(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2010-0021868
(43) 공개일자 2010년02월26일

(51) Int. Cl.
G06F 12/08 (2006.01)
(21) 출원번호 10-2008-0080510
(22) 출원일자 2008년08월18일
심사청구일자 없음

(71) 출원인
삼성전자주식회사
경기도 수원시 영통구 매탄동 416
(72) 발명자
서동영
경기 화성시 병점동 주공아파트 704동 1402호
신동군
서울 관악구 봉천6동 1681-21 로얄빌라 101호
(74) 대리인
권혁수, 송윤호, 오세준

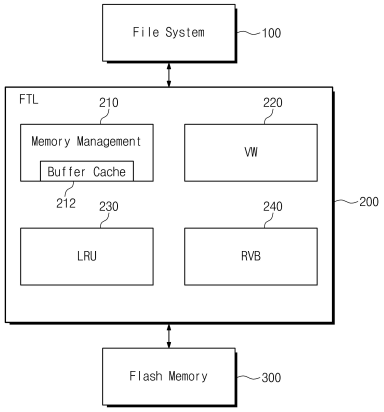
전체 청구항 수 : 총 8 항

(54) 플래시 메모리 장치를 위한 버퍼 캐쉬 관리 방법

(57) 요 약

본 발명의 버퍼 캐쉬 관리 방법은, 버퍼 캐쉬가 꽉 찼을 때 상기 버퍼 캐쉬에 저장된 페이지 데이터 중 빅팀 윈도우 내 이전 빅팀 블록에 대응하는 페이지 데이터를 우선적으로 제거한다. 그러므로 플래시 메모리에서 로그 블록에 저장된 데이터와 데이터 블록의 데이터를 머지하는 횟수를 감소시킬 수 있다

대 표 도 - 도1



특허청구의 범위

청구항 1

플래시 메모리를 포함하는 메모리 시스템의 버퍼 캐쉬 관리 방법에 있어서:

버퍼 캐쉬에 기입될 페이지 데이터를 입력받는 단계와;

상기 버퍼 캐쉬에 저장된 페이지 데이터 중 상기 플래시 메모리로 기입될 페이지 데이터를 제거하는 단계를 포함하되;

상기 제거 단계는, 상기 버퍼 캐쉬의 빅팀 윈도우 내 이전 빅팀 블록에 대응하는 페이지 데이터를 제거하는 것을 특징으로 하는 버퍼 캐쉬 관리 방법.

청구항 2

제 1 항에 있어서,

상기 빅팀 윈도우는,

상기 버퍼 캐쉬에 가장 오래 전에 저장된 페이지 데이터부터 소정의 페이지 데이터를 포함하는 것을 특징으로 하는 버퍼 캐쉬 관리 방법.

청구항 3

제 1 항에 있어서,

상기 버퍼 캐쉬로부터 제거되는 페이지 데이터의 블록을 상기 이전 빅팀 블록의 정보로서 저장하는 단계를 더 포함하는 것을 특징으로 하는 버퍼 캐쉬 관리 방법.

청구항 4

제 1 항에 있어서,

상기 메모리 시스템은,

상기 이전 빅티 블록의 정보 및 상기 빅팀 윈도우 그리고 상기 버퍼 캐쉬 내 가장 오래 전에 사용된 페이지 데이터에 대한 정보를 각각 저장하기 위한 레지스터들 포함하는 것을 특징으로 하는 버퍼 캐쉬 관리 방법.

청구항 5

제 1 항에 있어서,

상기 제거 단계는 상기 버퍼 캐쉬 내 어떤 페이지 데이터도 선택될 수 있을 때 상기 버퍼 캐쉬 내 가장 오래 전에 사용된 페이지 데이터를 제거하는 것을 특징으로 하는 버퍼 캐쉬 관리 방법.

청구항 6

제 1 항에 있어서,

상기 제거 단계는, 상기 버퍼 캐쉬의 빅팀 윈도우 내 이전 빅팀 블록에 대응하는 페이지 데이터의 수가 복수 개일 때 상기 빅팀 윈도우 내 가장 오래 전에 사용된 페이지 데이터를 제거하는 것을 특징으로 하는 버퍼 캐쉬 관리 방법.

청구항 7

제 1 항에 있어서,

상기 제거 단계는,

상기 버퍼 캐쉬가 꽉 찼을 때 상기 버퍼 캐쉬에 저장된 페이지 데이터 중 상기 플래시 메모리로 기입될 페이지 데이터를 제거하는 것을 특징으로 하는 버퍼 캐쉬 관리 방법.

청구항 8

제 1 항에 있어서,

상기 플래시 메모리는 낸드 플래시 메모리인 것을 특징으로 하는 버퍼 캐쉬 관리 방법.

명 세 서

발명의 상세한 설명

기 술 분 야

[0001] 본 발명은 메모리 시스템에 관한 것으로, 좀 더 구체적으로는 플래시 메모리 장치를 위한 버퍼 캐쉬 관리 방법에 관한 것이다.

배 경 기 술

[0002] 플래시 메모리는 비휘발성이며 저전력 소모 저장 장치이다. 하드 디스크 드라이브에 비해 고속 액세스 성능을 갖는 플래시 메모리는 MP3 플레이어, 디지털 카메라 및 PDA 등과 같은 임베디드 시스템들용 저장 장치로서 널리 사용되고 있다. 최근 MP3 플레이어, 디지털 카메라의 사용이 증대되고, 이러한 장치들에서 대용량의 데이터 저장 장치를 요구함에 따라서 낸드 플래시(NAND flash) 시장이 급격하게 성장하였다. 또한 데스크탑 PC와 같은 퍼스널 시스템들에서도 낸드 플래시 메모리가 사용된다. 예컨대, 하이브리드 하드 디스크(hybrid hard disk) 및 터보 메모리(turbo memory)는 하드 디스크 드라이브의 불휘발성 캐쉬로서 플래시 메모리를 사용한다. 낸드 플래시 기반 SSD(solid-state disk)는 가까운 장래에 하드 디스크를 대체할 것으로 기대된다.

[0003] 마그네틱 하드디스크와 달리, 검색 시간(seek time)이 불필요한 플래시 메모리는 고속 읽기 성능(high read performance)을 제공한다. 그러나, 플래시 메모리는 기입 성능(write performance)를 저하시키는 두 가지 특성을 갖는다. 첫 번째는 "기입전 소거(erase befor write)" 구조라고 불리는, 데이터가 블록에 기입되기 전 블록이 소거되어야 한다는 것이다. 또다른 특성은 기입 동작이 페이지 단위로 수행되는 반면 소거 동작은 블록 단위로 수행되는 것이다. 블록은 복수의 페이지 데이터의 묶음이다. 예컨대, 삼성의 대형 블록 낸드 플래시 메모리에 있어서, 한 페이지 데이터의 크기는 2킬로바이트(KB)이고, 한 블록 데이터의 크기는 128 킬로바이트(64 페이지 데이터)이다.

[0004] 이러한 두 가지 특성 때문에 대부분의 시스템은 파일 시스템으로부터의 논리적 페이지 어드레스를 플레시 메모리 장치 내 물리적 어드레스로 매핑하는 FTL(flash translation layter)을 사용한다. FTL의 어드레스 매핑 스킴은 세 가지 즉, 블록-레벨 매핑, 페이지-레벨 매핑 그리고 하이브리드 매핑으로 분류된다. 블록-레벨 매핑은, 매핑 페이블이 논리적 블록 어드레스 및 물리적 블록 어드레스 사이의 매핑 정보를 유지한다. 그러므로 논리적 페이지가 순차 위치(in-place) 스킴에 의해서 기입된다. 이것은 블록 내 페이지 오프셋에 의해서 정의된 블록의 고정된 위치에 페이지 데이터가 기입됨을 의미한다. 블록 레벨 매핑은 작은 크기의 매핑 테이블을 필요로 한다. 그러나, 특정 페이지에서 데이터가 수정될 때 특정 블록은 소거되어야 하고, 더욱이 변경되지 않아야 하는 페이지들은 새로운 블록으로 복사되어야 한다. 이러한 제약은 많은 페이지 데이터를 이동시키는 결과를 초래하고, 기입 성능을 저하시킨다.

[0005] FTL의 세가지 어드레스 매핑 스킴 즉, 블록-레벨 매핑, 페이지-레벨 매핑 그리고 하이브리드 매핑은 다음과 같은 특징을 갖는다.

[0006] 페이지 레벨 매핑에서, 매핑 테이블은 논리적 페이지 어드레스와 물리적 페이지 어드레스 간의 매핑 정보를 유지한다. 그러므로, 논리적 페이지는 임의 위치(out-of-pace) 스킴에 의해서 매핑된다. 즉, 페이지 데이터가 블록 내 어떠한 물리적 페이지에도 기입될 수 있다. 만일 플래시 메모리에 이미 기입된 데이터에 대한 갱신(update) 요구가 전송되면, FTL은 새로운 데이터를 다른 비어있는 페이지에 기입하고, 플래시 메모리의 여유 공간에 표시하는 것에 의해서 이전 페이지 데이터를 무효화한다. 이러한 페이지 레벨 매핑의 단점은 매핑 테이블의 크기가 크다는 것이다.

[0007] 하이브리드 매핑은 페이지 매핑 및 블록 매핑 모두를 사용한다. 이 스킴에서 모든 물리적 블록들은 로그 블록들(log blocks)과 데이터 블록들(data blocks)로 분리된다. 로그 블록들은 로그 버퍼(log buffer)라고도 불리운다. 그러므로 하이브리드 매핑 스킴을 사용하는 FTL은 로그 버퍼 기반 FTL이라고도 불리운다. 로그 블록들은

페이지 레벨 매핑과 임의 위치(out-of-pace) 스킴을 사용하고, 데이터 블록들은 블록-레벨 매핑 및 순차 위치(in-place) 스킴에 의해서 처리된다. 기입 요청에 대해서 FTL은 데이터를 로그 블록으로 전송하고, 데이터 블록 내 대응하는 이전 데이터(old data)는 무효화한다.

[0008] 만일 로그 블록이 꽉차서 빈 공간이 없다면, 어느 하나의 로그 블록이 제거(victim)를 위해서 선택되고, 로그 블록 내 모든 유효한 페이지들은 데이터 블록들로 이동된다. 이 단계에서, 로그 블록은 로그 블록과 관련있는 데이터 블록들과 머지된다. 따라서 이 단계는 블록 머지라 불리운다. 블록 머지는 세 가지 즉, 완전 머지(full merge), 부분 머지(partial merge) 그리고 스위치 머지(switch merge)로 분류된다. 부분 머지 및 스위치 머지는 블록 내 모든 페이지들이 순차 위치 스킴에 의해서 기입된 경우에 수행될 수 있다. 완전 머지는 많은 페이지 복사 및 블록 소거를 필요로하는 반면, 부분 머지 및 스위치 머지는 적은 페이지 이동 비용을 유발한다. 하이브리드 매핑은 블록 매핑에 비해 페이지 이동 비용을 감소시킬 수 있으나 작은 크기의 매핑 테이블을 요구한다.

[0009] 플래시 메모리 시스템의 입출 성능을 향상시키기 위하여 블록 머지에 의해서 유발되는 오버헤드는 감소되어야 한다. 그러므로, 대부분의 FTL 스킴은 블록 머지의 횟수를 감소시키는 것을 목표로 한다. 그러나, 플래시 메모리 시스템이 MP3 플레이어 및 디지털 카메라와 같은 멀티미디어 시스템들을 타겟으로 하기 때문에 현재 FTL 기술은 순차적 기입 패턴에 초점이 맞추어져 있다. 그러나, 최근 플래시 메모리 장치들은 데스크탑 PC 등과 같은 범용 시스템들에 사용되고 있으므로, 순차적 및 랜덤 기입 모두에 대한 요구가 증대되고 있다.

발명의 내용

해결 하고자하는 과제

[0010] 따라서 본 발명의 목적은 플래시 메모리 장치에 데이터를 효율적으로 기입할 수 있는 방법을 제공하는데 있다.

[0011] 본 발명의 다른 목적은 플래시 메모리 장치에서의 머지 발생을 최소화할 수 있는 버퍼 캐쉬 관리 방법을 제공하는데 있다.

과제 해결수단

[0012] 이와 같은 목적을 달성하기 위한 본 발명의 일 특징에 의하면, 플래시 메모리를 포함하는 메모리 시스템의 버퍼 캐쉬 관리 방법은: 상기 버퍼 캐쉬에 기입될 페이지 데이터를 입력받는 단계와, 상기 버퍼 캐쉬가 꽉 찼을 때 상기 버퍼 캐쉬에 저장된 페이지 중 상기 플래시 메모리로 기입될 페이지 데이터를 제거하는 단계를 포함한다. 상기 제거 단계는, 상기 버퍼 캐쉬의 빅팀 윈도우 내 이전 빅팀 블록에 대응하는 페이지 데이터를 제거한다.

[0013] 이 실시예에 있어서, 상기 빅팀 윈도우는, 상기 버퍼 캐쉬에 가장 오래 전에 저장된 페이지 데이터부터 소정의 페이지 데이터를 포함한다.

[0014] 이 실시예에 있어서, 상기 버퍼 캐쉬로부터 제거되는 페이지 데이터의 블록을 상기 이전 빅팀 블록의 정보로서 저장하는 단계를 더 포함한다.

[0015] 이 실시예에 있어서, 상기 메모리 시스템은, 상기 이전 빅티 블록의 정보 및 상기 빅팀 윈도우 그리고 상기 버퍼 캐쉬 내 가장 오래 전에 사용된 페이지 데이터에 대한 정보를 각각 저장하기 위한 레지스터들을 포함한다.

[0016] 이 실시예에 있어서, 상기 제거 단계는 상기 버퍼 캐쉬 내 어떤 페이지 데이터도 선택될 수 있을 때 상기 버퍼 캐쉬 내 가장 오래 전에 사용된 페이지 데이터를 제거한다.

[0017] 이 실시예에 있어서, 상기 제거 단계는, 상기 버퍼 캐쉬의 빅팀 윈도우 내 이전 빅팀 블록에 대응하는 페이지 데이터의 수가 복수 개일 때 상기 빅팀 윈도우 내 가장 오래 전에 사용된 페이지 데이터를 제거한다.

[0018] 이 실시예에 있어서, 상기 플래시 메모리는 낸드 플래시 메모리이다.

효 과

[0019] 이와 같은 본 발명에 의하면, 버퍼 캐쉬에서 이전에 빅팀된 페이지 데이터를 우선적으로 제거함으로써 플래시 메모리에서 로그 블록에 저장된 데이터와 데이터 블록의 데이터를 머지하는 횟수를 감소시킬 수 있다.

발명의 실시를 위한 구체적인 내용

[0020] 이하 본 발명의 바람직한 실시예를 첨부된 도면들을 참조하여 상세히 설명한다.

[0021] 본 발명에 따른 플래시 메모리 저장 장치의 버퍼 캐쉬 관리 방법은 버퍼 캐쉬가 최근에 로그 버퍼에 기입된 페이지 데이터만을 버퍼 캐쉬에서 제거(eviction)하도록 강제한다.

[0022] 로그 버퍼 기반 FTL은 블록 관련 정책(block association policy)에 따라서 1:1 로그 블록 매핑과 1:N 로그 블록 매핑의 두 가지로 나뉜다. 블록 관련 정책은 얼마나 많은 데이터 블록들이 로그 블록으로 사용될 수 있는가를 의미한다. 1:1 스킴에서, 로그 블록은 오직 하나의 데이터 블록으로 할당된다. 1:1 스킴에서 로그 블록들의 수보다 다양한 블록들에 대한 페이지 데이터 입/출력이 발생할 때 로그 블록에 대한 머지가 수행되는 로그 블록 쓰래싱(log block thrathing)이 발생한다. 즉, 로그 블록에 페이지 데이터를 저장할 공간이 남아 있음에도 불구하고 로그 블록에 저장된 페이지 데이터를 플래시 메모리에 기입하여 로그 블록을 비워야 하는 것이다.

[0023] 반면, 1:N 로그 블록 매핑은 하나의 로그 블록에 여러 데이터 블록들이 매핑될 수 있도록 한다. 로그 블록이 머지될 때 연관된 데이터 블록의 수는 로그 블록에 몇 개의 데이터 블록에 대한 페이지 데이터가 저장되었는 가에 따라서 결정된다. 이 때 하나의 로그 블록을 머지할 때 여러 개의 데이터 블록들과의 머지가 수행되면 머지 코스트가 증가한다.

[0024] 최근에는 N:N 스킴 즉, 수퍼블록(superblock) 스킴이 제안되고 있다. 수퍼블록 스킴은 1:1 매핑 스킴과 1:N 매핑 스킴의 혼합형이다. 본 발명의 버퍼 캐쉬 관리 방법은 다양한 블록 관련 정책 모두에 적용될 수 있다.

[0025] 도 1은 본 발명의 바람직한 실시예에 따른 플래시 메모리 시스템에 포함되는 FTL을 보여주는 도면이다.

[0026] 도 1을 참조하면, 플래시 메모리를 포함하는 플래시 메모리 시스템은 플래시 메모리를 효율적으로 관리하기 위한 소프트웨어 모듈을 포함한다. 이 소프트웨어 모듈이 FTL(flash translation layer)이다. FTL(200)은 파일 시스템 또는 어플리케이션 소프트웨어로부터 전달받은 섹터 어드레스와 섹터 갯수를 파라미터로 하여 플래시 메모리(300)의 기입/독출 동작을 위한 어드레스 변환을 수행한다. 파일 시스템 또는 어플리케이션 소프트웨어로부터 전달받은 섹터 어드레스는 호스트로부터 요청될 수 있다.

[0027] 본 발명의 FTL(200)은 파일 시스템(100)과 플래시 메모리(300) 사이에 위치하며, 플래시 메모리(300)에서의 머지 동작을 최소화할 수 있도록 동작한다. FTL(200)은 호스트와 독립된 하드웨어 형태로 구성되거나 또는 호스트의 내부에 디바이스 드라이버 형태로 구현될 수 있다.

[0028] FTL(200)은 파일 시스템(100)로부터 기입 명령이 입력되면, 논리주소를 플래시 메모리(300) 상의 이미 소거된 영역에 대한 물리주소로 변환한다. FTL(200)는 비교적 수행시간이 오래 걸리는 소거 동작을 감추고 I/O를 하나의 단위(atomic operation)로 처리함으로써 플래시 메모리(300)를 효율적으로 제어할 수 있다.

[0029] FTL(200)은 메모리 관리기(memory management)(210), 빅팀 윈도우(victim window, VW) 레지스터(220), LRU(Least Recently Used) 레지스터(230) 그리고 RVB(recent victim block, RVB) 레지스터(240)를 포함한다. 메모리 관리기(210)는 플래시 메모리(300)에 기입될 데이터를 임시 저장하기 위한 버퍼 캐쉬(212)를 포함한다.

[0030] 버퍼 캐쉬(212)를 관리하는 기법에는 여러 가지가 있을 수 있다. 플래시 메모리 기입 패턴은 버퍼 캐쉬의 빅팀 페이지 선택 정책에 의해서 결정되며, 몇몇 정책들은 플래시 메모리의 기입 비용을 최소화하기 위해 제안되었다.

[0031] 본 발명의 버퍼 캐쉬(212) 관리 방법은 FaPE(Flash-aware Page Eviction) 스킴으로 불리며, 이것은 최근 제거(eviction)된 페이지 데이터를 고려하여 버퍼 캐쉬로부터 교체될 빅팀 페이지 데이터를 결정하는 방식이다.

[0032] 도 2는 도 1에 도시된 버퍼 캐쉬 및 플래시 메모리의 관계를 구체적으로 보여주는 도면이다.

[0033] 도 2에 도시된 예에서, 버퍼 캐쉬(212)는 최대 6 개의 페이지 데이터를 저장할 수 있는 크기를 갖는다. 플래시 메모리(300)는 실제 데이터가 저장되는 데이터 영역(310)과 데이터를 임시로 저장하는 로그 영역(320)을 포함한다. 이 실시예에서, 데이터 영역(310)는 최대 5 개의 데이터 블록들(B0-B5)을 저장할 수 있으며, 로그 영역(320)은 최대 2 개의 로그 블록들(L0, L1)을 저장할 수 있다. 데이터 블록과 로그 블록 각각은 최대 4 개의 페이지 데이터를 포함한다.

[0034] 예컨대, 버퍼 캐쉬(212)에 LRU 순으로 페이지 데이터 p0, p4, p9, p13, p1 및 p12가 저장되어 있고, 두 개의 로그 블록들(L0, L1)은 페이지 데이터 p8 및 p12를 저장하고, 로그 블록들(L0, L1)은 1:1 로그 블록 매핑 스킴으로 관리된다고 가정하자.

[0035] 만일 LRU 페이지 교체 정책(page replacement policy)을 사용한다면, 버퍼 캐쉬(212) 내 페이지 데이터는 p0, p4, p9, p13, p1, p5 순으로 제거되고, 플래시 메모리(300) 에서는 6 번의 머지 동작이 수행될 것이다.

[0036] 본 발명은 플래시 메모리(300)의 머지 동작을 최소화하기 위하여 새로운 방법으로 버퍼 캐쉬(212)를 운영한다.

[0037] 플래시 메모리(300) 내 데이터 블록들(B2, B3)은 최근에 로그 블록들(L0, L1)에 기입되었기 때문에 데이터 블록들(B2, B3)의 페이지 데이터(p9, p13)을 버퍼 캐쉬(212)로부터 제거하는 것이 더 바람직하다. 만일 버퍼 캐쉬((212)가 페이지 데이터를 p9, p13, p0, p4, p1, p5 순으로 제거하면, 오직 두 번의 머지 동작이 요구된다.

[0038] 다시 말하면, 만일 버퍼 캐쉬((212)가 페이지 데이터를 p9, p13을 우선 로그 블록들(L0, L1)에 기입한다면, 로그 블록(L0)에 저장된 페이지 데이터(p8)와 새로 저장될 페이지 데이터(p9)는 동일한 블록(B2)에 저장될 데이터이고, 마찬가지로 로그 블록(L1)에 저장된 페이지 데이터(p12)와 새로 저장될 페이지 데이터(p13)는 동일한 블록(B3)에 저장될 데이터이므로 머지 동작이 불필요하다.

[0039] 블록(B0)에 대응하는 새로운 페이지 데이터(p0)를 로그 영역(320)에 기입하기 위해서는 로그 블록들(L0, L1) 중 하나를 데이터 영역과 머지해야 한다. 예컨대, 로그 블록(L0)을 머지한다면, 로그 블록(L0)에 저장된 페이지 데이터(p8, p9)을 데이터 블록(B2) 내 이미 저장된 페이지 데이터과 머지한 후 로그 블록(L0)을 소거한다. 이와 같은 방법에 의하면, 버퍼 캐쉬(212)가 페이지 데이터를 p9, p13, p0, p4, p1, p5 순으로 제거할 때 오직 두 번의 머지 동작이 요구된다.

[0040] 로그 블록들(L0, L1)이 1:N 로그 블록 매핑 스킴으로 관리되는 경우 본 발명의 FaPE(Flash-aware Page Eviction) 스킴에 따른 버퍼 캐쉬 관리 방법은 다음과 같다.

[0041] 도 3은 1:N 로그 블록 매핑 스킴에 적용된 본 발명의 버퍼 캐쉬 관리 방법을 설명하기 위한 버퍼 캐쉬와 플래시 메모리의 관계를 예시적으로 보여주는 도면이다.

[0042] 도 3을 참조하면, 초기에 로그 영역(320)의 로그 블록(L0)에 페이지 데이터(p8, 12)이 저장된 것으로 가정하자. 로그 영역(320a)은, 버퍼 캐쉬(212)에 저장된 페이지 데이터가 LRU 방식으로 p0, p4, p9, p13, p1, p5 순으로 제거된 경우 페이지 데이터의 저장 상태를 보여주며, 로그 영역(320b)은, 본 발명의 바람직한 실시예에 따른 방식으로 버퍼 캐쉬(212)에 저장된 페이지 데이터가 p9, p13, p0, p4, p1, p5 순으로 제거된 경우 페이지 데이터의 저장 상태를 보여준다.

[0043] 로그 영역(320a)의 로그 블록들(L0, L1)은 4의 블록 관련성(block associativity)을 가지며, 로그 영역(320b)의 로그 블록들(L0, L1)의 블록 관련성은 2로 감소한다. 이와 같이 본 발명의 버퍼 캐쉬 관리 스킴은 로그 블록의 블록 관련성과 블록 머지의 횟수를 감소시킨다.

[0044] 본 발명의 버퍼 캐쉬 관리 방법은 크게 세가지 특징을 갖는다.

[0045] 첫 번째는 블록 레벨 페이지 데이터 제거이다. 블록 머지 또는 블록 관련성을 감소시키기 위하여 버퍼 캐쉬(212)로부터 빅팀 블록들의 페이지 데이터만 제거된다. 즉, LRU 교체 정책을 통해 구현된다. 두 번째는 가능하면 로그 블록들과 관련있는 데이터 블록들 즉, 빅팀 블록들이 유지되도록 한다. 이것은 블록의 모든 페이지 데이터가 동시에 제거되도록 함으로써 블록 머지 비용을 감소시킨다. 그리고 세 번째는 최신의 페이지 데이터 레벨을 고려하는 것이다. 즉, 최근에 사용된 페이지 데이터가 제거되는 것을 방지하기 위하여 빅팀 페이지 데이터는 최근에 사용되지 않은 페이지 데이터 중 선택된다.

[0046] 도 4는 본 발명의 바람직한 실시예에 따른 버퍼 캐쉬 관리 기법에 따라서 도 1에 도시된 메모리 관리기가 버퍼 캐쉬에 저장된 페이지 데이터 중 빅팀 블록을 선택하는 것을 예시적으로 보여주는 도면이다.

[0047] 도 4를 참조하면, 버퍼 캐쉬(128a)는 8 개의 페이지 데이터를 저장할 수 있고, 각 페이지 데이터는 LRU 순으로 저장되는 것으로 가정한다. 초기에, 도 1에 도시된 최근 빅팀 블록(RVB)을 저장하는 레지스터(240)는 비어있다. 로그 블록 쓰레싱을 방지하기 위하여, 레지스터(240)에 저장되는 최근 빅팀 블록(RVB)의 수는 플래시 메모리 내 로그 블록들의 수보다 작아야 한다. 예컨대, FTL의 세가지 어드레스 매핑 스킴 즉, 블록-레벨 매핑, 페이지-레벨 매핑 그리고 하이브리드 매핑은 다음과 같은 특징을 갖는다.

[0048] 페이지 레벨 매핑에서, 매핑 테이블은 논리적 페이지 어드레스와 물리적 페이지 어드레스 간의 매핑 정보를 유지한다. 그러므로, 논리적 페이지는 임의 위치 스킴에 의해서 매핑된다. 즉, 페이지 데이터가 블록 내 어떠한 물리적 페이지에도 기입될 수 있다. 만일 플래시 메모리에 이미 기입된 데이터에 대한 갱신(update) 요구가 전송되면, FTL은 새로운 데이터를 다른 비어있는 페이지에 기입하고, 플래시 메모리의 여유 공간에 표시하는 것에

의해서 이전 페이지 데이터를 무효화한다. 이러한 페이지 레벨 매핑의 단점은 매핑 테이블의 크기가 크다는 것이다.

[0049] 하이브리드 매핑은 페이지 매핑 및 블록 매핑 모두를 사용한다. 이 스킴에서 모든 물리적 블록들은 로그 블록들(log blocks)과 데이터 블록들(data blocks)로 분리된다. 로그 블록들은 로그 버퍼(log buffer)라고도 불리운다. 그러므로 하이브리드 매핑 스킴을 사용하는 FTL은 로그 버퍼 기반 FTL이라고도 불리운다. 로그 블록들은 페이지 레벨 매핑과 임의 위치 스킴을 사용하고, 데이터 블록들은 블록-레벨 매핑 및 순차 위치 스킴에 의해서 처리된다. 기입 요청에 대해서 FTL은 데이터를 로그 블록으로 전송하고, 데이터 블록 내 대응하는 이전 데이터(old data)는 무효화한다.

[0050] 만일 로그 블록이 꽉차서 빈 공간이 없다면, 어느 하나의 로그 블록이 제거(victim)를 위해서 선택되고, 로그 블록 내 모든 유효한 페이지 데이터는 데이터 블록들로 이동된다. 이 단계에서, 로그 블록은 로그 블록과 관련 있는 데이터 블록들과 머지된다. 따라서 이 단계는 블록 머지라 불리운다. 블록 머지는 세 가지 즉, 완전 머지(full merge), 부분 머지(partial merge) 그리고 스위치 머지(switch merge)로 분류된다. 부분 머지 및 스위치 머지는 블록 내 모든 페이지 데이터가 순차 위치 스킴에 의해서 기입된 경우에 수행될 수 있다. 완전 머지는 많은 페이지 데이터 복사 및 블록 소거를 필요로하는 반면, 부분 머지 및 스위 머지는 적은 페이지 데이터 이동 비용을 유발한다. 하이브리드 매핑은 블록 매핑에 비해 페이지 데이터 이동 비용을 감소시킬 수 있으나 작은 크기의 매핑 테이블을 요구한다.

[0051] 플래시 메모리 시스템의 입출 성능을 향상시키기 위하여 블록 머지에 의해서 유발되는 오버헤드는 감소되어야 한다. 그러므로, 대부분의 FTL 스킴은 블록 머지의 횟수를 감소시키는 것을 목표로 한다. 그러나, 플래시 메모리 시스템이 MP3 플레이어 및 디지털 카메라와 같은 멀티미디어 시스템들을 타겟으로 하기 때문에 현재 FTL 기술은 순차적 기입 패턴에 초점이 맞추어져 있다. 그러나, 최근 플래시 메모리 장치들은 데스크탑 PC 등과 같은 범용 시스템들에 사용되고 있으므로, 순차적 및 랜덤 기입 모두에 대한 요구가 증대되고 있다.의 수는 2인 것으로 가정한다. 최근에 사용된 페이지 데이터가 제거되는 것을 방지하기 위하여 빅팀 윈도우(VW)가 사용된다. 빅팀 윈도우(VW)는 레지스터(220)에 설정된다. 이 실시예에서 빅팀 윈도우(VW)는 75%이다. 즉, 8개의 페이지 데이터 중 오래 전에 사용된 순으로 6 개의 페이지 데이터가 빅팀 윈도우(VW) 내에 포함된다. 즉, 오래 전에 사용된 순으로 6 개의 페이지 데이터 중 최근 빅팀 블록(RVB)에 대응하는 2 개의 블록들이 검색된다.

[0052] 만일 버퍼 캐쉬(212a) 내 어떤 페이지 데이터도 선택될 수 있는 상황이면 버퍼 캐쉬(212a) 내 가장 오래 전에 사용된 페이지 데이터가 선택된다. 예컨대, 로그 버퍼(320)가 비어있거나, 빅팀 윈도우 내 페이지 데이터가 동일한 블록 내 페이지 데이터인 경우 등이 이에 해당한다. 도 1의 LRU 레지스터(230)는 가장 오래전에 사용된 페이지 데이터 정보를 저장한다. 만일 버퍼 캐쉬(212a)에 최근 빅팀 블록(RVB) 내 대응하는 블록의 페이지 데이터가 빅팀 윈도우(VW) 밖에 존재하면 다른 페이지 데이터가 제거되고 나서 그 페이지 데이터가 빅팀 윈도우(VW) 내 들어갈 수 이다.

[0053] 도 4는 새로운 페이지 데이터(p2, p6, p10 및 p14)이 버퍼 캐쉬 내 삽입될 때 페이지 데이터(p8, p12, p8 및 p13)이 제거(축출)되는 것을 보여준다. 모든 빅팀 페이지 데이터가 플래시 메모리(300)로 플러쉬되어서 빅팀 페이지 데이터 리스트 내에 어떤 페이지 데이터도 없을 때, 새로운 최근 빅팀 블록(RVB)이 구성되어야 한다. 새로운 최근 빅팀 블록들(RVB)은 B0, B1이다.

[0054] 도 5는 본 발명의 바람직한 실시예에 따른 버퍼 캐쉬 관리 방법을 보여주는 플로우차트이다.

[0055] 도 5를 참조하면, 앞서 설명한 바와 같이, 파일 시스템(100)으로부터 FTL(200)로 플래시 메모리(300)에 기입될 페이지 데이터가 입력되면(510), 메모리 관리기(210)는 버퍼 캐쉬(212)가 꽉 찼는지 판별한다(520). 만일 버퍼 캐쉬(212)가 꽉차지 않았다면 수신된 페이지 데이터를 버퍼 캐쉬(212)에 삽입한다. 만일 버퍼 캐쉬(212)가 꽉 찼다면, 버퍼 캐쉬(212)에 저장된 페이지 데이터 중 하나를 제거해야만 한다. 이 때 메모리 관리기(210)는 앞서 설명한 바와 같이 빅팀 윈도우 내 이전 빅팀 블록에 대응하는 페이지 데이터를 제거한다(530). 페이지 데이터 제거에 의해서 버퍼 캐쉬에 빈 공간이 생기면 메모리 관리기(210)는 빈 공간에 새로운 페이지 데이터를 삽입한다(540).

[0056] 본 발명의 버퍼 캐쉬 관리 방법은 소정의 빅팀 블록들의 페이지 데이터만을 제거하기 때문에 1대1 페이지 데이터 제거의 효과가 모든 빅팀 페이지 데이터를 제거하는 것과 유사하다. 대신에, 1대1 페이지 데이터 제거는 빅팀 페이지 데이터 상의 갱신 요청이 발생할 때 버퍼 캐쉬의 미스(miss) 횟수를 감소시킬 수 있으므로 플래시 메모리의 기입 요청의 횟수가 감소된다.

[0057] 빅팀 윈도우(VW)의 크기는 기입 패턴의 위치를 고려하여 주의깊게 선택되어야만 한다. 만일 빅팀 윈도우(VW)의 크기가 너무 크면 최근에 사용된 페이지 데이터가 제거되어서 버퍼 캐쉬의 미스율(miss ratio)이 증가한다. 만일 빅팀 윈도우(VW)의 크기가 너무 작으면 종래의 LRU 스킴과 유사하게 동작하게 되므로 로그 블록의 쓰래싱을 유발할 수 있다. 데스크탑 벤치마킹 어플리케이션들을 이용한 테스트를 통하여 빅팀 윈도우(VW)의 크기는 버퍼 캐쉬의 전체 크기의 약 75%가 바람직하다.

[0058] 상술한 바와 같이 본 발명의 캐쉬 버퍼 관리 방법은 빅팀 블록의 페이지 데이터가 버퍼 캐쉬에서 제거되도록 강제한다. 이러한 관리 방법은 블록 쓰레싱 및 블록 관련성을 최소화함으로써 로그 영역 내 로그 블록들이 데이터 블록과 머지되는 횟수를 감소시킨다. 그 결과, 플래시 메모리에 대한 액세스 속도가 향상될 수 있다. 더욱이, 필요에 따라서 빅팀 윈도우의 크기와 빅팀 블록을 조절할 수 있다.

도면의 간단한 설명

[0059] 도 1은 본 발명의 바람직한 실시예에 따른 플래시 메모리 시스템에 포함되는 파일 변환 계층(FTL)을 보여주는 도면이다.

[0060] 도 2는 도 1에 도시된 버퍼 캐쉬 및 플래시 메모리의 관계를 구체적으로 보여주는 도면이다.

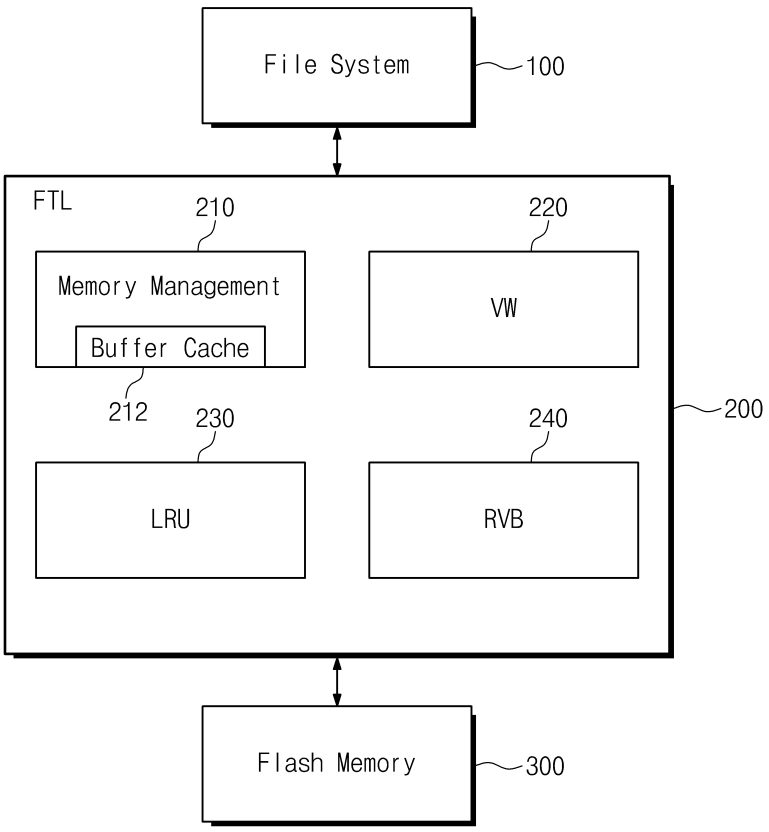
[0061] 도 3은 1:N 로그 블록 매핑 스킴에 적용된 본 발명의 버퍼 캐쉬 관리 방법을 설명하기 위한 버퍼 캐쉬와 플래시 메모리의 관계를 예시적으로 보여주는 도면이다.

[0062] 도 4는 본 발명의 바람직한 실시예에 따른 버퍼 캐쉬 관리 기법에 따라서 도 1에 도시된 메모리 관리기가 버퍼 캐쉬에 저장된 페이지 데이터 중 빅팀 블록을 선택하는 것을 예시적으로 보여주는 도면이다.

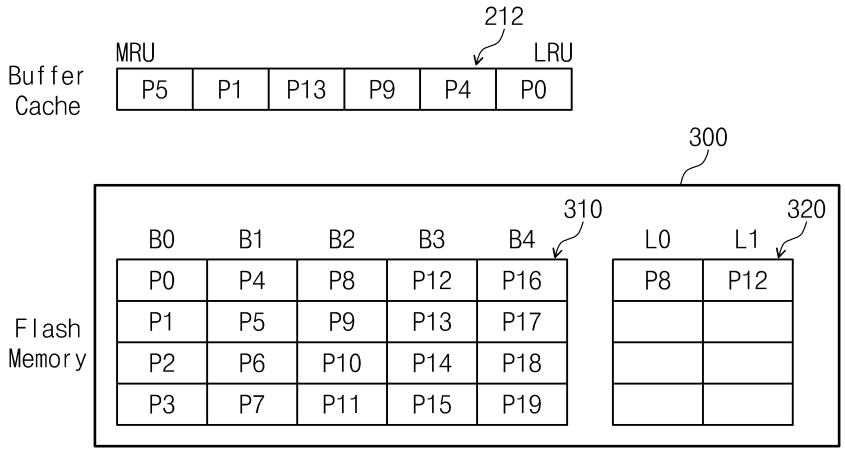
[0063] 도 5는 본 발명의 바람직한 실시예에 따른 버퍼 캐쉬 관리 방법을 보여주는 플로우차트이다.

도면

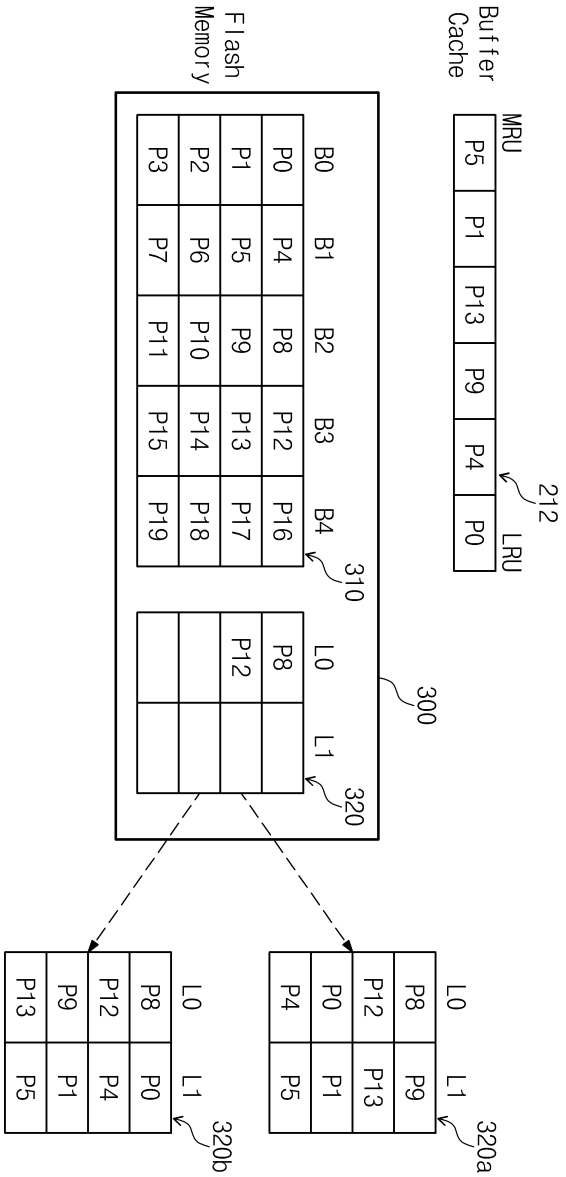
도면1



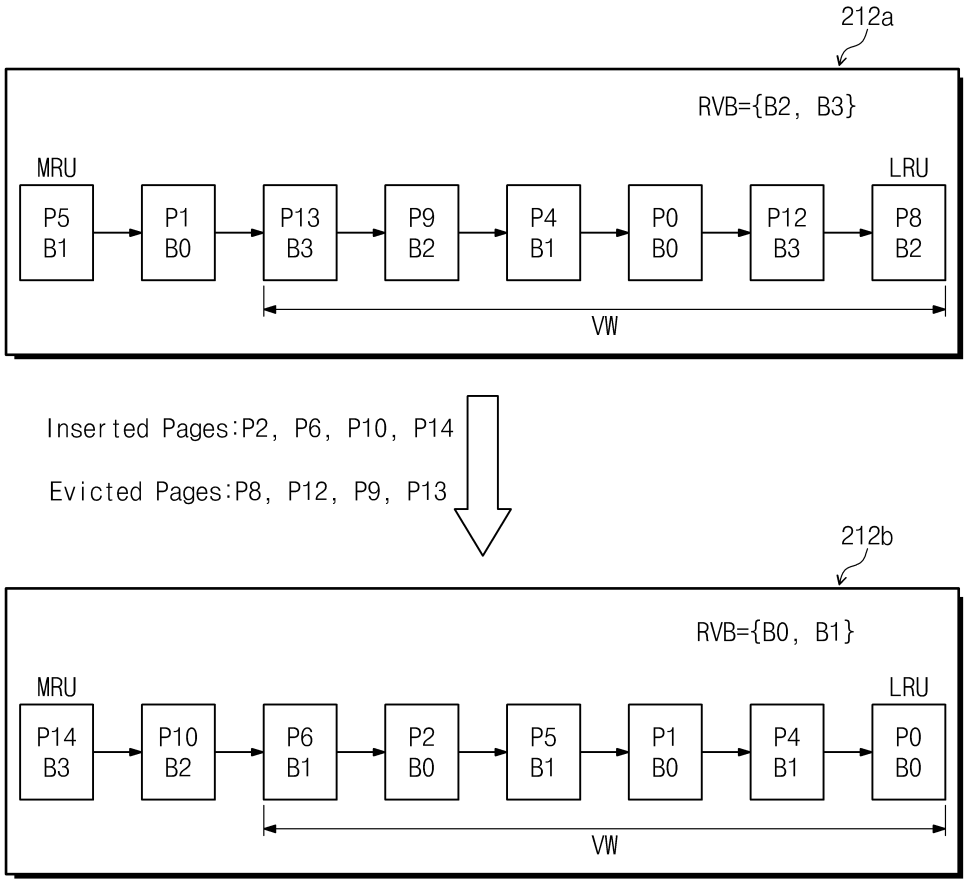
도면2


212
MRU
LRU
Buffer Cache
P5
P1
P13
P9
P4
P0
300
310
320
B0
B1
B2
B3
B4
L0
L1
Flash Memory
P0
P4
P8
P12
P16
P8
P12
P1
P5
P9
P13
P17
P2
P6
P10
P14
P18
P3
P7
P11
P15
P19

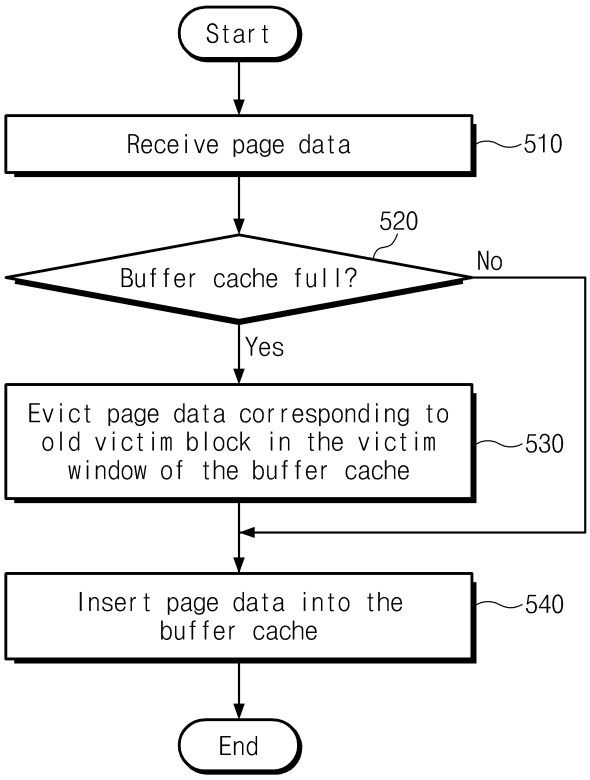
도면3


Buffer Cache
MRU
P5
P1
P13
P9
P4
P0
LRU
212
Flash Memory
B0
B1
B2
B3
B4
P0
P4
P8
P12
P16
P1
P5
P9
P13
P17
P2
P6
P10
P14
P18
P3
P7
P11
P15
P19
310
L0
L1
P8
P12
320
300
L0
L1
P8
P9
P12
P13
P0
P1
P4
P5
320a
L0
L1
P8
P0
P12
P4
P1
P13
P5
P9
320b

도면4


212a
RVB={B2, B3}
MRU
LRU
P5 B1
P1 B0
P13 B3
P9 B2
P4 B1
P0 B0
P12 B3
P8 B2
VW
Inserted Pages:P2, P6, P10, P14
Evicted Pages:P8, P12, P9, P13
212b
RVB={B0, B1}
MRU
LRU
P14 B3
P10 B2
P6 B1
P2 B0
P5 B1
P1 B0
P4 B1
P0 B0
VW

도면5


Start
Receive page data
510
520
Buffer cache full?
No
Yes
Evict page data corresponding to old victim block in the victim window of the buffer cache
530
Insert page data into the buffer cache
540
End