(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년03월21일
(11) 등록번호 10-1840996
(24) 등록일자 2018년03월15일

(51) 국제특허분류(Int. Cl.)
G06F 17/30 (2006.01) *G06F 9/06* (2018.01)
(52) CPC특허분류
G06F 17/30345 (2013.01)
G06F 17/30283 (2013.01)
(21) 출원번호 10-2017-7033663(분할)
(22) 출원일자(국제) 2011년06월01일
심사청구일자 2017년11월21일
(85) 번역문제출일자 2017년11월21일
(65) 공개번호 10-2017-0132338
(43) 공개일자 2017년12월01일
(62) 원출원 특허 10-2012-7032646
원출원일자(국제) 2011년06월01일
심사청구일자 2016년04월27일
(86) 국제출원번호 PCT/US2011/038811
(87) 국제공개번호 WO 2011/159476
국제공개일자 2011년12월22일
(30) 우선권주장
12/815,418 2010년06월15일 미국(US)
(56) 선행기술조사문헌
US20090094582 A1
US20070276878 A1
US7721062 A
US7401093 A

(73) 특허권자
마이크로소프트 테크놀로지 라이센싱, 엘엘씨
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
카길 조나단 엠
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 페이턴츠 마이크로소프트 코포레이션
밀러 토마스 제이
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 페이턴츠 마이크로소프트 코포레이션
팁톤 윌리엄 알
미국 워싱턴주 98052-6399 레드몬드 원 마이크로소프트 웨이 엘씨에이 - 인터내셔널 페이턴츠 마이크로소프트 코포레이션
(74) 대리인
제일특허법인

전체 청구항 수 : 총 37 항 심사관 : 경연정

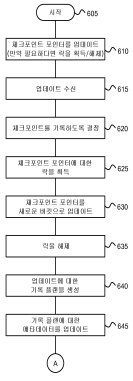
(54) 발명의 명칭 **파일 시스템에 대한 체크포인트**

(57) 요 약

본 명세서에 기술된 청구사항의 측면들은 파일 시스템에 대한 체크포인트에 관련된다. 측면들에서, 파일 시스템에 대한 업데이트들은 체크포인트 버킷들로 조직화된다. 체크포인트가 요구될 때, 후속 업데이트들이 다른 체크포인트 버킷으로 연결된다. 현재 체크포인트 버킷에서의 업데이트들에 대해서 글로벌 테이블이 업데이트된

(뒷면에 계속)

대 표 도 - 도6



후에, 글로벌 테이블의 논리적 카피가 생성된다. 이러한 논리적 카피는 체크포인트 데이터의 일부로서 저장된다. 복구를 돕기 위해서, 체크포인트 매니저는 최종 체크포인트 데이터를 스토리지에 기록하기 전에 현재 체크포인트 버킷의 모든 업데이트들이 스토리지에 기록될 때까지 기다릴 수 있다. 이러한 최종 체크포인트 데이터는 글로벌 테이블의 논리적 카피를 지칭할 수 있고, 체크포인트 데이터가 올바른 것인지를 입증하도록 확인 코드를 포함할 수 있다.

(52) CPC특허분류
G06F 17/30321 (2013.01)
G06F 17/30327 (2013.01)
G06F 17/30589 (2013.01)
G06F 9/06 (2013.01)

명 세 서

청구범위

청구항 1

하드웨어 컴퓨팅 장치에서 수행되는 방법으로서,

체크포인트를 복수의 파일 시스템 객체의 업데이트와 연관시키는 단계- 상기 업데이트는 트랜잭션에 대응하고, 상기 파일 시스템 객체는 저장 장치 상의 대응하는 제1 위치에 저장되고, 상기 업데이트는 상기 파일 시스템 객체의 적어도 일부의 수정된 논리적 카피를 포함함 -와,

상기 제1 위치와 다른 제2 위치에서 상기 업데이트를 상기 저장 장치에 기록하기 위한 기록 플랜(write plan)을 생성하되 상기 생성과 병행하여, 상기 연관시키는 단계에 후속하여 발생하는 후속 업데이트에 대해 다른 기록 플랜이 또한 생성되는 것을 허용하는 단계- 상기 생성된 기록 플랜은 상기 업데이트가 상기 트랜잭션의 컨텍스트에서 서로를 바인딩함을 지정하고, 상기 생성된 기록 플랜은 상기 제2 위치를 나타냄 -와,

상기 생성된 기록 플랜에 기초하여 또한 상기 업데이트가 상기 생성된 기록 플랜에 따라 상기 제2 위치에서 상기 저장 장치에 기록되는 것에 응답하여, 체크포인트 정보를 상기 저장 장치에 기록하는 단계- 상기 체크포인트 정보는 상기 생성된 기록 플랜 및 체크포인트 레코드를 포함함 -를

포함하는 방법.

청구항 2

제 1 항에 있어서,

상기 연관시키는 단계에 앞서, 상기 업데이트가 상기 체크포인트와 연관될 것이라는 표시를 제공하는 단계를 더 포함하는

방법.

청구항 3

제 1 항에 있어서,

상기 후속 업데이트가 후속 체크포인트와 연관될 것이라는 표시를 제공하는 단계를 더 포함하는

방법.

청구항 4

제 1 항에 있어서,

상기 업데이트를 상기 제2 위치에서 상기 저장 장치에 기록하는 단계를 더 포함하는

방법.

청구항 5

제 1 항에 있어서,

기록된 상기 체크포인트 레코드는 상기 기록된 업데이트를 인증하는데 사용되도록 구성된 확인 코드를 포함하는

방법.

청구항 6

제 5 항에 있어서,

상기 확인 코드는 복구 동작에 사용되도록 더 구성되는

방법.

청구항 7

컴퓨팅 장치에 의해 실행되는 경우 상기 컴퓨팅 장치로 하여금 동작들을 수행하게 하는 컴퓨터 실행가능 명령어를 저장하는 적어도 하나의 컴퓨터 저장 매체로서,

상기 동작들은

체크포인트를 복수의 파일 시스템 객체의 업데이트와 연관시키는 동작- 상기 업데이트는 트랜잭션에 대응하고, 상기 파일 시스템 객체는 저장 장치 상의 대응하는 제1 위치에 저장되고, 상기 업데이트는 상기 파일 시스템 객체의 적어도 일부의 수정된 논리적 카피를 포함함 -과,

상기 제1 위치와 다른 제2 위치에서 상기 업데이트를 상기 저장 장치에 기록하기 위한 기록 플랜을 생성하되 상기 생성과 병행하여, 상기 연관시키는 동작에 후속하여 발생하는 후속 업데이트에 대해 다른 기록 플랜이 또한 생성되는 것을 허용하는 동작- 상기 생성된 기록 플랜은 상기 업데이트가 상기 트랜잭션의 컨텍스트에서 서로를 바인딩함을 지정하고, 상기 생성된 기록 플랜은 상기 제2 위치를 나타냄 -과,

상기 생성된 기록 플랜에 기초하여 또한 상기 업데이트가 상기 생성된 기록 플랜에 따라 상기 제2 위치에서 상기 저장 장치에 기록되는 것에 응답하여, 체크포인트 정보를 상기 저장 장치에 기록하는 동작- 상기 체크포인트 정보는 상기 생성된 기록 플랜 및 체크포인트 레코드를 포함함 -을 포함하는

컴퓨터 저장 매체.

청구항 8

제 7 항에 있어서,

상기 동작들은 상기 연관시키는 동작에 앞서, 상기 업데이트가 상기 체크포인트와 연관될 것이라는 표시를 제공하는 동작을 더 포함하는

컴퓨터 저장 매체.

청구항 9

제 7 항에 있어서,

상기 동작들은 상기 후속 업데이트가 후속 체크포인트와 연관될 것이라는 표시를 제공하는 동작을 더 포함하는

컴퓨터 저장 매체.

청구항 10

제 7 항에 있어서,

상기 동작들은 상기 업데이트를 상기 제2 위치에서 상기 저장 장치에 기록하는 동작을 더 포함하는

컴퓨터 저장 매체.

청구항 11

제 7 항에 있어서,

기록된 상기 체크포인트 레코드는 상기 기록된 업데이트를 인증하는데 사용되도록 구성된 확인 코드를 포함하는

컴퓨터 저장 매체.

청구항 12

제 11 항에 있어서,

상기 확인 코드는 복구 동작에 사용되도록 더 구성되는

컴퓨터 저장 매체.

청구항 13

동작들을 수행하도록 구성된 컴퓨팅 장치로서,

상기 동작들은

체크포인트를 복수의 파일 시스템 객체의 업데이트와 연관시키는 동작- 상기 업데이트는 트랜잭션에 대응하고, 상기 파일 시스템 객체는 저장 장치 상의 대응하는 제1 위치에 저장되고, 상기 업데이트는 상기 파일 시스템 객체의 적어도 일부의 수정된 논리적 카피를 포함함 -과,

상기 제1 위치와 다른 제2 위치에서 상기 업데이트를 상기 저장 장치에 기록하기 위한 기록 플랜을 생성하되 상기 생성과 병행하여, 상기 연관시키는 동작에 후속하여 발생하는 후속 업데이트에 대해 다른 기록 플랜이 또한 생성되는 것을 허용하는 동작- 상기 생성된 기록 플랜은 상기 업데이트가 상기 트랜잭션의 컨텍스트에서 서로를 바인딩함을 지정하고, 상기 생성된 기록 플랜은 상기 제2 위치를 나타냄 -과,

상기 생성된 기록 플랜에 기초하여 또한 상기 업데이트가 상기 생성된 기록 플랜에 따라 상기 제2 위치에서 상기 저장 장치에 기록되는 것에 응답하여, 체크포인트 정보를 상기 저장 장치에 기록하는 동작- 상기 체크포인트 정보는 상기 생성된 기록 플랜 및 체크포인트 레코드를 포함함 -을 포함하는

컴퓨팅 장치.

청구항 14

제 13 항에 있어서,

상기 컴퓨팅 장치는 상기 연관시키는 동작에 앞서, 상기 업데이트가 상기 체크포인트와 연관될 것이라는 표시를 제공하도록 더 구성되는

컴퓨팅 장치.

청구항 15

제 13 항에 있어서,

상기 컴퓨팅 장치는 상기 후속 업데이트가 후속 체크포인트와 연관될 것이라는 표시를 제공하도록 더 구성되는

컴퓨팅 장치.

청구항 16

제 13 항에 있어서,

상기 컴퓨팅 장치는 상기 업데이트를 상기 제2 위치에서 상기 저장 장치에 기록하도록 더 구성되는

컴퓨팅 장치.

청구항 17

제 13 항에 있어서,

기록된 상기 체크포인트 레코드는 상기 기록된 업데이트를 인증하는데 사용되도록 구성된 확인 코드를 포함하고, 상기 확인 코드는 복구 동작에 사용되도록 더 구성되는

컴퓨팅 장치.

청구항 18

적어도 하나의 처리 장치 및 메모리를 포함하는 적어도 하나의 컴퓨팅 장치에서 수행되는 방법으로서,

상기 적어도 하나의 컴퓨팅 장치에 의해, 파일 시스템에 대한 업데이트를 제2 위치에 기록하기 위한 기록 플랜을 생성하는 단계- 상기 업데이트는 제1 위치에 저장된 객체에 대한 것이며, 상기 객체는 상기 파일 시스템의 객체이고, 상기 업데이트는 체크포인트와 연관되고, 상기 제2 위치는 상기 제1 위치와 다름 -와,

상기 업데이트가 상기 생성된 기록 플랜에 따라 상기 제2 위치에 기록되는 것과 연계하여 상기 적어도 하나의 컴퓨팅 장치에 의해, 상기 체크포인트에 대한 정보를 상기 제2 위치에 기록하는 단계- 상기 기록된 정보는 상기 생성된 기록 플랜 및 체크포인트 레코드를 포함함 -를 포함하는

방법.

청구항 19

제 18 항에 있어서,

상기 체크포인트 정보는 상기 파일 시스템에 대한 적어도 하나의 업데이트를 나타내는

방법.

청구항 20

제 18 항에 있어서,

상기 체크포인트 정보는 상기 기록된 체크포인트 정보가 정확히 기록되었음을 판정하기에 충분한 확인 코드를 포함하는

방법.

청구항 21

제 18 항에 있어서,

상기 기록 플랜은 상기 파일 시스템에 대한 적어도 하나의 업데이트를 위한 위치를 나타내는

방법.

청구항 22

제 18 항에 있어서,

상기 체크포인트 정보는 상기 객체의 저장 위치를 나타내는

방법.

청구항 23

제 18 항에 있어서,

상기 체크포인트 정보는 공간 할당 정보를 포함하는

방법.

청구항 24

제 18 항에 있어서,

상기 기록하는 단계는 상기 객체에 대한 업데이트가 상기 제1 위치에 기록되는 것에 후속하는

방법.

청구항 25

함께 동작들을 수행하도록 구성된 적어도 하나의 컴퓨팅 장치 및 적어도 하나의 프로그램 모듈을 포함하는 시스템으로서,

상기 적어도 하나의 컴퓨팅 장치는 적어도 하나의 처리 장치 및 메모리를 포함하고,

상기 동작들은

상기 적어도 하나의 컴퓨팅 장치에 의해, 파일 시스템에 대한 업데이트를 제2 위치에 기록하기 위한 기록 플랜을 생성하는 동작- 상기 업데이트는 제1 위치에 저장된 객체에 대한 것이며, 상기 객체는 상기 파일 시스템의 객체이고, 상기 업데이트는 체크포인트와 연관되고, 상기 제2 위치는 상기 제1 위치와 다름 -과,

상기 업데이트가 상기 생성된 기록 플랜에 따라 상기 제2 위치에 기록되는 것과 연계하여 상기 적어도 하나의 컴퓨팅 장치에 의해, 상기 체크포인트에 대한 정보를 상기 제2 위치에 기록하는 동작- 상기 기록된 정보는 상기 생성된 기록 플랜 및 체크포인트 레코드를 포함함 -을 포함하는

시스템.

청구항 26

제 25 항에 있어서,

상기 체크포인트 정보는 상기 파일 시스템에 대한 적어도 하나의 업데이트를 나타내는

시스템.

청구항 27

제 25 항에 있어서,

상기 체크포인트 정보는 상기 기록된 체크포인트 정보가 정확히 기록되었음을 판정하기에 충분한 확인 코드를 포함하는

시스템.

청구항 28

제 25 항에 있어서,

상기 기록 플랜은 상기 파일 시스템에 대한 적어도 하나의 업데이트를 위한 위치를 나타내는

시스템.

청구항 29

제 25 항에 있어서,

상기 체크포인트 정보는 상기 객체의 저장 위치를 나타내는

시스템.

청구항 30

제 25 항에 있어서,

상기 체크포인트 정보는 공간 할당 정보를 포함하는

시스템.

청구항 31

제 25 항에 있어서,

상기 기록하는 동작은 상기 객체에 대한 업데이트가 상기 제1 위치에 기록되는 것에 후속하는

시스템.

청구항 32

적어도 하나의 처리 장치 및 메모리를 포함하는 적어도 하나의 컴퓨팅 장치에 의해 실행되는 경우 상기 적어도 하나의 컴퓨팅 장치로 하여금 동작들을 수행하게 하는 컴퓨터 실행가능 명령어를 저장하는 적어도 하나의 컴퓨터 저장 매체로서,

상기 동작들은

상기 적어도 하나의 컴퓨팅 장치에 의해, 파일 시스템에 대한 업데이트를 제2 위치에 기록하기 위한 기록 플랜을 생성하는 동작- 상기 업데이트는 제1 위치에 저장된 객체에 대한 것이며, 상기 객체는 상기 파일 시스템의 객체이고, 상기 업데이트는 체크포인트와 연관되고, 상기 제2 위치는 상기 제1 위치와 다름 -과,

상기 업데이트가 상기 생성된 기록 플랜에 따라 상기 제2 위치에 기록되는 것과 연계하여 상기 적어도 하나의 컴퓨팅 장치에 의해, 상기 체크포인트에 대한 정보를 상기 제2 위치에 기록하는 동작- 상기 기록된 정보는 상기 생성된 기록 플랜 및 체크포인트 레코드를 포함함 -을 포함하는

컴퓨터 저장 매체.

청구항 33

제 32 항에 있어서,

상기 체크포인트 정보는 상기 파일 시스템에 대한 적어도 하나의 업데이트를 나타내는

컴퓨터 저장 매체.

청구항 34

제 32 항에 있어서,

상기 체크포인트 정보는 상기 기록된 체크포인트 정보가 정확히 기록되었음을 판정하기에 충분한 확인 코드를 포함하는

컴퓨터 저장 매체.

청구항 35

제 32 항에 있어서,

상기 기록 플랜은 상기 파일 시스템에 대한 적어도 하나의 업데이트를 위한 위치를 나타내는

컴퓨터 저장 매체.

청구항 36

제 32 항에 있어서,

상기 체크포인트 정보는 상기 객체의 저장 위치를 나타내거나, 상기 체크포인트 정보는 공간 할당 정보를 포함하는

컴퓨터 저장 매체.

청구항 37

제 32 항에 있어서,

상기 기록하는 동작은 상기 객체에 대한 업데이트가 상기 제1 위치에 기록되는 것에 후속하는

컴퓨터 저장 매체.

발명의 설명

기 술 분 야

배 경 기 술

[0001] 스토리지 디바이스에 데이터를 기록하는 중간에 정전 또는 시스템 고장이 발생할 수 있다. 이러한 상황이 발생하면, 데이터는 손실되거나 비일관적일 수 있다. 예를 들어, 만약 계좌 소유자가 ATM으로부터 돈을 인출하는 중간에 시스템이 고장난다면 트랜잭션은 은행 또는 계좌 소유자에게 부당한 호의를 베풀게 될 수 있다. 다른 예시로서, 만약 시스템이 디스크 액세스를 포함하는 매우 긴 계산 중에 고장난다면, 그 계산을 다시 하는데에 상당한 시간이 걸릴 수 있다.

[0002] 본 명세서에서 청구된 청구사항은 임의의 단점들을 해결하거나 전술된 바와 같은 환경에서만 동작하는 실시예들로 제한되지 않는다. 오히려, 이러한 배경기술은 단지 본 명세서에 기술된 일부 실시예들이 실시될 수 있는 하나의 예시적인 기술 영역을 설명하도록 제공된 것이다.

발명의 내용

[0003] 간략하게, 본 명세서에 기술된 청구사항의 측면들은 파일 시스템에 대한 체크포인트와 관련된다. 측면들에서, 파일 시스템에 대한 업데이트가 체크포인트 버킷들(checkpoint buckets)로 편성된다. 체크포인트가 요구되면, 후속 업데이트가 다른 체크포인트 버킷에 연결된다. 현재 체크포인트 버킷 내의 업데이트에 대해서 다음 글로벌 테이블(global table)이 업데이트된 후에, 글로벌 테이블의 논리적 카피가 생성된다. 이러한 논리적 카피는 체크포인트 데이터의 일부로서 저장된다. 복구를 돕기 위해서, 체크포인트 매니저는 최종 체크포인트 데이터를 스토리지에 기록하기에 앞서 현재 체크포인트 버킷의 모든 업데이트가 스토리지에 기록될 때까지 기다릴 수 있다. 이러한 최종 체크포인트 데이터는 글로벌 테이블의 논리적 카피를 참조할 수 있고, 체크포인트 데이터가 정확하다는 것을 확인하기 위한 확인 코드(validation code)를 포함할 수 있다.

[0004] 이 요약부는 아래의 상세한 설명에서 추가로 기술되는 개념들의 선택을 간략한 형태로 소개하고자 제공되었다. 본 요약부는 특허청구범위의 중요 특징 또는 기본 특징을 식별하고자 하는 것은 아니며, 청구범위의 범주를 제한하는데에 사용되는 것 또한 아니다.

[0005] "본 명세서에 기술된 청구사항"이라는 구절은 명확하게 언급되지 않는 한 상세한 설명에 기술된 사항들을 지칭한다. "측면"이라는 용어는 "적어도 하나의 측면"으로 해석되어야 한다. 상세한 설명에 기술된 청구사항의 측면들을 식별하는 것은 청구사항의 중요한 특징 또는 기본 특징을 식별하기 위한 것은 아니다.

[0006] 전술된 측면들과 본 명세서에 기술된 청구사항의 다른 측면들이 첨부된 도면에 제한적이 아닌 예시적인 방식으로 도시되었으며, 도면에서 동일한 참조 번호는 유사한 요소를 지칭한다.

도면의 간단한 설명

[0007] 도 1은 본 명세서에 기술된 청구사항의 측면들이 결합될 수 있는 예시적인 범용 컴퓨팅 환경을 나타내는 블록도;

도 2는 본 명세서에 기술된 청구사항의 측면들이 동작할 수 있는 시스템의 구성요소들의 예시적인 배치를 나타내는 블록도;

도 3은 본 명세서에 기술된 청구사항의 측면들을 도시한 블록도;

도 4는 본 명세서에 기술된 청구사항의 측면들에 따른 파일 시스템에 대한 업데이트를 일반적으로 나타내는 도면;

도 5는 본 명세서에 기술된 청구사항의 측면들에 따른 예시적인 체크포인트 버킷들을 도시한 블록도;

도 6 내지 8은 본 명세서에 기술된 청구사항의 측면들에 따라 발생할 수 있는 예시적인 작업들을 일반적으로 나타내는 흐름도.

발명을 실시하기 위한 구체적인 내용

[0008] 정의

[0009] 본 명세서에서 사용되는 "포함하는"이라는 용어 및 그에서 변형 단어들은 "-를 포함하지만 그것으로 제한되는 것은 아닌"이라는 의미의 제한을 두지 않는 용어로서 해석되어야 한다. "또는"이라는 용어는 명확하게 언급되지 않는 한 "및/또는"의 의미로 해석되어야 한다. "-에 기초하여"라는 용어는 "적어도 일부분 기초하여"로 읽혀져야 한다. "일 실시예" 및 "실시예"라는 용어들은 "적어도 하나의 실시예"로서 읽혀져야 한다. "다른 실시예"라는 용어는 "적어도 하나의 다른 실시예"로 읽혀져야 한다. 명시적이거나 암시적인 그외의 정의들이 아래에 포함될 수 있다.

[0010] 예시적인 동작 환경

[0011] 도 1은 본 명세서에 기술된 청구사항의 측면들이 구현될 수 있는 적합한 컴퓨팅 시스템 환경(100)의 예시를 도시한다. 컴퓨팅 시스템 환경(100)은 단지 적합한 컴퓨팅 환경의 일례일 뿐이며, 본 명세서에 기술된 청구사항

의 측면들의 이용 또는 기능의 범주와 관련된 어떠한 제한을 두기 위한 것도 아니다. 컴퓨팅 환경(100)이 예시적인 운영 환경(100) 내에 도시된 구성요소들 중 임의의 하나 또는 구성요소들의 조합과 관련하여 어떠한 종속성 또는 필요성을 가지는 것으로도 해석되어서는 안된다.

[0012] 본 명세서에 기술된 청구사항의 측면들은 다수의 다른 범용 또는 전용 컴퓨팅 시스템 환경 또는 구성과 운용가능하다. 본 명세서에 기술된 청구사항의 측면들과 함께 사용하기 적합할 수 있는 잘 알려진 컴퓨팅 시스템, 환경, 또는 구성의 예시는, 개인 컴퓨터, 서버 컴퓨터, 휴대용 또는 랩탑 디바이스, 멀티프로세서 시스템, 마이크로컨트롤러 기반의 시스템, 셋톱 박스, 프로그램가능한 소비자 전자기기, 네트워크 PC, 미니컴퓨터, 메인프레임 컴퓨터, PDA, 게임 디바이스, 프린터, 셋톱과 미디어 센터를 포함하는 어플라이언스(appliance), 또는 그 외의 어플라이언스, 차량 장착 또는 부착된 컴퓨팅 디바이스, 그 외의 모바일 디바이스, 전술된 임의의 시스템들 또는 디바이스들을 포함하는 분산 컴퓨팅 환경 등을 포함한다.

[0013] 본 명세서에 기술된 청구사항의 측면들은 컴퓨터에 의해 실행되는 프로그램 모듈과 같은 컴퓨터 실행가능한 명령들의 일반적인 맥락에서 기술될 수 있다. 일반적으로, 프로그램 모듈은 특정한 태스크를 수행하거나 특정한 추출 데이터 타입을 구현하는 루틴, 프로그램, 객체, 컴포넌트, 데이터 구조 등을 포함한다. 본 명세서에 기술된 청구사항의 측면들은 또한 통신 네트워크를 통해 링크된 원격 프로세싱 디바이스에 의해 태스크가 수행되는 분산 컴퓨팅 환경에서 실시될 수도 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 메모리 스토리지 디바이스를 포함하는 로컬 및 원격 컴퓨터 스토리지 매체 모두에 위치될 수 있다.

[0014] 도 1을 참조하면, 본 명세서에 기술된 청구사항의 측면들을 구현하는 예시적인 시스템은 컴퓨터(110)의 형태로 범용 컴퓨팅 디바이스를 포함한다. 컴퓨터는 명령을 실행할 수 있는 임의의 전자 디바이스를 포함할 수 있다. 컴퓨터(110)의 구성요소들은 프로세싱 유닛(120), 시스템 메모리(130) 및 시스템 메모리를 포함하는 다양한 시스템 구성요소들을 프로세싱 유닛(120)에 연결시키는 시스템 버스(121)를 포함할 수 있다. 시스템 버스(121)는 메모리 버스 또는 메모리 컨트롤러를 포함하는 몇몇 유형의 버스 구조들, 주변 버스 및 다양한 버스 아키텍처 중 임의의 것을 이용하는 로컬 버스 중 임의의 것일 수 있다. 예를 들어, 이러한 아키텍처는 산업 표준 아키텍처(ISA; Industry Standard Architecture) 버스, 마이크로 채널 아키텍처(MCA; Micro Channel Architecture) 버스, 개선된 ISA(EISA) 버스, 비디오 전자기기 표준 협의(VESA; Video Electronics Standards Association) 로컬 버스, 메자닌 버스(Mezzanine bus)로도 알려져 있는 주변 컴포넌트 상호접속(PCI; Peripheral Component Interconnect) 버스, 주변 컴포넌트 상호접속 확장(PCI-X) 버스, 발전된 그래픽 포트(AGP; Advanced Graphics Port) 및 PCI 익스프레스(PCIe)를 포함하지만 이것으로 제한되지는 않는다.

[0015] 컴퓨터(110)는 전형적으로 다양한 컴퓨터 판독가능한 매체를 포함한다. 컴퓨터 판독가능한 매체는 컴퓨터(110)에 의해 액세스될 수 있고 휘발성 매체와 비휘발성 매체 및 제거가능한 매체와 제거 불가능한 매체 모두를 포함하는 임의의 입수가능한 매체일 수 있다. 예를 들면, 컴퓨터 판독가능한 매체는 컴퓨터 스토리지 매체 및 통신 매체를 포함할 수 있지만, 이것으로 제한되지는 않는다.

[0016] 컴퓨터 스토리지 매체는 컴퓨터 판독가능한 명령들, 데이터 구조, 프로그램 모듈, 또는 다른 데이터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현되는 휘발성 및 비휘발성 매체, 제거가능 및 제거 불가능 매체 모두를 포함한다. 컴퓨터 스토리지 매체는 RAM, ROM, EEPROM, 플래시 메모리 또는 다른 메모리 기술, CD-ROM, DVD 또는 다른 광디스크 스토리지, 자기 카세트, 자기 테이프, 자기 디스크 스토리지 또는 다른 자기 스토리지 디바이스, 또는 원하는 정보를 저장하는데에 사용될 수 있고 컴퓨터(110)에 의해 액세스될 수 있는 임의의 다른 매체를 포함한다.

[0017] 통신 매체는 전형적으로 컴퓨터 판독가능한 명령들, 데이터 구조, 프로그램 모듈, 또는 반송파 또는 다른 전송 메커니즘과 같은 수정된 데이터 신호의 다른 데이터를 구현하고, 임의의 정보 전달 매체를 포함한다. "수정된 데이터 신호(modulated data signal)"라는 용어는 자신의 하나 이상의 형질 세트(characteristics set)를 구비하거나, 또는 신호 내의 정보를 인코딩하는 방식으로 변경된 신호를 의미한다. 예를 들면, 통신 매체는 유선 네트워크 또는 직통(direct-wire) 통신과 같은 유선 매체 및 음향, RF, 적외선 및 그외의 무선 매체와 같은 무선 매체를 포함하지만, 이것으로 제한되는 것은 아니다. 전술된 것들의 임의의 조합들 또한 컴퓨터 판독가능한 매체의 범주 내에 포함되어야 한다.

[0018] 시스템 메모리(130)는 판독 전용 메모리(ROM)(131) 및 랜덤 액세스 메모리(RAM)(132)와 같은 휘발성 및/또는 비휘발성 메모리의 형태로 컴퓨터 스토리지 매체를 포함한다. 예컨대 시작(start-up) 동안 컴퓨터(110) 내의 소자들 간의 정보 전송을 돕는 베이직 루틴들을 포함하는 베이직 입력/출력 시스템(133)(BIOS)이 전형적으로 ROM(131) 내에 저장된다. RAM(132)은 전형적으로 데이터 및/또는 즉시 액세스될 수 있고/있거나 현재 프로세싱

유닛(120)에 의해 동작되고 있는 프로그램 모듈들을 포함한다. 예를 들면, 도 1은 운영 시스템(134), 애플리케이션 프로그램(135), 다른 프로그램 모듈들(136) 및 프로그램 데이터(137)를 포함하지만, 이것으로 제한되는 것은 아니다.

[0019] 컴퓨터(110)는 또한 다른 제거가능/제거 불가능, 휘발성/비휘발성 컴퓨터 스토리지 매체를 포함할 수도 있다. 예시적으로, 도 1은 제거 불가능한 비휘발성 자기 매체로/로부터 기록 또는 판독하는 하드 디스크 드라이브(141), 제거가능한 비휘발성 자기 디스크(152)로/로부터 기록 또는 판독하는 자기 디스크 드라이브(151) 및 CD-ROM 또는 다른 광학적 매체와 같은 제거가능한 비휘발성 광디스크(156)로/로부터 기록 또는 판독하는 광디스크 드라이브(155)를 도시한다. 예시적인 운영 환경에서 사용될 수 있는 다른 제거가능/제거 불가능, 휘발성/비휘발성 컴퓨터 스토리지 매체는, 자기 테이프 카세트, 플래시 메모리 카드, 디지털 다용도 디스크, 다른 광디스크, 디지털 비디오 테이프, 솔리드 스테이트(solid state) RAM, 솔리드 스테이트 ROM 등을 포함한다. 하드 디스크 드라이브(141)는 전형적으로 인터페이스(140)와 같은 제거 불가능한 메모리 인터페이스를 통해서 시스템 버스(121)에 접속되고, 자기 디스크 드라이브(151) 및 광디스크 드라이브(155)는 전형적으로 인터페이스(150)와 같은 제거가능한 메모리 인터페이스에 의해 시스템 버스(121)에 접속된다.

[0020] 도 1에 도시되고 전술된 드라이브들 및 그와 연관된 컴퓨터 스토리지 매체들은, 컴퓨터 판독가능한 명령어들, 데이터 구조들, 프로그램 모듈들 및 컴퓨터(110)에 대한 다른 데이터의 스토리지를 제공한다. 도 1에서, 예를 들면, 하드 디스크 드라이브(141)는 운영 시스템(144), 애플리케이션 프로그램(145), 다른 프로그램 모듈들(146) 및 프로그램 데이터(147)를 저장하는 것으로 도시되었다. 이들 구성요소들은 운영 시스템(134), 애플리케이션 프로그램(135), 다른 프로그램 모듈들(136) 및 프로그램 데이터(137)와 동일할 수도 있고 다른 것일 수도 있음을 인식해야 한다. 본 명세서에서, 운영 시스템(144), 애플리케이션 프로그램(145), 다른 프로그램 모듈들(146) 및 프로그램 데이터(147)는 최소한, 운영 시스템(134), 애플리케이션 프로그램(135), 다른 프로그램 모듈들(136) 및 프로그램 데이터(137)와는 다른 복제물들이라는 것을 나타내고자 상이한 참조번호가 주어졌다.

[0021] 사용자는 키보드(162) 및 흔히 마우스, 트랙볼, 또는 터치 패드로 지칭되는 포인팅 디바이스(161)와 같은 입력 디바이스를 통해 커맨드(command) 또는 정보를 컴퓨터(110)에 입력할 수 있다. (도시되지 않은) 다른 입력 디바이스는 마이크로폰, 조이스틱, 게임 패드, 위성 접시, 스캐너, 터치 스크린, 입력 태블릿(writing tablet) 등을 포함할 수 있다. 이들 입력 디바이스들과 다른 입력 디바이스들은 종종 시스템 버스에 연결된 사용자 입력 인터페이스(160)를 통해 프로세싱 유닛(120)에 접속되지만, 병렬 포트, 게임 포트 또는 USB와 같은 버스 구조 및 다른 인터페이스를 통해서 접속될 수도 있다.

[0022] 모니터(191) 또는 다른 유형의 디스플레이 디바이스 또한 비디오 인터페이스(190)와 같은 인터페이스를 통해서 시스템 버스(121)에 접속된다. 모니터뿐 아니라, 컴퓨터는 출력 주변 인터페이스(195)를 통해 접속될 수 있는 프린터(196) 및 스피커(197)와 같은 다른 주변 출력 디바이스도 포함할 수 있다.

[0023] 컴퓨터(110)는 원격 컴퓨터(180)과 같은 하나 이상의 원격 컴퓨터에 대한 논리적 접속을 이용하여 네트워킹된 환경에서 동작할 수 있다. 원격 컴퓨터(180)는 개인 컴퓨터, 서버, 라우터, 네트워크 PC, 피어 디바이스(peer device) 또는 다른 일반적인 네트워크 노드일 수 있으며, 전형적으로 컴퓨터(110)와 관련하여 전술된 대다수의 요소들 또는 모든 요소들을 포함하지만, 도 1에는 오직 메모리 스토리지 디바이스(181)만이 도시되었다. 도 1에 도시된 논리적 접속들은 로컬 영역 네트워크(LAN)(171) 및 광역 네트워크(WAN)(173)를 포함하지만, 다른 네트워크들도 포함할 수 있다. 이러한 네트워킹 환경은 사무실, 전사적(enterprise-wide) 컴퓨터 네트워크, 인트라넷 및 인터넷에서 매우 흔하다.

[0024] LAN 네트워킹 환경에서 사용될 때, 컴퓨터(110)는 네트워크 인터페이스 또는 어댑터(170)를 통해서 LAN(171)에 접속된다. WAN 네트워킹 환경에서 사용될 때, 컴퓨터(110)는 모뎀(172) 또는 인터넷과 같은 WAN(173) 상에서의 통신을 확립하는 다른 수단을 포함할 수 있다. 내부 또는 외부 모뎀일 수 있는 모뎀(172)은, 사용자 입력 인터페이스(160) 또는 다른 적절한 메커니즘을 통해서 시스템 버스(121)에 접속될 수 있다. 네트워킹된 환경에서, 컴퓨터(110)와 관련하여 도시된 프로그램 모듈들, 또는 그 일부는 원격 메모리 스토리지 디바이스에 저장될 수 있다. 예를 들면, 도 1은 원격 애플리케이션 프로그램(185)이 메모리 디바이스(181) 상에 존재하는 것으로 도시하였지만, 이것으로 제한되는 것은 아니다. 도시된 네트워크 접속이 예시적인 것이며, 컴퓨터들 사이의 통신 링크를 확립하는 다른 수단이 사용될 수 있음을 이해할 것이다.

[0025] 체크포인팅(Checkpointing)

[0026] 전술된 바와 같이, 정전 및 시스템 고장이 스토리지 디바이스에 데이터를 기록하는 동안 발생할 수 있다. 이는

스토리지 디바이스 상에 저장된 데이터를 비일관적인 상태로 남겨둘 수 있다. 이러한 문제와 다른 문제들을 해결하기 위해, 체크포인트가 스토리지 디바이스에 기록될 수 있다.

[0027] 도 2는 본 명세서에 기술된 청구사항의 측면들이 동작할 수 있는 시스템의 구성요소들의 예시적인 배치를 나타낸 블록도이다. 도 2에 도시된 구성요소들은 예시적인 것으로, 필요하거나 포함될 수 있는 모든 구성요소들을 포함하는 것으로 해석되어서는 안된다. 다른 실시예들에서, 도 2와 관련하여 기술된 구성요소들 및/또는 기능들이 본 명세서에 기술된 청구사항의 측면들의 범주 또는 사상으로부터 벗어나지 않고 (도시되거나 도시되지 않은) 다른 구성요소에 포함될 수 있거나 또는 하위 구성요소 내에 배치될 수 있다. 일부 실시예들에서, 도 2와 관련하여 기술된 구성요소들 및/또는 기능들이 복수의 디바이스에 걸쳐 분산될 수도 있다.

[0028] 도 2를 참조하면, 시스템(205)은 하나 이상의 애플리케이션(210), API(215), 파일 시스템 구성요소(220), 스토어(250), 통신 메커니즘(255) 및 (도시되지 않은) 다른 구성요소들을 포함할 수 있다. 시스템(205)은 하나 이상의 컴퓨팅 디바이스를 포함할 수 있다. 이러한 디바이스들은, 예를 들면, 개인 컴퓨터, 서버 컴퓨터, 휴대용 또는 랩탑 컴퓨터, 멀티프로세서 시스템, 마이크로컨트롤러 기반의 시스템, 셋톱 박스, 프로그램가능한 소비자 전자기기, 네트워크 PC, 미니컴퓨터, 메인프레임 컴퓨터, 셀룰러 폰, PDA, 게임 디바이스, 프린터, 셋톱 및 미디어 센터를 포함하는 어플라이언스, 또는 그 외의 어플라이언스, 차량 장착 또는 부착된 컴퓨팅 디바이스, 그 외의 모바일 디바이스, 전술된 시스템들 또는 디바이스들 중 임의의 것을 포함하는 분산 컴퓨팅 환경 등을 포함할 수 있다.

[0029] 시스템(205)이 단일 디바이스를 포함하는 경우, 시스템(205)으로서 작업하도록 구성될 수 있는 예시적인 디바이스는 도 1의 컴퓨터(110)를 포함한다. 시스템(205)이 복수의 디바이스를 포함하는 경우, 복수의 디바이스의 각각은 유사하게 또는 서로 다르게 구성된 도 1의 컴퓨터(110)를 포함할 수 있다.

[0030] 파일 시스템 구성요소(220)는 복구 매니저(225), 체크포인트 매니저(230), I/O 매니저(235), 기록 플랜 매니저(write plan manager)(237) 및 도시되지 않은 다른 구성요소들을 포함할 수 있다. 본 명세서에서 사용되는 "구성요소"라는 용어는 디바이스 전체 또는 일부, 하나 이상의 소프트웨어 모듈의 컬렉션 또는 그 일부, 하나 이상의 소프트웨어 모듈 또는 그 일부와 하나 이상의 디바이스 또는 그 일부의 소정의 조합 등을 포함하는 것으로 읽혀져야 한다.

[0031] 통신 메커니즘(255)은 시스템(205)이 다른 엔티티들과 통신할 수 있게 한다. 예를 들어, 통신 메커니즘(255)은 시스템(205)이 원격 호스트 상에서 애플리케이션들과 통신하게 할 수 있다. 통신 메커니즘(255)은 네트워크 인터페이스 또는 어댑터(170), 모뎀(172), 또는 도 1과 관련하여 기술된 바와 같은 통신을 확립하기 위한 임의의 다른 메커니즘일 수 있다.

[0032] 스토어(store)(250)는 데이터에 대한 액세스를 제공할 수 있는 임의의 스토리지 매체이다. 스토어는 휘발성 메모리(예컨대, 캐시) 및 비휘발성 메모리(예컨대, 지속성 스토리지(persistent storage))를 포함할 수 있다. "데이터"라는 용어는 하나 이상의 컴퓨터 스토리지 소자에 의해 표현될 수 있는 임의의 것을 포함하도록 넓은 범위로 해석되어야 한다. 논리적으로, 데이터는 휘발성 또는 비휘발성 메모리에서 일련의 1과 0으로 표현될 수 있다. 비이진식(non-binary) 스토리지 매체를 구비하는 컴퓨터들에서, 데이터는 스토리지 매체의 능력에 따라 표현될 수 있다. 데이터는 숫자, 문자 등과 같은 단순한 데이터 타입, 계층적 데이터 타입, 링크된 데이터 타입, 또는 그 외의 연관성 데이터 타입, 복수의 다른 데이터 구조들 또는 단순한 데이터 타입을 포함하는 데이터 구조 등을 포함하는 서로 다른 타입의 데이터 구조들로 조직화될 수 있다. 데이터의 일부 예는 정보, 프로그램 코드, 프로그램 상태, 프로그램 데이터, 그 외의 데이터 등을 포함한다.

[0033] 스토어(250)는 하드 디스크 스토리지, 다른 비휘발성 스토리지, RAM과 같은 휘발성 메모리, 다른 휘발성 스토리지, 이들의 소정의 조합 등을 포함할 수 있으며, 복수의 디바이스에 걸쳐 분산될 수도 있다. 스토어(250)는 외부 스토어이거나 내부 스토어일 수 있고, 또는 시스템(205)에 대해 내부에 있는 구성요소와 외부에 있는 구성요소를 모두 포함한다.

[0034] 스토어(250)는 스토리지 컨트롤러(240)를 통해 액세스될 수 있다. 본 명세서에서 사용되는 "액세스"라는 용어는 데이터 판독, 데이터 기록, 데이터 삭제, 데이터 업데이트, 이 중 둘 이상을 포함하는 조합 등을 포함할 수 있다. 스토리지 컨트롤러(240)는 스토어(250)에 액세스하라는 요청을 수신할 수 있고 이러한 요청을 적절하게 충족시킬 수 있다. 스토리지 컨트롤러(240)는 데이터가 수신된 순서대로 스토어(250)에 기록되는 것을 보장하지 않도록 구성될 수도 있다. 또한, 스토리지 컨트롤러(240)는 자신이 스토어(250)의 비휘발성 메모리에 데이터를 실제로 기록하기 이전에, 요청된 데이터를 기록하였다고 나타낼 수도 있다.

[0035] 하나 이상의 애플리케이션(210)은 데이터의 생성, 데이터 삭제 또는 데이터 업데이트와 관련될 수 있는 임의의 프로세스를 포함한다. 이러한 프로세스들은 사용자 모드 또는 커널 모드(kernel mode)에서 실행할 수 있다. 본 명세서에서 사용되는 "프로세스"라는 용어 및 이로부터 변형된 형태의 용어들은 하나 이상의 통상적인 프로세스, 스레드(thread), 컴포넌트, 라이브러리, 태스크를 수행하는 객체 등을 포함할 수 있다. 프로세스는 하드웨어, 소프트웨어, 또는 하드웨어와 소프트웨어의 조합에서 구현될 수 있다. 하지만 실시예에서, 프로세스는 작업을 수행할 수 있거나 수행하는데에 사용되는 임의의 메커니즘을 지칭한다. 프로세스는 복수의 디바이스 또는 단일 디바이스에 걸쳐 분산될 수 있다. 하나 이상의 애플리케이션(210)은 API(215)를 통해서 I/O 매니저(235)로 (예컨대, 기능/방법 호출에 의한) 파일 시스템 요청을 할 수 있다.

[0036] I/O 매니저(235)는 어떤 I/O 요청 또는 요청들을 스토리지 컨트롤러(240)(또는 일부 다른 중간 구성요소)에게 발행해야 할지를 결정할 수 있다. I/O 매니저(235)는 또한 데이터를 파일 시스템 요청 진행, 완료, 또는 실패와 연관된 동작들로서 하나 이상의 애플리케이션(210)에 반환할 수 있다. 파일 시스템 요청이 트랜잭션을 포함하는 경우, I/O 매니저(235)는 (도시되지 않은) 트랜잭션 매니저가 적절하게 트랜잭션을 관리할 수 있도록 트랜잭션 매니저에게 알려줄 수 있다. 일부 실시예들에서, 트랜잭션 매니저의 기능은 I/O 매니저(235)에 포함될 수도 있다.

[0037] 파일 시스템 구성요소(220)는 파일 시스템 객체들 또는 파일 시스템 객체들과 관련된 메타데이터를 스토어(250)에 기록할 때, Copy-on-Write, Write-in-Place 및 이들의 조합을 사용할 수 있다. "파일"이라는 용어는 디렉토리, 하위객체(children)를 갖지 않는 파일 시스템 객체(예컨대, 때때로 파일로서 생각되는 것), 다른 파일 시스템 객체 등을 포함할 수 있다.

[0038] Copy-on-Write 시에, 파일의 데이터가 수정되기 전에, 수정될 데이터의 카피(copy)가 다른 위치로 복사된다. Write-in-Place 시에, 파일의 데이터가 다른 위치에 오리지널 데이터를 복사하지 않은 채 그 자리에서 수정될 수 있다. Copy-on-Write 및 Write-in-Place의 혼합은 파일과 관련된 메타데이터에 대해 Copy-on-Write를 수행하는 것과 동시에 파일 내에 포함된 데이터에 대해 Write-in-Place를 수행하는 것을 포함할 수 있다.

[0039] 파일 시스템의 객체들은 트랜잭션의 맥락으로 업데이트될 수 있다. 트랜잭션은 예컨대 원자성(atomic), 일관성(consistent), 고립성(isolated) 및 내구성(durable)을 포함하는 다양한 속성들에 의해 기술될 수 있는 동작들의 그룹이다. 본 명세서에서 사용되는 "트랜잭션(transaction)"이라는 용어는 적어도 일관성의 속성에 의해서 정의될 수 있으며, 전술된 다른 속성들 중 하나 이상에 의해서 정의될 수도 있다.

[0040] 일관성 속성은 하나 이상의 파일에 대해서 데이터의 허용 상태(allowed state)를 지칭한다. (트랜잭션 중에는 비허용 상태를 거치더라도) 트랜잭션이 시작하기 전과 트랜잭션이 완료된 후에, 파일 시스템의 파일들은 허용 상태에 있게 된다. 예를 들어, 은행 송금은 하나의 계좌로부터의 인출 및 다른 계좌로의 입금을 포함하는 두 동작의 세트로서 실행될 수 있다. 이 예시에서의 일관성은 은행과 계좌 소유자의 통합된 계좌 잔고가 상수를 갖는 것으로서 정의될 수 있다(예컨대, T=A+B, 이때 T는 상수, A=은행 잔고, B=계좌 소유자 잔고). 이 예시에서의 일관성을 구현하기 위해서, 인출 및 입금 동작들은 동일한 금액에 대해 수행되어야 하고 각 계좌 상에서 두 동작 모두가 완료되거나 또는 두 동작 모두가 완료되지 않아야 한다.

[0041] 체크포인트는 파일 시스템의 일관성 상태(consistent state)를 나타내도록 기록될 수 있다. 체크포인트는 체크포인트 및/또는 체크포인트와 연관된 데이터가 디스크에 올바르게 기록되었는지 여부를 판정하는데에 사용될 수 있는 하나 이상의 확인 코드(validation code)(예컨대, 하나 이상의 체크섬, 해시, 또는 다른 데이터)를 포함할 수 있다. 복구를 할 때, 마지막으로 기록된 체크포인트의 위치를 찾을 수 있다. 그 다음 체크포인트의 확인 코드(들)는 체크포인트 및/또는 체크포인트와 연관된 데이터가 디스크에 올바르게 기록되었는지 여부를 판정하는데에 사용될 수 있다. 만약 그렇지 않다면, 유효한 체크포인트가 발견될 때까지 이전의 체크포인트의 위치를 찾아서 유효성을 검사할 수 있다. 가장 최근의 유효 체크포인트가 발견되면, 파일 시스템의 마지막 일관성 상태가 알려진다. 이 포인트 이후에 발생한 파일 시스템 동작들은 폐기될 수 있거나 또는 원한다면 추가적인 복구 작업들이 수행될 수 있다.

[0042] 일 실시예에서, 파일 시스템 상의 객체는 D_n으로 표기될 수 있으며, 이때 n은 시스템에 대한 객체를 식별한다.

파일 시스템 상의 객체들은 직렬화가능하거나(serializable)(예컨대, 스토어(250) 상의 데이터로서 표현될 수 있음) 역직렬화가능하다(deserializable). 객체 테이블은 각 객체 식별자를 스토어(250) 상의 그의 위치에 연관시킨다.

[0043] 처음으로 D_n이 수정 트랜잭션에서 업데이트되면, D_n은 n을 이용하여 객체 테이블 내의 자신의 위치를 찾음으로써

발견된다. 예를 들면, 스토어(250) 상의 D_n의 스토리지 위치는 L_1로 지칭된다.

[0044] 그 다음 L_1의 콘텐츠가 스토어(250)로부터 판독되고, 객체는 역직렬화될 수 있으며(예컨대, 직렬화된 포맷으로부터 객체의 구조로 변환됨), 수정될 객체의 일부분들이 메인 시스템 메모리로 복사된다. 업데이트들은 메모리 내의 일부분들(또는 그들의 카피들) 상에서 수행된다. 메모리 내의 일부분들이 수정되는 것과 관련하여, 스토어(250) 상에 하나 이상의 새로운 위치들(L_2로 지칭됨)이 수정된 일부분들을 위해 지정된다.

[0045] 메인 시스템 메모리 내의 이러한 카피들은 본 명세서에서 때때로 객체들의 "논리적 카피(logical copy)"로 지칭된다. 객체의 논리적 카피는 객체를 나타내는데에 사용될 수 있는 하나 이상의 데이터 구조를 포함한다. 논리적으로, 논리적 카피는 객체의 사본이다. 물리적으로, 논리적 카피는 객체의 사본을 생성하는데에 사용될 수 있는 (다른 데이터에 대한 포인터를 포함하는) 데이터를 포함한다. 예를 들어, 일 구현에서, 논리적 카피는 객체의 실제 카피(예컨대, 비트-바이-비트 카피)이거나 또는 객체를 생성하는데에 사용될 수 있는 데이터를 포함하는 데이터 구조일 수 있다.

[0046] 다른 구현에서, 수정되지 않은 논리적 카피가 오리지널 객체를 참조하는 하나 이상의 포인터를 포함할 수 있다. 논리적 카피가 수정되면, 논리적 카피 내의 포인터는 (예컨대, 논리적 카피의 변경된 일부분에 있어서) 새로운 메모리 위치를 참조할 수 있는 반면, 다른 포인터들은 (예컨대, 논리적 카피의 변경되지 않은 일부분에 있어서) 오리지널 객체의 일부분을 참조할 수 있다. 포인터를 이용하여, 수정된 카피는 수정된 데이터와 함께 오리지널 객체의 수정되지 않은 데이터를 사용하여 구성될 수 있다. 예를 들면, 객체의 사본을 생성하는데에 필요한 스토리지를 감소시키기 위해서 논리적 카피의 생성이 수행될 수 있다.

[0047] 또한, 본 명세서에서 직렬화 및 역직렬화가 언급되었지만, 본 명세서에 기술된 청구사항의 측면들이 관례적으로 직렬화 및 역직렬화로 생각되는 것으로 제한하는 것은 아니다. 일 실시예에서, 직렬화된 버전은 역직렬화된 버전과 매 비트마다 동일할 수 있다. 다른 실시예에서, 직렬화된 버전의 비트들은 역직렬화된 버전의 비트들과는 다른 포맷과 순서로 패키징될 수 있다. 실제로, 일 실시예에서, 직렬화 및 역직렬화는 스토어로부터의 객체들을 나타내는 데이터를 저장 및 검색하는 임의의 메커니즘을 의미하는 것으로 이해되어야 한다. 다른 메커니즘은, 예를 들어, 객체들의 속성을 텍스트 포맷으로 스토어에 기록하는 것, 객체들의 속성을 마크업(markup) 언어로 스토어에서 인코딩하는 것, 스토어 상의 객체들의 속성들과 다른 특징들을 저장하는 것, 등을 포함할 수 있다.

[0048] 시스템의 재량에 따라(예컨대, 트랜잭션 실행 후에 또는 일부 다른 시간에), 시스템은 수정된 논리적 카피를 안정된 매체의 위치 L_2에 다시 직렬화할 수 있다. 수정된 논리적 카피를 새로운 위치에 다시 기록하려는 것을 기록 플랜(write plan)이라고 지칭한다. 기록 플랜은 하나 이상의 객체에 대한 임의의 수의 업데이트들을 알아보게 할 수 있다. 기록 플랜은 둘 이상의 트랜잭션에서 발생하는 변화들을 참조할 수 있다. 복수의 기록 플랜이 단일 기록 플랜으로 결합될 수도 있다.

[0049] 기록 플랜 매니저(237)는 다양한 업데이트들에 대한 기록 플랜을 생성하는 것과 관련될 수 있다. 기록 플랜이 복수의 파일 시스템 객체를 포함할 때(예컨대, 트랜잭션의 맥락에서), 기록 플랜 매니저(237)는 파일 시스템에 대한 일관성 상태를 유지하기 위해 트랜잭션에 관련된 모든 파일 시스템 객체들의 스토리지 상의 위치들을 나타내는 기록 플랜을 생성하도록 동작할 수 있다.

[0050] 체크포인트 직후에 수정가 발생하는 경우, (복수의 위치에서 복제될 수 있는) 복구 블록(recovery block)이라 불리는 블록이 수정된 논리적 카피의 시작점(예컨대, L_2)을 가리키도록 수정될 수 있다. L_2에서의 객체 내의 필드는 다음에 기록될 위치를 가리킨다. 이러한 필드는 체크포인트들 사이에서 발생하는 기록 플랜들의 사슬 중 하나의 고리(a link in a chain of write plans)를 나타낸다.

[0051] 논리적 카피를 기록하라는 요청을 전송하는 것과 관련해서, 객체 테이블이 수정될 수 있다. 특히, 객체의 식별자에 의해 인덱스된 위치값은 수정된 논리적 카피가 저장될 위치(즉, L_2)의 위치값으로 설정될 수 있다. 이에 따라 객체 D_n의 후속하는 위치 검색이 객체의 새로운 버전인 위치 L_2를 참조할 것이다.

[0052] 만약 트랜잭션이 예컨대 D_i 및 D_j의 둘 이상의 객체를 수정한다면, 객체들은 서로 "원자적으로 묶인(atomically bound)" 것으로 간주되어, 하나의 기록 플랜에 기록된다. 기록 플랜은 (예컨대, 관련된 객체들에 대한 고리들에서) 이러한 관계를 식별한다.

[0053] 임의의 수의 객체들은 이러한 방식으로 지속될 수 있다. 주기적으로, 객체 테이블 또한 임의의 다른 객체와 동일한 방식으로 스토어(250)에 기록될 수 있다.

[0054] 객체 테이블을 스토어(250)에 기록하라는 요청을 전송하는 것과 관련하여, 플러쉬 커맨드(flush command)도 스토리지 컨트롤러(240)에 전송될 수 있다. 플러쉬 커맨드는 스토리지 컨트롤러(240)가 스토어(250)의 비휘발성 메모리에 이미 기록되지 않은 자신의 휘발성 메모리로부터의 모든 데이터를 기록하도록 명령한다.

[0055] 주기적으로, 체크포인트는 아래에서 보다 자세히 기술되는 바와 같이 스토리지에 기록될 수 있다. 체크포인트는 스토어(250)에 의해 저장되는 체크포인트 기록에 의해 표시될 수도 있다. 체크포인트는 어떠한 시간에도 기록될 수 있으며, 플러쉬 후에는 안정적이고 내구성 있게 될 수 있다. 안정적/내구성 있다는 것은 스토어의 비휘발성 메모리 상에 저장된 체크포인트를 지칭한다.

[0056] 체크포인트가 안정적/내구성 있게 된 후에, 임의의 오래되고 사용되지 않은 객체들의 카피들에 대해 사용된 공간이 재사용될 수 있다. 플러쉬가 완료된 후에, 복구 블록이 다음 기록 플랜들의 사슬의 시작점을 가리키게 된다. 일 실시예에서, 복구 블록은 기록 플랜들의 사슬의 시작점이 객체 테이블의 새로운 위치를 가리키게 할 수 있다.

[0057] 보다 구체적인 예시가, 본 명세서에 기술된 청구사항의 측면들을 도시한 블록도인 도 3과 관련하여 기술되었다. 도시된 바와 같이, 도 3은 메인 메모리(305) 및 스토어(250)를 나타낸다. 라인(307)은 메인 메모리(305)와 스토어(250) 사이의 분할을 나타낸다. 라인(307) 위의 객체들은 메인 메모리에 있는 반면, 라인(307) 아래의 객체들은 스토어(250)의 휘발성 또는 비휘발성 메모리에 있다.

[0058] 객체들(314-316)은 메인 메모리(305) 내에 도시되었다. 구현에서, 객체들(314-316)은 각각 객체들(319-321)의 역직렬화된 논리적 카피들일 수 있다. 객체(319)는 스토어(250) 상의 위치(1550)에 위치되어 있고, 객체(320)는 스토어(250) 상의 위치(200)에 위치되어 있으며, 객체(321)는 스토어(250) 상의 위치(800)에 위치되어 있다.

[0059] 객체 테이블(310)은 스토어(250) 상의 객체들(319-321)의 위치를 나타내는 키값의 쌍들(key value pairs)을 포함한다. 키값의 쌍들은 객체(319-321)의 식별자(n)를 이용하여 인덱스된다.

[0060] 트랜잭션이 객체(316)를 (예컨대 객체(316)의 네임(name)을 foo.txt로 변경함으로써) 수정시킬 때, 일관성 구성요소(예컨대, 도 2의 일관성 구성요소(220))가 업데이트된 객체를 위한 새로운 스토리지 위치(예컨대, 위치(801))를 결정할 수 있다. 만약 그 객체가 파일이라면, 트랜잭션의 맥락에서 파일의 이름을 업데이트하는 것은 그 파일을 포함하는 디렉토리 또한 트랜잭션에 포함되게 할 수 있다. 예를 들면, 파일명이 변경되었을 때, 그 파일을 나타내는 객체와 그 파일을 포함하는 디렉토리를 나타내는 개체 모두가 트랜잭션에 포함되어야만 할 수 있다. 이러한 경우에, 객체를 포함하는 디렉토리는 객체(314)로서 표현되고, 업데이트된 디렉토리(예컨대, 객체(318))의 논리적 카피는 스토어(250) 내에 객체(323)로서 표현된다. 또한, 테이블(310)은 수정된 객체들(즉, 객체(317, 318))의 새로운 스토리지 위치들(즉, 위치(801, 1000))을 나타내기 위해서 테이블(311)로 논리적으로 업데이트되었다.

[0061] 트랜잭션의 맥락에서의 객체의 수정가 다른 객체에도 영향을 미친다는 사실은, 예컨대 도 2의 I/O 매니저(235) 또는 일부 다른 구성요소에 의해서 명백하게 표시되거나 결정될 수 있다.

[0062] 둘 이상의 객체들이 트랜잭션의 업데이트에 연관되는 경우에, 객체들은 앞서 언급된 바와 같이 "원자적으로 묶인" 것으로 간주된다. 복구 동작에서, 트랜잭션의 맥락에서 변경된 모든 객체들에 대한 변화들이 스토어(250) 내에서 발견되지 않는 한, 발견된 모든 변화들은 폐기된다. 다시 말하면, 만약 객체들 중 하나에 대한 변화가 발견되었지만 다른 객체들에 대한 변화가 발견되지 않았다면, 객체들 중 하나에 대한 변화는 폐기된다.

[0063] 둘 이상의 객체들을 원자적으로 묶기 위해서, 일 실시예에서, 포인터가 저장되거나, 그외의 방법으로 스토어(250) 내의 각 객체와 연관될 수 있다. 포인터는 트랜잭션에 연관된 다른 객체(또는 그의 일부분)의 스토리지 위치를 나타낼 수 있다. 만약 트랜잭션에 추가적인 객체가 없다면, 포인터는 "데드 블록(dead block)"을 가리키거나 또는 다른 기록 플랜의 "헤드(head)" 객체의 스토리지 위치를 나타낼 수 있다. 이러한 헤드 객체는 기록 플랜, 기록 플랜의 수정된 객체(또는 그의 일부분) 등을 포함할 수 있다.

[0064] 다음 스토리지 위치로의 포인터뿐만 아니라, "포인팅된" 객체의 올바른 콘텐츠를 나타내기 위해 데이터도 스토어(250)에 저장될 수 있다. 예를 들면, 포인팅된 객체의 올바른 콘텐트를 나타내는 해시(hash)가 저장될 수 있다.

[0065] 도 3에서 나타낸 예시에서, 객체(322)와 연관된 포인터는 객체(323)와 연관된 스토리지 위치를 가리킬 수 있다.

포인터는 두 객체들을 함께 묶는다. 만약 복구 중에 이들 객체 중 하나가 발견되지 않거나 이들 객체가 올바른 콘텐트를 갖지 않으면, 발견된 객체들에 의해 표시되는 변화들은 폐기될 수 있다.

[0066] 스토어(250)의 성질로 인해, 어느 객체가 스토어(250)의 비휘발성 메모리에 처음으로 기록될 것인지에 대한 보장이 없을 수 있다. 만약 객체(322)가 먼저 기록되었고 객체(323)가 기록되지 않았다면, 객체(322)로부터의 포인터는 거짓(spurious) 데이터를 가질 수 있는 스토리지 위치를 가리키게 될 것이다. 그러나, 그 스토리지 위치에서 데이터의 해시를 계산하고 계산된 해시를 객체(322)와 저장된 해시와 비교함으로써, 위치(1000)에서의 데이터는 유효하지 않은 데이터를 갖는 것으로 검출될 수 있다. 이러한 경우에, 복구하는 동안 복구 매니저(예컨대, 도 2의 복구 매니저(225))가 객체들(322, 323)에 의해 나타내어지는 변화들을 폐기할 수 있다.

[0067] 복구 블록(330)은 체크포인트 이후에 데이터가 저장된 것으로 추정되는 제 1 스토리지 위치(이 경우에는 위치(801))를 가리킨다. 복구 블록(330)은 또한 제 1 스토리지 위치에 저장된 객체의 올바른 콘텐츠를 이용하여 계산된 해시를 포함하거나 해시와 연관될 수 있다.

[0068] 도 4는 본 명세서에 기술된 청구사항의 측면들에 따라 파일 시스템 상에서 발생하는 업데이트를 일반적으로 나타낸 도면이다. 글로벌 테이블(405)은 스토어 상의 객체들의 위치를 식별하는 객체 테이블 및 할당된 스토어(250) 상의 공간과 관련된 할당 데이터를 포함한다. 진행 중인 업데이트들(410)도 도시되었다. 업데이트가 시간 축(415)을 만날 때 업데이트가 완료되며, 더 이상 어떠한 글로벌 테이블(405)도 수정할 필요가 없다. 업데이트들(410)의 각각의 업데이트 라인이 복수의 업데이트를 나타낼 수 있다. 일관성을 유지하기 위해서 복수의 업데이트가 함께 이루어져야 할 경우, 복수의 업데이트는 트랜잭션의 맥락에서 이루어질 수 있다.

[0069] 체크포인트가 유효하기 위해서, 체크포인트는 일관성 상태에서 기록되어야만 한다. 기록 파일 시스템 상의 카피를 이용해서, 객체가 업데이트될 때 수정되는 객체의 논리적 카피가 파일 시스템의 새로운 위치에 저장된다. 이러한 새로운 위치는 객체 테이블에 대한 업데이트를 통해 객체 테이블 내에 반영된다. 일관성에 있어서는, 업데이트가 시스템 고장 전에 디스크에 완전하게 기록되지 않았을 수 있기 때문에, 디스크에 아직 기록되지 않은 업데이트를 객체 테이블에 반영하는 것이 올바르지 않을 수 있다. 유사하게, 업데이트가 완료되어 디스크에 기록되고 다른 트랜잭션-관련 업데이트들이 완료되지만 객체 테이블이 업데이트를 나타내지 않는 것 또한 올바르지 않을 수 있다.

[0070] 일관성을 보장하기 위해서, 체크포인트는 업데이트를 위한 메타데이터가 글로벌 테이블 내에 반영될 때에 선택되어야만 한다. 만약 업데이트들(410)을 나타내는 각각의 선들이 글로벌 테이블(405)이 그 업데이트를 위해서 업데이트될 수 있는 기간을 나타낸다면, 시간(520)에서 체크포인트를 수행하는 것은 비일관성 상태를 산출할 수 있지만 시간(525)에 체크포인트를 수행하는 것은 일관성 상태를 산출할 것이다.

[0071] 도 5는 본 명세서에 기술된 청구사항의 측면들에 따른 예시적인 체크포인트 버킷들을 도시한 블록도이다. 전술된 이슈들과 그 외의 이슈들을 해결하기 위해서, 각 업데이트가 체크포인트 버킷(bucket)(예컨대, 버킷들(515) 중 하나)과 연관될 수 있다. 체크포인트 버킷은, 체크포인트의 체크포인트 데이터가 디스크에 기록되기에 앞서, 적어도 체크포인트 버킷와 연관된 업데이트들의 기록 플랜들을 설명하기 위해서 글로벌 테이블이 업데이트되어야 한다는 것을 나타내는 논리적 개념이다. 다시 말하면, 업데이트들이 어떤 위치에 현재 기록되었거나 또는 기록되지 않았더라도 글로벌 테이블은 버킷의 업데이트들의 위치 및 할당 정보를 설명하기 위해 업데이트되어야 한다.

[0072] 주기적으로(예컨대, 복구 윈도우에 기초하여 체크포인트 타이머가 만료되었을 때, 소정의 수의 기록들이 발생된 후, 일부 다른 임계값이 초과된 후, 등), 체크포인트를 생성하라는 결정이 내려질 수 있다. 체크포인트 생성 결정이 내려지면, 체크포인트 매니저는 체크포인트 버킷을 나타내는 데이터(예컨대 데이터 구조(510))가 후속하는 업데이트와 연관되도록 업데이트할 수 있다. 예를 들면, 체크포인트 매니저는 현재의 체크포인트 버킷을 나타내는 데이터(예컨대, 데이터 구조(510))에 대한 배타적 락(exclusive lock)(예컨대, 락(505))을 획득할 수 있다. 체크포인트 매니저가 데이터에 대한 배타적 락을 획득한 후에, 체크포인트 매니저는 후속하는 업데이트들에 대한 새로운 체크포인트 버킷을 나타내도록 데이터를 업데이트할 수 있다. 데이터가 후속하는 업데이트들에 대한 다른 체크포인트 버킷을 나타내도록 변경될 때까지 모든 후속하는 업데이트들은 새로운 체크포인트 버킷과 연관된다.

[0073] 체크포인트 버킷은 논리적 개념으로 생각될 수 있으며 다양한 방식으로 구현될 수 있다. 예를 들어 일 구현에서, 체크포인트 버킷은 체크포인트 버킷과 연관된 각각의 업데이트로의 포인터를 구비하는 리스트와 같은 데이터 구조로서 구현될 수 있다. 다른 예시로서, 체크포인트 버킷은 각 업데이트에 대해 유지되는 데이터로서 구

현될 수 있으며, 이러한 데이터는 그 업데이트와 연관된 체크포인트를 나타낸다. 다른 예시로서, 체크포인트 버킷은 계수 세마포어(counting semaphore)로서 구현될 수 있다. 이러한 예시에서, 어떤 업데이트들이 디스크에 기록되어야 하는지는 알려지지 않을 수 있지만, 여전히 디스크에 기록되어야만 하는 업데이트들의 수는 알려진다. 판독/기록 락이 이러한 예시에서 사용될 수도 있다.

[0074] 전술된 예시들은 체크포인트 버킷을 구현하는 포괄적인 또는 독점적인 방식을 의도하는 것은 아니다. 실제로, 본 명세서에서의 내용에 기초하여, 당업자는 체크포인트 버킷을 구현하는 다수의 다른 메커니즘을 인식할 수 있을 것이다.

[0075] (예컨대 데이터 구조(510)를 변경함으로써) 후속하는 업데이트들에 대한 체크포인트 버킷을 나타낸 후에, 체크포인트 매니저는 현재 체크포인트 버킷 내의 모든 업데이트들에 대한 기록 플랜들이 생성되기를 기다릴 수 있다. 현재 체크포인트 버킷 내의 모든 업데이트들에 대한 기록 플랜들이 생성된 후에(그러나 어쩌면 스토리지에는 기록되지 않았을 경우에), 체크포인트 매니저는 도 4의 글로벌 테이블(405)의 스냅샷(snapshot)을 찍고, 글로벌 테이블(405)의 스냅샷을 스토어에 기록하도록 기록 플랜을 생성할 수 있다. 스냅샷은 Copy-on-Write 또는 다른 메커니즘을 통해서 글로벌 테이블(405)의 논리적 카피로서 생성될 수 있다.

[0076] 도 4를 다시 참조하면, 체크포인트 매니저는 현재 체크포인트 버킷 내의 모든 업데이트들이 생성되기를 기다리고, 체크포인트 매니저가 체크포인트를 기록하기 위한 기록 플랜을 생성하는 동안, 체크포인트에 후속하는 업데이트들에 대한 기록 플랜들이 생성되어 디스크에 기록될 수 있다. 그러나, 체크포인트 매니저가 글로벌 테이블의 스냅샷을 획득하기 위해 찾을 때, 체크포인트 매니저는 스냅샷 생성 이전에 글로벌 테이블(405)에 대한 배타적 락(exclusive lock)을 획득할 수 있다. 체크포인트 매니저가 배타적 락을 갖는다 해도, 다른 업데이트들에 대한 기록 플랜들이 여전히 생성될 수 있고 이러한 기록 플랜들은 스토어 상에 저장될 수도 있지만, 체크포인트 매니저가 자신의 배타적 락을 해제하기 전까지는 글로벌 테이블(예컨대, 객체 테이블)이 이러한 기록 플랜들을 가리키도록 업데이트될 수 없다. 락의 해제와 관련하여, 체크포인트 매니저는 후속하는 체크포인트가 이용가능하게 되었으며 후속하는 업데이트들이 글로벌 테이블을 업데이트할 수 있음을 나타내는 신호(예컨대, Raise an event)를 전송할 수 있다.

[0077] 복구를 돕기 위해, 체크포인트는 아래의 규칙에 따라 체크포인트를 확인하는 확인 코드와 함께 디스크에 기록될 수 있다:

[0078] 1. 디스크에 기록될 기록 플랜들에 의해 표시되는 데이터를 기다린다(예컨대, 디스크에 기록될 체크포인트와 연관된 모든 업데이트들을 기다린다);

[0079] 2. 체크포인트와 연관된 모든 데이터가 디스크에 기록되도록 요청한다(예컨대, 메타데이터의 논리적 카피가 디스크에 기록되도록 요청한다);

[0080] 3. 플러쉬를 발행 또는 대기하고, 플러쉬가 성공적으로 완료되었다는 답신을 기다린다.

[0081] 4. 디스크에 기록된 체크포인트 데이터에 대한 확인 코드를 생성한다. 일 실시예에서, 확인 코드는 디스크에 기록된 데이터의 서브셋(subset)에 대한 것일 수 있다. 예를 들어, 만약 트리의 각 노드가 자신의 자식(children)에 대한 확인 코드를 포함하는 트리에 파일에 대한 데이터가 저장된다면, 확인 코드는 트리의 루트 노드(root node)에 대한 것일 수 있다. 이러한 실시예에서, 확인 코드는 루트 노드와 함께 기록될 수 있고 또한 확인 코드가 올바르다는 것을 증명하는데에 사용될 수 있다.

[0082] 5. 확인 코드(및 루트 노드와 같은 임의의 연관된 데이터)가 디스크에 기록되도록 요청한다. 확인 코드는 시스템 고장 전에 실제로 디스크에 도달하지 않을 수도 있음을 인지해야 한다. 만약 그렇지 않으면, 체크포인트는 유효한 체크포인트가 아니다.

[0083] 이들 규칙들에 따르면, 복구 중에 만약 체크포인트가 스토리지 상에서 발견되고 체크포인트의 내부 확인 코드가 유효하다면, 체크포인트와 연관된 다른 데이터가 스토리지 상에 저장되었고 그것이 유효할 것이 기대된다. 만약 확인 코드가 루트 노드에 포함되었다면, 루트 노드 내의 다른 데이터(예컨대, 트리 내의 다른 노드들로의 포인터)가 체크포인트에 상응하는 나머지 데이터를 발견하는데에 사용될 수 있다.

[0084] 대안으로, 체크포인트와 연관된 각 업데이트에 대한 확인 코드가 스토리지에 기록될 수 있다. 예를 들면, 체크포인트는 그 체크포인트에 앞서, 그리고 이전의 체크포인트 이후에 발생하도록 되어 있던 모든 업데이트들의 블록들을 나타낼 수 있다. 나타내어진 각각의 블록에 대해서, 체크포인트는 블록의 올바른 콘텐츠를 나타내는 확인 코드를 저장할 수 있다. 이러한 대안에서의 복구 중에, 체크포인트를 인증하기 위해서 각 블록이 그 체크포

인트의 연관된 확인 코드에 대해 확인될 수 있다.

[0085] 도 2를 참조하면, 일 실시예에서, 체크포인트 매니저(230)는 아래의 작업들을 수행하도록 동작할 수 있다:

[0086] 1. 파일 시스템 객체들을 업데이트하라는 요청들과 연관시키도록 제 1 체크포인트를 결정한다. 전술된 바와 같이, 체크포인트 매니저(230)는 데이터 구조(예컨대, 도 5의 데이터 구조(510))가 새로운 체크포인트 버킷을 가리키도록 업데이트함으로써 이것을 수행할 수 있다. 그 다음 각각의 후속하는 업데이트 요청이 수신됨에 따라, 이러한 요청이 새로운 체크포인트 버킷에 할당될 수 있다.

[0087] 본 명세서에서 사용되는 "제 1의"라는 용어는 가장 첫 번째 체크포인트를 의미하는 것이 아니며, "제 2의" 체크포인트로부터 구분하도록 사용된 것이다. 다시 말하면, N개의 체크포인트들이 존재하는 경우, 제 1 체크포인트는 임의의 X일 수 있으며, 이때 $1 \leq X \leq N$이고, 제 2 체크포인트는 임의의 Y일 수 있으며, 이때 $1 \leq Y \leq N$이고 $X<Y$ 또는 $X>Y$이다.

[0088] 2. 파일 시스템의 스토리지에 체크포인트와 연관된 체크포인트 데이터를 언제 기록할지를 결정한다. 예를 들어, 체크포인트 타이머가 만료될 수 있거나, 업데이트의 수가 초과될 수 있거나, 또는 일부 다른 임계값이 체크포인트 데이터를 기록할 시간임을 결정하는데에 사용될 수 있다.

[0089] 3. 파일 시스템 객체들을 업데이트하라는 후속 요청들에 대한 제 2 체크포인트를 결정한다. 전술된 바와 같이, 체크포인트 매니저(230)는 데이터 구조의 배타적 락(예컨대, 락(505))을 획득한 후에 데이터 구조(예컨대, 도 5의 데이터 구조(510))를 업데이트함으로써 이것을 수행할 수 있다.

[0090] 4. 파일 시스템의 일관성 상태(consistent state)를 기다리는 한편, 후속 요청들을 위해 데이터 기록을 준비하도록 한다. 일관성 상태는 현재 체크포인트 버킷과 연관된 모든 업데이트들이 스토리지 상에 나타날 때 (예컨대, 스토리지에 성공적으로 기록되었을 때) 발생한다. 후속 요청들을 위해 데이터 기록을 준비하도록 하는 것은, 기록 플랜들이 생성되어 후속 요청들을 위해 스토리지 상에 기록되는 것을 허용하지만 메타데이터(예컨대, 글로벌 테이블)의 논리적 카피가 생성되기 전까지는 메타데이터가 업데이트되는 것을 허용하지 않는 것을 포함한다.

[0091] 5. 파일 시스템의 메타데이터의 논리적 카피를 생성한다. 이것은 전술된 것처럼 글로벌 테이블의 스냅샷을 찍음으로써 수행될 수 있다.

[0092] 6. 메타데이터의 논리적 카피를 스토리지에 기록한다. 일 실시예에서, 이것은 논리적 카피가 스토리지에 기록되도록 요청하고 논리적 카피가 스토리지에 기록되었다는 컨펌(confirmation)을 기다리는 것을 포함할 수 있다. 다른 실시예에서, 메타데이터의 논리적 카피를 스토리지에 기록하는 것은, 업데이트들을 허용하기 이전에 메타데이터에 대한 후속 업데이트들이 Copy-on-Write를 발생시키도록, 스토리지 상의 카피가 깨끗하다고 마킹하는 것을 포함할 수 있다.

[0093] 7. 스토리지에 적어도 하나의 확인 코드를 기록한다. 전술된 바와 같이, 확인 코드는 체크포인트 이전의 업데이트가 스토리지에 기록되었는지 여부와 체크포인트 기록 자신이 유효한지 여부를 판정하는데에 사용가능할 수 있다.

[0094] API(215)는 트랜잭션에 연관된 객체를 수정하라는 요청을 수신할 수 있다. 응답시에, I/O 매니저(235)는 객체를 스토어의 스토리지 위치(예컨대, L_1)에 위치시키고, 객체의 논리적 카피를 생성하고, 트랜잭션의 맥락에서 객체를 변경하고, 변경된 논리적 카피를 저장하기 위한 제 2 스토리지 위치(예컨대, L_2)를 결정하고, 변경된 논리적 카피를 스토리지 컨트롤러(240)에 기록하라는 요청을 전송하며, 논리적 카피가 제 2 스토리지 위치에 저장되었음을 나타내도록 휘발성 데이터 구조(예컨대, 객체 테이블(310))를 업데이트할 수 있다.

[0095] 만약 API(215)가 트랜잭션에 연관된 다른 객체를 수정하라는 요청을 수신하면, I/O 매니저(235)는 추가적인 작업들을 수행할 수 있으며, 이러한 추가적인 작업은 상기 다른 객체와 제 1 객체를 함께 묶는 연계(예컨대, 기록 플랜)를 생성하는 것을 포함한다. 그 다음, 객체들의 수정를 스토리지에 기록하라는 요청을 전송하는 것과 관련하여, I/O 매니저(235)는 상기 연계성을 스토리지 컨트롤러(240)에 기록하라는 요청도 전송할 수 있다.

[0096] 도 6 내지 8은 본 명세서에 기술된 청구사항의 측면들에 따라 발생할 수 있는 예시적인 작업들을 일반적으로 나타내는 순서도이다. 설명의 단순화를 위해서, 도 6 내지 8과 관련하여 기술된 방법론은 일련의 행동들로 도시 및 기술되었다. 본 명세서에 기술된 청구사항의 측면들이 도시된 행동들 및/또는 행동들의 순서에 의해 제한되는 것이 아님을 이해하고 인식해야 한다. 일 실시예에서, 행동들은 아래에 기술된 바와 같은 순서대로 발생한

다. 그러나 다른 실시예들에서, 행동들은 동시에 발생할 수 있거나, 다른 순서로 발생할 수 있고/있거나 본 명세서에 제시 및 기술되지 않은 다른 행동들과 함께 발생할 수 있다. 또한, 본 명세서에 기술된 청구사항의 측면들에 따른 방법론을 구현하는데에 도시된 모든 행동들이 요구되는 것은 아닐 수 있다. 또한, 당업자는 이러한 방법론이 상태 다이어그램을 통한 일련의 상호관계적 상태들로서 또는 이벤트들로서 다르게 표현될 수 있다는 것을 이해하고 인식할 것이다.

[0097] 도 6을 참조하면, 블록(605)에서 작업이 개시된다. 블록(610)에서, 제 1 세트의 업데이트들이 제 1 체크포인트와 연관될 것임이 표시된다. 이것은 후속 업데이트들이 제 1 체크포인트와 연관될 것임을 나타내도록 데이터 구조를 수정함으로써 수행될 수 있다. 이것은, 예를 들어 전술된 바와 같이 락(lock)의 획득 및 해제와 체크포인트 버킷을 참조하기 위해 포인터 또는 다른 데이터 구조를 업데이트하는 것을 포함할 수 있다. 여기에서 "제 1의"라는 용어가 파일 시스템의 임의의 체크포인트를 의미할 수 있으며 후속 체크포인트로부터 해당 체크포인트를 구분하도록 사용된다는 것을 다시 인지해야 한다. 예를 들어, 도 2 내지 5를 참조하면, 체크포인트 매니저(230)는 데이터 구조(510)에 대한 락(505)을 획득할 수 있고 체크포인트 버킷들(515) 중 하나를 가리키도록 포인터를 업데이트할 수 있다.

[0098] 블록(615)에서, 업데이트들이 수신되어 제 1 체크포인트와 연관된다. 예를 들어, 도 2를 참조하면, I/O 매니저(235)는 API(215)를 통해 애플리케이션(들)(210)으로부터의 업데이트 요청들을 수신할 수 있다. 업데이트들이 수신되면, 체크포인트와 연관될 수 있다.

[0099] 블록(620)에서, 제 1 체크포인트의 체크포인트 데이터를 파일 시스템의 스토리지에 기록하라는 결정이 이루어진다. 예를 들어, 도 2를 참조하면, 체크포인트 매니저(230)는 체크포인트 타이머가 만료되었다고 결정할 수 있고 그에 기초하여 체크포인트가 스토어(250)에 기록될 것임을 결정할 수 있다.

[0100] 블록(625)에서, 후속 업데이트들에 대한 체크포인트를 나타내기 위해 데이터 구조에 대한 락이 획득된다. 예를 들어, 도 2 및 5를 참조하면, 체크포인트 매니저(230)가 데이터 구조(510)에 대한 락(505)을 획득할 수 있다.

[0101] 블록(630)에서, 데이터 구조가 다른 체크포인트를 참조하도록 업데이트된다. 이러한 데이터 구조를 수정하는 것은, 제 1 세트의 업데이트들에 후속하여 발생하는 임의의 업데이트가 후속 체크포인트와 연관될 것임을 나타낸다. 예를 들어, 도 2 및 5를 참조하면, 체크포인트 매니저(230)는 체크포인트 버킷들(515) 중 다른 체크포인트 버킷을 참조하도록 데이터 구조(510)를 업데이트할 수 있다.

[0102] 블록(635)에서, 락이 해제된다. 예를 들어, 도 2 및 5를 참조하면, 체크포인트 매니저(230)는 락(505)을 해제할 수 있다.

[0103] 블록(640)에서, 업데이트들에 대한 기록 플랜들이 생성된다. 각 기록 플랜은 적어도 제 1 세트의 업데이트들 중 적어도 하나를 나타내는 데이터에 대한 스토리지 상의 계획된 위치를 나타낸다. 예를 들어, 도 2를 참조하면, 기록 플랜 매니저(237)는 체크포인트와 연관된 업데이트들에 대한 기록 플랜들을 생성하는 것과 관련되었을 수 있다.

[0104] 블록(645)에서, 메타데이터는 기록 플랜들에 대해서 업데이트된다. 이러한 메타데이터는 (기록 플랜들이 스토리지에 기록되었거나 아직 기록되지 않았을 수 있을지라도) 기록 플랜들에 대한 스토리지 위치를 나타낸다. 예를 들어, 도 2를 참조하면, 기록 플랜 매니저(237)는 기록 플랜들에 의해 수정된 객체의 스토리지 위치를 나타내도록 글로벌 테이블을 업데이트할 수 있다.

[0105] 블록(645) 후에, 도 7의 블록(705)에서 작업들이 계속된다. 도 7을 참조하면, 블록(705)에서, 메타데이터에 대해서 락이 획득된다. 예를 들어, 도 2 및 4를 참조하면, 체크포인트 매니저(230)는 글로벌 테이블(405)에 대한 락을 획득할 수 있다. 체크포인트 매니저(230)는 메타데이터가 제 1 세트의 업데이트들 내의 모든 업데이트들에 대한 스토리지 위치를 반영할 때까지 기다릴 수 있다(제 1 세트의 업데이트들 모두가 이러한 스토리지 위치에 기록되었거나 기록되지 않았을 수 있더라도).

[0106] 블록(710)에서, 메타데이터의 논리적 카피가 생성된다. 전술된 바와 같이, 이것은 새로운 카피의 메타데이터를 생성하는 것, 메타데이터에 대한 후속 업데이트가 Copy-on-Write를 발생시키도록 메타데이터를 깨끗하다고 마킹하는 것, 또는 일부 다른 논리적 복제 메커니즘을 포함할 수 있다. 예를 들어, 도 2 및 4를 참조하면, 체크포인트 매니저(230)는 글로벌 테이블(405)의 논리적 카피를 생성할 수 있다.

[0107] 블록(715)에서, 락이 해제된다. 예를 들어, 도 2 및 4를 참조하면, 체크포인트 매니저(230)는 글로벌 테이블(405)에 대한 락을 해제할 수 있다.

[0108] 블록(720)에서, 제 1 체크포인트 데이터를 기록하기 위한 기록 플랜이 생성된다. 이러한 기록 플랜을 생성하는 것은, 현재 기록 플랜에 상응하는 체크포인트와 데이터가 디스크에 기록되는 것에 이어지는 업데이트들에 대해 기록 플랜들이 생성되는 것(그리고 디스크에 기록되는 것)과 동시에 발생한다. 예를 들어, 도 2를 참조하면, 체크포인트 매니저(230)는 제 1 체크포인트의 체크포인트 데이터에 대한 기록 플랜을 생성하도록 기록 플랜 매니저(237)를 사용할 수 있다. 이러한 데이터는 앞서 언급된 글로벌 테이블의 논리적 카피를 포함할 수 있다.

[0109] 블록(725)에서, 일 실시예에 따르면, 체크포인트 매니저가 제 1 세트의 업데이트들의 모든 업데이트들이 성공적으로 스토리지에 기록되기를 기다릴 수 있다. 모든 업데이트들이 성공적으로 스토리지에 기록된 후에, 업데이트 매니저는 확인 코드를 포함하는 최종 체크포인트 레코드(record)을 기록할 수 있다. 전술된 바와 같이, 이것은 복구시 체크포인트에 상응하는 모든 업데이트들이 스토리지에 기록된 것으로 예상되는지를 결정하게 위해서 간단하게 확인 코드를 검사할 수 있도록 한다.

[0110] 다른 실시예에서, 체크포인트 매니저는 체크포인트 레코드 내에 몇몇 확인 코드를 기록할 수 있다. 이러한 확인 코드들은 제 1 세트의 업데이트들의 업데이트의 스토리지 위치와 연관될 수 있다. 이러한 실시예에서, 체크포인트 매니저는 이들 업데이트들이 스토리지에 기록되기를 기다릴 수 있거나 또는 기다리지 않고 체크포인트 레코드를 기록할 수 있다. 만약 후자가 선택된다면, 유효 체크포인트 레코드가 디스크 상에 있음을 입증하기보다는 복구 동안에 적절한 체크포인트를 찾는 것이 더 포함될 수 있다.

[0111] 블록(730)에서, 체크포인트 데이터는 스토리지에 기록될 수 있다. 이것은 예를 들어 체크포인트 데이터와 연관된 기록 플랜을 스토리지에 기록하는 것을 포함할 수 있다. 다른 예시로서, 이것은 글로벌 테이블의 논리적 카피를 참조하는 스토리지에 체크포인트 기록을 기록하는 것을 포함할 수 있다. 예를 들어, 도 2를 참조하면, 체크포인트 매니저(230)는 체크포인트 데이터에 상응하는 기록 플랜이 스토리지에 기록될 것을 요청할 수 있다.

[0112] 블록(735)에서, 적어도 하나의 확인 코드가 스토리지에 기록된다. 적어도 하나의 확인 코드를 스토리지에 기록하는 것은, 글로벌 테이블의 논리적 카피를 참조하는 스토리지에 체크포인트 레코드를 기록하는 것과 결합될 수 있다. 예를 들어, 도 2를 참조하면, 체크포인트 매니저(230)는 글로벌 테이블의 논리적 카피를 참조하며 체크포인트 레코드의 콘텐츠를 인증하기 위한 확인 코드를 포함하는 스토리지에 체크포인트 레코드를 기록할 수 있다.

[0113] 블록(740)에서, 만약 존재한다면, 다른 작업들이 수행될 수 있다.

[0114] 도 8을 참조하면, 블록(805)에서 작업들이 시작된다. 블록(810)에서, 복구 요청이 수신된다. 예를 들어, 도 2를 참조하면, 복구 매니저(225)가 스토어(250)에 저장된 데이터에 대한 복구를 수행하라는 복구 요청을 수신할 수 있다.

[0115] 블록(815)에서, 체크포인트 데이터가 발견된다. 예를 들어, 도 2를 참조하면, 복구 매니저(225)는 스토어(250)(또는 다른 스토어) 상에 저장된 최후의 체크포인트 데이터를 발견할 수 있다.

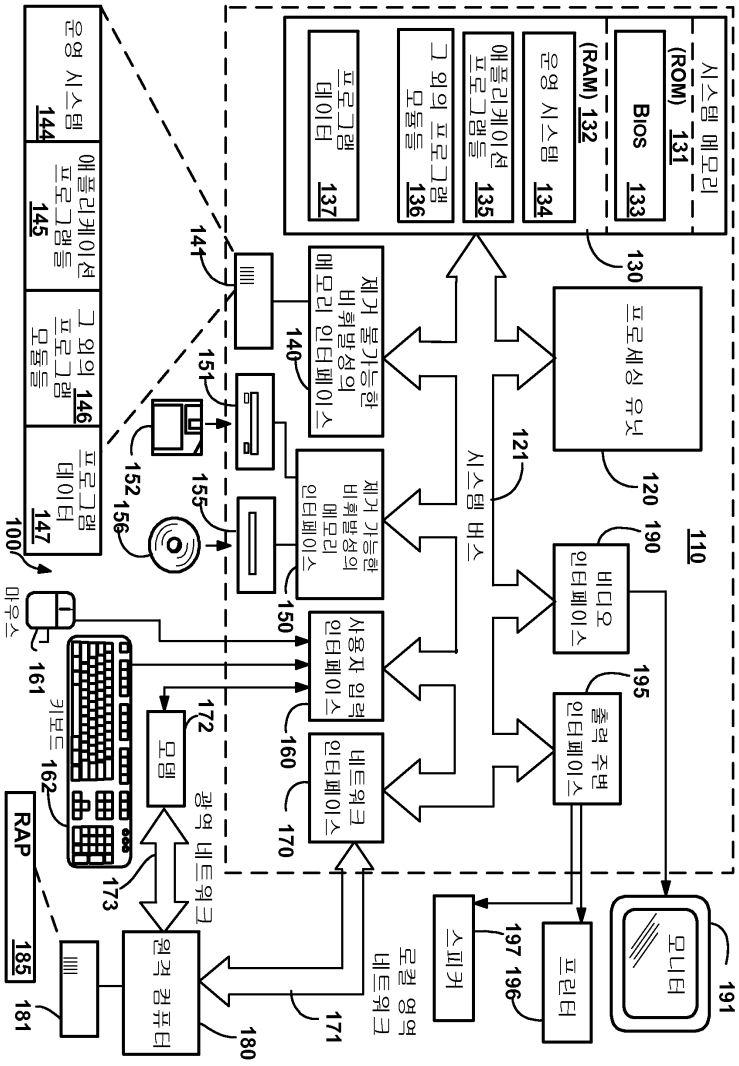
[0116] 블록(820)에서, 체크포인트 데이터는 확인 코드를 이용하여 인증된다. 예를 들어, 도 2를 참조하면, 복구 매니저(225)는 체크포인트 데이터의 체크섬을 계산하고 이러한 체크섬을 체크포인트 데이터와 저장된 체크섬에 비교할 수 있다. 만약 체크섬이 일치하면, 체크포인트는 유효한 것으로 간주될 수 있다. 만약 추가적인 확인이 요구된다면, 복구 매니저는 체크포인트 데이터에 의해 참조되는 글로벌 테이블에 의해 나타내어진 하나 이상의 객체를 확인하고자 시도할 수 있다.

[0117] 블록(825)에서, 만약 존재한다면, 다른 작업들이 수행된다.

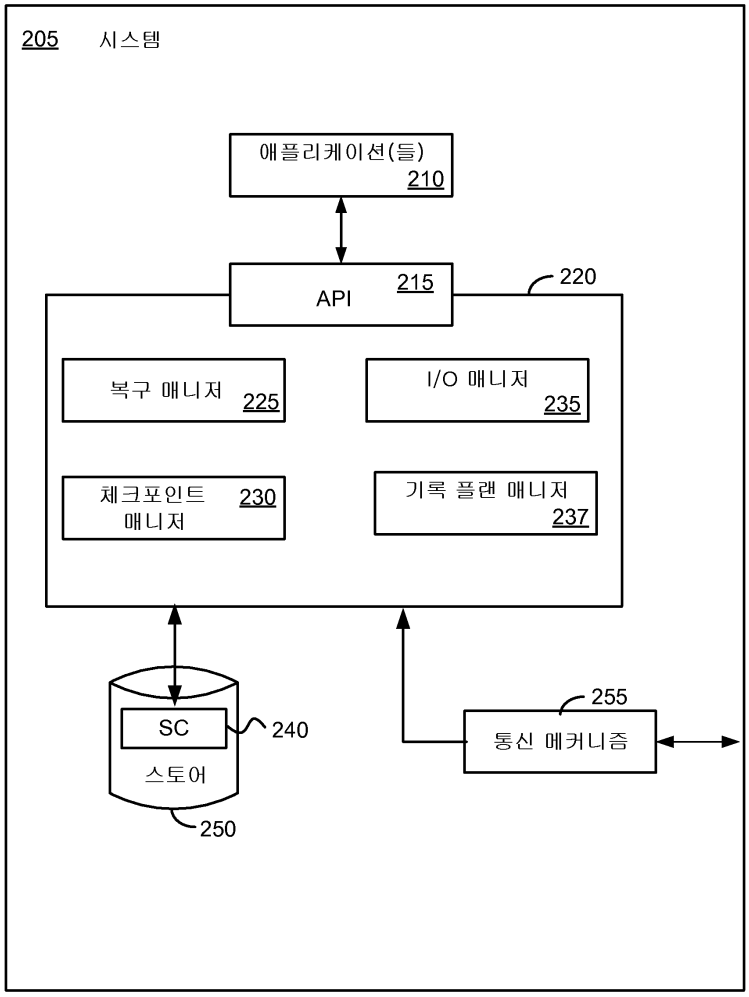
[0118] 전술된 상세한 설명에서 볼 수 있는 바와 같이, 파일 시스템에 대한 체크포인트와 관련된 측면들이 기술되었다. 본 명세서에 기술된 청구사항의 측면들이 다양한 변경사항과 대안적인 구조에 대해 민감하지만, 본 명세서에 설명된 소정의 실시예들이 도면에 도시되었으며 위에서 상세하게 기술되었다. 그러나, 청구된 청구사항의 측면들을 개시된 특정한 형태로 제한하고자 하는 것이 아니며, 그와 반대로 본 명세서에 기술된 청구사항의 다양한 측면들의 범주 및 사상 내에 포함되는 모든 변경사항, 대안적인 구성들 및 그의 동등물을 커버하고자 한다는 것을 이해해야 한다.

도면

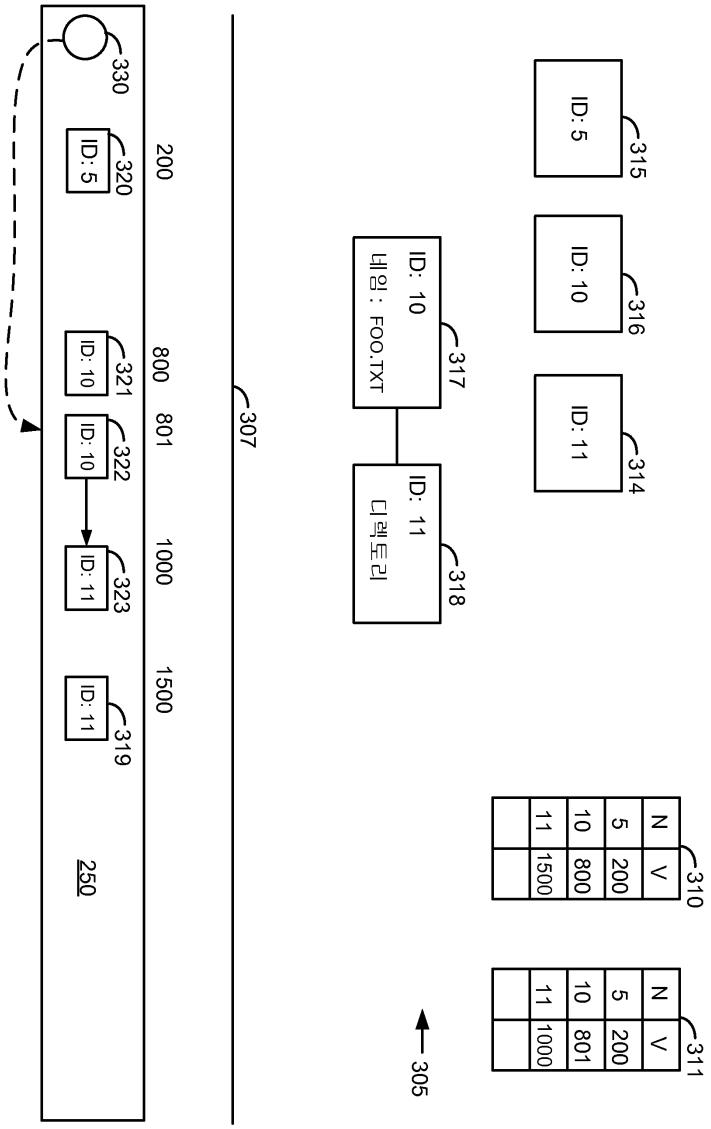
도면1


시스템 메모리
(ROM) 131
Bios 133
(RAM) 132
운영 시스템 134
애플리케이션 프로그램들 135
그 외의 프로그램 모듈들 136
프로그램 데이터 137
130
프로세싱 유닛
120
190
110
121
시스템 버스
비디오 인터페이스
195
출력 주변 인터페이스
191
모니터
197
스피커
196
프린터
171
로컬 영역 네트워크
180
원격 컴퓨터
181
RAP 185
173
162
키보드
161
마우스
100
운영 시스템 144
애플리케이션 프로그램들 145
그 외의 프로그램 모듈들 146
프로그램 데이터 147
141
151
152
155
156
140
제거 불가능한 비휘발성의 메모리 인터페이스
150
제거 가능한 비휘발성의 메모리 인터페이스
사용자 입력 인터페이스
160
네트워크 인터페이스
170
172
모뎀
광역 네트워크

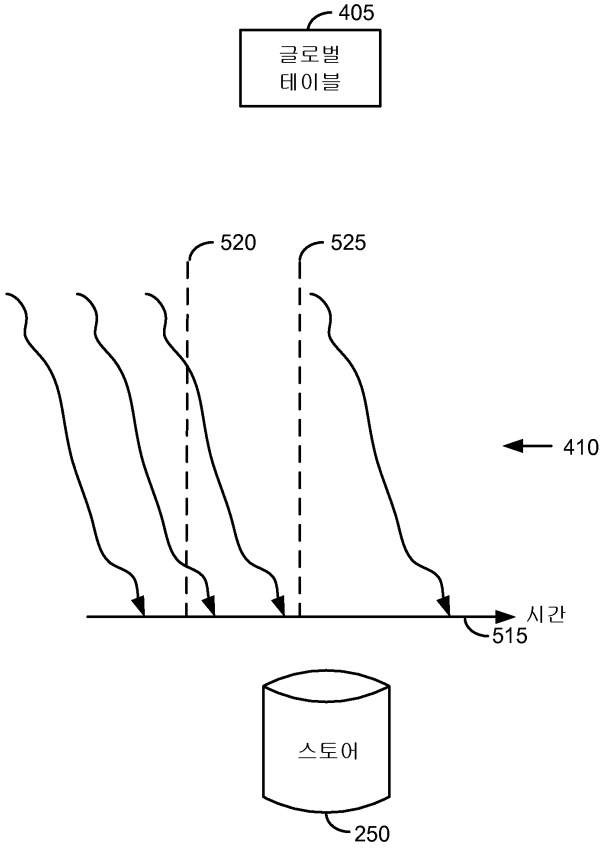
도면2


205 시스템
애플리케이션(들)
210
API
215
220
복구 매니저
225
I/O 매니저
235
체크포인트
매니저
230
기록 플랜 매니저
237
SC
240
스토어
250
255
통신 메커니즘

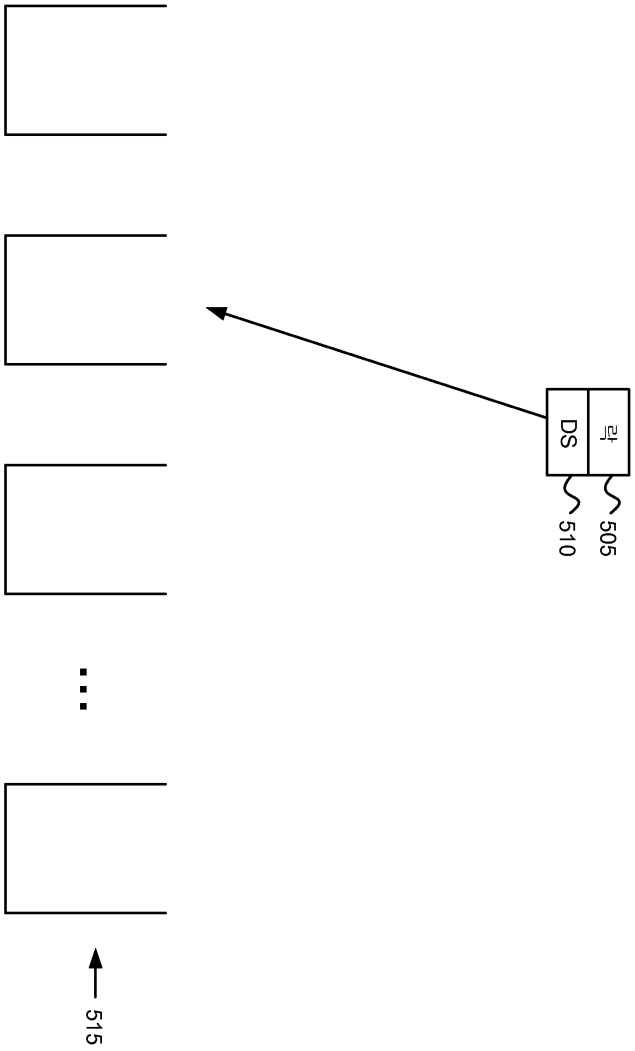
도면3


330
ID: 5
320
200
ID: 10
321
800
ID: 10
322
801
ID: 11
323
1000
ID: 11
319
1500
250
307
ID: 5
315
ID: 10
316
ID: 11
314
ID: 10
네임 : FOO.TXT
317
ID: 11
디렉토리
318
305
N
V
5
200
10
800
11
1500
310
N
V
5
200
10
801
11
1000
311

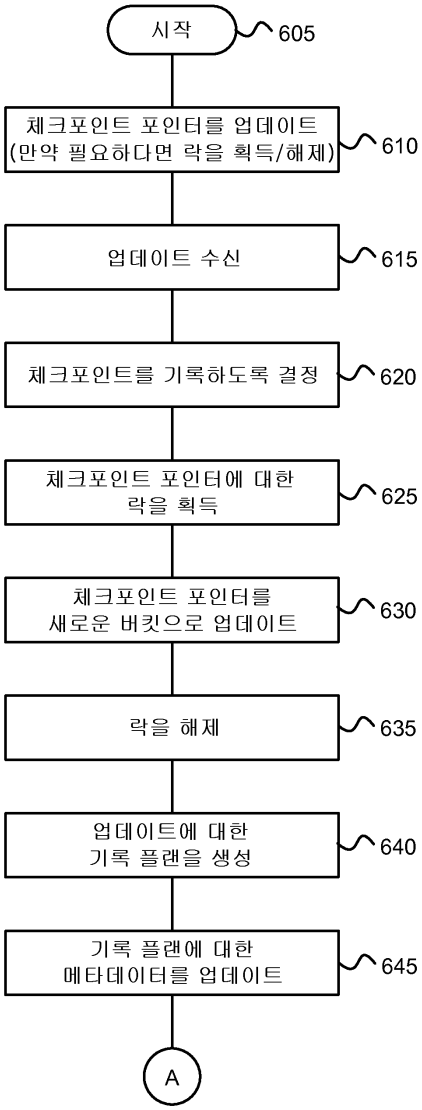
도면4



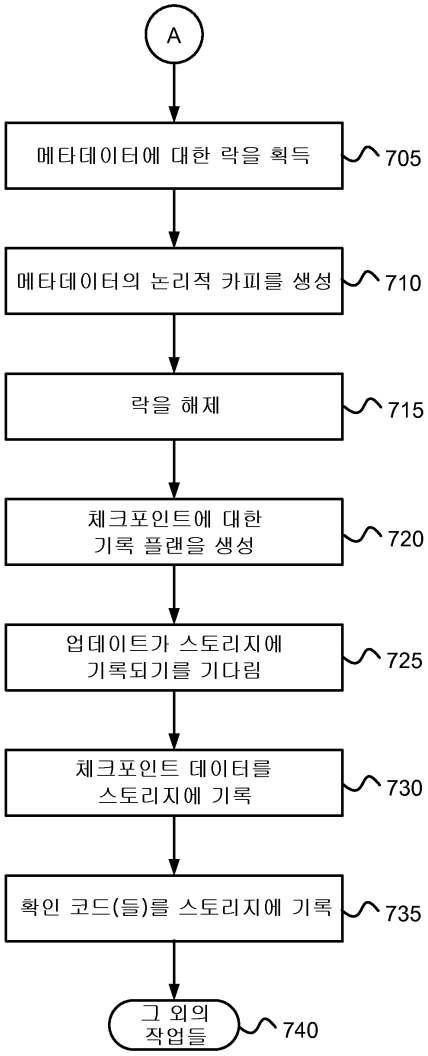
도면5



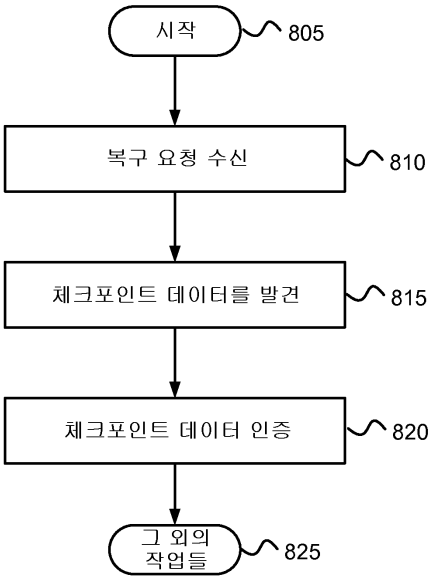
도면6


시작
605
체크포인트 포인터를 업데이트
(만약 필요하다면 락을 획득/해제)
610
업데이트 수신
615
체크포인트를 기록하도록 결정
620
체크포인트 포인터에 대한
락을 획득
625
체크포인트 포인터를
새로운 버킷으로 업데이트
630
락을 해제
635
업데이트에 대한
기록 플랜을 생성
640
기록 플랜에 대한
메타데이터를 업데이트
645
A

도면7


A
메타데이터에 대한 락을 획득
705
메타데이터의 논리적 카피를 생성
710
락을 해제
715
체크포인트에 대한
기록 플랜을 생성
720
업데이트가 스토리지에
기록되기를 기다림
725
체크포인트 데이터를
스토리지에 기록
730
확인 코드(들)를 스토리지에 기록
735
그 외의
작업들
740

도면8


시작
805
복구 요청 수신
810
체크포인트 데이터를 발견
815
체크포인트 데이터 인증
820
그 외의
작업들
825