



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2010년04월07일
(11) 등록번호 10-0951107
(24) 등록일자 2010년03월29일

- (51) Int. Cl.
G06F 9/06 (2006.01) G06F 15/76 (2006.01)
G06F 12/00 (2006.01)
- (21) 출원번호 10-2006-7025847
(22) 출원일자 2005년05월17일
심사청구일자 2007년10월22일
(85) 번역문제출일자 2006년12월08일
(65) 공개번호 10-2007-0024573
(43) 공개일자 2007년03월02일
(86) 국제출원번호 PCT/IL2005/000511
(87) 국제공개번호 WO 2005/111777
국제공개일자 2005년11월24일
- (30) 우선권주장
10/952,837 2004년09월30일 미국(US)
60/571,850 2004년05월17일 미국(US)
- (56) 선행기술조사문헌
US05404485 A1*
US20030200002 A1*
*는 심사관에 의하여 인용된 문헌

- (73) 특허권자
샌디스크 아이엘 엘티디
이스라엘 44425 크파 사바 아틸 예다 스트리트 7
- (72) 발명자
라세르 메나켄
이스라엘 44864 코하브 예어 돌레브 4
- (74) 대리인
김정욱, 박종혁, 송봉식, 정삼영

전체 청구항 수 : 총 31 항

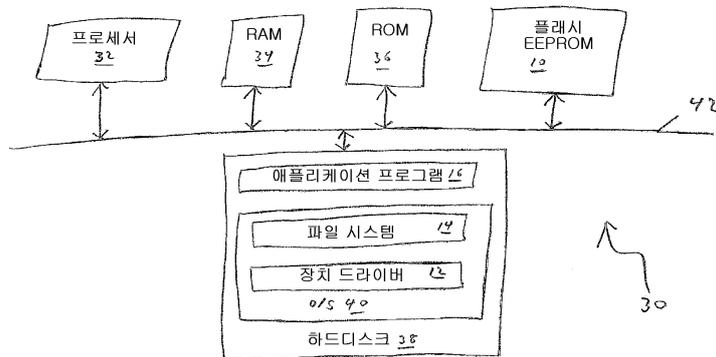
심사관 : 복진요

(54) 최적의 성능을 위한 파일 관리 방법

(57) 요약

메모리에 파일을 저장하는 방법. 개선된 기록 성능, 개선된 판독 성능 또는 개선된 레이턴시 성능등의 파일의 원하는 성능 특성에 따라 파일을 저장하기 위해 명시적 명령이 파일 시스템에 발생된다. 파일 시스템은 원하는 성능 특성에 따라 메모리에 파일을 저장한다. 바람직하게는 원하는 성능 특성은 메모리의, 물리적으로 인접한 일부나 메모리의 논리적으로 인접한 일부에 파일을 저장함으로써 달성된다. 복수의 파일의 각각에 대하여 명시적 명령이 발생된 후, 파일은 동시에 저장될 수 있다.

대표도



특허청구의 범위

청구항 1

메모리에 파일을 저장하는 방법에 있어서, 상기 방법은:

복수의 선택가능한 성능 특성으로부터 선택되는 제1 성능 특성에 따라 파일을 저장하기 위한 명령을 수신하는 단계;

상기 제1 성능 특성에 기초하여 상기 메모리의 제1 부분을 할당하는 단계; 및

상기 파일을 상기 메모리의 상기 제1 부분에 저장하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 2

제 1 항에 있어서, 상기 메모리의 상기 제1 부분은 상기 메모리의 제1의 물리적으로 인접한 부분을 포함하고, 상기 제1의 물리적으로 인접한 부분은 적어도 상기 파일에 대한 파일 크기 만큼 큰 것을 특징으로 하는 방법.

청구항 3

삭제

청구항 4

삭제

청구항 5

제 1 항에 있어서, 상기 메모리의 상기 제1 부분은 상기 메모리의 물리적으로 인접한 부분을 포함하고, 상기 물리적으로 인접한 부분은 적어도 상기 메모리의 소정 크기를 갖는 삭제 유닛만큼 크기가 큰 것을 특징으로 하는 방법.

청구항 6

삭제

청구항 7

제 1 항에 있어서, 상기 메모리의 제1 부분을 할당하는 단계가,

상기 메모리의 상기 제1 부분이 되기 위한 상기 메모리의 사용가능한 부분을 식별하는 단계로서, 상기 사용가능한 부분은 물리적으로 인접하고, 적어도 상기 파일에 대한 파일 크기 만큼 크고, 그리고 데이터를 저장하도록 이용가능한, 상기 단계; 및

상기 사용가능한 부분이 식별되지 않으면, 상기 메모리의 상기 제1 부분을 생성하는 단계를 포함하고,

상기 메모리의 상기 제1 부분을 생성하는 단계는,

데이터를 저장하도록 이용가능하고 적어도 상기 파일 크기 만큼 큰, 상기 메모리의 한 부분을 식별하는 단계; 및

메모리의 이 식별된 부분을 조각모음하여 상기 메모리의 상기 제1 부분을 만드는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 8

제 1 항에 있어서, 상기 메모리의 상기 제1 부분이 복수의 물리적으로 인접한 섹터를 포함하는 것을 특징으로 하는 방법.

청구항 9

제 1 항에 있어서, 상기 파일을 저장하는 단계 후에, 다른 데이터를 저장하기 위해 상기 메모리의 상기 제1 부

본의 사용되지 않은 나머지 부분을 릴리스하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 10

제 1 항에 있어서, 상기 파일에 관한 데이터를 수신하기 전에 상기 메모리의 상기 제1 부분을 할당하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 11

삭제

청구항 12

삭제

청구항 13

삭제

청구항 14

삭제

청구항 15

삭제

청구항 16

제 1 항에 있어서, 상기 메모리의 상기 제1 부분이 상기 메모리의 논리적으로 인접한 부분을 포함하는 것을 특징으로 하는 방법.

청구항 17

제 1 항에 있어서, 상기 제1 성능 특성은 개선된 기록 성능인 것을 특징으로 하는 방법.

청구항 18

제 1 항에 있어서, 상기 제1 성능 특성은 개선된 관독 성능인 것을 특징으로 하는 방법.

청구항 19

제 1 항에 있어서, 상기 제1 성능 특성은 저 레이턴시인 것을 특징으로 하는 방법.

청구항 20

제 1 항에 있어서, 상기 파일이 생성되기 전에 상기 명령이 발생하는 것을 특징으로 하는 방법.

청구항 21

제 1 항에 있어서, 상기 파일이 생성된 후에 그리고 상기 파일에 저장될 데이터를 제공하기 전에, 상기 명령이 발생하는 것을 특징으로 하는 방법.

청구항 22

제 1 항에 있어서, 상기 파일에 저장될 데이터를 수신한 후에 그리고 상기 파일을 닫기 전에, 상기 명령이 발생하는 것을 특징으로 하는 방법.

청구항 23

제 1 항에 있어서, 상기 메모리의 상기 제1 부분을 할당하는 단계가, 상기 메모리의 적어도 일부분을 조각모음하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 24

제 1 항에 있어서, 상기 파일을 닫은 후에 상기 명령이 발생하는 것을 특징으로 하는 방법.

청구항 25

삭제

청구항 26

데이터 프로세싱 디바이스에 있어서,

메모리; 및

파일 시스템으로서, 복수의 선택가능한 성능 특성으로부터 선택된 제1 성능 특성에 따라 파일을 저장하기 위한 명령을 수신하도록, 그리고 상기 제1 성능 특성에 기초하여 선택된 상기 메모리의 제1 부분에 상기 파일을 저장하도록 구성된 파일 시스템;을 포함하는 것을 특징으로 하는 데이터 프로세싱 디바이스.

청구항 27

컴퓨터 판독가능 저장매체에 있어서,

상기 저장매체는, 파일에 할당된 제1 성능 특성에 따라 메모리에 상기 파일을 저장하기 위한 제1 명령을 포함하는 컴퓨터 판독가능 코드를 포함하고,

상기 제1 명령은 복수의 명령으로부터 선택가능하고, 상기 복수의 명령의 각각은 개별 성능 특성과 관련되어 있는 것을 특징으로 하는 컴퓨터 판독가능 저장매체.

청구항 28

메모리에 복수의 파일을 저장하는 방법에 있어서,

복수의 명령을 수신하는 단계로서, 각 명령은 대응하는 파일에 할당된 개별 성능 특성에 따라 상기 메모리에 상기 대응하는 파일을 저장하기 위한 것인, 상기 단계; 및

복수개의 상기 파일을 상기 메모리에 동시에 저장하는 단계를 포함하고,

상기 복수개의 파일의 각각은 그 대응하는 명령에 따라 저장되는 것을 특징으로 하는 메모리에 복수의 파일을 저장하는 방법.

청구항 29

데이터 프로세싱 디바이스에 있어서,

메모리; 및

복수의 명령(각각의 명령은 대응하는 성능 특성에 따라 복수의 파일중 대응하는 파일을 저장하기 위한 것이고, 각각의 대응하는 성능 특성은 복수의 성능 특성으로부터 선택됨)을 수신하고, 그리고 상기 복수의 파일을 상기 메모리에 동시에 저장하는 파일 시스템;을 포함하고,

각각의 파일은 상기 대응하는 성능 특성에 따라 저장되는 것을 특징으로 하는 데이터 프로세싱 디바이스.

청구항 30

컴퓨터 판독가능 저장매체에 있어서, 상기 저장매체는,

프로세서에 의해 실행될 때 프로세서로 하여금:

복수의 명령(각각의 명령은 복수의 성능 특성으로부터 선택되는 대응하는 성능 특성에 따라 복수의 파일중 대응하는 파일을 저장하기 위한 것임)을 수신하게 하고;

상기 대응하는 성능 특성에 적어도 부분적으로 기초하여, 메모리의 대응하는 메모리 부분을 각 파일에 할당하게 하고; 및

메모리의 상기 대응하는 메모리 부분에 상기 복수의 파일의 각각을 동시에 저장하게 하는 프로그램 코드를 저장하는 것을 특징으로 하는 컴퓨터 판독가능 저장매체.

청구항 31

메모리에 복수의 파일을 저장하는 방법에 있어서,

복수의 명령(각각의 명령은 복수의 성능 특성으로부터 선택되는 대응하는 성능 특성에 따라 복수의 파일중 대응하는 파일을 저장하기 위한 것임)을 수신하는 단계;

상기 대응하는 성능 특성에 적어도 부분적으로 기초하여, 상기 메모리의 대응하는 부분을 각 파일에 할당하는 단계; 및

상기 복수의 파일(각 파일은 상기 메모리의 상기 대응하는 부분에 저장됨)을 메모리에 동시에 저장하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 32

제 26 항에 있어서, 상기 제1 성능 특성이 상기 메모리 내의 파일 액세스에 관련되는 것을 특징으로 하는 데이터 프로세싱 디바이스.

청구항 33

제 1 항에 있어서, 상기 제1 성능 특성이 상기 메모리 내의 파일 액세스에 관련되는 것을 특징으로 하는 방법.

청구항 34

제 29 항에 있어서, 상기 복수의 성능 특성의 각각이 상기 메모리 내의 파일 액세스에 관련되는 것을 특징으로 하는 데이터 프로세싱 디바이스.

청구항 35

제 27 항에 있어서, 상기 제1 명령이 상기 메모리에서의 파일 액세스에 관련되는 것을 특징으로 하는 컴퓨터 판독가능 저장매체.

청구항 36

제 28 항에 있어서, 각 성능 특성이 상기 메모리 내의 파일 액세스에 관련되는 것을 특징으로 하는 방법.

청구항 37

제 36 항에 있어서, 상기 성능 특성의 적어도 하나가 저 레이턴시, 개선된 판독 성능, 또는 개선된 기록 성능을 포함하는 것을 특징으로 하는 방법.

청구항 38

제 29 항에 있어서, 상기 복수의 성능 특성이, 감소된 액세스 레이턴시, 개선된 판독 성능, 및 개선된 기록 성능 중 적어도 하나를 포함하는 것을 특징으로 하는 데이터 프로세싱 디바이스.

청구항 39

제 1 항에 있어서, 상기 제1 성능 특성이 감소된 액세스 레이턴시, 개선된 판독 성능, 또는 개선된 기록 성능을 포함하는 것을 특징으로 하는 방법.

청구항 40

제 26 항에 있어서, 상기 제1 성능 특성이 감소된 액세스 레이턴시, 개선된 판독 성능, 또는 개선된 기록 성능을 포함하는 것을 특징으로 하는 데이터 프로세싱 디바이스.

명세서

기술분야

[0001] 본 발명은 파일 시스템에 관한 것이고 특히 파일의 원하는 성능 특성에 따라 파일을 저장하기 위한 명시적 명령 (explicit command)을 지원하는 파일 시스템에 관한 것이다.

배경기술

[0002] 대형 메인프레임이건 소형의 내장 마이크로컨트롤러이건 거의 모든 컴퓨터 시스템은 시스템의 전원이 나갔을 때 데이터가 상실되지 않도록 데이터를 저장하여야 한다. 따라서 보통 이러한 컴퓨터는 프로그램을 실행하기 위해 사용하는 임의의 휘발성 메모리는 물론 일종의 비휘발성 메모리(NVM: Non Volatile Memory)를 포함한다. NVM은 자기 디스크, 플래시 메모리 칩, 또는 기타 다른 비휘발성 저장 엘리먼트일 수 있다.

[0003] 도 1은 이러한 저장장치가 어떻게 액세스되는지에 대한 일반적인 구조를 도시하고 있다. 도 1의 하단에 물리적 저장 매체(10)가 있는데 이것은 물리적 저장을 구현하는 하드웨어 계층이다. 각각의 저장장치는 그 자체의 독특한 인터페이스와 작업을 불편하게 만드는 특성을 가질 수 있기 때문에 운영 시스템에 포함된 소프트웨어 장치 드라이버(12)가 컴퓨터에서 작동하게끔(운영 시스템이 사용되지 않는 경우 베어 하드웨어에서 작동하게끔)하는 것이 일반적인 경우인데, 이러한 장치 드라이버(12)는 장치에 액세스하기 원하는 기타 소프트웨어 컴포넌트를 위한 간단하고 표준화된 인터페이스를 제공한다. 파일을 저장하는 데 사용되는 저장장치(10)(즉, 디스크, 디스켓 등)-그러나 저장장치에만 국한되지 않는다-를 위해, 장치 드라이버(12)에 의해 제공된 인터페이스는 보통 "블록 장치 드라이버"로 알려진 타입이다. 이러한 장치 드라이버(12)는 단일 바이트보다는 데이터 블록을 사용하여 클라이언트와 상호작용한다. 이것은 입력 및 출력 동작, 즉, 판독과 기록 모두에 적용된다. 블록 장치(10)의 가장 일반적인 예는 하드웨어 인터페이스가 보통 512바이트 이상 등과 같이 (여기서는 보통 "섹터"라고 부르는) 완전한 블록만을 전달하도록 구성되는 자기 디스크이다. 물리적 저장장치(10)가 블록 장치 인터페이스를 제공하는 장치 드라이버(12)를 가지기 위해 물리적으로 블록 동작으로 제한될 필요는 없음을 주목해야한다. 예컨대, 배터리 백업식 RAM디스크는 물리적으로 블록으로 제한되지 않고 메모리 바이트의 각각을 물리적으로 판독하고 기록할 수 있다. 또한 장치 드라이버(12)는 자기 디스크와 호환가능하고 상호변경가능하도록 시스템의 나머지에 블록 장치 인터페이스를 제공한다. 따라서, 본 발명을 위한 블록 장치는 실제의 물리적 구조와 무관하게 그 드라이버(12)가 블록 장치 인터페이스를 제공하는 임의의 장치이다.

[0004] 블록 장치는 사용자에게 어떤 고정된 크기를 갖는 블록의 선형 어레이로 보인다. 이러한 블록의 각각은 도 2에 도시된 바와 같이 어레이내의 인덱스를 사용하여 기타 블록과 독립적으로 판독되거나 기록될 수 있다. (또한 여기서 사용되는) 일반적인 경우는 블록 번호 0(21)에서 개시하여 블록 번호 N-1(22)에서 끝나도록 블록을 넘버링하는 것이다(여기서, N은 장치 드라이버에 의해 익스포트된 블록 번호). 이러한 선형 어레이 구조는 물리적 장치 레벨에서 반드시 존재하는 것은 아니라는 점을 주목해야한다. 예컨대, 플래시 디스크 블록 장치 드라이버(12)는 또한 이러한 선형 어레이 이미지를 제공하지만, 플래시 메모리내의 기록 제한과 배드 블록의 존재 가능성 때문에 내부적으로 플래시 매체(10)상의 물리적 블록은 보통 랜덤 순으로 분산된다(따라서 블록 번호 0은 물리적으로 플래시 매체(10)의 중간이나 끝에 위치할 수 있다). 또한 블록 장치 드라이버(12)는 상위 소프트웨어 계층(14 및 16)에 의해 블록에 들어간 콘텐츠를 알지 못한다는 점을 이해해야한다.

[0005] 도 1을 참조하면, 장치 드라이버(12)위에 파일 시스템(FS) 소프트웨어 계층(14)이 위치함을 알 수 있다. FS(14)는 일반적인 프로그래머나 사용자에게 훨씬 자연스럽게 편리한 파일 컨셉만을 사용하여 애플리케이션 프로그램(16)이 저장장치(10)와 상호작용할 수 있게 함으로써 물리적 장치(10)로부터 더 격리시키는 소프트웨어 컴포넌트이다. FS(14)는 블록 장치(10)상의 사용자 데이터를 몇몇 로직 구조로 조직하고 파일 데이터를 포함하는 블록을 파일 속성(즉, 파일명, 생성시간, 액세스 허용 등)과 연관시킴으로써 이러한 추상화(abstraction)를 수행한다. 이를 위해, FS(14)는 장치(10)내에 사용자에게 직접 보이지 않고 FS(14)가 사용자 파일을 추적하고 액세스할 수 있는 FS(14)의 내부 북 키퍼 정보를 포함하는 메타 데이터를 저장한다. 예컨대, 가장 간단한 상용 FS중 하나인 마이크로소프트사의 DOS FAT(12)파일 시스템은 저장장치(10)에 (장치(10)의 첫번째 블록에 있어야 하는) 기타 메타 데이터 구조의 로케이션을 가능하게 하는 몇몇 기초 파라미터, 장치(10)의 할당 맵인 파일 할당 테이블(FAT)의 하나 이상의 카피, 및 파일명으로 파일을 위치시키기 위한 루트 디렉토리 구조를 포함하는 부트 섹터를 저장한다. 애플리케이션 프로그램(16)은 "파일 열기", "파일 삭제", "파일 쓰기"등과 같은 명령을 생성함으로써 파일 레벨상에서 FS(14)와 상호작용한다. 따라서 애플리케이션 프로그램(16)은 그 아래의 블록 구조를 완전히 무시한다. 오늘날 그 내부 구조와 특성이 크게 다른 많은 파일 시스템(14)이 사용되고 있다. (리눅스 운영 시스템에 의하는 것과 같은) 많은 경우에 운영 시스템은 사용자에게 몇몇 파일 시스템(14)을 제공

하여 필요에 가장 적합한 하나를 선택한다.

- [0006] 대부분의 종래의 저장 시스템은 도 1에 도시된 장치 드라이버(12)와 파일 시스템(14)간 계층 모듈 분리를 따르지만, 이러한 명확한 분리가 존재하지 않고 장치 드라이버와 파일 시스템 기능이 하나의 소프트웨어 모듈로 결합되어 있는 저장 시스템이 있다. 이러한 시스템을 일반적으로 "모놀리식" 저장 시스템이라고 한다. 본 명세서에서 레이어드 모델을 사용하여 모든 설명을 하고 있지만, 본 발명은 모놀리식 모델에도 같이 적용될 수 있고 이 경우 "파일 시스템"으로 언급하는 것은 모놀리식 소프트웨어의 파일 처리부를 언급하는 것으로 해석되어야 하고, "장치 드라이버"로 언급되는 것은 모놀리식 소프트웨어의 장치 처리부를 언급하는 것으로 해석되어야함을 이해하여야한다.
- [0007] 파일 시스템에 의한 파일의 처리에 관련된 고려점에 대한 보다 자세한 설명은 미국 특허 출원 제 10/849,234호-2004년 5월 20일 출원, 발명의 명칭 "A File System that Manages Files According to Content", 발명자 Mosek-에 나타나 있다.
- [0008] 종래의 파일 시스템(14)과 저장장치 드라이버(12)는 모든 파일에 대하여 같이 저장 공간 할당을 처리한다. 파일이 물리적 공간에 할당되는 방식이 성능에 영향을 미칠 수 있더라도 사용자는 파일에 액세스할 때 알 수 있다는 것을 이해해야한다.
- [0009] 일 예로 많은 섹터에 걸쳐있는 (섹터는 매체(10)로부터 판독되거나 매체(10)에 기록된 데이터의 최소 청크로서 1섹터는 대개 512바이트이다) 파일을 판독하는 경우를 고려해보자. 파일 시스템(14)에 의해 사용되는 저장 매체(10)는 고체 상태 플래시 디스크라고 가정하고 파일 시스템(14)은 파일의 첫번째 16KB를 판독하도록 요청된다고 하자. 이는 첫번째 32데이터 섹터를 판독하도록 번역된다. 사용자가 논리적으로 이러한 16KB를 볼 때 연속 데이터 스트림으로 매체(10)상에 정주하는 것처럼 보이더라도, 이것이 데이터가 매체(10)상에 물리적으로 저장되는 필연적 방식인 것은 아니다. 파일 시스템(14)은 일반적으로 새로운 데이터를 기록할 필요가 있을때 매체 공간을 할당하는데 이러한 할당 폴리는 비순차적일 수 있다. 이러한 것의 간단한 일 예가 DOS와 여러 윈도우 운영 시스템에 사용된 것인, 널리 사용되고 있는 마이크로소프트사의 FAT파일 시스템이다. FAT은 "클러스터"내에 공간을 할당하는데, 각각의 클러스터는 몇개의 섹터를 포함한다(대개 클러스터당 4개의 섹터를 포함한다). 같은 클러스터내의 섹터가 매체(10)상의 물리적으로 순차적 위치에 할당되더라도, 논리적으로 인접한 클러스터(즉, 인접한 논리적 주소를 갖는 클러스터)가 물리적으로 매체(10)상에 서로 가까울 필요는 없다. 파일 시스템(14)은 또다른 파일에 의해 사용된 공간 가운데에 비어있는 클러스터를 할당할 것을 판단할 수 있고, 따라서 연속 범위의 논리적 순차 데이터를 매체(10)의 물리적 비순차 청크로 매핑할 수 있다. (마이크로소프트는 디스크 드라이브를 "조각모음(defragmenting)"하기 위한 유틸리티를 제공한다. 이러한 유틸리티는 매체상의 파일을 재정리하여 파일이 "연속적"이게 한다. 그러나, 조각모음 유틸리티는 매우 시간 소모적이고 시스템 사용자에게 의해 대개 몇주에 한번만 동작되는 반면, 주기 중간에는 디스크가 점점 "비연속"상태로 되돌아가게 된다. 또한, 논리적으로 연속적인 저장과 물리적으로 연속적이거나 인접하는 저장을 구별하여야한다. 플래시 디스크에서는 이것은 같지 않다. 배드 블록의 플래시 제한과 큰 청크의 공간을 삭제하지 않고 재기록하는 것이 불가능하기 때문에, 플래시 메모리 관리 시스템은 논리적 주소를 물리적 주소로 매핑하는 것을 채용하고, 따라서 장치 드라이버가 연속 할당을 보고하더라도, 물리적으로는 그러하지 않는다). FAT등의 파일 시스템이 저장 할당을 처리하는 방식에 대한 보다 상세한 설명은, 미국 특허 출원 제 10/397,378호, 2003년 3월 27일 출원, 발명의 명칭 "Robust Self-Maintaining File System"에 나타나 있다.
- [0010] (일반적으로 고체 상태 디스크나 메모리 카드에 사용되는) 많은 낸드 타입 플래시 메모리 장치는 "연속 판독"의 구조를 갖는다. 이것은 플래시 페이지의 마지막 바이트(장치 하드웨어 레벨에서 데이터 섹터의 등가물)가 장치로부터 판독될 때, 이 장치는 물리적으로 다음 페이지의 메모리 셀로부터 내부 데이터 버퍼를 로딩하기 시작함을 의미한다. NAND플래시에서는, 저장된 데이터로 데이터 버퍼를 로딩하는데 필요한 시간은 비교적 길고 메모리 시스템의 판독 성능에 주요한 영향을 미친다. 버퍼를 로딩하는데 걸리는 전형적인 시간값은 15마이크로초이다. 100나노초 버스 사이클을 사용하는 16비트 넓이의 장치에 대하여, 버스상에서 512바이트의 페이지를 전달하는데 필요한 시간은 $256 \times 0.1 = 25.6$ 마이크로초이고, 이는 버퍼 로딩 시간과 같은 정도의 크기이다. 따라서 버퍼 로딩을 미리 함으로써 섹터의 판독을 상당히 가속시킬 수 있고, 결과적으로 파일의 판독을 가속시킬 수 있다. 그러나 "연속 판독" 구조를 사용한 이러한 시간 절약은 판독될 다음 페이지가 이전에 판독된 페이지에 물리적으로 순차적일 때만 가능하다. 판독 성능은 저장 할당 폴리에 영향을 받고 "영리한" 할당 결정에 의해 유리하게 될 수 있다.
- [0011] 상기에 기초하여, 파일을 매체(10)상에 물리적으로 연속적이게 하는 것이 유리함을 알 수 있지만, 왜 모든 파일

을 그렇게 할당하지 않는 이유는 적당히 복잡한 파일 시스템(14)에서도 매체(10)와 병렬로 상주하는 수백개의 파일을 지원해야하기 때문이다. 이러한 파일은 생성, 기록, 첨가, 삭제 또는 많은 복잡하고도 예상불가의 이벤트 시퀀스로 변할 수 있다. 모든 파일을 물리적으로 순차적(sequential)으로 유지하는 것은 가장 간단한 시스템의 경우를 제외하면 불가능하다. 예컨대, 사용자가 기존 텍스트 파일에 몇 개의 라인을 부가함으로써 파일 시스템(14)이 파일 길이를 확장하게 한다. 그러나 파일을 바로 따르는 공간이 현재 다른 파일에 의해 사용되고 있다면, 이용가능한 유일한 옵션은 업데이트된 파일을 비순차적 부분으로 깨거나, 전체 다른 파일을 또다른 위치로 이동시켜 구 공간을 업데이트된 파일을 위해 사용하거나, 전체 업데이트된 파일을 충분한 공간이 있는 다른 위치로 이동시키는 것이다. 대부분의 경우에 바람직한 선택은 연속성을 깨는 것인데, 나머지 옵션들은 훨씬 더 시간 소모적이기 때문이다.

[0012] 또 다른 예로서, 플래시 디스크에 파일을 기록하는 경우를 고려하자. 플래시 메모리 장치는 기록용으로 목표된 블록이 이전에 삭제되지 않았다면 페이지가 기록될 수 없는 특성을 갖는다. 이것은 디스크가 연속적으로 사용되는 한편 새로운 데이터를 위한 공간을 만들기 위해 디스크의 블록을 리사이클(삭제)할 필요가 있다. 이러한 리사이클링은 사용되지 않는 공간이 즉시 필요할 때만 수행될 수 있거나 주기적으로 또는 시스템이 대기 상태(idle time)일때 수행될 수도 있다. 플래시 메모리 할당 및 리사이클링의 문제에 대한 보다 나은 이해를 위해, 미국 특허 제 5,404,485호, 발명의 명칭 "Flash File System", 특허일 1995. 4. 4을 참조하자.

[0013] 저장 시스템이 파일을 기록하는데 걸리는 시간은 할당에 바로 이용가능한 사용되지 않는 공간이 있는지 또는 몇몇 공간이 먼저 이용가능한 공간을 생성하기 위해 리사이클되어야하는지에 좌우된다. 플래시 블록을 삭제하는 것은 다소 느린 동작이고(NAND플래시에서 수 밀리초, NOR플래시에서 수백 밀리초), 따라서 파일의 기록 성능에 주된 영향을 미칠 수 있다. 파일을 기록하는데 필요한 스토리지가 미리 예약되어 리사이클링을 요하지 않는다는 것을 보장할 수 있다면, 기록 성능은 보다 높은 것으로 보장될 것이다.

[0014] 실제, 기록하기 전에 플래시 공간을 예약하는 아이디어는 새로운 것이 아니다-인텔 사는 사용자가 스토리지의 "패킷"을 "초기화"하고 패킷을 데이터로 채우게할 수 있는 플래시 데이터 인테그레이터(또는 줄여서 "FDI")라 부르는 플래시 관리 소프트웨어 패키지를 제공한다. 초기화는 사용되지 않는 공간의 예비할당을 포함하고, 이것은 필요하다면 리사이클링을 일으킨다. 그러나, 이러한 아이디어의 FDI는 다소 원시적이고도 제한적이다-이것은 파일에는 전혀 적용되지 않고 숫자 식별자에 의해 식별된 원시 구조인 몇몇 타입의 오브젝트에 적용된다. 또한, 하나의 패킷 스트림만이 단 한번의 입력을 위해 개방될 수 있다.

[0015] 공간 할당에 관하여 모든 파일을 동일하게 다루는 이러한 종래의 저장 시스템은 사용자가 특정 파일에 대한 최적의 성능을 획득하게 할 수 없다.

[0016] 따라서 응용프로그램(16)이 최적의 성능을 위해, 매체(10)상의 파일의 특별한 애드 호크 스토리지를 요청할 수 있게 하는 파일 시스템(14)의 필요성에 대한 인식이 널리 재고되었고 이러한 파일 시스템(14)을 갖는 것이 매우 유리할 수 있다.

[0017] 여기서 "성능"은 상기한 예에서 나타난 기록 성능이나 판독 성능만을 의미하는 것은 아님을 분명히 해야한다. 성능은 시간에 관련된 저장 시스템 동작의 실행과 연계된 임의의 다른 특성을 포함하도록 해석되어야한다. 예컨대, 저장 시스템의 레이턴시 또한 성능 특성이다. 레이턴시는 저장 시스템이 단일 콜을 완성하는데 걸리는 시간량을 의미한다. 일부의 소위 "하드" 실시간 시스템에 있어서는, 단일 콜은 소정 반복주기를 초과하지 않아야하고 따라서 몇몇 타입 크리티컬한 동작을 수행하는데 지연이 전혀 없어야한다는 것이 필수조건이다. 따라서 이러한 시스템에서는 저 레이턴시가 필요한 성능 특성이다.

[0018] 또한 본 명세서에 제공된 모든 예에서 매체(10)가 고체 상태 플래시 디스크로 되어 있지만, 본 발명은 자기 디스크를 포함한 다른 타입의 저장 매체에도 똑같이 적용될 있음을 분명히 해야한다. 파일에 대한 최적의 할당을 구성하는 것에 대한 고려는 대체로 서로 다른 매체 타입에 대해 다르지만, 같은 아이디어가 여전히 적용된다. 예컨대, 자기 디스크에서는 기록 및 판독에 적용되는 서로 다른 고려사항으로 저장 영역에 액세스하는데 사용된 판독/기록 헤드의 기계적 이동을 최적화하는 문제가 있다. 따라서 이하 기술되고 청구되는 본 발명은 이러한 경우에도 적용될 수 있다.

발명의 상세한 설명

[0019] 본 발명에 따라 (a) 파일의 원하는 성능 특성에 따라 메모리에 파일을 저장하기 위한 파일 시스템에 명시적 명령을 발생시키는 단계; 및 (b) 상기 파일 시스템에 의하여, 상기 원하는 성능 특성에 따라 메모리에 파일을 저

장하는 단계;를 포함하는, 메모리에 파일을 저장하는 방법이 제공된다.

- [0020] 본 발명에 따라 (a) 메모리; 및 (b) 상기 메모리에 파일을 저장하고, 상기 파일의 원하는 성능 특성에 따라 상기 메모리에 상기 파일을 저장하기 위한 명시적 명령을 지원하는 파일 시스템을 포함하는 데이터 프로세싱 장치가 제공된다.
- [0021] 본 발명에 따라 파일 시스템을 위한 컴퓨터 판독가능 코드를 구현한 컴퓨터 판독가능 저장 매체가 제공되고, 상기 컴퓨터 판독가능 코드는 파일의 원하는 성능 특성에 따라, 메모리에 파일을 저장하기 위한 명시적 명령을 지원하는 프로그램 코드를 포함한다.
- [0022] 본 발명에 따라 메모리에 복수의 파일을 저장하는 방법으로서, (a) 각각의 파일에 대해, 각각의 원하는 성능 특성에 따라 메모리에 상기 각 파일을 저장하기 위한 명시적 명령을 파일 시스템에 발생시키는 단계; 및 (b) 상기 파일 시스템에 의하여 실질적으로 동시에 상기 원하는 성능 특성에 따라 메모리에 상기 파일을 저장하는 단계를 포함하는, 메모리에 복수의 파일을 저장하는 방법이 제공된다.
- [0023] 본 발명에 따라 (a) 메모리; 및 (b) 실질적으로 동시에 파일 시스템에 복수의 파일을 저장하고, 상기 각각의 파일의 각각의 원하는 성능 특성에 따라 상기 메모리에 상기 파일의 각각을 저장하기 위한 명시적 명령을 지원하는 파일 시스템을 포함하는 데이터 프로세싱 장치가 제공된다.
- [0024] 본 발명에 따라 파일 시스템을 위한 컴퓨터 판독가능 코드를 구현한 컴퓨터 판독가능 저장 매체가 제공되고, 상기 컴퓨터 판독가능 코드는 각각의 원하는 성능 특성에 따라, 실질적으로 동시에 메모리에 복수의 파일을 저장하기 위한 명시적 명령을 지원하는 프로그램 코드를 포함하고, 상기 명시적 명령은 상기 각각의 파일에 대하여 별개로 발생 된다.
- [0025] 본 발명에 따라 메모리에 복수의 파일을 저장하는 방법이 제공되고, 본 방법은 (a) 메모리에 파일을 저장하기 위한 디폴트 프로시저를 확립하는 단계; (b) 각각의 원하는 성능 특성에 따라 메모리에 파일을 저장하기 위한, 상기 디폴트 프로시저와 다른 제 2 프로시저를 확립하는 단계; (c) 상기 복수의 파일의 각각의 파일에 대하여, 상기 각각의 원하는 성능 특성에 따라 메모리에 상기 각 파일을 저장하기 위한 명시적 명령을 파일 시스템에 발생시키는 단계; 및 (d) 상기 파일 시스템에 의해, 실질적으로 동시에 상기 각각의 원하는 성능 특성에 따라 메모리에 복수의 파일을 저장하는 단계;를 포함한다.
- [0026] 본 발명의 기본적인 방법은 파일의 원하는 성능 특성을 개선하는 식으로 메모리에 파일을 저장하기 위한 파일 시스템에 명령을 발생시킴으로써 메모리에 파일을 저장한다. 이후 파일 시스템은 파일 시스템이 수신한 명령에 따라 메모리에 파일을 저장한다. 보다 정확하게는, 파일 시스템의 코드와 메모리의 장치 드라이버의 코드를 실행하는 프로세서가 파일 시스템이 수신한 명령에 따라 메모리에 파일을 저장한다.
- [0027] 바람직하게는, 메모리에 파일을 저장하기 위해 두개의 프로시저가 확립되는데, 그 중 하나는 소정의 파일에 대하여 명시적 명령이 발생되지 않은 경우에 사용되는 디폴트 프로시저(보통 종래의 프로시저)이고 나머지 하나는 소정의 파일에 대하여 명시적 명령이 발생하는 경우 사용되는 다른 제 2 프로시저이다.
- [0028] 보다 바람직하게는, 상기 제 2 프로시저는 메모리의 물리적으로 인접하는 부분 또는 메모리의 논리적으로 인접하는 부분에 파일의 적어도 일부를 저장하는 것을 포함한다. "물리적으로 인접하는"과 "물리적으로 연속하는"은 본 명세서에서 서로 바꾸어 사용될 수 있음을 유의한다.
- [0029] 가장 바람직하게는, 메모리의 물리적으로 인접하는 부분은 디폴트 프로시저에 의해 사용되는 메모리의 가장 작은 물리적으로 인접하는 부분보다 크다.
- [0030] 또한, 가장 바람직하게는, 단일 삭제 동작에서 삭제되는 메모리 공간의 가장 작은 청크가 단일 판독 또는 기록 동작으로 판독 내지 기록되는 메모리의 청크보다 큰 플래시 메모리 등의 메모리의 경우, 메모리의 물리적으로 인접한 부분은 단일 삭제 동작에서 삭제되는 메모리의 적어도 그러한 가장 작은 청크이다. 이와 같이 단일 삭제 동작에서 삭제되는 메모리의 가장 작은 청크를 본 명세서에서는 "삭제 유닛"이라 한다.
- [0031] 또한 가장 바람직하게는, 전체 파일이 메모리의 물리적으로 인접한 부분에 저장된다.
- [0032] 가장 바람직하게는, 파일을 저장하기 위하여, 파일 시스템은 메모리의 물리적으로 인접한 부분에 저장될 파일의 적어도 일부를 수용하기에 충분히 큰 메모리의 물리적으로 인접하면서 비어있는 부분을 탐색한다. 파일 시스템이 이와 같은 메모리의 충분히 크고, 물리적으로 인접하며 비어있는 부분을 찾는 데 실패하면, 파일 시스템은 필요한 물리적으로 인접하고 비어있는 부분을 생성한다. 예컨대, 파일 시스템은 플래시 메모리의 장치 드라이

버를 거쳐 플래시 메모리로 메모리에 현재 저장되어 있는 파일의 스토리지를 재정렬하여 필요한 물리적으로 인접한 부분을 삭제(free up)하도록 한다. 또한 가장 바람직하게는, 메모리의 물리적으로 인접한 부분은 메모리의 복수의 물리적으로 인접한 섹터이다. 또한 가장 바람직하게는, 파일이 메모리에 저장된 후에, 메모리의 물리적으로 인접한 부분의 사용되지 않은 나머지 부분을 다른 파일의 저장을 위해 릴리스한다.

[0033] 보다 바람직하게는, 상기 제 2 프로시저는 파일 시스템이 저장할 파일 데이터의 임의의 데이터를 실제로 수신하기 전에 파일의 적어도 일부를 위해 메모리에 공간을 할당하는 것을 포함한다. 디폴트 공간량이 할당되거나 명시적 명령의 파라미터에 따라 결정된 공간량이 할당된다. 가장 바람직하게는, 할당된 공간은 물리적으로 인접하는, 예컨대, 메모리의 복수의 물리적으로 인접하는 섹터이다. 또한 가장 바람직하게는, 파일이 메모리에 저장된 후에, 물리적으로 인접하는 할당된 공간의 사용되지 않은 나머지 부분이 다른 파일의 저장을 위해 릴리스된다.

[0034] 바람직하게는, 원하는 성능 특성은 파일 시스템의 디폴트 방법으로 매체에 저장된 같은 크기의 파일과 관련된, 개선된 기록 성능, 개선된 판독 성능, 또는 개선된 레이턴시 성능이다.

[0035] 바람직하게는, 명령은 파일이 생성되기 전에 발생된다. 대안으로, 명령은 파일이 생성된 이후이지만 파일에 저장할 데이터가 제공되기 전, 파일에 저장할 데이터가 제공된 후이지만 파일이 닫히기 전, 또는 파일이 닫힌 후에 발생 된다. 명령이 파일에 저장할 데이터가 제공된 이후, 파일이 닫히기 전 또는 이후에 발생 되면, 가장 바람직하게는 파일을 저장하는 것은 파일이 위치되는 메모리의 적어도 일부분을 조각 모음하는 것을 포함한다.

[0036] 본 발명의 데이터 프로세싱 장치는 메모리에 파일을 저장하기 위한 본 발명의 파일 시스템과 메모리를 포함한다. 파일 시스템은 파일의 원하는 성능 특성에 따라 메모리에 파일을 저장하기 위한 명시적 명령을 지원한다.

[0037] 본 발명의 컴퓨터 판독가능 저장 매체는 본 발명의 파일 시스템의 컴퓨터 판독가능 코드가 구현된다.

[0038] 또한 본 발명에 따라, 복수의 파일의 각각에 대하여 명시적 명령이 발생된 이후, 파일은 실질적으로 동시에 저장된다. 본 발명의 파일 시스템은 이러한 동시 스토리지를 지원한다. 이러한 동시 스토리지는 한번에 단 하나의 패킷 스트림만을 처리하는, 종래의 FDI와 본 발명을 구분 짓는 본 발명의 특징 중 하나이다. FDI와 본 발명을 구분 짓는 본 발명의 또 다른 특징은 본 발명의 명시적 명령은 원하는 성능 특성을 어떻게 얻는지를 명시적으로 정의함이 없이 원하는 성능 특성을 정의한다는 점이다. FDI에서는, 사용자가 기록에 앞서 Y바이트의 X청크의 각각이 기록될 것을 정의한다. 이후 사용자는 매번 기록되는 Y바이트 정도로, X회 기록할 수 있다. 또한, FDI하에서 기록된 데이터는 FDI기록 명령에 있어서, 명명된 파일과 연관되지 않는다.

[0039] 본 발명과 상기한 미국 특허 출원 제 10/849,234호가 교시하는 것과 차이가 있음을 주목하여야 한다. 양 발명은 서로 다른 파일에 대한 서로 다른 폴리를 사용하여 파일 시스템내에서 처리하는 파일을 최적화하는 법을 다루고 있지만, 파일 시스템이 파일의 원하는 성능 특성에 대하여 러닝(learn)하는 방법은 동일하지 않다. 상기한 60/549,891 출원에서는 파일 시스템이 (예컨대 특정 파일의 확장자 또는 특정 디렉토리 명을 감시하여) 파일을 처리하기 위한 적당한 폴리를 선택하기 위해, 예컨대 파일 타입 등, 파일의 같은 특성의 파일 인스턴스를 알아야 한다. 이후 파일 시스템은 폴리를 결정하기 위한 특성에 대응하는 관리 프로토콜에 문의한다. 따라서 파일에 어떤 폴리가 적용될지에 대한 호출 애플리케이션의 지시는 어느 관리 프로토콜에 문의하는지를 파일 시스템에 지시하는 파일 특성을 통해 암시하고 있다. 반대로, 본 발명에서는 호출 애플리케이션이 명시적인 호출 폴리를 파일 시스템에 발생시킴으로써 원하는 임의의 파일을 위한 원하는 성능 특성을 지시하기 때문에, 특정 처리 폴리를 파일에 적용하기 위하여 파일의 임의의 특성을 파일 시스템이 알 필요가 없다. 따라서, 양 발명의 방법들이 둘 다 사용되는 구현예가 있을 수 있지만 어느 하나의 발명의 방법만이 사용되는 구현예도 있을 수 있다.

실시예

[0043] 이하 본 발명의 실시예를 첨부 도면을 참조하여 보다 자세히 설명하기로 한다.

[0044] 본 발명은 파일의 성능 특성의 애드 호크 최적화를 지원하는 파일 시스템에 관한 것이다. 상세하게는, 본 발명은 선택된 파일의 판독, 기록 또는 레이턴시 특성을 최적화하는데 사용될 수 있다.

[0045] 본 발명에 따른 파일 시스템의 원리와 동작은 발명의 상세한 설명과 첨부도면을 참조하여 보다 잘 이해될 것이다.

제 1 예: 고 기록 성능

- [0046]
- [0047] 휴대폰의 비디오 스트리밍 애플리케이션(16)이 무선으로 비디오 파일을 다운로드하여 그 비디오 파일을 이후 재생을 위하여 휴대폰의 로컬 스토리지(10)내에 저장하는 세션을 개시한다. 다운로드 프로토콜은 소스가 중지하거나 재전송하도록 요청하기 위한 메커니즘없이 소스에 의해 전송될 때 임의의 패킷의 데이터를 애플리케이션(16)이 수용하도록 요구한다. 휴대폰의 램 메모리는 용량이 제한되고 매체(10)에 영구적으로 저장될 때까지 유입 데이터를 버퍼링하는데 사용될 수 없다. 따라서 이러한 조건하에서 성공적인 동작을 위해 고 기록 성능(및 저 레이턴시 시간)이 필요불가결하다.
- [0048] 휴대폰은 본 발명의 방법에 따라 구현된 저장 시스템을 채용한다. 소스와의 연결을 확립한 바로 이후(및 데이터 다운로드를 개시하기 전) 애플리케이션(16)은 유입 데이터를 저장하기 위한 새로운 파일을 생성한다. 이것은 다음 파일 시스템 콜 또는 이와 유사한 파일 시스템 콜을 본 발명의 파일 시스템(14)에 발생시킴으로써 수행된다.
- [0049] `fileHandle=CreateFile("NewsVideo")`
- [0050] 여기서, "NewsVideo"는 새로운 파일의 이름이고 "fileHandle"은 이후에 콜을 참조하기 위한 애플리케이션(16)에 의해 사용된 핸들이다.
- [0051] 다음, 애플리케이션(16)은 아래의 파일 시스템 콜 또는 이와 유사한 파일 시스템을 사용하여 새로운 파일의 원하는 특성에 대하여 본 발명의 파일 시스템(14)으로 알린다.
- [0052] `SetFileForFastWrite(fileHandle, EstimatedSize)`
- [0053] 여기서 "fileHandle"은 CreateFile 콜에 의해 이전에 반환된 핸들이고 "EstimatedSize"는 본 발명의 파일 시스템(14)으로 애플리케이션(16)이 파일의 크기가 어떻게 될 것이라는 것을 알려주는 (선택적인) 입력이다.
- [0054] 현재 본 발명의 파일 시스템(14)은 NewsVideo파일을 기록할 때 요구되는 것을 알고 있다. 본 발명의 전형적인 실시예에서, 파일 시스템(14)은 파일에 EstimatedSize와 같은 크기의 공간량(또는 평가된 크기가 제공되지 않는다면 디폴트 크기)을 즉시 할당한다. 파일은 그 파일을 위해 데이터가 제공되고 있지 않고 있어도 이러한 할당을 즉시 얻는다.
- [0055] 지금 애플리케이션(16)이 파일 시스템 콜:
- [0056] `WriteFile(fileHandle, pBuffer, iLength)`
- [0057] 또는 유사한 파일 시스템 콜을 발생시킬 때, 본 발명의 파일 시스템(14)은 이미 할당된 공간에 pBuffer로 지적된, 크기 iLength의 콘텐츠를 버퍼에 기록한다. (예상 크기가 초과되지 않는 한) 필요한 공간은 임의의 시간소모적이고 느린 저장 리사이클링을 할 필요없이 이용가능함을 보증한다.
- [0058] 이러한 많은 WriteFile콜 이후 세션은 종료하고 애플리케이션(16)은 파일 시스템 콜:
- [0059] `CloseFile(fileHandle)`
- [0060] 또는 유사한 파일 시스템 콜을 발생시킨다. 본 발명의 파일 시스템(14)은 예컨대 할당되고 사용되지않은 임의의 잉여 공간이 릴리스되고 실제 파일 크기와 디렉토리 엔트리 길이 필드간의 임의의 오차를 수정하는 등 "클린업"을 수행한다.
- [0061] 요컨대, 애플리케이션(16)은 시스템 콜 시퀀스:
- [0062] `fileHandle=CreateFile("NewsVideo")`
- [0063] `SetFileForFastWrite(fileHandle, EstimatedSize)`
- [0064] `WriteFile(fileHandle, pBuffer, iLength)`
- [0065] .
- [0066] .
- [0067] .
- [0068] `WriteFile(fileHandle, pBuffer, iLength)`

- [0069] CloseFile(fileHandle)
- [0070] 를 사용하여 유입하는 비디오 파일을 위한 최적의 기록 성능을 획득한다.
- [0071] 이것은 대응하는 종래의 파일 시스템(14)을 위한 시스템 콜 시퀀스와 같은 파일 시스템 콜의 디폴트 시퀀스:
- [0072] fileHandle=CreateFile("NewsVideo")
- [0073] WriteFile(fileHandle, pBuffer, iLength)
- [0074] .
- [0075] .
- [0076] .
- [0077] WriteFile(fileHandle, pBuffer, iLength)
- [0078] CloseFile(fileHandle)
- [0079] 와 대조적이다.
- [0080] 본 예에서 본 발명에 의해 제공된 유일하게 새로운 파일 시스템 콜은 SetFileForWrite콜이다. SetFileForFastWrite콜은 새 파일을 기록하는 동안 최적화된 기록 성능과 레이턴시 성능을 제공함을 주목하자.
- [0081] 제 2 예: 고 관독성능
- [0082] 게임 콘솔은 RAM에 로딩되어 사용자의 의지로 수행되는 게임 애플리케이션을 저장하기 위한 저장 시스템을 포함한다. 기록 성능이 중요하지 않지만(게임 애플리케이션은 단 한번만 저장된다), 실행할 게임을 사용자가 선택할 때부터 게임이 실제로 수행될 때까지의 시간에 영향을 미치기 때문에 관독 성능은 매우 중요하다. 장치의 운영 시스템내에는 메모리 페이징 서포트가 없고 따라서 전체 애플리케이션은 실행을 개시할 수 있기 전에 로딩되어야한다.
- [0083] 콘솔은 본 발명의 방법에 따라 구현된 저장 시스템을 채용한다. (예컨대, 제조장소에서 또는 인터넷에서 새로운 게임을 다운로드할 때와 같이) 장치에 게임 애플리케이션을 저장할 때, 저장 애플리케이션(16)이 게임 코드를 저장하기 위한 새로운 파일을 생성한다. 이것은 파일 시스템 콜:
- [0084] fileHandle=CreateFile("PacMan")
- [0085] 또는 유사한 파일 시스템 콜을 발생시킴으로써 수행된다(여기서, "PacMan"은 새로운 파일의 이름이고 "fileHandle"은 이후의 콜에서 파일을 참조하기 위한 애플리케이션(16)에 의해 사용된 핸들이다.
- [0086] 이후 애플리케이션(16)은 파일 시스템 콜:
- [0087] SetFileForFastRead(fileHandle, EstimatedSize)
- [0088] 또는 유사한 파일 시스템 콜을 사용하여 새로운 파일의 원하는 특성에 대한 본 발명의 파일 시스템(14)을 알린다(여기서 "fileHandle"은 CreateFile콜에 의해 이전에 반환된 핸들이고, "EstimatedSize"는 애플리케이션(16)이 파일 크기가 어떻게 될 것이라는 것을 본 발명의 파일 시스템(14)에게 알려주는 (선택적인) 입력이다.
- [0089] 지금 본 발명의 파일 시스템(14)은 PacMan파일을 기록할 때 요구되는 것을 알고 있다. 본 발명의 파일 시스템(14)의 일 전형적인 실시예에서, 이 단계에서는 요청을 "기억하는 것"외에는 어떤 것도 수행되지 않는다.
- [0090] 현재 애플리케이션(16)이 파일 시스템 콜:
- [0091] WriteFile(fileHandle, pBuffer, iLength)
- [0092] 또는 유사한 파일 시스템 콜을 발생시킬 때, 본 발명의 파일 시스템(14)은 (필요하다면) 공간을 할당하여 버퍼에 새롭게 할당된 공간에 pBuffer에 의해 지적된, 크기 iLength의 내용을 기록한다. 이 파일은 고속 관독을 위해 마킹된다는 것을 주목하면, 이러한 할당은 파일 할당 테이블에서 임의의 이용가능한 사용되지 않는 클러스터를 포착함은 물론 바람직하게는 플래시 매체(10)의 전체 삭제가능한 유닛을 포괄하는 인접하는 사용되지 않는 클러스터 군을 찾는다. 일단 발견되면, 현재의 기록 요청에 필요한 것보다 훨씬 클 수 있더라도, 완전한 군이 할당된다.
- [0093] 후속의 WriteFile콜에서, 현재 기록되고 있는 인접하는 사용되지 않는 클러스터군이 새로운 데이터를 위한 충분

한 공간이 있는지에 따라, 보다 많은 공간이 할당되거나 할당되지 않을 수 있다.

- [0094] 이러한 많은 WriteFile콜 이후에 세션은 종료하고 애플리케이션(16)은 파일 시스템 콜:
- [0095] CloseFile(fileHandle)
- [0096] 또는 유사한 파일 시스템 콜을 발생시킨다. 본 발명의 파일 시스템(14)은 예컨대 할당되고 사용되지않은 임의의 잉여 공간이 릴리스되고 실제 파일 크기와 디렉토리 엔트리 길이 필드간의 임의의 오차를 수정하는 등 "클린업"을 수행한다. 저장 시스템의 저 레벨 관독 기능이 관독 요청에서 탐지된 임의의 순차성을 자동적으로 이용하도록 설정될때 종료 결과는 개시에서 종료까지 연속적으로 관독될 때 평균 관독 성능보다 훨씬 빠른 성능을 달성하는 인접의 큰 청크에 할당된 파일이다.
- [0097] 요컨대, 애플리케이션(16)은 시스템 콜 시퀀스:
- [0098] fileHandle=CreateFile("PacMan")
- [0099] SetFileForFastRead(fileHandle, EstimatedSize)
- [0100] WriteFile(fileHandle, pBuffer, iLength)
- [0101] .
- [0102] .
- [0103] .
- [0104] WriteFile(fileHandle, pBuffer, iLength)
- [0105] CloseFile(fileHandle)
- [0106] 를 사용하여 유입하는 게임 애플리케이션을 위한 최적의 관독 성능을 획득한다.
- [0107] 이것은 대응하는 종래의 파일 시스템(14)을 위한 시스템 콜 시퀀스와 같은 파일 시스템 콜의 디폴트 시퀀스:
- [0108] fileHandle=CreateFile("PacMan")
- [0109] WriteFile(fileHandle, pBuffer, iLength)
- [0110] .
- [0111] .
- [0112] .
- [0113] WriteFile(fileHandle, pBuffer, iLength)
- [0114] CloseFile(fileHandle)
- [0115] 와 대조적이다.
- [0116] 본 예에서 본 발명에 의해 제공된 유일하게 새로운 파일 시스템 콜은 SetFileForRead콜이다.
- [0117] 몇몇 SetFileForFastWrite 또는 SetFileForFastRead 명령은 (서로 다른 파○일을 위한) fileHandle의 서로 다른 값으로 발생할 수 있고, 이후 다양한 파일에 대한 WriteFile명령이 인터리빙될 수 있음을 주목하자. 즉, 몇몇 파일은 동시에 기록될 수 있다.
- [0118] 이러한 두예는 애플리케이션(16)을 콜하는 것이 파일을 생성한 후와 데이터를 파일에 기록하기를 개시하기전에 원하는 성능 특성을 요청하는 본 발명의 실시예를 설명한다. 본 발명은 이러한 실시예로 제한되지 않는다. 이하의 방법 중 각각의 하나는 본 발명의 범위내에 있다.
- [0119] a. 애플리케이션(16)은 예컨대 "생성될 다음 파일은 명기된 특성을 가져야 한다"는 것을 의미하는 파일 시스템 콜을 발생시킴으로써 파일을 생성하기 전에 안내를 제공한다.
- [0120] b. 애플리케이션(16)은 (상기 두 예에서 처럼) 파일을 생성한 후지만 파일 데이터를 저장 시스템에 제공하기 전에 안내를 제공한다.
- [0121] c. 애플리케이션(16)은 파일의 데이터를 저장 시스템에 제공한 후이지만 파일을 닫기 전에 안내를 제공한다.

이것은 물론 판독 성능을 최적화하기 위해서만 유용하고 대개 저장 시스템으로 하여금 파일이 비최적적으로 이미 기록된 후 파일의 재정렬을 하도록 요구한다.

[0122] d. 애플리케이션(16)은 파일을 닫은 후 안내를 제공한다. 이 또한 판독 성능을 최적화하기 위해서만 유용하고 대개 스토리지내에서 파일의 재정렬을 요구한다.

[0123] 방법 c와 d를 구현하는 일 방법은 (상기한 바와 같이, 전형적으로 매우 느린 프로세스인, 전체 매체(10)를 조각모음하는 종래 방법과 대조적인 것으로) 파일을 조각모음하는 것이다.

[0124] 상기한 두 예는 제 1 예의 SetFileForFastWrite콜이나 제 2 예의 WriteFile콜에 의해, 요청될때 매체(10)에서 물리적으로 인접한 스토리지가 이용가능함을 전제로한다. 이러한 물리적으로 인접한 스토리지가 매체(10)에서 이용될 수 없는 경우, 당업자는 본 발명의 애플리케이션(16)이나 파일 시스템(14)의 어느 하나 또는 양 소프트웨어 컴포넌트에 의해 취해질 조치를 용이하게 생각할 수 있을 것이다. 예컨대, iError=SetFileForFastWrite(fileHandle, EstimatedSize)형태의 수정된 SetFileForFastWrite콜이 에러 코드("iError")를 요청된 물리적으로 인접한 스토리지가 이용가능하고 할당되는지(iError=0) 혹은 할당되지 않는지(iError=1)를 나타내는 애플리케이션(16)에 반환한다. 후자의 경우에, 애플리케이션(16)은 기록 성능 최적화가 없는 디폴트 파일 기록 콜 시퀀스로 되돌리거나, 본 발명의 파일 시스템(14)에 적당한 콜을 갖는 (예컨대 선입 선출 순으로) 구 파일을 삭제하고 최적화된 기록을 위하여 (예컨대 "OpenFile(fileHandle)타입의 콜을 포함하는 콜을 갖는) 새로운 파일을 재오픈하는, 새로운 파일을 닫는 옵션을 갖는다. 본 발명의 파일 시스템(14)은 애플리케이션(16)으로부터 적당한 콜에 수신된 콜을 구동 드라이버(12)로 번역하여 새로운 파일을 수용시키도록 매체(10)내에 충분히 인접한 물리적 공간을 클리어링한다. iError=SetFileForFastRead(fileHandle, EstimatedSize)형태의 수정된 SetFileForFastRead콜은 유사하게 동작한다.

[0125] 대안으로, SetFileForFastWrite명령과 SetFileForFastRead명령은 에러 코드를 반환하지 않고 "최고의 노력" 명령으로 기능한다. 파일 시스템(14)이, 예컨대 충분하지 않은 인접 공간이 할당될 수 있다는 이유로, 원하는 성능 특성을 얻을 수 없다고 판단하면, 파일 시스템(14)은 저장 장치(10)에 데이터를 저장하는 종래 방법 중 하나로 되돌린다.

[0126] 도 3은 본 발명의 데이터 프로세싱 장치(30)의 고레벨 부분 블록도이다. 데이터 프로세싱 장치(30)는 프로세서(32), 휘발성 메모리(특히, RAM)(34), 그리고 비휘발성 메모리들-ROM(36), 하드디스크(38), 그리고 물리적 저장 매체(10)로 기능하는 플래시 EEPROM등을 포함한다. 하드디스크(38)는 애플리케이션 프로그램(16), 그리고 운영 시스템(40)의 일부로서, 플래시 EEPROM(10)과 본 발명의 파일 시스템(14)을 위한 구동 드라이버(12)를 저장한다. 프로세서(32), 휘발성 메모리(34) 그리고 비휘발성 메모리(36, 38, 10)는 버스(42)를 거쳐 서로 통신하게 된다. 데이터 프로세싱 장치(30)에 전원이 공급되면, 프로세서(32)는 ROM(36)으로부터 부트 코드를 실행시킨다. 부트 코드의 기능 중 하나는 하드디스크(38)로부터 RAM(34)으로 운영 시스템(40)을 로딩하는 것이다. 운영 시스템(40)이 RAM(34)에 로딩되면, 프로세서(32)는 애플리케이션 프로그램(16)중 하나를 RAM(34)에 로딩하여 애플리케이션 프로그램(16)을 실행시키는 것을 진행한다. 실행된 애플리케이션 프로그램(16)은 (도시안된) 종래의 수단에 의해 데이터 프로세싱 장치(30)가 연결되어 있는 (도시안된) 네트워크등을 통해 외부 데이터 프로세싱 장치(30)로부터 파일을 다운로드하여 플래시 EEPROM(10)에 저장하여, 본 발명의 파일 시스템(14)에 적당한 콜을 사용하여 필요한 파일의 판독, 기록 및/또는 레이턴시 성능을 최적화시킨다. 상기한 바와 같이, 파일은 플래시 EEPROM(10)에 동시에 저장될 수 있다.

[0127] 하드디스크(38)도 본 발명의 파일 시스템(14)을 구현한 컴퓨터 판독가능 저장 매체를 대표한다.

[0128] 본 발명이 제한된 수의 실시예에 관하여 설명되었지만, 본 발명에 많은 변형, 변경 및 기타 응용이 가능함을 이해하여야한다.

도면의 간단한 설명

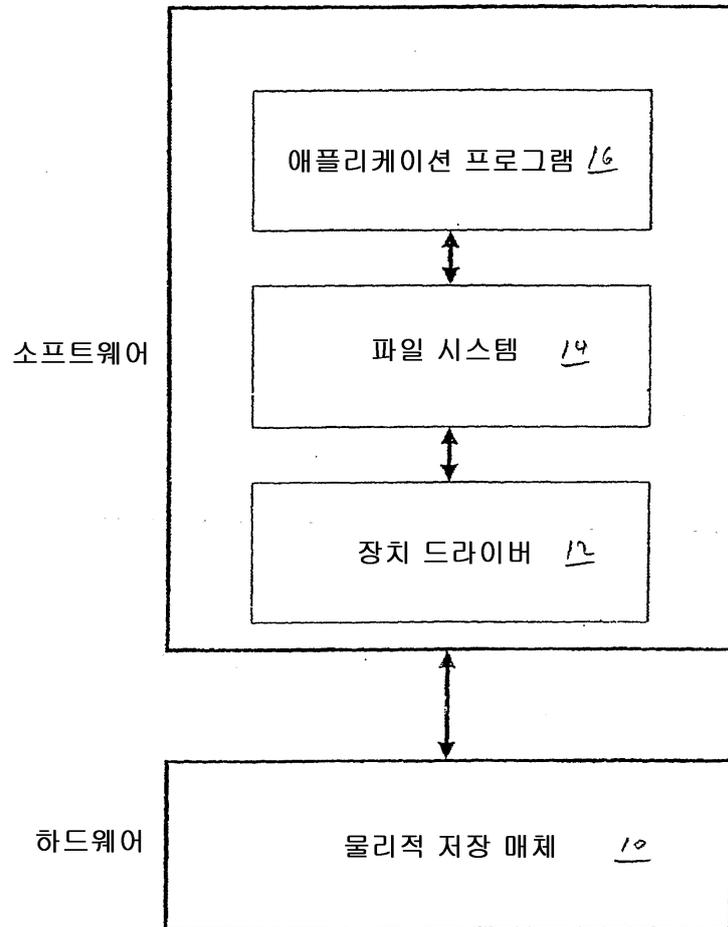
[0040] 도 1은 종래 기술 및 본 발명에 따라 데이터 저장 매체가 소프트웨어에 의해 어떻게 액세스되는지를 도시한 도면.

[0041] 도 2는 블록 장치의 구조를 도시한 도면.

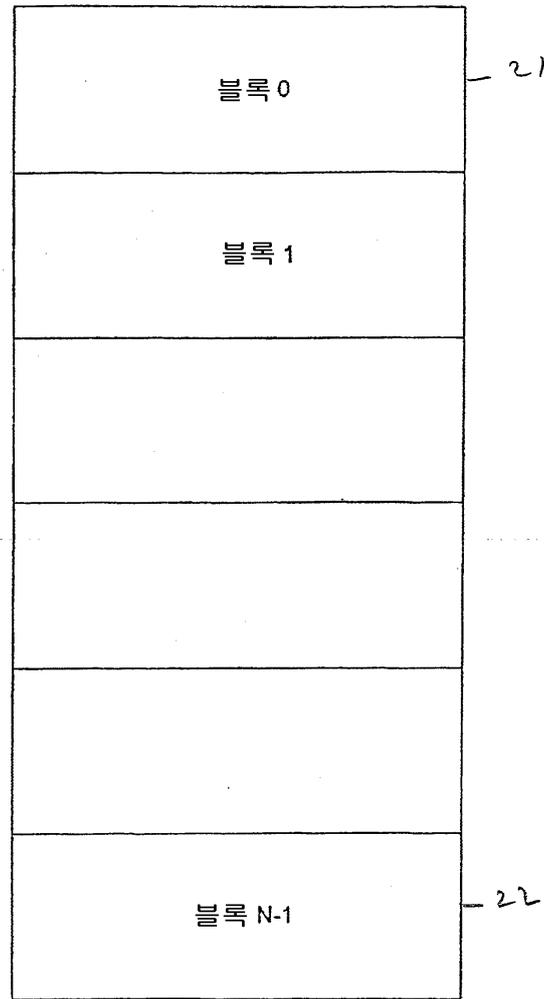
[0042] 도 3은 본 발명의 데이터 프로세싱 장치의 고 레벨 부분 블록도.

도면

도면1



도면2



도면3

