적 주소(physical address)로 변환하는 기법을 사용하고 있다.

[0008] 플래시 메모리는 읽기 연산의 속도가 매우 빠른 반면 쓰기 연산은 읽기에 비해서 느리다는 단점이 있다. 또한, 쓰기는 결국에 플래시 메모리 블록에 대한 삭제를 유발하게 되고, 삭제를 많이 할수록 플래시 메모리의 수명을 소진하게 된다. 그러므로 잦은 데이터의 갱신(update)은 플래시 메모리 저장 장치의 성능과 수명에 매우 나쁜 영향을 미친다.

발명의 내용

해결하려는 과제

[0009] 따라서 본 발명에서는 플래시 메모리로 RAID를 구성하는 경우 플래시 메모리의 수명을 증대시킬 수 있는 방법 및 그 장치를 제공한다.

[0010] 또한 본 발명에서는 플래시 메모리로 RAID를 구성하는 경우 플래시 메모리의 성능을 향상시킬 수 있는 방법 및 그 장치를 제공한다.

[0011] 또한 본 발명에서는 플래시 메모리로 RAID를 구성하는 경우 저장된 데이터에 발생한 오류의 복구를 효율적으로 할 수 있는 방법 및 장치를 제공한다.

과제의 해결 수단

[0012] 본 발명의 일 실시 예에 따른 방법은, RAID 장치에서 데이터 관리 방법으로, 저장 장치들에 저장되어 있는 데이터의 갱신이 필요한 경우 패리티 캐쉬에 갱신할 데이터의 스트라이프에 대응하는 부분 패리티 엔트리가 저장되어 있는가를 검사하는 과정과, 상기 부분 패리티 엔트리가 존재하고, 갱신할 데이터가 언커미트 데이터가 아닌 경우 새로운 부분 패리티를 계산하여 부분 패리티를 변경하는 과정과, 상기 갱신이 필요한 데이터를 상기 저장 장치들 중 해당 저장 장치에 기록하는 과정을 포함한다.

[0013] 본 발명의 일 실시 예에 따른 장치는, RAID 기법을 이용하여 데이터를 관리하기 위한 장치로, 각 저장 장치들로부터 데이터를 읽어오는 읽기 처리부와, 읽어온 데이터로부터 부분 패리티를 계산하기 위한 연산 처리부와, 신규 데이터 또는 갱신 데이터 또는 패리티 갱신 시에 상기 각 저장 장치로 정보를 기록하는 쓰기 처리부와, 상기 부분 패리티를 저장하며, 상기 커미트가 되지 않은 데이터의 주소와 이전 데이터의 주소를 가지는 내부 메모리를 포함한다.

발명의 효과

[0014] 본 발명에서와 같이 플래시 메모리를 이용하여 RAID를 구성하는 경우 데이터의 관리가 용이해지며, 플래시 메모리의 수명을 연장할 수 있고, 데이터 처리 속도를 증대시킬 수 있으며, 플래시 메모리에 오류 시 데이터 복구가 용이한 이점이 있다.

도면의 간단한 설명

[0015] 도 1은 일반적인 RAID 기법을 사용하는 경우 데이터 저장 및 갱신 과정을 설명하기 위한 개념도이다.

도 2는 본 발명에 따라 RAID 기법에 사용되는 제어기의 내부 기능 블록 구성도이다.

도 3은 본 발명에 따라 패리티 캐쉬에 저장되는 부분 패리티의 구성도이다.

도 4는 본 발명에 따른 패리티 캐쉬에서 부분 패리티가 변경되는 경우의 한 예를 설명하기 위한 개념도이다.

도 5는 본 발명에 따른 패리티 캐쉬에서 부분 패리티가 변경되는 경우의 다른 예를 설명하기 위한 개념도이다.

도 6은 본 발명에 따라 부분 패리티의 갱신 과정을 설명하기 위한 제어 흐름도이다.

도 7은 본 발명의 바람직한 실시 예에 따른 패리티 커미트 과정을 설명하기 위한 제어 흐름도이다.

도 8은 본 발명에 따라 스트라이프의 부분 패리티가 패리티 캐쉬에 존재하며 복구할 데이터가 부분 패리티에 언커미티드 데이터가 포함된 경우의 복구 예이다.

도 9는 본 발명에 따라 스트라이프의 부분 패리티가 패리티 캐쉬에 존재하며 복구할 데이터가 부분 패리티에 언커미티드 데이터에 포함되지 않은 경우의 복구 예이다.

발명을 실시하기 위한 구체적인 내용

[0016] 이하 첨부된 도면을 참조하여 본 발명을 설명한다. 본 발명을 설명함에 있어 당업자에게 자명한 부분에 대하여는 본 발명의 요지를 흩뜨리지 않도록 생략하기로 한다. 또한 이하에서 설명되는 각 용어들은 본 발명의 이해를 돕기 위해 사용된 것일 뿐이며, 각 제조 회사 또는 연구 그룹에서는 동일한 용도임에도 불구하고 서로 다른 용어로 사용될 수 있음에 유의해야 한다.

[0017] 1. RAID 기법에서 데이터 저장 및 갱신

[0018] 먼저 RAID 기법은 앞에서 살핀 바와 같이 다양한 종류가 있다. 대체로 RAID 기법에서는 데이터의 안정성을 위해서 리던던트 데이터(redundant data)로 패리티를 저장한다. 이러한 패리티를 공유하는 디스크(disk)들을 하나의 스트라이프(stripe)라고 부른다. 또한 RAID 4와 RAID 5는 보통 N+1개의 disk로 구성된다. 그러면 이러한 예를

도면을 참조하여 살펴보기로 한다.

[0019] 도 1은 일반적인 RAID 기법을 사용하는 경우 데이터 저장 및 갱신 과정을 설명하기 위한 개념도이다. 이하 도 1을 참조하여 RAID 4의 경우 데이터가 저장되는 경우와 갱신되는 경우의 각 과정에 대하여 살펴보기로 하자.

[0020] 먼저 도 1의 왼쪽에 도시한 도면을 살펴보면, 디스크들(101, 102, 103, 104, 105)은 5개로 구성되어 있음을 알 수 있다. 그 중 4개의 디스크들(101, 102, 103, 104)은 데이터를 저장하기 위한 디스크들이고, 나머지 하나의 디스크(105)는 패리티 데이터를 저장하기 위한 디스크이다. 즉, 도 1의 구성은 N이 4인 경우이다.

[0021] 디스크 0(101)에는 데이터의 저장을 위한 영역으로 D_0, D_4, D_8, D_{12}가 있으며, 패리티를 저장하기 위한 영역으로 P_4가 있고, 디스크 1(102)에는 데이터의 저장을 위한 영역으로 D_1, D_5, D_9, D_{16}이 있으며, 패리티를 저장하기 위한 영역으로 P_3가 있고, 디스크 2(103)에는 데이터의 저장을 위한 영역으로 D_2, D_6, D_{13}, D_{17}이 있으며, 패리티를 저장하기 위한 영역으로 P_2가 있으며, 디스크 3(104)에는 데이터의 저장을 위한 영역으로 D_3, D_{10}, D_{14}, D_{18}이 있으며, 패리티를 저장하기 위한 영역으로 P_1이 있고, 디스크 4(105)에는 데이터의 저장을 위한 영역으로 D_7, D_{11}, D_{15}, D_{19}가 있으며, 패리티를 저장하기 위한 영역으로 P_0이 있다.

[0022] 한편, 앞에서 설명한 바와 같이 특정의 데이터를 기록할 때는 스트라이프 단위로 이루어진다. 도 1에서 각 스트라이프들(111, 112, …)은 서로 다른 디스크들(101, 102, 103, 104, 105)의 각 데이터 저장을 위한 영역들을 가짐을 알 수 있다. 이때, 스트라이프 0(111)에 D_0의 데이터 변경이 필요하면, 제어기(120)는 해당 데이터와 패리티 P_0를 읽어온다. 이후 제어기(120)는 10의 연산 과정을 거쳐 새로운 패리티 P'_0를 생성한다. 10의 연산 과정은 아래 <수학식 1>과 같이 표현할 수 있다.

수학식 1

[0023] $$P'_0 = D_0 \oplus D'_0 \oplus P_0$$

[0024] 이후 제어기(120)는 도 1의 오른편과 같이 스트라이프 0(111)에서 해당하는 디스크 0(101) 및 디스크 4(105)에 D'_0와 P'_0를 기록한다.

[0025] 다시 도 1을 참조하여 살펴보면, RAID 4의 경우 전체 5개의 디스크로 구성되며, 각 디스크에서 동일한 논리주소 공간에 있는 D_0, D_1, D_2 D_3, P_0은 하나의 스트라이프를 구성하고, P_0은 D_0~D_3에 대한 패리티가 된다. 또한 각 디스크는 독립적인 읽기/쓰기가 가능한 플래시 메모리 plane이나 die, chip이 될 수도 있고 제어기(120)까지 포함한 solid-state disk 형태일 수도 있다.

[0026] 이러한 RAID 구성에서는 패리티를 유지하기 위해서 데이터를 기록할 때마다 패리티를 갱신한다. Parity 갱신 과정은 갱신되는 데이터의 이전 값과 기존의 패리티를 도 1의 왼편에서 설명한 바와 같이 읽어온다. 이후 참조부호 10과 같이 갱신할 데이터와 이전 데이터 및 패리티를 배타적 논리합(xor) 연산을 하여 새로운 패리티를 생성한다.

[0027] 이와 같은 과정을 거치게 되는 경우 즉, D_0를 D'_0으로 변경하기 위해 D'_0을 디스크 0(101)에 기록하려면, D_0 와 P_0를 읽어서 P'_0를 앞에서 설명한 <수학식 1>과 같이 계산한 뒤에, P'_0를 D'_0와 함께 해당하는 디스크들에 각각 기록해야 한다. 그러므로 하나의 데이터를 기록할 때마다 2번의 읽기와 한 번의 쓰기가 추가로 필요하다. 만약 하나의 스트라이프가 모두 갱신되었다면, N개의 데이터에 대해서 읽기와 쓰기가 각각 2배씩 수행된다.

[0028] 이와 같은 일반적인 RAID 기법은 종래기술에서 설명한 바와 같이 데이터 기록에 대한 오버헤드(overhead)가 크다는 것을 알 수 있다. 또한 플래시 메모리 장치에서 쓰기 연산의 횟수가 많다는 점은 저장 장치를 빨리 마모시키기 때문에 매우 부적합하게 된다.

[0029] **2. 본 발명에 따른 기법의 개요**

[0030] 본 발명에서는 플래시 메모리로 구성된 저장 장치를 RAID로 사용할 때 패리티를 임시로 저장하는 캐시(cache)를

사용한다. 또한 본 발명에서는 패리티를 갱신 시마다 바로 기록하지 않고 지연하여 기록(delayed write)하는 기법을 적용한다. 이를 통해 본 발명에서는 종래기술보다 효율적인 데이터 복구가 가능해지는 데이터 복구 방법이 제시된다. 뿐만 아니라 본 발명과 같이 지연하여 패리티를 기록하는 방법을 사용하면, 플래시 메모리로 구성된 RAID 저장장치는 패리티 갱신을 위한 읽기/쓰기 연산 횟수를 줄여 성능을 향상시키고 저장 장치의 수명을 증대시키는 것이 가능하다.

[0031] **2-1. 패리티를 지연하여 갱신하는 기법(Delayed Parity Update Scheme)**

[0032] 앞에서 살펴본 바와 같이 기존의 RAID 기법에서는 데이터를 기록할 때마다 매번 패리티의 쓰기 연산이 발생한다. 따라서 RAID 기법에서 기록(Write) 연산의 비용이 크며 물리적으로 갱신이 불가능한 플래시 메모리의 특성상 이는 성능의 저하 및 저장 장치의 수명에 악영향을 미친다. 그러므로 플래시 메모리 장치에서 빈번하게 패리티를 기록하게 되는 문제를 해결하기 위해 패리티를 지연하여 갱신하는 기법을 사용한다.

[0033] 먼저 본 발명에서는 RAID의 성능 및 수명의 저하를 방지하기 위해서 패리티를 임시로 저장하는 공간을 사용하며, 이러한 메모리 공간을 "패리티 캐쉬(parity cache)"라고 한다. 패리티 캐쉬는 RAID 제어기에 포함되며, 각 디스크들의 고장 시에도 영향을 받지 않고 가용성이 보장되도록 구현한다. 예를 들어, 전원이 갑자기 차단되었을 때도 데이터가 손실되지 않도록 패리티 캐시의 공간을 NVRAM 예를 들어 PRAM, MRAM, FeRAM, battery-backed RAM 등을 사용하여 구성하는 것이 바람직하다.

[0034] 도 2는 본 발명에 따라 RAID 기법에 사용되는 제어기의 내부 기능 블록 구성도이다.

[0035] 플래시 메모리부는 도 1에서 설명한 각각의 디스크들이 플래시 메모리부에 대응하며, 본 발명에 따른 제어기는 도 1에서 설명한 제어기(120)를 대체하기 위한 구성이다. 먼저 읽기 처리부(210)는 플래시 메모리의 데이터 갱신이 필요한 경우 플래시 메모리부로부터 해당하는 디스크와 해당하는 스트라이프에서 갱신할 데이터 및 패리티를 읽어오는 동작을 수행한다. 이와 같이 읽어온 데이터 및 패리티는 연산 처리부(220)에서 종래기술과 동일한 방법을 통해 새롭게 갱신되는 패리티(이하 '갱신 패리티'라 함)를 계산한다. 이와 같이 계산된 갱신 패리티는 본 발명에 따른 내부 메모리(230)의 패리티 캐쉬(232)의 영역에 저장된다. 또한 내부 메모리(230)에는 패리티 캐쉬(232)에 저장되는 갱신 패리티에 대한 정보 및 갱신되기 전의 패리티(이하 '이전 패리티'라 함) 정보를 알 수 있도록 하는 매핑 테이블(231)을 포함한다. 내부 메모리(230)와 매핑 테이블(231) 및 패리티 캐쉬(232)에 대하여는 이하에서 더 살펴보기로 한다. 또한 제어기에서 플래시 메모리로 데이터 또는 갱신 패리티를 저장하는 경우 쓰기 처리부(240)에서 플래시 메모리에 해당 영역에 데이터 또는/및 패리티를 기록/갱신/삭제를 수행한다.

[0036] 패리티 연산을 효율적으로 수행하고 플래시 메모리 장치에 접근을 최소화하기 위해서 본 발명에 따른 패리티 캐쉬(232)에는 각 stripe의 전체 데이터에 대한 패리티가 아닌 새로 갱신된 데이터의 패리티만을 계산하여 저장한다. 이러한 패리티를 이하에서는 부분 패리티(partial parity)라 칭한다. 본 발명에서와 같이 부분 패리티를 사용하며, RAID를 구성하면, 도 1에서 도시한 디스크들 중 하나가 고장을 일으켜 데이터를 읽을 수 없는 경우에, 다른 디스크에 저장된 이전 버전(old version)이나 부분 패리티를 이용하여 데이터를 복구할 수 있다.

[0037] 도 3은 본 발명에 따라 패리티 캐쉬에 저장되는 부분 패리티의 구성도이다.

[0038] 먼저 구성을 살펴보면, 패리티 캐쉬(232)는 C개의 부분 패리티 엔트리(partial parity entry)(310)를 보관할 수 있다. C 개의 부분 패리티 엔트리들(310)에서 각 부분 패리티 엔트리는 부분 패리티가 어떤 스트라이프에 속해 있는지를 알기 위한 스트라이프 인덱스 영역(311)을 포함하고 있으며, 부분 패리티가 어떤 데이터들로 구성되어 있는지 정보를 보관하기 위한 부분 패리티 비트맵 영역(312)과 부분 패리티를 저장하기 위한 영역(313)을 포함한다.

[0039] 이때 전체 패리티 캐쉬의 크기를 C는 다음과 같은 방법으로 구할 수 있다. 먼저 RAID 디스크 내의 전체 스트라이프의 개수를 S라고 하면, 스프라이프를 위해 $\log_2 S$ bits가 필요하다. 또한 부분 패리티가 어떤 데이터들로 구성되어 있는지 정보를 보관하기 위한 부분 패리티 비트맵 정보를 N bits라 하고, 부분 패리티 데이터P bits라 하면, 전체 패리티 캐쉬(232)의 크기는 $C(\log_2 S + N + P)$ bits 가 된다.

[0040] 이러한 부분 패리티는 앞에서 설명한 바와 같이 데이터가 갱신될 때 생성 및 수정된다. 패리티 캐쉬(232)에 데이터가 속한 스트라이프의 패리티가 존재하지 않을 때는 새로운 부분 패리티 엔트리가 생성된다. 만약에 패리티

캐쉬에 새로운 엔트리를 생성할 공간이 없는 경우 후술할 패리티 캐쉬 대체(parity cache replacement) 과정을 통해서 오래된 패리티를 커미트(commit)하여 빈 공간을 확보한다. 이에 대하여는 아래에서 더 살피기로 한다.

[0041] 부분 패리티는 패리티 비트의 특성을 이용하여 해당 스트라이프에 대한 부분 패리티가 생성된 이후에 갱신된 최신의 데이터의 패리티만을 유지한다. 이와 같이 새로 갱신되어 부분 패리티에 그 값이 적용된 데이터를 아직 플래시 메모리부의 저장 장치 내에 위치한 패리티에 값이 반영되지 않았다는 의미에서 언커미티드(uncommitted) 패리티라 한다.

[0042] 한편, RAID를 구성하는 각 디스크는 덮어쓰기가 안되는 플래시 메모리의 특성 때문에 논리 주소를 물리 주소로 변환해 주는 매핑 테이블(Mapping table)을 일반적으로 가지고 있다. 본 발명과 같이 패리티를 지연하여 기록하는(delayed parity write) 기법에서도 매핑 테이블(231)을 가진다. 이러한 매핑 테이블(231)은 아래 <표 1>과 같이 구성할 수 있다.

표 1

[0043]

lpn	ppn	old ppn
0	25	-
1	26	-
2	53	51
3	14	104
...	...	...
n	64	41

[0044] 위의 <표 1>과 같은 매핑 테이블에서는 이전 데이터(old data) 필드를 추가한다. 예를 들어, 논리주소(lpn) 2번지의 데이터가 원래 물리 주소(ppn) 51번지에 기록되어 있었는데, 해당 데이터가 변경되어 물리 주소 53번지에 기록되면 매핑 테이블은 원래의 물리 주소(ppn)에 새로운 물리주소를 기록하면서, 이전 물리주소(old ppn)에 이전 데이터가 기록된 물리 주소를 기록한다. 이와 같이 매핑 테이블을 구성하는 것은 언커미티드 스트라이프(uncommitted stripe)에서 에러가 발생하면, 이전 물리주소(old data)를 사용하여 이를 복구하기 위해서이다. 이러한 데이터 복구 방법은 아래에서 다시 살펴보기로 한다. 즉, 이전 물리주소는 커미트(commit)가 되기 전의 패리티와 연관된 데이터들의 주소를 기록하고 있다.

[0045] 다음으로 패리티 캐쉬(232)에서 부분 패리티가 변경되는 경우에 대하여 살펴보기로 한다. 부분 패리티가 변경되는 경우는 아래의 2가지 경우이다.

[0046] 첫 번째로, 기록할 데이터가 포함되는 스트라이프의 부분 패리티가 패리티 캐쉬(232)에 존재하며, 갱신할 해당 데이터가 부분 패리티의 언커미티드 데이터(uncommitted data)에 포함되지 않는 경우이다. 이를 첨부된 도 4를 참조하여 살펴보기로 한다.

[0047] 도 4는 본 발명에 따른 패리티 캐쉬에서 부분 패리티가 변경되는 경우의 한 예를 설명하기 위한 개념도이다.

[0048] 도 4에서는 먼저 패리티 캐쉬(232)에 D'_1에 대한 부분 패리티(401)가 존재하는 상태이다. 즉, 저장 장치에 이전의 D_1의 데이터는 유효하지 않은(invalid) 데이터가 되고, 새롭게 갱신된 D_1'의 데이터(411)가 유효한 데이터로 저장되어 있으며, 패리티 캐쉬(232)에 해당 스트라이프의 엔트리가 존재하는 상태이다. 도 4에서는 새로운 데이터로부터 이전 데이터로(D_1)의 화살표를 표시하여 이전 데이터(old data)(412)가 유효하지 않은 데이터임을 도면에 표시하였다. 따라서 패리티 캐쉬(232)에는 해당 스트라이프에 대한 부분 패리티가 존재하는 상태이다. 이러한 경우 해당하는 스트라이프의 다른 디스크에 저장된 즉, D_2'(431)과 같은 데이터 갱신이 요구되면(421), 패리티 캐쉬(232)에 해당하는 엔트리에서 D_1'과 D_2'의 새로운 부분 패리티를 생성한다. 이러한 부분 패리티는 도 4

에 도시한 바와 같이 배타적 논리합을 이용하여 계산할 수 있다. 또한 저장 장치에는 참조부호 431과 같이 갱신이 요구된 데이터(D_2')를 저장하며, 이전 데이터(D_2)는 유효하지 않은 데이터가 된다.

[0049] 즉, 이상에서 설명한 바와 같이 해당 스트라이프에 대한 부분 패리티가 존재하는 경우 해당하는 패리티와 배타적 논리합 연산을 수행하여 부분 패리티를 계산할 수 있다. 이 경우에는 저장 장치에 별도로 접근할 필요가 없으며, 또한 저장 장치에 기록된 기존의 패리티에 영향을 미쳤던 이전 데이터(old data)를 데이터 복구(data recovery) 과정에서 활용하기 위해서 상술한 매핑 테이블의 이전 페이지(old page) 정보를 이전 데이터(old data)를 참조할 수 있게 갱신한다.

[0050] 패리티 캐쉬(232)에서 부분 패리티가 변경되는 두 번째 경우는 기록할 데이터가 포함되는 스트라이프의 부분 패리티가 패리티 캐쉬(232)에 존재하고 해당 데이터가 부분 패리티의 언커미티드 데이터(uncommitted data)에 포함되는 경우이다.

[0051] 도 5는 본 발명에 따른 패리티 캐쉬에서 부분 패리티가 변경되는 경우의 다른 예를 설명하기 위한 개념도이다.

[0052] 도 5는 앞서 설명한 도 4의 경우에 부가하여 갱신이 1회 이상 이루어진 데이터를 다시 갱신하는 경우가 된다. 따라서 참조부호 511과 같이 패리티 캐쉬(232)에 해당하는 엔트리가 존재하는 경우이며, 이때, 해당 엔트리 중 갱신된 데이터에 대하여 재갱신이 요구되는 즉, 도 5에서는 D_2''의 갱신이 요구되는 경우이다. 이러한 경우에는 앞에서와 마찬가지 방법으로 D_2''가 갱신되며, 패리티 캐쉬(232)에는 새로운 부분 패리티가 저장된다.

[0053] 부분 패리티에 대하여 좀 더 살펴보기로 한다. 패리티를 계산하기 위해서 이전 데이터와 새로운 데이터의 차이를 알아야한다. 그러므로 저장 장치로부터 이전 데이터를 읽어온 다음에 기존 부분 패리티와 이전 데이터, 새로운 데이터를 배타적 논리합(xor)하여 새로운 부분 패리티를 계산한다. 이 과정에서는 이전 데이터를 읽기 위해서 1번의 읽기(read)가 필요하다. 위의 예제에서 D_2''(501)을 기록할 때 이전 데이터인 D_2'(431)이 언커미티드된 데이터(uncommitted data)이기 때문에 D_2'와 D_2''의 차이를 계산해야한다. 그러므로 부분 패리티 및 D_2'(431)와 D_2''(501)을 배타적 논리합(xor)하여 새로운 부분 패리티를 계산한다.

[0054] 만일 하나의 스트라이프의 모든 데이터가 언커미티드 데이터가 되면, 즉 해당하는 스트라이프의 모든 데이터가 갱신되어 스트라이프의 모든 데이터의 패리티가 패리티 캐쉬(232)에 계산된 경우에는 이 패리티를 전체 패리티(full parity)라고 한다. 전체 패리티는 패리티 커미트(parity commit) 연산을 할 때 저장 장치에 추가적인 접근을 하지 않고도 패리티를 갱신할 수 있다.

[0055] 도 6은 본 발명에 따라 부분 패리티의 갱신 과정을 설명하기 위한 제어 흐름도이다. 이하 도 6을 참조하여 본 발명에 따른 패리티 갱신 과정을 살펴보기로 한다.

[0056] 먼저 제어기는 600단계에서 패리티 캐쉬(232)에 데이터의 패리티가 존재하는가를 검사한다. 600단계의 검사결과 패리티 캐쉬(232)에 데이터의 패리티가 존재하는 경우 602단계로 진행하고 그렇지 않은 경우 610단계로 진행한다.

[0057] 그러면 먼저 패리티 캐쉬(232)에 데이터의 패리티가 존재하는 경우 즉, 602단계로 진행하는 경우에 대하여 살펴보기로 한다. 제어기는 602단계로 진행하면, 데이터가 언커미티드 데이터인가를 검사한다. 602단계의 검사결과 저장된 데이터가 언커미티드된 데이터인 경우 604단계로 진행하여 데이터의 이전 버전을 읽어온 후 606단계로 진행한다. 반면에 602단계의 검사결과 데이터가 언커미티드된 데이터가 아닌 경우 바로 606단계로 진행하여 새로운 부분 패리티를 계산한다. 이때 604단계에서 606단계로 진행하면, 이전 버전의 데이터를 함께 이용하여 부분 패리티를 계산한다. 즉, 앞에서 전술한 도 5의 과정과 같이 새로운 패리티를 계산한다. 이후 제어기는 608단계에서 패리티 캐쉬(232)에 부분 패리티를 변경하여 저장한다.

[0058] 한편 600단계의 검사결과 패리티 캐쉬(232)에 데이터의 패리티가 없어 610단계로 진행하는 경우 제어기는 610단계에서 패리티 캐쉬(232)가 꽉 찬 상태인가를 검사한다. 즉, 패리티 캐쉬(232)에 더 이상 새로운 엔트리를 생성할 수 없는 상태인가를 검사하는 것이다. 610단계의 검사결과 패리티 캐쉬(232)가 꽉 찬 상태인 경우 612단계로 진행하며, 그렇지 않은 경우 616단계로 진행한다.

[0059] 먼저 패리티 캐쉬(232)가 꽉 찬 상태로 검사되어 612단계로 진행하면, 제어기는 저장 장치에 대상 부분 패리티

를 선정한다(victim partial parity). 이후 제어기는 614단계로 진행하여 선정된 부분 패리티들을 저장 장치에 기록한다(parity commit). 이러한 과정을 통해 패리티 캐쉬(232)에 저장이 가능한 엔트리들을 생성할 수 있다. 또한 본 발명에서는 저장 장치에 저장할 패리티를 선정하는 기준 등에 대하여는 특별한 제한을 두지 않기로 한다.

[0060] 이후 610단계에서 또는 614단계에서 616단계로 진행하면, 제어기는 새로운 패리티 캐시 공간을 할당한다. 그런 후 제어기는 618단계로 진행하여 새롭게 할당된 패리티 캐시 공간에 도 3에 도시한 바와 같은 형태로 패리티 엔트리에 부분 패리티를 기록한다.

[0061] **2-2. 패리티 커미트 방법**

[0062] 패리티 캐쉬(232)에 보관된 패리티 데이터는 시간이 지나면 저장 장치에 기록되어야 한다. 이러한 연산을 패리티 커미트라고 한다. 패리티 커미트는 다음의 2가지 경우에 발생한다.

[0063] 패리티 커미트가 발생하는 첫 번째 경우는, 언커미티드 데이터(uncommitted data)의 이전 데이터(old data)를 가진 플래시 메모리 블록(block)이 불필요 데이터 수집(Garbage collection)에 의해서 삭제되면 더 이상 이전 데이터를 이용한 데이터 복구를 할 수 없으므로 패리티 커미트가 필요하다. 패리티 커미트가 발생하는 두 번째 경우는 패리티 캐쉬(232)가 가득 차 새로운 부분 패리티를 생성할 공간이 없어 하나의 패리티 데이터를 대상(victim)으로 선정하여 저장 장치에 기록해야 할 때 선정된 패리티에 대한 커미트가 발생한다.

[0064] 만일 전체 패리티가 커미트 될 때에는 이미 패리티가 모두 패리티 캐쉬(232)에서 계산되었기 때문에 별도의 연산 없이 바로 해당 전체 패리티를 저장 장치에 기록할 수 있다. 그러나 부분 패리티에 대하여 커미트가 발생하면, 스트라이프 내에서 언커미티드 데이터의 값만이 패리티에 적용되었기 때문에 나머지 데이터의 값을 취합하여 전체 패리티를 계산할 필요가 있다. 이 때 저장장치의 접근 횟수를 줄이기 위해서 본 발명에서는 스트라이프의 언커미티드 데이터의 수에 따라 패리티 연산 방법을 달리 한다.

[0065] 본 발명에서는 언커미티드 데이터의 수에 따라 아래와 같은 방법으로 패리티를 계산함으로써 패리티 커미트 연산에 필요한 저장 장치에 대한 읽기 연산을 전체 스트라이프의 1/2 이하로 유지할 수 있다.

[0066] (1) 해당 부분 패리티의 언커미티드 데이터의 수가 전체 스트라이프 데이터의 1/2 이상인 경우 : 이러한 경우 모든 디스크들에서 나머지 변경되지 않은 데이터들을 읽어 부분 패리티와 배타적 논리합 연산을 수행함으로써 새로운 패리티를 계산할 수 있다. 예를 들어 RAID 4인 경우 D_0, D_1, D_2, D_3, P가 각각의 디스크에 저장되어 있고, D_1, D_2, D_3들의 데이터가 언커미티드 데이터라면, 부분 패리티는 D_1, D_2, D_3 각각에 대응하여 갱신된 데이터들을 가진 부분 패리티이다. 따라서 실제로 갱신되지 않은 데이터인 D_0에 대하여만 배타적 논리합을 수행함으로써 실제로 저장 장치에 기록할 패리티를 계산할 수 있다.

[0067] (2) 해당 부분 패리티의 언커미티드 데이터의 수가 전체 스트라이프 데이터의 1/2보다 작은 경우에는 언커미티드 데이터의 이전 데이터(old data)와 이전 패리티(old parity)를 읽어 들여 부분 패리티(partial parity)와 배타적 논리합(xor)하여 새로운 패리티를 계산할 수 있다. 예를 들어 RAID 4인 경우 D_0, D_1, D_2, D_3, P가 각각의 디스크에 저장되어 있고, D_1의 데이터가 언커미티드 데이터라면, 부분 패리티는 D_1에 대하여만 부분 패리티가 된다. 따라서 이전 패리티인 P와 D_1 및 D_1의 갱신된 데이터인 D_1'을 배타적 논리합을 통해 실제로 저장 장치에 기록할 새로운 패리티를 계산할 수 있다.

[0068] 이상과 같은 패리티 연산 과정을 마치면 그 결과를 다시 플래시 메모리에 기록하고, 해당 패리티에 연관된 언커미티드 데이터들에 대한 매핑 테이블의 이전 데이터 필드를 삭제(clear)하는 것으로 패리티 커미트 과정이 완료된다.

[0069] 도 7은 본 발명의 바람직한 실시 예에 따른 패리티 커미트 과정을 설명하기 위한 제어 흐름도이다.

[0070] 먼저 제어기는 패리키 커미트가 필요하면 700단계에서 패리티 캐시(232)에 저장된 해당 엔트리의 패리티가 전체 패리티 상태인가를 검사한다. 700단계의 검사결과 전체 패리티 상태라면, 제어기는 710단계로 진행하여 패리티 데이터를 저장 장치에 기록하도록 제어한다. 즉, 전체 패리티는 앞에서 설명한 바와 같이 더 이상의 연산이 필요하지 않기 때문에 바로 패리티 데이터를 저장 장치에 기록할 수 있다.

[0071] 반면에 700단계의 검사결과 패리티 캐시(232)에 저장된 해당 엔트리의 패리티가 전체 패리티가 아닌 경우 제어

기는 702단계로 진행하여 해당 스트라이프의 언커미티드 데이터의 개수가 패리티에 속하지 않은 데이터의 수보다 큰가를 검사한다. 상기 검사결과 스트라이프의 언커미티드 데이터 개수가 패리티에 속하지 않는 데이터의 수보다 큰 경우 704단계로 진행하고, 이하인 경우 706단계로 진행한다.

[0072] 먼저 704단계로 진행하는 경우는 앞에서 설명한 첫 번째 경우에 해당한다. 따라서 제어기는 704단계에서 언커미티드 데이터를 제외한 데이터를 읽어온다. 그런 후 제어기는 708단계에서 앞의 첫 번째 과정에서 설명한 바와 같이 새로운 패리티를 계산하고, 710단계에서 저장 장치에 패리티 데이터를 기록한다. 반면에 706단계로 진행하는 경우는 앞에서 설명한 두 번째 경우에 해당한다. 따라서 제어기는 706단계에서 언커미티드 데이터의 이전 데이터(old data)와 이전 패리티(old parity)를 읽어온다. 그런 후 708단계로 진행하면 제어기는 706단계에서 읽어온 데이터들을 이용하여 새로운 패리티를 계산한다. 이후 제어기는 710단계에서 새로운 패리티를 저장 장치에 기록한다.

[0073] **2-3. 부분 패리티의 불필요 데이터 수집**

[0074] 플래시 메모리 저장 장치에서는 관리 기법에 따라 저장 공간을 확보하기 위해서 오래된 데이터를 삭제하는 불필요 데이터 수집(garbage collection, 이하 "GC"라 함) 연산 등이 발생한다. 이 과정에서 대상(victim)으로 선정된 블록에 어떤 스트라이프의 이전 패리티(old parity)나 이전 페리티를 구성하는 페이지(page)가 포함되어 있고, 이에 대한 부분 패리티가 패리티 캐쉬(232)에 있다고 가정하자. 이 경우에는 해당 페이지가 삭제됨에 따라 더 이상 이전 데이터를 사용한 데이터 복구 방법을 사용할 수 없다. 그러므로 해당 페이지들에 대한 패리티 커미트가 필요하다. 그러므로 패리티 커미트에 의하여 일어나는 플래시 메모리 연산을 최소화하기 위해서 FTL 기법에서도 블록을 삭제할 때 해당 블록에 어느 부분 패리티의 언커미티드 데이터에 대한 이전 데이터가 있는 블록 보다는 그렇지 않은 블록을 대상 블록(victim block)으로 선정될 우선순위를 부여한다.

[0075] **2-4. 패리티 캐쉬 대체(Parity cache replacement)**

[0076] 패리티 캐쉬 대체에 의해서 패리티 캐쉬(232)에서 제거될 패리티 데이터는 커미트가 필요하다. 그러므로 최대한 패리티 커미트의 수를 줄이고 그 비용을 절감할 수 있도록 대상(victim)을 선정하여야 한다. 패리티 데이터는 연관된 페이지 데이터가 갱신되면, 갱신될 필요가 있으므로 더 이상 갱신되지 않을 것으로 보이는 데이터들의 패리티를 대상으로 선정하여 패리티 커미트의 수를 줄이기 위해서 LRU(Least-recently-updated) 알고리즘을 기반으로 하는 것이 적합하다.

[0077] 또한 패리티 커미트의 비용은 부분 패리티의 구성에 따라 달라진다. 예를 들어 전체 패리티의 경우에는 스트라이프의 모든 갱신된 데이터에 대한 패리티가 계산되어 있으므로 추가 연산 비용 없이 패리티를 기록할 수 있다. 그리고 부분 패리티의 경우에는 몇 개의 데이터의 패리티가 적용되었는지에 따라 새로운 패리티 데이터를 계산하는 비용이 달라지므로 이를 기준으로 비용이 적은 패리티 데이터를 대상으로 선정하는 것이 좋다. 결론적으로 커미트 비용이 적고 향후 변경될 가능성이 적은 부분 패리티에 우선순위를 주어서 대상을 선정한다.

[0078] **2-5. 데이터 복구(Data recovery) 방법**

[0079] 일부 데이터 또는 하나의 플래시 메모리 장치에서 내/외부적인 요인에 의하여 데이터에 에러가 발생한 경우에는 다음과 같은 조건에 따라 각 데이터를 복구할 수 있다.

[0080] (1) 스트라이프의 부분 패리티가 패리티 캐쉬(232)에 존재하는 경우 :

[0081] (a) 복구해야 할 데이터가 부분 패리티의 언커미티드 데이터에 포함되는 경우는 다른 언커미티드 데이터와 부분 패리티를 이용하여 데이터를 복구할 수 있다. 이를 도 8을 참조하여 살펴보기로 한다. 도 8은 본 발명에 따라 스트라이프의 부분 패리티가 패리티 캐쉬에 존재하며 복구할 데이터가 부분 패리티에 언커미티드 데이터가 포함된 경우의 복구 예이다.

[0082] 도 8에서는 앞에서와 같이 RAID 4를 가정하여 도시하였다. 참조부호 810과 같이 디스크 1에 데이터 오류가 발생하였고, 그 외의 다른 데이터의 갱신이 있으며, 패리티 캐쉬(232)에 해당하는 스트라이프에 대응하는 패리티(801)가 존재하는 경우이다. 이러한 경우 갱신된 D_1'의 데이터는 참조부호 820에 도시한 바와 같이 D_1'과 D_2'의

배타적 논리합인 패리티 캐쉬(232)에 저장된 부분 패리티(801)와 디스크 2에 갱신되어 저장된 D_2'를 이용하여 계산함으로써 데이터를 복구할 수 있다.

[0083] (b) 복구해야 할 데이터가 부분 패리티의 언커미티드 데이터에 포함되지 않는 경우는 이전 데이터와 이전 패리티를 이용하여 복구할 수 있다. 이를 도 9를 참조하여 살펴보기로 한다.

[0084] 도 9는 본 발명에 따라 스트라이프의 부분 패리티가 패리티 캐쉬에 존재하며 복구할 데이터가 부분 패리티에 언커미티드 데이터에 포함되지 않은 경우의 복구 예이다.

[0085] 참조부호 910과 같이 디스크 0에 데이터 오류가 발생하였고, 오류가 발생하지 않은 디스크들 중 하나 이상의 디스크에 저장된 데이터의 갱신이 있으며, 패리티 캐쉬(232)에 해당하는 스트라이프에 대응하는 패리티(901)가 존재하는 경우이다. 이러한 경우 갱신된 D_0의 데이터는 참조부호 920에 도시한 바와 이전 패리티인 P와 이전 데이터들인 D_1, D_2 및 D_3를 배타적 논리합하여 D_0의 데이터를 복구할 수 있다. 즉, 에러가 발생한 데이터가 부분 패리티의 언커미티드 데이터에 포함되지 않으면 이 데이터에 대한 패리티 정보는 이전 패리티에 기록되어 있으므로 이전 데이터와 이전 패리티를 사용하여 이를 복구할 수 있는 것이다. 이 과정에서 D_1과 D_2는 언커미티드 데이터의 이전 데이터이므로 매핑 테이블 정보의 이전 데이터 필드를 참조한다.

[0086] (2) 스트라이프의 부분 패리티가 패리티 캐쉬(232)에 존재하지 않은 경우:

[0087] 스트라이프의 부분 패리티가 패리티 캐쉬(232)에 존재하지 않는 경우는 패리티 캐쉬에 복구해야 할 데이터가 포함되는 스트라이프의 부분 패리티가 존재하지 않을 때이다. 따라서 이러한 경우는 해당 스트라이프의 패리티가 정상적으로 커미트된 경우이므로 기존의 RAID와 마찬가지로 저장 장치의 데이터와 패리티를 사용하여 데이터를 복구할 수 있다.

[0088] 이상에서 설명한 본 발명에 대하여 다시 살펴보기로 한다.

[0089] 기존 RAID 장치에서는 데이터가 쓰일 때마다 패리티를 계산한다. 이 연산은 해당 데이터의 이전 데이터와 패리티 데이터를 읽어 현재 데이터와 배타적 논리합한 다음에 갱신된 패리티를 기록하는 과정으로 이루어진다. 그러므로 하나의 데이터의 추가적인 패리티의 기록 때문에 2번의 독취(read)와 1번의 오버헤드가 발생한다. 즉 데이터 기록(data write)의 비용이 $2 * (T_{read} + T_{write})$ 이다.

[0090] 그러나 본 발명에 따른 방법에서는 패리티 캐쉬를 사용하여 패리티를 지연하여 기록하기 때문에 읽기 및 쓰기 비용을 절약할 수 있다. 또한 데이터를 기록할 때, 패리티 캐쉬가 해당 데이터의 이전 데이터에 대한 패리티를 가지고 있을 때는 한 번의 독취(read)와 한 번의 데이터 기록(write)만이 필요하며, 이전 데이터에 대한 패리티가 없다면 한 번의 데이터 기록만이 필요하다.

[0091] 패리티 커미트 연산에서 가장 비용이 많이 소모되는 경우는 N개의 저장 장치 가운데 1/2 개의 데이터만이 갱신되어 부분 패리티를 구성하고 있을 때이다. 이때 완전한 패리티를 계산하기 위해서 $0.5 * N$ 번의 읽기가 필요하다. 반면 전체 패리티가 패리티 캐쉬에서 계산된 경우에는 저장 장치에 접근할 필요가 없다. 패리티를 계산한 다음에는 이를 저장 장치에 기록하는 1번의 기록만이 발생한다. 즉 T_{write}와 $0.5 * N * T_{read} + T_{write}$ 사이의 비용이 필요하다.

[0092] 또한 본 발명에 따른 방법의 성능은 특히 패리티 캐쉬에서 많은 갱신이 발생할 때 극대화된다. 일반적인 컴퓨팅 환경에서 데이터들은 특정 시간 동안 같은 데이터가 반복해서 접근되는 시간 지역성(temporal locality)을 가지므로 본 발명의 장점을 충분히 활용할 수 있다는 것을 알 수 있다.

[0093] 뿐만 아니라 본 발명에 따른 방법 및 장치는 패리티 캐쉬에서 전체 패리티가 만들어지는 경우에는 패리티를 계산하기 위한 추가적인 오버헤드가 전혀 발생하지 않는다. 대용량의 데이터가 저장되는 경우에는 연속적인 쓰기가 발생하는 경우가 많기 때문에 전체 패리티가 계산되는 경우에도 많은 비용을 줄이는 것이 가능하다.

부호의 설명

[0094] 101, 102, 103, 104 : RAID를 구성하는 디스크들

111, 112 : 스트라이프들

120 : 제어기, 210 : 읽기 처리부

220 : 연산 처리부, 230 : 내부 메모리

231 : 매핑 테이블, 232 : 패리티 캐쉬

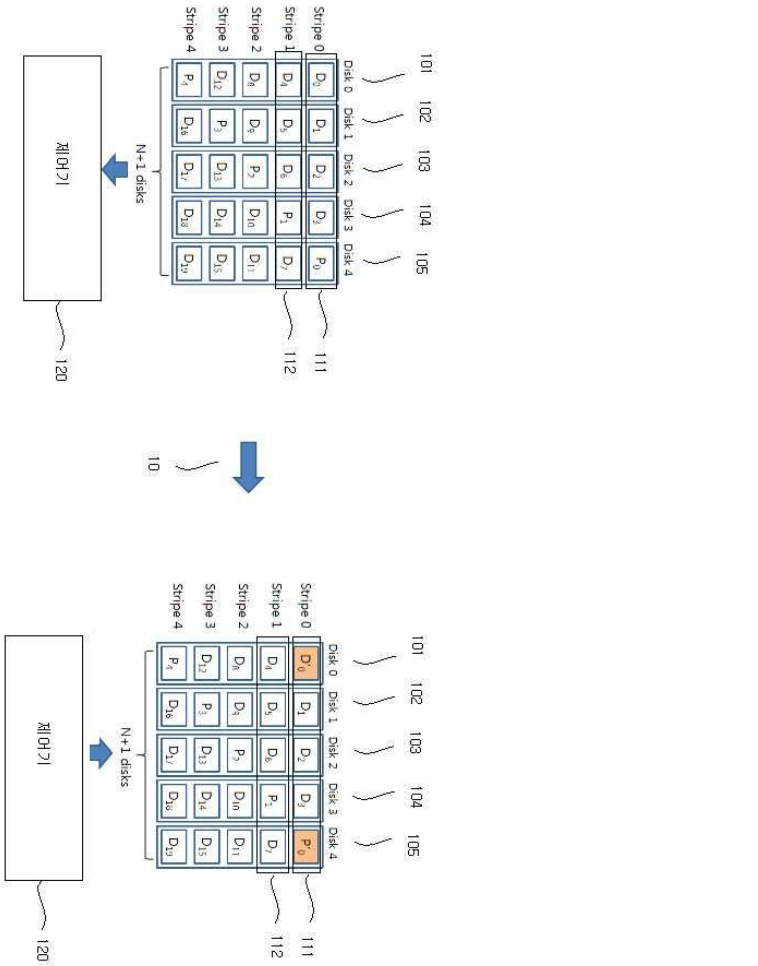
240 : 쓰기 처리부, 310 : 패리티 캐쉬의 부분 패리티 수

311 : 스트라이프 인덱스, 312 : 부분 패리티 비트 맵

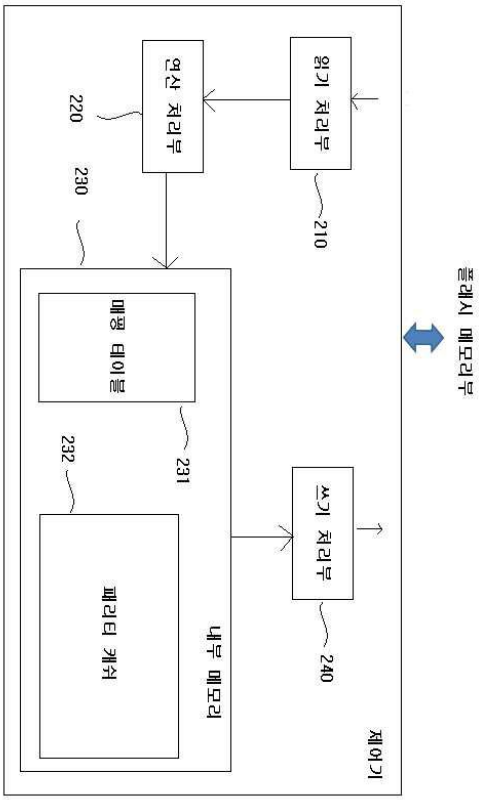
313 : 부분 패리티의 데이터 및 그 크기

도면

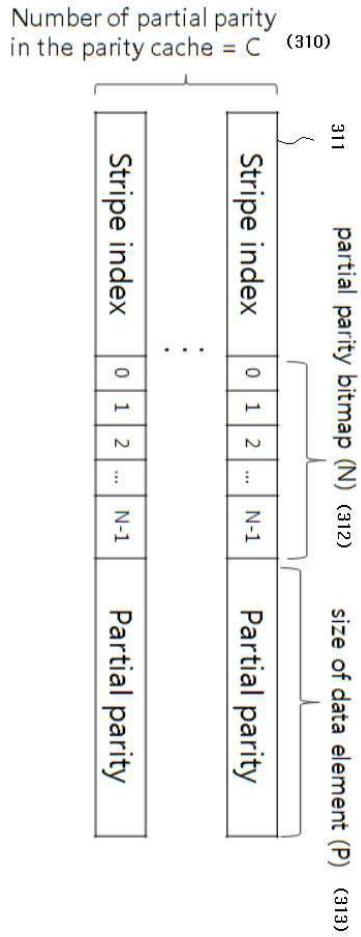
도면1



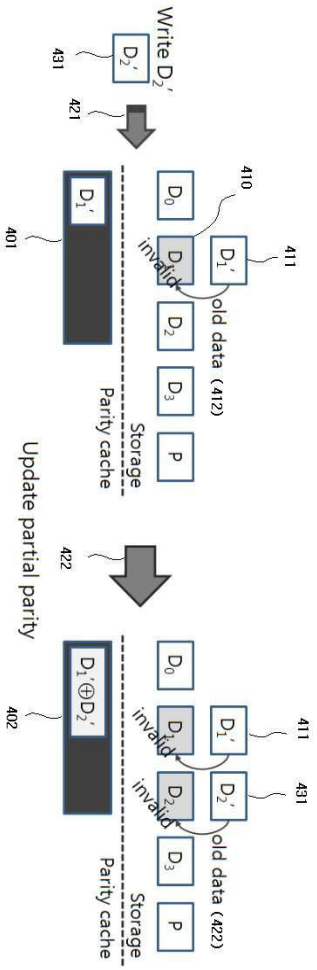
도면2


제어기
내부 메모리
매핑 테이블
231
232
패리티 캐쉬
220
230
연산 처리부
읽기 처리부
210
쓰기 처리부
240
플래시 메모리부

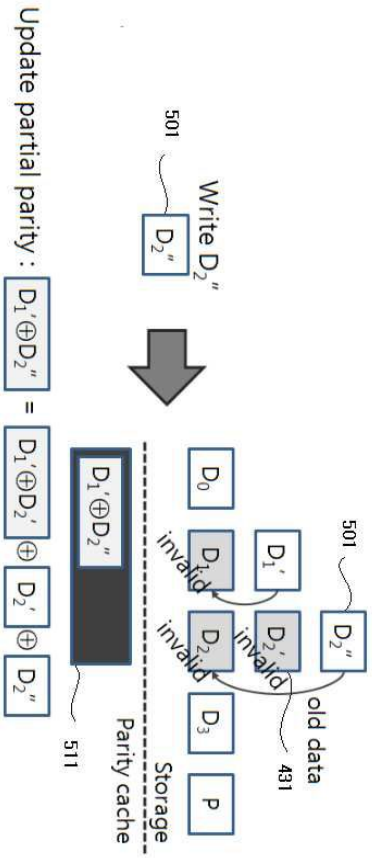
도면3


Number of partial parity
in the parity cache = C
(310)
311
Stripe index
0
1
2
...
N-1
Partial parity
partial parity bitmap (N)
(312)
size of data element (P)
(313)

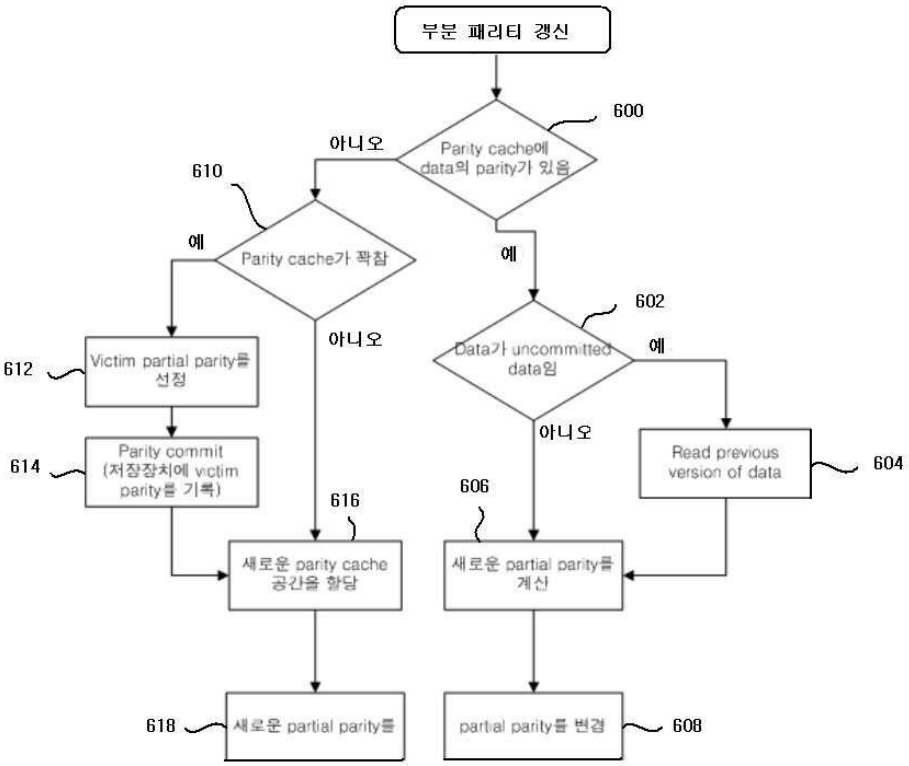
도면4


Write D2'
431
421
410
D0
D1
invalid
D1'
411
old data (412)
D2
D3
P
Storage
Parity cache
401
Update partial parity
422
D1⊕D2'
402
D0
D1
invalid
D1'
411
D2
invalid
D2'
431
old data (422)
D3
P
Storage
Parity cache

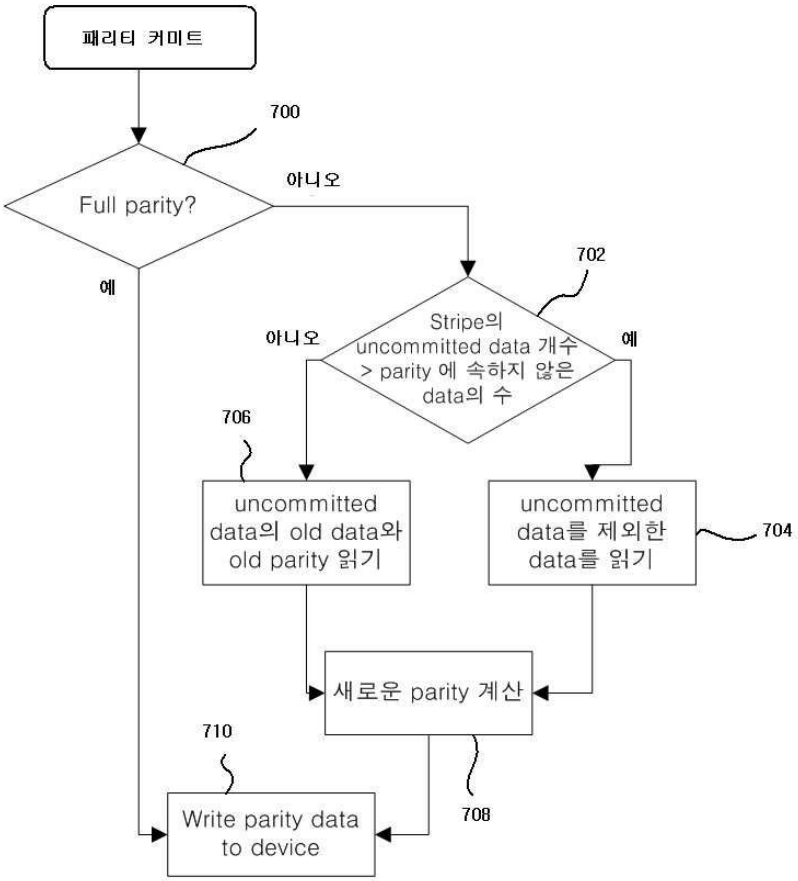
도면5


Update partial parity :
D1'⊕D2"
=
D1'⊕D2'
⊕
D2'
⊕
D2"
Write D2"
501
D2"
D1'⊕D2"
511
Parity cache
Storage
D0
D1
D1'
invalid
D2
D2'
invalid
D3
P
501
D2"
old data
431

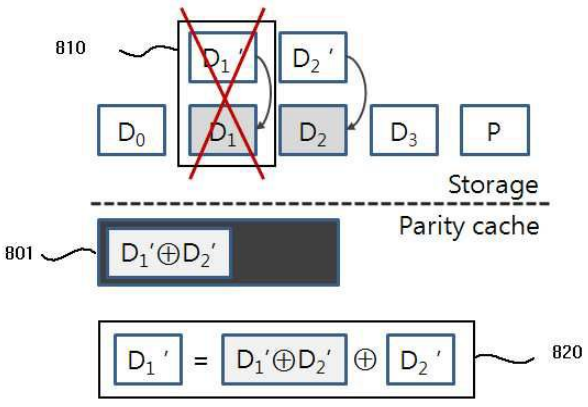
도면6


부분 패리티 갱신
600
Parity cache에 data의 parity가 있음
아니오
예
610
Parity cache가 꽉참
예
아니오
612
Victim partial parity를 선정
614
Parity commit (저장장치에 victim parity를 기록)
616
새로운 parity cache 공간을 할당
618
새로운 partial parity를
602
Data가 uncommitted data임
예
아니오
Read previous version of data
604
606
새로운 partial parity를 계산
partial parity를 변경
608

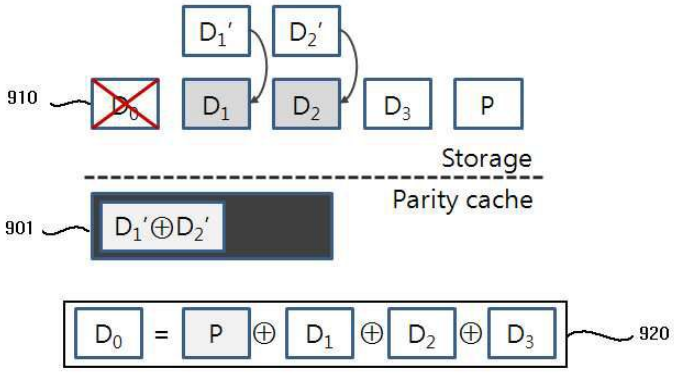
도면7


패리티 커미트
700
Full parity?
아니오
예
702
Stripe의 uncommitted data 개수 > parity 에 속하지 않은 data의 수
아니오
예
706
uncommitted data의 old data와 old parity 읽기
uncommitted data를 제외한 data를 읽기
704
새로운 parity 계산
710
708
Write parity data to device

도면8


810
D1 '
D2 '
D0
D1
D2
D3
P
Storage
Parity cache
801
D1'⊕D2'
D1 ' = D1'⊕D2' ⊕ D2 '
820

도면9


D1′
D2′
910
D0
D1
D2
D3
P
Storage
Parity cache
901
D1′⊕D2′
D0 = P ⊕ D1 ⊕ D2 ⊕ D3
920