



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2009-0090570
(43) 공개일자 2009년08월26일

(51) Int. Cl.

G06F 12/06 (2006.01) G06F 9/00 (2006.01)

(21) 출원번호 10-2008-0015857

(22) 출원일자 2008년02월21일

심사청구일자 2008년02월21일

(71) 출원인

인하대학교 산학협력단

인천 남구 용현동 253 인하대학교

(72) 발명자

유태경

서울 영등포구 대림3동 신동아아파트 3동 1003호

김기창

인천 연수구 동춘동 924번지 대림2차아파트 207동 404호

(74) 대리인

황광연, 강동호, 오정환

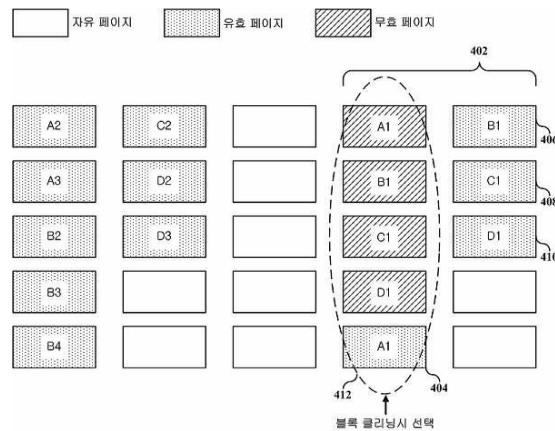
전체 청구항 수 : 총 5 항

(54) 메모리의 페이지 할당 방법 및 이를 수행하기 위한프로그램이 기록된 컴퓨터로 판독 가능한 기록 매체

(57) 요약

논리 페이지에 대한 업데이트 정보를 가지는 윈도우 정보로써 메모리의 페이지를 할당하는 본 발명의 일 실시예에 따른 메모리의 페이지 할당 방법은, 물리 페이지(Physical Page)로써 구성되는 복수의 블록(Block)을 포함하는 메모리의 페이지 할당 방법에 있어서, 상기 복수의 블록을 제1 블록 및 제2 블록으로 구분하는 단계; 특정 논리 페이지에 대한 업데이트 요청을 수신하면, 상기 특정 논리 페이지의 업데이트 횟수를 산출하는 단계; 및 상기 업데이트 요청에 상응하여 업데이트되는 특정 논리 페이지에 대해, 상기 산출된 업데이트 횟수에 따라 상기 제2 블록의 물리 페이지 할당 여부를 결정하는 단계를 포함하는 것을 특징으로 한다.

대표도 - 도4



특허청구의 범위

청구항 1

물리 페이지(Physical Page)로써 구성되는 복수의 블록(Block)을 포함하는 메모리의 페이지 할당 방법에 있어서,

상기 복수의 블록을 제1 블록 및 제2 블록으로 구분하는 단계;

특정 논리 페이지에 대한 업데이트 요청을 수신하면, 상기 특정 논리 페이지의 업데이트 횟수를 산출하는 단계; 및

상기 업데이트 요청에 상응하여 업데이트되는 특정 논리 페이지에 대해, 상기 산출된 업데이트 횟수에 따라 상기 제2 블록의 물리 페이지 할당 여부를 결정하는 단계;

를 포함하는 것을 특징으로 하는 메모리의 페이지 할당 방법.

청구항 2

제1항에 있어서, 상기 물리 페이지 할당 여부 결정 단계에서,

상기 특정 논리 페이지를 포함하는 파일은 복수의 논리 페이지로 구성되되, 상기 업데이트 횟수가 상기 복수의 논리 페이지의 평균 업데이트 횟수 이상이면, 상기 제2 블록의 물리 페이지를 할당하는 것으로 결정하는 것을 특징으로 하는 메모리의 페이지 할당 방법.

청구항 3

제2항에 있어서,

상기 파일을 구성하는 논리 페이지 각각에 대해 업데이트 횟수, 논리 페이지 번호, 및 상기 파일에 대한 링크 정보 중 적어도 하나를 포함하는 윈도우 정보를 생성하는 단계를 더 포함하되, 상기 평균 업데이트 횟수는 상기 윈도우 정보를 사용하여 산출되는 것을 특징으로 하는 메모리의 페이지 할당 방법.

청구항 4

제3항에 있어서,

상기 업데이트 요청 수신 시, 기 생성된 상기 윈도우 정보의 수가 기 설정된 생성 가능한 윈도우 정보의 최대치와 동일하고 상기 특정 논리 페이지에 해당하는 윈도우 정보가 존재하지 않는 경우, 상기 기 생성된 상기 윈도우 정보 중에서 업데이트 횟수가 최소이거나 가장 먼저 생성된 윈도우 정보를 삭제하고, 상기 특정 논리 페이지에 해당하는 윈도우 정보를 생성하는 단계를 더 포함하는 것을 특징으로 하는 메모리의 페이지 할당 방법.

청구항 5

제1항 내지 제4항 중 어느 하나의 항에 기재된 방법을 수행하기 위한 프로그램이 기록된 컴퓨터로 판독 가능한 기록 매체.

명세서

발명의 상세한 설명

기술분야

<1> 본 발명은 메모리의 페이지 할당 방법 및 이를 수행하기 위한 프로그램이 기록된 컴퓨터로 판독 가능한 기록 매체에 관한 것으로서 보다 상세하게는 논리 페이지에 대한 업데이트 정보를 가지는 윈도우 정보로써 메모리의 페이지를 할당하는 방법 및 이를 위한 프로그램이 기록된 기록 매체에 관한 것이다.

배경기술

<2> 일반적으로 낸드 플래시 메모리는 복수의 블록(Block)과 블록을 구성하는 복수의 페이지(Page)로서 구분된다. 파일(File)은 복수의 페이지에 저장될 수 있고, 하나의 파일이 복수의 블록에 존재하거나 한 블록에 복수의 파

일이 저장될 수 있다.

- <3> 도 1의 일반적인 낸드 플래시 메모리(102)의 구조를 나타낸 도면으로서, 한 블록(106)이 32개의 페이지(104)로 구성되고, 각 페이지(104)는 512바이트의 데이터 영역과 16바이트의 스페어 영역을 가진다. 전체 낸드 플래시 메모리(102)는 4096개의 블록(106)으로 이루어 지고, 각 블록(106)은 32개의 528바이트 크기의 페이지(104)로 이루어지므로 66메가 바이트(528메가 비트)의 크기를 가진다.
- <4> 그런데, 낸드 플래시 메모리는 기존 저장 매체와 구별되는 다음의 제한 사항들을 가진다. 첫째, 덮어 쓰기를 하기 위해서는 삭제 연산이 선행 되어야 하고, 둘째, 삭제 연산과 읽기, 쓰기 연산의 단위가 다르다. 셋째, 삭제 연산이 다른 연산에 비해 느리고 전력 소비가 크며, 넷째, 각 블록마다 삭제 연산의 횟수가 제한된다.
- <5> 상술한 제한 사항을 극복하기 위해 LFS(Log-structured File System)에서는 덮어 쓰기 연산을 추가 연산으로 변형하여 처리하는 외부 갱신 기법을 사용한다. 이것은 수정된 데이터를 다른 빈 영역에 쓰고, 기존 데이터를 무효화시킴으로써, 덮어 쓰기 시에 선행되어야 할 삭제 연산을 제거한다. 그러나 그 과정에서 생긴 무효화 데이터들을 회수하여 다시 쓰기 가능한 상태로 전환하는 블록 클리닝이 요구되고, 그 블록 클리닝 비용 및 횟수를 줄이는 것이 파일 시스템의 성능에 큰 영향을 미친다.
- <6> 블록 클리닝은 먼저 삭제할 희생 블록을 선택하고, 그 블록의 유효 데이터들을 다른 블록에 재복사한 후, 그 블록을 삭제하는 순서로 이루어 진다.
- <7> 그런데, 업데이트된 논리 페이지를 위해 다른 물리 페이지를 할당하고, 기존 물리 페이지를 무효화는 과정에서 일반적인 방법은 물리 페이지를 순차적으로 할당하여, 유효 페이지와 무효 페이지가 메모리의 전 영역에 걸쳐 퍼져 있게 된다. 따라서, 블록 클리닝 시에 무효 페이지들이 존재하는 블록을 희생 블록으로 선택한다면 블록 클리닝을 위한 읽고, 쓰는 시간이 크다는 문제점이 있었다.
- <8> 또한, 블록 클리닝 시에 유효 페이지가 많은 블록을 희생 블록으로 선택한다면, 블록 클리닝으로 회수되어 사용할 수 있는 물리 페이지가 적어, 블록 삭제 횟수가 많아 진다는 문제점이 있었다.

발명의 내용

해결 하고자하는 과제

- <9> 본 발명은 상술한 문제점을 해결하기 위한 것으로서, 업데이트된 논리 페이지에 대해 효율적으로 물리 페이지를 할당하여 블록 클리닝을 위한 비용이 적은 메모리의 페이지 할당 방법을 제공하는 것을 기술적 과제로 한다.
- <10> 또한, 본 발명은 블록 클리닝 이후에 회수되는 물리 페이지가 많아 블록 삭제 횟수가 적은 메모리의 페이지 할당 방법을 제공하는 것을 다른 기술적 과제로 한다.

과제 해결수단

- <11> 상술한 목적을 달성하기 위한 본 발명의 일 측면에 따른 메모리의 페이지 할당 방법은 물리 페이지(Physical Page)로써 구성되는 복수의 블록(Block)을 포함하는 메모리의 페이지 할당 방법에 있어서, 상기 복수의 블록을 제1 블록 및 제2 블록으로 구분하는 단계; 특정 논리 페이지에 대한 업데이트 요청을 수신하면, 상기 특정 논리 페이지의 업데이트 횟수를 산출하는 단계; 및 상기 업데이트 요청에 상응하여 업데이트되는 특정 논리 페이지에 대해, 상기 산출된 업데이트 횟수에 따라 상기 제2 블록의 물리 페이지 할당 여부를 결정하는 단계를 포함하는 것을 특징으로 한다.

효 과

- <12> 상술한 바와 같이 본 발명에 따르면, 자주 업데이트 되는 핫 페이지(Hot Page)를 다른 블록에 위치시킴으로써 블록 클리닝 시에 삭제할 블록의 유효 페이지를 최대한 감소시켜 전체적인 블록 클리닝 비용을 감소시킬 수 있는 메모리의 페이지 할당 방법을 구현하는 효과가 있다.
- <13> 또한, 본 발명에 따르면, 블록 클리닝 시 많은 무효 페이지가 회수되어 사용할 수 있는 물리 페이지를 증가시킴으로써, 블록 삭제 횟수가 적고 가비지 콜렉션(Garbage Collection)이 적게 수행되게 하는 메모리의 페이지 할당 방법을 구현하는 다른 효과가 있다.

발명의 실시를 위한 구체적인 내용

- <14> 본 발명의 실시예에 대한 상세한 설명을 하기 이전에 도 2를 참조하여 파일의 수정에 따른 플래시 메모리의 일반적인 페이지(Page) 할당 방법, 가비지 콜렉션(Garbage Collection), 및 블록 클리닝(Block Cleaning)에 대해 간략히 설명한다.
- <15> 도 2는 파일 수정 후, 일반적인 페이지 할당 방법에 따른 메모리의 상태를 나타낸 도면이다. 일 예를 낸드 플래시 전용 파일 시스템인 YAFFS(Yet Another Flash File System)으로써 설명한다.
- <16> YAFFS에서는 파일 시스템이 마운트된 이후 파일이 처음 생성되면, 메모리에서 파일의 정보를 저장할 yaffs_Object와 파일의 위치를 저장할 yaffs_Tnode를 할당 받는다. 그리고, 페이지 할당 정책에 의해 자유 페이지를 할당 받아 실제 데이터를 쓰고, 그 페이지의 물리 주소를 yaffs_Tnode에 기록한다. 따라서, yaffs_Object에 파일의 논리 페이지 주소가 저장되고 yaffs_Tnode 트리(Tree)를 사용하여 물리 페이지 주소를 찾게 된다.
- <17> 또한, 파일의 크기, 데이터 페이지 개수, 파일의 생성 및 수정 시간들의 파일의 정보를 yaffs_Object에 업데이트하고, 그것을 바탕으로 yaffs_ObjectHeader의 내용을 구성하여 자유 페이지를 할당 받아 플래시 메모리에 기록한다.
- <18> 파일의 데이터 중 일부가 수정되면, 먼저 자유 페이지가 할당되고, 수정된 데이터는 할당된 자유 페이지에 기록된다. 그 다음 이전 데이터가 저장된 페이지를 무효 페이지로 만든다. 그리고 논리 주소로 Tnode 트리를 검색하여 수정된 데이터가 저장된 페이지의 물리 주소를 업데이트 한다. 따라서, 파일의 데이터 중 일부가 수정된다는 것은 파일을 구성하는 페이지들 중 업데이트 대상인 페이지가 무효 페이지로 바뀌고, 새롭게 할당된 자유 페이지에 업데이트된 데이터가 저장됨을 의미한다.
- <19> 도 2의 메모리 구조에는 5개의 페이지를 구성 요소로 포함하는 블록이 5개 존재하고, A파일, B파일, C파일, 및 D파일이 저장되어 있다. 페이지 A1, A2 및 A3은 A파일을 구성하고, 마찬가지로 나머지 파일이 구성된다. A1(202), A2, A3, B1(206) 및 B2가 첫 번째 블록을 구성하며, 마찬가지로 한 페이지 열(Column)이 한 블록이 되어, 나머지 블록이 구성된다.
- <20> A파일의 A1에 대한 업데이트 요청이 수신되면, 기존의 A1은 무효 페이지 A1(202)로 변환되고, 새로운 자유 페이지가 할당되어 업데이트된 데이터는 유효 페이지 A1(204)에 저장된다. 마찬가지로 방식으로 무효 페이지 B1(206) 및 유효 페이지 B1(208)이 생성된다.
- <21> 그런데, 도 2에서는 A1, B1 및 C1의 업데이트를 위해 세 번째 블록에서 D3의 다음에 위치하는 자유 페이지들이 순차적으로 할당된 것을 확인할 수 있다. 따라서, 페이지들에 대한 업데이트가 계속 진행됨에 따라 A1(202) 및 B1(206)과 같은 무효 페이지들이 플래시 메모리의 전 영역에 흩어져 존재하게 된다.
- <22> 이로 인해, 사용할 수 있는 자유 블록의 개수가 점점 줄어 들게 된다. 그래서 데이터를 저장할 공간이 부족하게 되면, 가비지 콜렉션을 수행하여, 지움 정책에 의해 희생 블록을 선택하여 그것을 지움으로써 공간을 확보한다
- <23> 상기 희생 블록을 선택하여 지움으로써 공간을 확보하는 것을 블록 클리닝이라 하며, 일 예로 도 2에서는 블록 클리닝을 위한 희생 블록으로서 두 번째 블록(210)이 선택되었다. 이하, 도 3을 참조하여, 일반적인 블록 클리닝 이후의 메모리의 상태를 설명하겠다.
- <24> 도 3를 참조하면, 두 번째 블록(302)의 페이지들은 클리닝되어 자유 페이지들로 변환되었고, 네 번째 블록의 B3(304), B4(306) 및 C2(308)는 이전의 두 번째 블록(210)에 포함된 B3, B4, 및 C2의 데이터가 블록 클리닝에 의해 새롭게 할당된 자유 페이지에 쓰여 져 생성된 것이다.
- <25> 도 2 및 도 3에 나타난 일반적인 페이지 할당 방법은 블록 클리닝 시에 많은 비용을 요구한다. 즉, 두 번째 블록(302)의 블록 클리닝 시에 B3(304), B4(306) 및 C2(308)를 생성하기 위해 세 번의 읽기와 세 번의 쓰기를 수행하고, 한 번의 블록 삭제 시간이 필요하게 된다.
- <26> 이하 첨부된 도면을 참조하여 본 발명의 실시예에 대해 상세히 설명한다.
- <27> 도 4는 본 발명의 일 실시예에 따른 페이지 할당 방법으로써 파일이 수정된 후, 메모리의 상태를 나타낸 도면이다.
- <28> 도시된 바와 같이, 본 발명의 일 실시예에 따른 할당 방법에서는 유효, 무효 페이지가 분리되어 존재 한다. 그 분리 방법은 논리 페이지가 업데이트 횟수가 많은 핫 페이지(Hot Page: A1,A2,A3,A4)인 경우, 해당 물리 페이지들을 핫 페이지를 위한 블록(402)에 할당하는 것이다.

- <29> 도 4를 참조하면, 파일 A, B, C 및 D의 논리 페이지인 A1, B1, C1 및 D1이 자주 업데이트 되는 핫 페이지인 경우, 업데이트 시 A1, B1, C1, 및 D1의 물리 페이지인 유효 페이지 A1(404), B1(406), C1(408) 및 D1(410)은 핫 페이지를 위한 블록(402)에 저장된다. 이 경우, 업데이트되기 이전의 A1, B1, C1, 및 D1의 물리 페이지들은 무효 페이지 A1, B1, C1 및 D1으로 변환된다.
- <30> 이 경우, 무효 페이지들이 존재하는 영역의 블록(412)을 희생 블록(412)으로 선택한다면 상대적으로 희생 블록(412)에 포함된 유효 페이지가 적고, 무효 페이지는 많기 때문에 도 2 및 도 3을 통해 진술한 일반적인 경우보다 더 적은 비용으로 더 많은 페이지를 회수할 수 있다. 블록 클리닝 시에 기존 방식은 $[3 * (\text{페이지 읽기 시간} + \text{페이지 쓰기 시간}) + \text{블록 삭제 시간}]$ 의 비용이 드는 반면, 본 발명의 일 실시예에 따른 방식은 $[1 * (\text{페이지 읽기 시간} + \text{페이지 쓰기 시간}) + \text{블록 삭제 시간}]$ 의 상대적으로 적은 비용이 든다.
- <31> 도 5는 본 발명의 일 실시예에 따른 페이지 할당 방법으로써 파일이 수정된 후, 블록 클리닝이 끝난 메모리의 상태를 나타낸 도면이다.
- <32> 도 5를 참조하면, 블록 클리닝이 끝난 블록(502)의 페이지들이 모두 자유 페이지로 전환되어 있다. 또한, 블록 클리닝 시 선택된 블록(412)의 A1은 핫 페이지를 위한 블록(402)의 물리 페이지를 할당 받고, 해당 물리 페이지에 저장된다. 따라서, 다섯 번째 블록의 유효 페이지 A1(504)가 생성되게 된다.
- <33> 결국, 도 2 및 도 3의 일반적인 경우와는 다르게, 블록 클리닝 시 유효 페이지 A1(504)만이 생성되므로, 블록 삭제 시간 외에 1번의 읽기와 1번의 쓰기 시간만이 소요됨을 알 수 있다.
- <34> 도 4 및 도 5에 의하면, 현재 사용할 수 있는 자유 페이지의 개수가 일반적인 방법에서는 11이고, 본 발명에 따른 일 실시예에서는 13이다. 왜냐하면 제안된 기법이 블록 클리닝 이후에 회수된 페이지가 많기 때문이다.
- <35> 가비지 콜렉션은 사용할 수 있는 영역이 미리 정의된 임계값 이하로 내려 갈 경우에 실행되어 자유 페이지 영역이 확보된다. 그러므로 본 발명의 일 실시예에서는 상대적으로 많은 무효 페이지가 회수되어 사용할 수 있는 영역이 더 많으므로, 가비지 콜렉션이 덜 수행되게 되고, 그로 인해 블록 삭제 횟수가 적어 진다.
- <36> 따라서 본 발명에서는 자주 업데이트 되는 핫 페이지를 다른 블록에 위치시킴으로써 블록 클리닝 시에 삭제할 블록의 유효 페이지를 최대한 감소시켜 전체적인 블록 클리닝 비용을 감소 시키고 블록 삭제 횟수를 감소시키는 효과가 있다.
- <37> 한편, 본 발명의 일 실시예에서는 업데이트 되는 논리 페이지가 핫 페이지인지 판단하기 위해, 논리 페이지의 업데이트 정보를 가지는 윈도우 및 윈도우들의 집합인 윈도우 리스트를 사용한다. 일 실시예에 있어서, 윈도우는 RAM에 생성되거나 플래시 메모리에 생성될 수 있다.
- <38> 도 6a는 본 발명의 일 실시예에 따른 논리 페이지의 업데이트 정보를 저장하기 위해 사용되는 윈도우 구조체를 나타내는 도면이다.
- <39> 도 6a를 참조하면, chunkInInode는 업데이트 되는 논리 페이지 번호를 저장하며, update_cnt는 업데이트 된 횟수를 저장하고, Link는 윈도우 리스트를 만들기 위한 링크로서 사용된다.
- <40> 도 6b는 본 발명의 일 실시예에 따른 윈도우 리스트를 나타내는 도면으로서, 특정 파일의 논리 페이지가 논리 페이지 번호 3, 4, 6, 8, 1, 18, 14, 15의 순서로 업데이트 되면서 생성된 윈도우 리스트를 나타낸다.
- <41> 윈도우 리스트는 미리 정의된 최대 개수(MAX_WINDOW_SIZE)이하의 윈도우로 구성되며, 파일 당 하나이다. 윈도우의 개수(윈도우 사이즈)가 최대일 경우 새로운 논리 페이지를 위한 윈도우를 할당하기 위해 업데이트 카운트가 가장 적은 윈도우 중에, 가장 오래된 윈도우를 희생 윈도우로 선택하여 삭제한 후에 새로운 윈도우를 추가 시킨다. 그리고 윈도우들의 업데이트 횟수의 총합이 미리 정의된 최대 횟수(MAX_TOTAL_UPDATE_CNT)이상이 되면 윈도우 리스트를 초기화 시킨다.
- <42> 도 6b를 참조하면, 윈도우는 파일 생성과 동시에 생성되는 메인 윈도우(602)와 페이지가 업데이트되는 경우 생성되어 윈도우 리스트에 추가되는 서브 윈도우들(604, 606)로 구분될 수 있다.
- <43> 메인 윈도우(602)의 windows_head를 통해 서브 윈도우들이 연결되고, windows_size에는 서브 윈도우의 수를, total_update_cnt에는 윈도우 리스트를 구성하고 있는 서브 윈도우들의 업데이트 횟수의 합을 저장하며, 각각 윈도우 리스트의 평균 업데이트 횟수를 계산할 때 사용된다.
- <44> 도시된 바와 같이, 논리 페이지 번호 3에 해당하는 페이지가 업데이트 되면, 해당 서브 윈도우(604)가

생성되고, 메인 윈도우(602)에 링크된다. 상기 서브 윈도우(604)의 chunkInNode(608)에는 논리 페이지 번호인 3이 저장되고, update_cnt(610)에는 1번 업데이트되는 것이므로 1이 저장된다.

- <45> 다음으로, 논리 페이지 주소 4에 해당하는 페이지가 업데이트 되면, 해당 서브 윈도우(606)가 생성되고, 논리 페이지 번호 3의 서브 윈도우(604)에 링크된다.
- <46> 마찬가지로, 논리 페이지 번호 6, 7, 1, 18에 대해서도 업데이트 시마다 해당 윈도우가 생성된다.
- <47> 도 6b의 윈도우 리스트에서는 메인 윈도우(602)의 windows_size는 6이고, total_update_cnt는 6이다.
- <48> 또한, 윈도우 리스트의 평균 업데이트 횟수(avg_update_cnt)는 $1(=total_update_cnt/windows_size)$ 이다.
- <49> 본 발명의 일 실시예에서는 업데이트되는 논리 페이지가 핫 페이지인지 판단하기 위해 상기 윈도우 리스트를 사용한다. 그리고, 핫 페이지로 판단되면 업데이트될 핫 페이지를 위한 블록(이하 '핫 블록' 한다)의 물리 페이지를 할당한다. 반대의 경우에 일반 블록의 물리 페이지를 할당한다.
- <50> 일 예로, 업데이트되는 논리 페이지가 번호 3에 해당하고, 이미 윈도우 리스트에 해당 윈도우가 존재하면, 핫 페이지인지 판단하는 방법은 업데이트 대상인 논리 페이지의 update_cnt(610)에서 획득한 논리 페이지의 업데이트 횟수가 메인 윈도우(602)의 windows_size 및 total_update_cnt로부터 계산된 평균 업데이트 횟수(avg_update_cnt) 이상인지 비교하는 것이다. 여기서, 비교 결과, 평균 업데이트 횟수 이상이면, 핫 페이지로 판단한다.
- <51> 도 7은 본 발명의 일 실시예에 따라 업데이트하려는 논리 페이지의 윈도우가 이미 윈도우 리스트에 존재하는 경우, 해당 윈도우의 이동을 나타내는 도면이다.
- <52> 도 6의 윈도우 리스트가 존재하는 경우, 특정 페이지에 대한 업데이트 요청이 수신되면, 먼저 특정 페이지의 서브 윈도우가 윈도우 리스트에 존재하는 지 판단한다. 도 7에서는 논리 페이지 번호 3에 해당하는 페이지의 업데이트 요청이 수신되었으므로, 이미 업데이트 대상이 되는 페이지의 서브 윈도우(702)가 존재하는 경우이다.
- <53> 그리고, 업데이트될 논리 페이지가 핫 페이지인지 판단하여, 핫 블록의 물리 페이지를 할당할 것인지 판단한다. 그런데, 도 6에서 상술한 바와 같이 평균 업데이트 횟수가 1이고, 현재 해당 서브 윈도우(702)의 update_cnt는 1이므로, 해당 페이지는 핫 페이지로 판단된다. 따라서, 핫 블록의 물리 페이지가 할당된다.
- <54> 이후, 해당 서브 윈도우(702)의 update_cnt를 1증가시켜 2(704)를 기록한다. 그리고, 해당 서브 윈도우(702)를 최신 위치(706)로 이동 시킨다.
- <55> 도 8은 본 발명의 일 실시예에 따라 윈도우 리스트에서 존재 가능한 윈도우 개수가 최대치인 경우, 업데이트 요청에 의해 새로운 윈도우를 추가하는 방법을 나타내는 도면이다.
- <56> 도 8은 도 7에서 업데이트된 윈도우 리스트에서 새롭게 논리 페이지 번호 5, 8, 9, 2, 15에 해당하는 페이지가 순차적으로 업데이트 되는 경우 변화되는 윈도우 리스트 구조를 나타낸다. 이하, 기 설정된 존재 가능한 서브 윈도우의 개수의 최대치(MAX_WINDOW_SIZE)가 10이라 가정하고 설명한다.
- <57> 논리 페이지 번호 5, 8, 9, 2에 해당하는 페이지가 순차적으로 업데이트되면, 해당 논리 페이지의 윈도우는 도 7의 윈도우 리스트에 존재하지 않으므로, 새롭게 해당 서브 윈도우들(802,804,806,808)이 순차적으로 최신 위치에 추가된다.
- <58> 그런데, 논리 페이지 번호 2에 해당하는 서브 윈도우(808)가 추가되는 시점에서 윈도우 리스트의 총 서브 윈도우 개수는 MAX_WINDOW_SIZE(=10)와 동일하게 된다. 따라서, 추가로 업데이트를 위해 논리 페이지 번호 15에 해당하는 서브 윈도우(812)를 추가하려면, 윈도우 리스트에서 하나의 서브 윈도우를 제거해야 한다.
- <59> 여기서, 업데이트 횟수가 가장 작고, 그 중에 가장 오래 전에 업데이트된 논리 페이지의 정보가 기록된 윈도우(810)를 제거한다. 그리고, 윈도우 전체 업데이트 횟수에서 그 윈도우(810)의 업데이트 횟수를 뺀 후, 논리 페이지 번호 15를 위한 새로운 윈도우(812)를 할당하고, 윈도우 리스트에 포함 시킨다.
- <60> 도 9는 본 발명의 일 실시예에 따른 메모리의 페이지 할당 방법을 나타낸 순서도이다.
- <61> 먼저, 메모리의 블록을 일반 블록과 핫 블록으로 구분한다(S902).
- <62> 다음으로, 특정 논리 페이지에 대한 업데이트 요청을 수신한 경우, 윈도우 리스트의 전체 업데이트 횟수가 기 설정된 최대치 이상인지 판단한다(S904). 여기서, 윈도우는 업데이트되는 논리 페이지에 대한 정보를 포함하는

구조체이며 윈도우 리스트는 파일 당 생성되며, 업데이트 순서에 따라 링크된 윈도우들의 집합이다.

- <63> 다음으로, 전체 업데이트 횟수가 기 설정된 최대치 이상이면, 윈도우 리스트를 초기화 하고(S906), 현재 존재하는 윈도우 리스트에 상기 특정 논리 페이지에 해당하는 윈도우가 존재하는지 판단한다(S908). 만약, 업데이트 횟수가 기 설정된 최대치 미만이면, S906 단계를 거치지 않고 S908 단계를 수행한다.
- <64> 이하, S908 단계에서 윈도우 리스트에 업데이트 대상인 논리 페이지에 대한 윈도우가 존재하지 않는 경우를 설명하겠다.
- <65> 논리 페이지에 대한 윈도우가 존재하지 않는 경우 윈도우 리스트의 윈도우 사이즈(윈도우의 수)가 허용 가능한 최대치인지 판단한다(S910).
- <66> 만약, 윈도우 사이즈가 허용 가능한 최대치이면, 업데이트 횟수가 가장 적고, 오래된 윈도우를 삭제한다(S912). 그리고, 업데이트 대상 논리 페이지를 위한 윈도우를 할당하고, 그 윈도우에 논리 페이지 번호를 기록한다(S914). 그 후, 윈도우의 업데이트 카운트를 1로 하고 윈도우 리스트에 연결한다(S916).
- <67> 다음으로, 업데이트 요청에 상응하여 업데이트 되는 논리 페이지를 위해 일반 블록의 물리 페이지를 할당한다(S918).
- <68> 만약, S910 단계에서 윈도우 사이즈가 허용 가능한 최대치가 아니면, S912 단계를 수행하지 않고, 업데이트 되는 논리 페이지를 위해 S914, S916, 및 S918 단계를 순차적으로 수행한다.
- <69> 이하, S908 단계에서 윈도우 리스트에 업데이트 대상인 논리 페이지에 대한 윈도우가 존재하는 경우를 설명하겠다.
- <70> 논리 페이지에 대한 윈도우가 존재하는 경우 해당 윈도우의 업데이트 카운트가 윈도우 리스트의 평균 업데이트 카운트 이상인지 판단한다(S920).
- <71> 만약, 해당 윈도우의 업데이트 카운트가 평균 업데이트 카운트 이상이면, 해당 윈도우의 업데이트 카운트를 증가시키고, 윈도우 리스트의 링크 구조에서 해당 윈도우를 최신 위치로 이동시킨다(S922).
- <72> 이어, 업데이트 요청에 상응하여 업데이트 되는 논리 페이지를 위해 핫 블록의 물리 페이지를 할당한다(S924).
- <73> 만약, 해당 윈도우의 업데이트 카운트가 평균 업데이트 카운트 미만이면, 해당 윈도우의 업데이트 카운트를 증가시키고, 윈도우 리스트의 링크 구조에서 해당 윈도우를 최신 위치로 이동시킨다(S926).
- <74> 이어, 업데이트 요청에 상응하여 업데이트 되는 논리 페이지를 위해 일반 블록의 물리 페이지를 할당한다(S928).
- <75> 상술한 메모리의 페이지 할당 방법은 다양한 컴퓨터 수단을 이용하여 수행될 수 있는 프로그램 형태로도 구현될 수 있는데, 이때 메모리의 페이지 할당 방법을 수행하기 위한 프로그램은 하드 디스크, CD-ROM, DVD, 롬(ROM), 램, 또는 플래시 메모리와 같은 컴퓨터로 판독할 수 있는 기록 매체에 저장된다.
- <76> 본 발명이 속하는 기술분야의 당업자는 본 발명이 그 기술적 사상이나 필수적 특징을 변경하지 않고서 다른 구체적인 형태로 실시될 수 있다는 것을 이해할 수 있을 것이다.
- <77> 그러므로, 이상에서 기술한 실시예들은 모든 면에서 예시적인 것이며 한정적인 것이 아닌 것으로 이해해야만 한다. 본 발명의 범위는 상기 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 등가 개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

도면의 간단한 설명

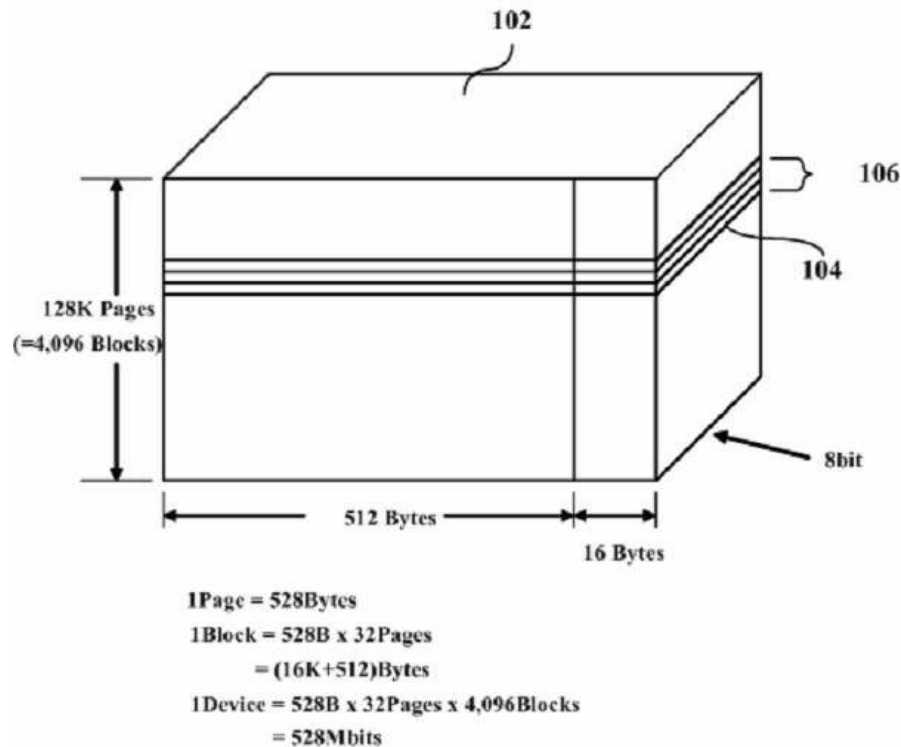
- <78> 도 1의 일반적인 낸드 플래시 메모리의 구조를 나타낸 도면.
- <79> 도 2는 파일 수정 후, 일반적인 페이지 할당 방법에 따른 메모리의 상태를 나타낸 도면.
- <80> 도 3은 일반적인 블록 클리닝 이후의 메모리의 상태를 나타낸 도면.
- <81> 도 4는 본 발명의 일 실시예에 따른 페이지 할당 방법으로써 파일이 수정된 후, 메모리의 상태를 나타낸 도면.
- <82> 도 5는 본 발명의 일 실시예에 따른 페이지 할당 방법으로써 파일이 수정된 후, 블록 클리닝이 끝난 메모리의 상

태를 나타낸 도면.

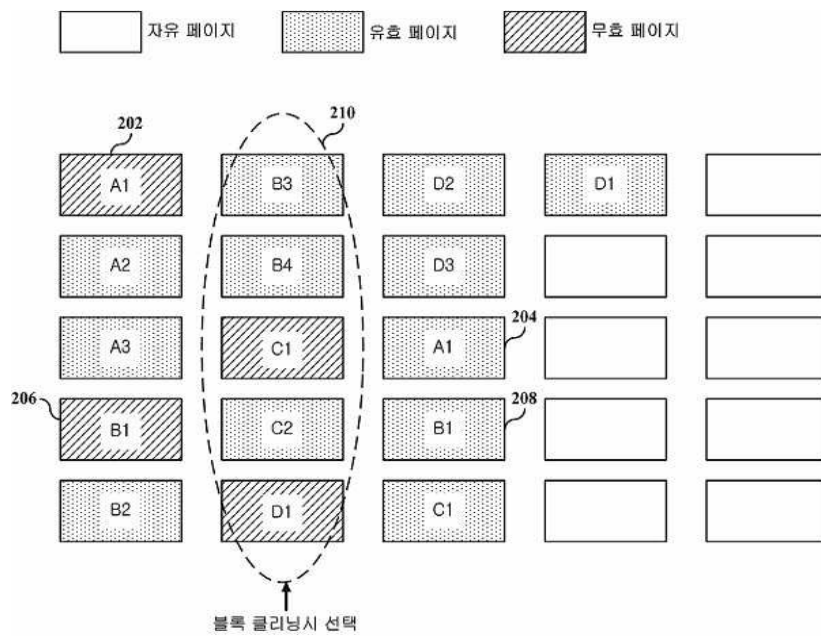
- <83> 도 6a는 본 발명의 일 실시예에 따른 논리 페이지의 업데이트 정보를 저장하기 위해 사용되는 윈도우 구조체를 나타내는 도면.
- <84> 도 6b는 본 발명의 일 실시예에 따른 윈도우 리스트를 나타내는 도면.
- <85> 도 7은 본 발명의 일 실시예에 따라 업데이트하려는 논리 페이지의 윈도우가 이미 윈도우 리스트에 존재하는 경우, 해당 윈도우의 이동을 나타내는 도면.
- <86> 도 8은 본 발명의 일 실시예에 따라 윈도우 리스트에서 존재 가능한 윈도우 개수가 최대치인 경우, 업데이트 요청에 의해 새로운 윈도우를 추가하는 방법을 나타내는 도면.
- <87> 도 9는 본 발명의 일 실시예에 다른 메모리의 페이지 할당 방법을 나타낸 순서도.
- <88> <도면의 주요부분에 대한 부호의 설명>
- <89> 402: 핫 페이지를 위한 블록 404: 유효 페이지 A1
- <90> 406: 유효 페이지 B1 308: 유효 페이지 C1
- <91> 410: 유효 페이지 D1
- <92> 412: 무효 페이지들이 존재하는 영역의 블록

도면

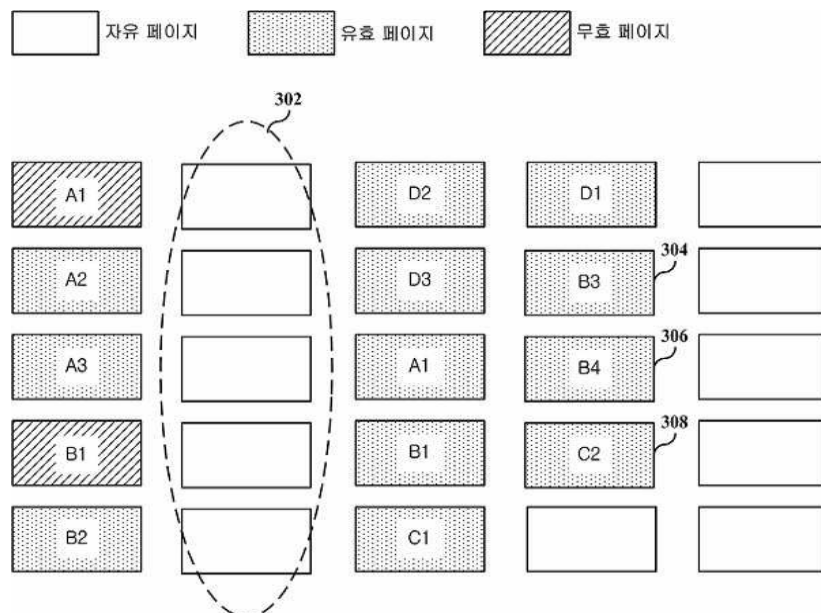
도면1



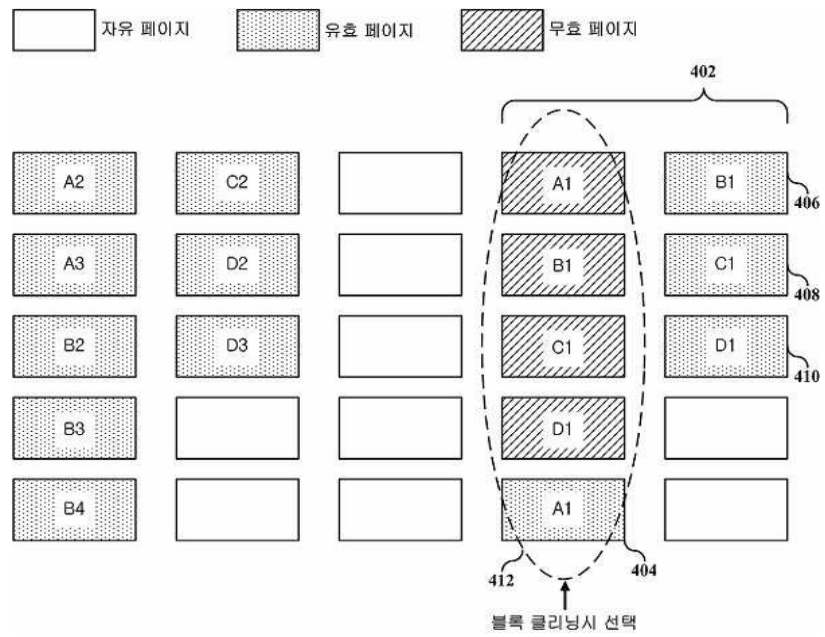
도면2



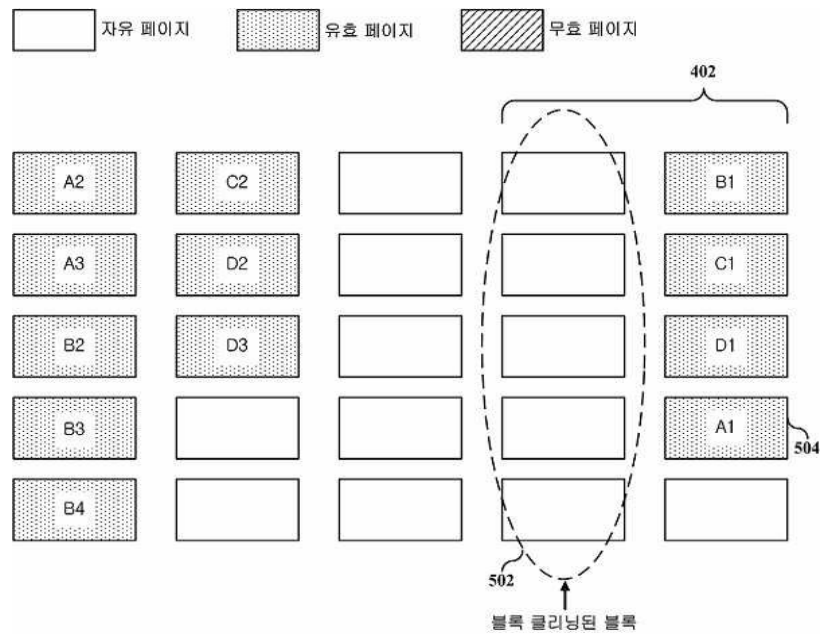
도면3



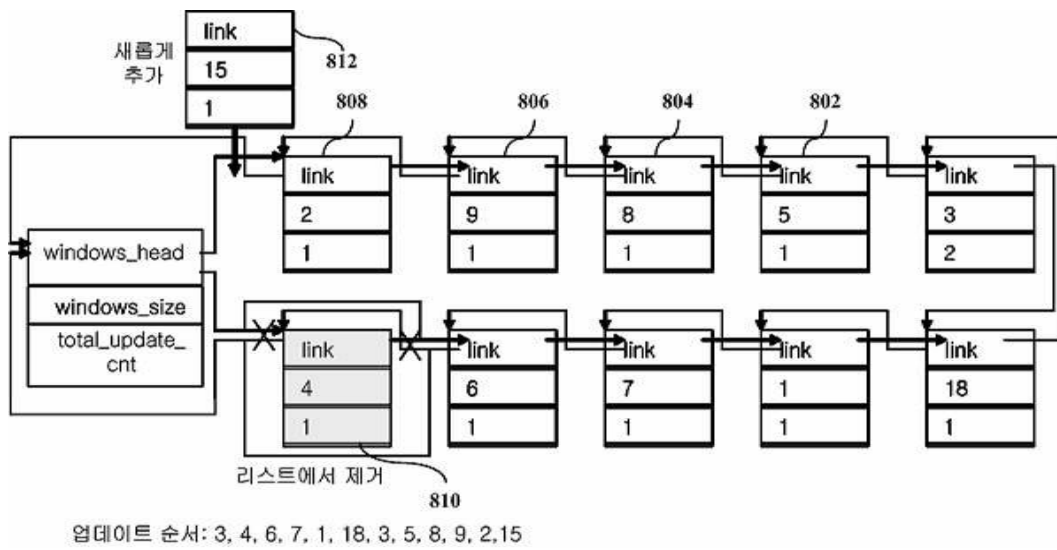
도면4



도면5



도면8



도면9

