(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2015년02월04일
(11) 등록번호 10-1490090
(24) 등록일자 2015년01월29일

(51) 국제특허분류(Int. Cl.)
G06F 17/30 (2006.01) *G06F 15/16* (2006.01)
(21) 출원번호 10-2014-7001183(분할)
(22) 출원일자(국제) 2006년11월30일
심사청구일자 2014년02월11일
(85) 번역문제출일자 2014년01월15일
(65) 공개번호 10-2014-0025580
(43) 공개일자 2014년03월04일
(62) 원출원 특허 10-2013-7021963
원출원일자(국제) 2006년11월30일
심사청구일자 2013년09월12일
(86) 국제출원번호 PCT/US2006/061431
(87) 국제공개번호 WO 2008/069811
국제공개일자 2008년06월12일
(30) 우선권주장
11/371,304 2006년03월08일 미국(US)
60/754,726 2005년12월29일 미국(US)
(56) 선행기술조사문헌
JP2000132441 A

(73) 특허권자
아마존 테크놀로지스, 인크.
미합중국 네바다 (우편번호 89507), 리노, 피.오.박스 8102
(72) 발명자
버뮬렌, 알랜, 에이치.
미국 98144-2734 워싱턴주 시애틀 스위트 1200 12번 애비뉴 사우쓰 1200
아틀라스, 알랜, 비.
미국 98144-2734 워싱턴주 시애틀 스위트 1200 12번 애비뉴 사우쓰 1200
(뒷면에 계속)
(74) 대리인
양영준, 백만기

전체 청구항 수 : 총 20 항 심사관 : 홍경아

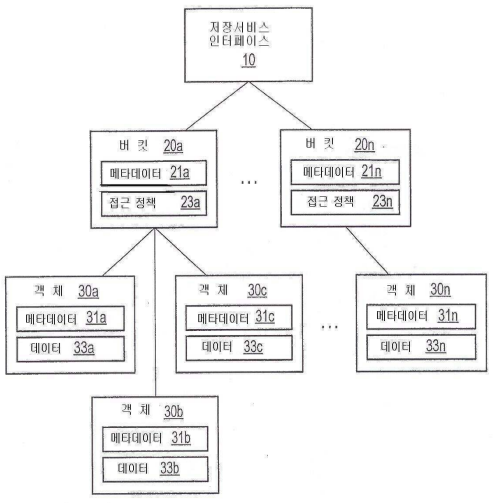
(54) 발명의 명칭 **웹 서비스 클라이언트 인터페이스를 갖는 분배형 저장 시스템**

(57) 요 약

시스템으로서, 적어도 하나의 프로세서; 및 상기 적어도 하나의 프로세서에 연결된 메모리를 포함하고, 상기 메모리는 명령들을 저장하고, 상기 명령들은, 상기 시스템이, 데이터 저장 웹 서비스에 접근하기 위해 어플리케이션 프로그래밍 인터페이스(API)에 따라, 데이터 객체들 중 주어진 하나를 저장하기 위한 하나의 주어진 클라이언

(뒷면에 계속)

대 표 도 - 도1



트 요청을 포함하는 데이터 객체들을 저장하기 위한 클라이언트 요청들을 수신하도록 구성된 웹 서비스 인터페이스; 상기 데이터 객체들의 복사(replica)들을 저장하도록 구성된 복수의 저장 노드들; 상기 주어진 데이터 객체의 하나 이상의 복사들을 상기 복수의 저장 노드들 중 하나 이상의 대응하는 노드들에 저장하기 위한 기입 계획을 동적으로 결정하도록 구성된 저장 노드 선택 로직 - 상기 기입 계획은 상기 복수의 저장 노드들의 현재 상태 정보에 따라 동적으로 결정됨 -; 상기 웹 서비스 인터페이스로부터 데이터 객체들을 저장하기 위한 상기 클라이언트 요청들을 수신하도록 구성된 조정자(coordinator) - 상기 주어진 클라이언트 요청에 응답하여, 상기 조정자는, 상기 주어진 데이터 객체의 상기 하나 이상의 복사들을 상기 기입 계획에 따라 상기 하나 이상의 대응하는 저장 노드들에 저장하도록 구성됨 - 에게 명령하도록, 상기 적어도 하나의 프로세서에 의해 실행가능한, 시스템에 제공된다.

(72) 발명자

바쓰, 데이비드, 엠.

미국 98144-2734 워싱턴주 시애틀 스위트 1200 12번 애비뉴 사우쓰 1200

코미에, 존, 데이비드

미국 98144-2734 워싱턴주 시애틀 스위트 1200 12번 애비뉴 사우쓰 1200

피츠맨, 에이미, 케이.

미국 98144-2734 워싱턴주 시애틀 스위트 1200 12번 애비뉴 사우쓰 1200

소렌슨, 제임스, 크리스토퍼 3세

미국 98144-2734 워싱턴주 시애틀 스위트 1200 12번 애비뉴 사우쓰 1200

와그너, 에릭, 엠.

미국 98144-2734 워싱턴주 시애틀 스위트 1200 12번 애비뉴 사우쓰 1200

특허청구의 범위

청구항 1

웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템으로서,

적어도 하나의 프로세서; 및

상기 적어도 하나의 프로세서에 연결된 메모리

를 포함하고,

상기 메모리는 명령들을 저장하고, 상기 명령들은,

상기 시스템이,

데이터 저장 웹 서비스에 접근하기 위해 어플리케이션 프로그래밍 인터페이스(API)에 따라, 데이터 객체들 중 주어진 하나를 저장하기 위한 하나의 주어진 클라이언트 요청을 포함하는 데이터 객체들을 저장하기 위한 클라이언트 요청들을 수신하도록 구성된 웹 서비스 인터페이스;

상기 데이터 객체들의 복사(replica)들을 저장하도록 구성된 복수의 저장 노드들;

상기 주어진 데이터 객체의 하나 이상의 복사들을 상기 복수의 저장 노드들 중 하나 이상의 대응하는 노드들에 저장하기 위한 기입 계획을 동적으로 결정하도록 구성된 저장 노드 선택 로직 - 상기 기입 계획은 상기 복수의 저장 노드들의 현재 상태 정보에 따라 동적으로 결정됨 -;

상기 웹 서비스 인터페이스로부터 데이터 객체들을 저장하기 위한 상기 클라이언트 요청들을 수신하도록 구성된 조정자(coordinator) - 상기 주어진 클라이언트 요청에 응답하여, 상기 조정자는, 상기 주어진 데이터 객체의 상기 하나 이상의 복사들을 상기 기입 계획에 따라 상기 하나 이상의 대응하는 저장 노드들에 저장하도록 구성됨 -

를 구현하도록, 상기 적어도 하나의 프로세서에 의해 실행가능한, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 2

제1항에 있어서,

상기 저장 노드 선택 로직은, 상기 조정자가 상기 주어진 클라이언트 요청을 수신하는 것에 응답하여 상기 기입 계획을 동적으로 결정하도록 또한 구성되는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 3

제1항에 있어서,

상기 현재 상태 정보는, 상기 복수의 저장 노드들에 각각 대응하고, 대응하는 저장 노드가 통상적인 동작 상태에 있는지 여부를 나타내는 동작 상태 정보를 포함하고,

상기 기입 계획을 동적으로 결정하기 위해, 상기 저장 노드 선택 로직은, 상기 하나 이상의 저장 노드들에 대응하는 상기 동작 상태 정보에 의존하여 상기 주어진 데이터 객체의 대응하는 복사를 저장하기 위한 상기 하나 이상의 저장 노드들을 선택하도록 또한 구성되는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 4

제1항에 있어서,

상기 현재 상태 정보는, 상기 복수의 저장 노드들에 각각 대응하고, 대응하는 저장 노드가 동작하는 부하의 레벨을 나타내는 부하 상태 정보를 포함하고,

상기 기입 계획을 동적으로 결정하기 위해, 상기 저장 노드 선택 로직은, 상기 하나 이상의 저장 노드들에 대응하는 상기 부하 상태 정보에 의존하여 상기 주어진 데이터 객체의 대응하는 복사를 저장하기 위한 상기 하나 이상의 저장 노드들을 선택하도록 또한 구성되는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 5

제4항에 있어서,

상기 복수의 저장 노드들 중 주어진 하나의 노드에 대해, 상기 부하 상태 정보는,

상기 주어진 저장 노드에 대응하는 프로세서 이용의 레벨;

상기 주어진 저장 노드에 대응하는 메모리 이용의 레벨;

상기 주어진 저장 노드에 대응하는 저장 장치 입력/출력 대역폭 이용의 레벨; 또는

상기 주어진 저장 노드에 대응하는 네트워크 인터페이스 대역폭 이용의 레벨 중 하나 이상을 나타내는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 6

제1항에 있어서,

상기 현재 상태 정보는, 상기 복수의 저장 노드에 각각 대응하고 대응하는 저장 노드와의 통신 비용을 나타내는 네트워크 비용 정보를 포함하고, 상기 기입 계획을 동적으로 결정하기 위해, 상기 저장 노드 선택 로직은, 상기 하나 이상의 저장 노드들에 대응하는 상기 네트워크 비용 정보에 의존하여 상기 주어진 데이터 객체의 대응 복사를 저장하기 위한 상기 하나 이상의 저장 노드들을 선택하도록 또한 구성되는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 7

제6항에 있어서,

상기 복수의 저장 노드들의 주어진 저장 노드에 대해, 상기 네트워크 비용 정보는,

상기 주어진 저장 노드에 대응하는 네트워크 통신 지연(latency)의 표시; 또는

상기 주어진 저장 노드에 대응하는 네트워크 통신 경제적 비용의 표시 중 하나 이상을 포함하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 8

제1항에 있어서,

상기 기입 계획을 동적으로 결정하기 위해, 상기 저장 노드 선택 로직은, 상기 기입 계획에 따라 상기 주어진 데이터 객체의 상기 하나 이상의 복사들을 저장하는 것과 관련된 비용을 최소화하도록 또한 구성되고, 상기 비용은 시간에 따라 변하고, 상기 비용을 최소화하기 위해, 상기 저장 노드 선택 로직은, 특정 복사들을 일정 시간 동안 특정 저장 노드에 저장하는 것과 관련된 비용에 의존하여, 상기 하나 이상의 복사들 중 특정 복사가 상기 복수의 저장 노드들 중 특정 저장 노드에 저장되는 상기 일정 시간을 식별하도록 또한 구성되는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 9

제1항에 있어서,

복수의 컴퓨팅 노드들은,

상기 대응하는 하나 이상의 저장 노드들로 저장되는 상기 주어진 데이터 객체의 상기 하나 이상의 복사들에 후속하여, 상기 하나 이상의 복사들 중 어느 것이 접근할 수 없는지를 결정하고;

상기 하나 이상의 복사들 중 어느 것이 접근할 수 없는지를 결정하는 것에 응답하여, 상기 주어진 데이터 객체의 하나 이상의 추가 복사들을 저장하기 위해 상기 저장 노드 선택 로직으로부터 새로운 기입 계획을 요청하고

- 상기 저장 노드 선택 로직은 상기 복수의 저장 노드의 현재 상태 정보에 따라 상기 새로운 기입 계획을 동적으로 결정하도록 구성됨 -; 및

상기 새로운 기입 계획을 수신하는 것에 후속하여, 상기 하나 이상의 추가 복사들을 상기 새로운 기입 계획에 따라 상기 저장 노드들 중 하나 이상의 대응하는 저장 노드들에 저장하도록 구성되는 복사기를 구현하도록 또한 구성되는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 10

제1항에 있어서,

상기 복수의 컴퓨팅 노드들은,

상기 대응하는 하나 이상의 저장 노드들에 저장되는 상기 주어진 데이터 객체의 상기 하나 이상의 복사들에 후속하여, 상기 주어진 데이터 객체의 상기 저장된 하나 이상의 복사들에 의해 만족되는 저장 정책들의 수가 불충분한지 결정하고;

상기 만족된 저장 정책들의 수가 불충분하다고 결정하는 것에 응답하여, 상기 주어진 데이터 객체의 하나 이상의 추가 복사들을 저장하기 위해 상기 저장 노드 선택 로직으로부터 새로운 기입 계획을 요청하고 - 상기 저장 노드 선택 로직은 상기 복수의 저장 노드들의 현재 상태 정보에 따라 상기 새로운 기입 계획을 동적으로 결정하고, 상기 기입 계획을 동적으로 결정하는 것은, 상기 새로운 기입 계획에 따라 상기 주어진 데이터 객체의 상기 하나 이상의 추가 복사들을 저장함으로써 만족되는 저장 정책들의 수를 최대화하는 것을 포함함 - ;

상기 새로운 기입 계획을 수신하는 것에 후속하여, 상기 하나 이상의 추가 복사들을 상기 새로운 기입 계획에 따라 상기 저장 노드들 중 하나 이상의 대응하는 저장 노드들에 저장하도록 구성되는 복사기를 구현하도록 또한 구성되는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 시스템.

청구항 11

웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법으로서,

웹 서비스 인터페이스에 따라, 데이터 객체들 중 주어진 데이터 객체를 저장하기 위한 주어진 클라이언트 요청을 포함하는 데이터 객체들을 저장하기 위한 클라이언트 요청들을 수신하는 단계;

상기 주어진 데이터 객체의 하나 이상의 복사들을 복수의 저장 노드들 중 하나 이상의 대응하는 저장 노드들에 저장하기 위한 기입 계획을 동적으로 결정하는 단계 - 상기 기입 계획은 상기 복수의 저장 노드들의 현재 상태 정보에 따라 동적으로 결정됨 -; 및

상기 주어진 클라이언트 요청에 응답하여, 상기 주어진 데이터 객체의 하나 이상의 복사들을 상기 기입 계획에 따라 상기 하나 이상의 대응하는 저장 노드들에 저장하는 단계를 하나 이상의 컴퓨터 시스템들에 의해 수행하는 것을 포함하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

청구항 12

제11항에 있어서,

상기 기입 계획을 동적으로 결정하는 단계는, 상기 주어진 클라이언트 요청을 수신하는 것에 응답하여 발생하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

청구항 13

제11항에 있어서,

상기 현재 상태 정보는, 상기 복수의 저장 노드들에 각각 대응하고, 대응하는 저장 노드가 통상적인 동작 상태에 있는지 여부를 나타내는 동작 상태 정보를 포함하고,

상기 기입 계획을 동적으로 결정하는 단계는, 상기 하나 이상의 저장 노드들에 대응하는 상기 동작 상태 정보에 의존하여 상기 주어진 데이터 객체의 대응하는 복사를 저장하기 위한 상기 하나 이상의 저장 노드들을 선택하는 단계를 더 포함하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

청구항 14

제11항에 있어서,

상기 현재 상태 정보는, 상기 복수의 저장 노드들에 각각 대응하고, 대응하는 저장 노드가 동작하는 부하의 레벨을 나타내는 부하 상태 정보를 포함하고,

상기 기입 계획을 동적으로 결정하는 단계는, 상기 하나 이상의 저장 노드들에 대응하는 상기 부하 상태 정보에 의존하여 상기 주어진 데이터 객체의 대응하는 복사를 저장하기 위해 상기 하나 이상의 저장 노드들을 선택하는 단계를 더 포함하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

청구항 15

제14항에 있어서,

상기 복수의 저장 노드들 중 주어진 저장 노드에 대해, 상기 부하 상태 정보는,

상기 주어진 저장 노드에 대응하는 프로세서 이용의 레벨;

상기 주어진 저장 노드에 대응하는 메모리 이용의 레벨;

상기 주어진 저장 노드에 대응하는 저장 장치 입력/출력 대역폭 이용의 레벨; 또는

상기 주어진 저장 노드에 대응하는 네트워크 인터페이스 대역폭 이용의 레벨 중 하나 이상을 나타내는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

청구항 16

제11항에 있어서,

상기 현재 상태 정보는, 상기 복수의 저장 노드에 각각 대응하고 대응하는 저장 노드와의 통신 비용을 나타내는 네트워크 비용 정보를 포함하고, 상기 기입 계획을 동적으로 결정하는 단계는 상기 하나 이상의 저장 노드들에 대응하는 상기 네트워크 비용 정보에 의존하여 상기 주어진 데이터 객체의 대응 복사를 저장하기 위한 상기 하나 이상의 저장 노드들을 선택하는 단계를 포함하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

청구항 17

제11항에 있어서,

상기 기입 계획을 동적으로 결정하는 단계는 상기 기입 계획에 따라 상기 주어진 데이터 객체의 하나 이상의 복사들을 저장함으로써 만족되는 저장 정책들의 수를 최대화하는 단계를 포함하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

청구항 18

제17항에 있어서,

상기 저장 정책들 중 주어진 정책은 상기 저장 노드들 중 대응하는 노드들에 저장될 상기 주어진 데이터 객체의 복사들의 최소수를 특정하거나, 또는 상기 주어진 데이터 객체의 상기 하나 이상의 복사들이 분산될 영역들의 최소수를 특정하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

청구항 19

제11항에 있어서,

상기 대응하는 하나 이상의 저장 노드들로 저장되는 상기 주어진 데이터 객체의 상기 하나 이상의 복사들에 후속하여, 상기 하나 이상의 복사들 중 어느 것이 접근할 수 없는지를 결정하는 단계;

상기 하나 이상의 복사들 중 어느 것이 접근할 수 없는지를 결정하는 단계에 응답하여, 상기 복수의 저장 노드의 현재 상태 정보에 따라 상기 주어진 데이터 객체의 하나 이상의 추가 복사들을 저장하기 위한 새로운 기입 계획을 동적으로 결정하는 단계; 및

상기 하나 이상의 추가 복사들을 상기 새로운 기입 계획에 따라 상기 저장 노드들 중 하나 이상의 대응하는 저

장 노드들에 저장하는 단계

를 더 포함하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

청구항 20

제11항에 있어서,

상기 대응하는 하나 이상의 저장 노드들에 저장되는 상기 주어진 데이터 객체의 상기 하나 이상의 복사들에 후속하여, 상기 주어진 데이터 객체의 상기 저장된 하나 이상의 복사들에 의해 만족되는 저장 정책들의 수가 불충분한지 결정하는 단계;

상기 만족된 저장 정책들의 수가 불충분하다고 결정하는 단계에 응답하여, 상기 복수의 저장 노드들의 현재 상태 정보에 따라 상기 주어진 데이터 객체의 하나 이상의 추가 복사들을 저장하기 위한 새로운 기입 계획을 동적으로 결정하는 단계 - 상기 기입 계획을 동적으로 결정하는 단계는 상기 새로운 기입 계획에 따라 상기 주어진 데이터 객체의 상기 하나 이상의 추가 복사들을 저장함으로써 만족되는 저장 정책들의 수를 최대화하는 단계를 포함함 - ;

상기 하나 이상의 추가 복사들을 상기 새로운 기입 계획에 따라 상기 저장 노드들 중 하나 이상의 대응하는 저장 노드들에 저장하는 단계

를 더 포함하는, 웹 서비스 인터페이스를 이용하여 데이터 객체들을 저장하기 위한 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 데이터 저장 시스템에 관한 것으로서, 더욱 상세하게는 웹 서비스로서 저장 장치에 접근을 제공하도록 구성된 저장 시스템에 관한 것이다.

배 경 기 술

[0002] 상이한 많은 컴퓨팅 어플리케이션은 다양한 종류의 어플리케이션 데이터의 지속적인 저장을 위해 일정 형태의 저장 매체에 의존한다. 예를 들면, 공통 오피스 어플리케이션 및 멀티미디어 어플리케이션은 다른 것 중에 서류들, 스프레드 시트들, 정지 영상들, 오디오 및 비디오 데이터와 같은 다양한 형태 및 포맷의 어플리케이션 데이터를 생성하여 사용한다. 종종, 이러한 데이터는 반복된 접근용으로 저장되거나 사용자를 위하여 사용된다. 예를 들면, 사용자는 시간 주기 동안 다수의 서류 또는 다른 데이터를 저장하고 작업하기를 바랄 수 있고, 필요한 경우 상기 데이터가 예측 가능한 상태로 용이하게 이용될 수 있다는 것을 기대할 수 있다.

[0003] 종래의 컴퓨팅 시스템에 있어서, 비록 광학 및 반도체 저장 장치들이 사용되더라도, 지속성 어플리케이션 데이터 저장용 어플리케이션들에 의해 사용된 상기 저장 매체는 가장 일반적으로는 자성 고정 드라이브 또는 "하드 드라이브"이다. 이러한 장치들은 상기 어플리케이션들을 수행하는 컴퓨터 시스템 내에 집적화되거나 근거리 주변 인터페이스 또는 네트워크를 통하여 상기 시스템에 접근할 수 있다. 통상적으로, 어플리케이션 저장 장치로서 역할을 하는 장치들은 장치-레벨 작용을 관리하는 동작 시스템에 의해 관리되어 파일 시스템 인터페이스와 같은 일관된 저장 인터페이스를 저장 접근을 필요로 하는 다양한 어플리케이션에게 제시한다.

[0004] 이러한 종래의 어플리케이션 저장 모델에는 수개의 제한이 있다. 먼저, 이것은 일반적으로 어플리케이션 데이터의 접근성을 제한한다. 예를 들면, 어플리케이션 데이터가 특정 컴퓨터 시스템의 근거리 하드 드라이브에 저장되면, 다른 시스템에서 실행되는 어플리케이션들에 접근할 수 없다. 비록 상기 데이터가 네트워크-접근 장치에 저장되더라도, 즉시 네트워크 밖 시스템들에서 실행되는 어플리케이션은 상기 장치에 접근할 수 없다. 예를 들면, 보안 이유 때문에, 기업들은 보통 그들의 근거리 망에 접근하는 것을 제한함으로써, 기업들 외부 시스템들은 상기 기업 내 시스템들 또는 자원에 접근할 수 없게 된다. 그래서, 휴대 장치들(예를 들면, 노트북 또는 휴대용 컴퓨터, 개인 휴대 단말기, 이동 전화 장치 등)에서 실행되는 어플리케이션들은 고정형 시스템 또는 네트워크와 지속적으로 연관된 데이터에 접근하는데 어려움을 경험할 수 있다.

[0005] 종래의 어플리케이션 저장 모델은 또한 저장된 데이터의 신뢰성을 충분히 보장할 수 없다. 예를 들면, 종래 동작 시스템은 통상적으로 디폴트에 의해 하나의 저장 장치에 어플리케이션 데이터의 하나의 복사를

저장함으로써, 데이터 리던던시가 바람직한 경우, 어플리케이션 데이터의 자체 복사들을 발생하고 관리하기 위한 사용자 또는 어플리케이션을 요구한다. 개별 저장 장치들 또는 제3자 소프트웨어가 리던던시를 어느 정도 제공하지만, 어플리케이션들에 이용할 수 있는 상기 저장 자원들이 어플리케이션 설치를 통해 널리 변할 수 있으므로, 이 특성들은 어플리케이션들에 일관성 있게 이용될 수는 없다. 상기 종래의 동작-시스템-중개 저장 모델은 또한 데이터의 교차-플랫폼 접근성을 제한할 수 있다. 예를 들면, 상이한 동작 시스템들은 상이한 비 호환성 포맷에서의 동일한 어플리케이션용 데이터를 저장하여, 하나의 플랫폼(예를 들면, 동작 시스템 및 기초 컴퓨터 시스템 하드웨어)에서 실행되는 어플리케이션들의 사용자들이 다른 플랫폼들에서 실행되는 어플리케이션에 의해 저장된 데이터에 접근하는 것을 어렵게 할 수 있다.

발명의 내용

[0006] 웹 서비스 기반 분배형 시스템의 다양한 실시예들이 개시되어 있다. 일 실시예에 따르면, 시스템은 웹 서비스 프로토콜에 따라 데이터 객체들에 접근하기 위한 클라이언트 요청들을 수신하도록 구성된 웹 서비스 인터페이스를 포함할 수 있다. 상기 데이터 객체들 중의 주어진 하나에 접근하기 위한 상기 클라이언트 요청들 중의 주어진 하나는 상기 주어진 데이터 객체에 대응하는 키 값을 포함할 수 있다. 상기 시스템은 또한 상기 데이터 객체들의 복사들을 저장하도록 구성되고, 상기 복사들 각각은 각 위치 표시자 값을 통하여 접근할 수 있는 다수의 저장 노드를 포함할 수 있고, 상기 위치 표시자 각각은 시스템 내에서 유일하다. 상기 시스템은 상기 데이터 객체들 각각에 대한 각 키맵 엔트리(keymap entry)를 저장하도록 구성된 키맵 인스턴스(keymap instance)를 포함할 수 있고, 상기 주어진 데이터 객체에 있어서 상기 각 키맵 엔트리는 상기 키(key) 값 및 상기 주어진 데이터 객체의 저장된 복사에 대응하는 각 위치 표시자 값을 포함한다. 상기 시스템은 상기 웹 서비스 인터페이스로부터 상기 데이터 객체들로 접근하기 위한 상기 클라이언트 요청들을 수신하고, 상기 주어진 클라이언트 요청에 응답하여, 상기 키맵 예에 접근하여 상기 키 값에 대응하는 적어도 하나의 위치 표시자 값을 확인하고, 상기 적어도 하나의 조정자 값의 특정 하나에 있어서 대응하는 저장 노드에 접근하여 대응하는 복사를 검색하도록 구성되는 조정자(coordinator)를 포함할 수 있다.

[0007] 상기 시스템의 특정 실행에 있어서, 상기 웹 서비스 인터페이스는 상기 웹 서비스 프로토콜에 따라 클라이언트 요청들을 수신하여 데이터 객체들을 저장하도록 더 구성되고, 상기 데이터 객체들의 특정 하나를 저장하기 위한 상기 클라이언트 요청 중의 특정 하나는 상기 특정 데이터 객체에 대응하는 키 값을 포함한다. 상기 조정자는 상기 클라이언트 요청들을 수신하여 상기 웹 서비스 인터페이스로부터의 데이터 객체들을 저장하고, 상기 특정 클라이언트 요청에 응답하여 상기 조정자는 상기 특정 데이터 객체의 적어도 하나의 복사를 상기 적어도 하나의 대응 저장 노드에 저장하도록 더 구성된다. 상기 특정 데이터 객체의 주어진 복사의 저장에 응답하여 상기 저장 노드들의 주어진 하나는 상기 주어진 복사에 대응하는 위치 표시자 값을 상기 조정자로 반환한다.

도면의 간단한 설명

[0008] 도 1은 웹 서비스로서 저장 장치를 사용자들에게 제공하는 저장 모델의 일 실시예를 나타낸 블록도이다.

도 2는 저장 서비스 시스템 아키텍처의 일 실시예를 나타낸 블록도이다.

도 3은 저장 서비스 시스템 구성 요소의 물리적 배치의 일 실시예를 나타낸 블록도이다.

도 4는 저장 노드의 일 실시예를 나타낸 블록도이다.

도 5는 데이터 객체들을 저장 노드 내에 구성하도록 구성된 데이터 구조들의 일 실시예를 나타낸 블록도이다.

도 6은 객체 획득 동작을 수행하는 방법의 일 실시예를 설명하는 흐름도이다.

도 7은 객체 풋 동작을 수행하는 방법의 일 실시예를 설명하는 흐름도이다.

도 8은 객체 방출 동작을 수행하는 방법의 일 실시예를 설명하는 흐름도이다.

도 9는 객체 저장 공간을 재패킹하는 방법의 일 실시예를 설명하는 흐름도이다.

도 10은 키맵 인스턴스 데이터 구조 한 세트의 일 실시예를 나타낸 블록도이다.

도 11A 내지 도 11D는 키맵 인스턴스의 계층적 실행의 일 실시예를 나타낸 도면들이다.

도 12는 키맵 인스턴스 내의 계층적 층들 사이의 관계를 개략적으로 나타낸 블록도이다.

도 13은 키맵 엔트리 풋 동작을 수행하는 방법의 일 실시예를 나타낸 흐름도이다.

도 14는 키맵 엔트리 획득 동작을 수행하는 방법의 일 실시예를 나타낸 흐름도이다.

도 15A는 업 데이트 전파를 이용한 키맵 인스턴스를 동기화하는 방법의 일 실시예를 나타낸 흐름도이다.

도 16은 반복자 키맵 엔트리의 일 실시예를 나타낸 블록도이다.

도 17은 불균형 인덱스 데이터 구조의 일 실시예를 나타낸 도면이다.

도 18은 불균형 데이터 구조에 사용되는 인덱스 노드의 일 실시예를 나타낸 도면이다.

도 19는 계층화된 인덱스 데이터 구조의 일 실시예를 나타낸 도면이다.

도 20은 불균형 인덱스 데이터 구조를 순회하는 방법의 일 실시예를 나타낸 흐름도이다.

도 21은 지문 앤티-엔트로피 프로토콜 메시지를 처리하는 방법의 일 실시예를 나타낸 흐름도이다.

도 22는 필터 앤티-엔트로피 프로토콜 메시지를 처리하는 방법의 일 실시예를 나타낸 흐름도이다.

도 23은 발견 및 고장 검출 데몬(DFDD)의 일 실시예를 나타낸 도면이다.

도 24는 DFDD 예에 의해 유지될 수 있는 글로벌 동작 상태 머신의 일 실시예를 나타낸 도면이다.

도 25는 잡담 기반 프로토콜에 따른 DFDD 예들을 동기화하는 방법의 일 실시예를 설명하는 흐름도이다.

도 26은 저장 서비스 시스템 내 저장 등급들의 동작 방법의 일 실시예를 설명하는 흐름도이다.

도 27은 저장 노드들의 현재 상태 정보에 따른 데이터 객체의 적어도 하나의 복사를 저장하기 위한 기입 계획을 동적으로 결정하는 방법의 일 실시예를 설명하는 흐름도이다.

도 28은 하나 이상의 복사가 저장 노드들 중에 이미 저장된 객체에 대한 기입 계획을 동적으로 결정하는 방법의 일 실시예를 나타낸 흐름도이다.

도 29는 컴퓨터 시스템의 바람직한 실시예를 설명하는 흐름도이다.

본 발명을 특정의 실시예들이 도면의 예에 의하여 그리고 아래에 자세히 설명하였으나, 본 발명은 다양한 변경 및 대체 형태를 적용할 수 있다. 하지만, 도면들 및 상세한 설명은 본 발명을 개시된 특정 형태에 한정되는 것이 아니라, 특허 청구의 범위에서 청구하는 본 발명의 요지를 벗어나지 않는 범위 내에서 여러 가지 개량, 변경, 대체 또는 부가하여 실시할 수 있다.

발명을 실시하기 위한 구체적인 내용

[0009] 서론

[0010] 컴퓨팅 어플리케이션이 지리적으로 분산된 뿐만 아니라 집중되는 더 많은 데이터가 됨에 따라, 어플리케이션 데이터에 대한 신뢰성 있는 위치-독립 접근에 대한 요구가 증가하고 있다. 예를 들면, 저작, 저장 및 재생 어플리케이션과 같은 멀티미디어는, 멀티미디어 컨텐트의 양과 질이 향상함에 따라 데이터 저장양의 증가를 요구한다. 또한, 데이터를 저장하는 장치의 위치에 무관한 다양한 위치로부터의 어플리케이션 데이터에 접근하는 것이 바람직할 수 있다. 예를 들면, 많은 컴퓨터가 본질적인 디스크 기반 저장 양을 포함하지만, 이러한 저장 장치를 균일하고 편리한 방법으로 원격에서 접근하는 것은 기술적이고 안전상의 어려움을 나타낸다.

[0011] 자신의 내부 저장 자원에 유일하게 의존하는 개별 컴퓨터를 구성하거나 근거리 네트워크-기반 저장 자원(예를 들면, 네트워크 부착 저장 장치(NAS), 저장 영역 네트워크(SAN) 등)를 제공하는 것에 반해, 인터넷 접속 데이터 저장 서비스는 예를 들면, 웹 서비스(WS) 프로토콜과 같은 인터넷 기반 프로토콜을 통하여 클라이언트들에게 포괄적인 저장 서비스를 제공하도록 구성될 수 있다. 웹 서비스 프로토콜과 같은 인터넷 기반 프로토콜은 일반적으로 기본 소프트웨어 또는 하드웨어와 무관한 기능을 하는 점에서 플랫폼-독립적이다. 따라서, 웹 서비스와 같은 데이터 저장 능력을 제공하는 것은 다종의 어플리케이션들이 어플리케이션들의 호스트 시스템 내 또는 근거리 네트워크 상에 구현되는 저장 자원에 무관한 임의 저장 양에 간단하게 접근할 수 있도록 한다. 또한, 웹 서비스-접근성 저장은 인터넷 접근을 제공하는 어느 위치로부터 접근할 수 있도록 한다. 웹 서비스-접근성 저장은 상이한 장치들 또는 어플리케이션들에 의한 공통 데이터에의 원격 접근, 실행 중 개별 어플리케이션들에 의해 널리 분배된 데이터에의 원격 접근, 협력하여 작업하는 분배형 사용자들 사이의 데이터 공유로의 접근, 및 상기 분배형 사용자들 사이의 어플리케이션 결과 데이터의 제공과 같은 다수의 상이한 컴퓨팅 특징, 및 많은 다른 유사 특징의 실행을 촉진할 수 있다.

[0012] 웹 서비스 기반 저장 시스템에 사용될 수 있는 데이터 저장 모델의 일 실시예가 다음의 검토에서 설명된다. 이어서, 데이터 저장 모델에 따라 저장 서비스를 제공하도록 구성된 저장 서비스 시스템이 개시되고, 이것의 다양한 구성 요소가 자세하게 설명된다.

[0013] 저장 서비스 사용자 인터페이스 및 저장 모델의 요약

[0014] 사용자들에게 웹 서비스와 같은 서비스로서 데이터 저장 장치를 제공하는 저장 모델의 일 실시예가 도 1에 도시되어 있다. 도시된 모델에서, 저장 서비스 인터페이스(10)는 저장 장치에 대한 컴퓨터- 또는 사용자-대면 인터페이스로서 제공된다. 인터페이스(10)에 의해 사용자에게 제시되는 모델에 따라, 저장 서비스는 인터페이스(10)를 경유하여 접근할 수 있는 임의 수의 버킷(bucket)(20a~20n)으로서 구성될 수 있다. 각 버킷(20)은 임의 수의 객체들(30a~30n)을 저장할 수 있고, 다음에 저장 서비스 사용자에 의해 지정된 데이터를 저장할 수 있다.

[0015] 아래에 상세하게 설명되는 바와 같이, 실시예에 의하면, 저장 서비스 인터페이스(10)는 웹 서비스 모델에 따라 저장 서비스와 그 사용자 사이의 상호 작용을 지지하도록 구성될 수 있다. 예를 들면, 일 실시예에서, 인터페이스(10)는 서비스 클라이언트들에 의해 발생한 웹 서비스 호출이 처리를 위하여 지향될 수 있는 균일자원 위치 표시자(URL), 예를 들면, http:/storageservice.domain.com을 갖는 웹 서비스 종료점으로서 클라이언트들에 의해 접근될 수 있다. 일반적으로 말해, 웹 서비스는 하이퍼텍스트 전송 프로토콜(HTTP) 또는 다른 적당한 프로토콜의 버전과 같은 적어도 하나의 인터넷 기반 어플리케이션 층 데이터 전송 프로토콜을 포함하는 요청 인터페이스를 통하여 요청 클라이언트가 이용할 수 있는 어느 방식의 컴퓨팅 서비스라고 불릴 수 있다.

[0016] 웹 서비스는 다양한 인에이블링 서비스 프로토콜을 이용한 다양한 아키텍쳐 스타일로 수행될 수 있다. 예를 들면, 표현 상태 변경(REST)-스타일 웹 서비스 아키텍쳐에 의하면, (예를 들면, 요청된 서비스 방식, 사용자 자격 증명서, 동작할 사용자 데이터 등을 지정하는) 웹 서비스 호출에 적절한 파라미터들은 파라미터들로서 HTTP 획득(GET) 또는 풋(PUT) 명령과 같은 웹 서비스 종료점에 웹 서비스 호출을 일으키는 데이터 전송 명령에 지정될 수 있다. 어느 실행에 의하면, REST-스타일 웹 서비스 아키텍쳐는, 각 웹 서비스 호출이 외부 상태 정보를 참조하지 않고 호출을 처리할 필요가 있는 모든 정보를 포함할 수 있다는 점에서 중요하지 않다. REST-스타일 웹 서비스 아키텍쳐에 반해, 서류 기반 또는 메시지 기반 웹 서비스 아키텍쳐는 웹 서비스 종료점에 전송되고 종료점에 의해 디코딩되고 작용할 수 있는 서류로서 웹 서비스에 적절한 파라미터들 및 데이터를 인코딩할 수 있다. 예를 들면, 확장성 마크업 언어(XML) 또는 다른 적합한 마크업 언어의 버전이 사용되어 웹 서비스 요청 서류를 포맷팅할 수 있다. 일부의 실시예에 의하면, 요청 서류를 포맷팅하는데 사용되는 마크업 언어는 요청의 처리를 제어하는 파라미터들의 범위를 정하고, 다른 실시예에서는 마크업 언어 자체(예를 들면, 어떤 태그들)의 어떤 특징이 요청 처리의 양상들을 직접 제어할 수 있다. 또한, 일부의 실시예에서는, 결과로 얻어지는 서류가 예를 들면, 종료점에 의한 웹 서비스 요청의 처리를 촉진하기 위하여, 단일 객체 접근 프로토콜(SOAP)의 버전과 같은 다른 프로토콜 내에 요약될 수 있다.

[0017] 다른 프로토콜들은 또한 웹 서비스 아키텍쳐의 다양한 실시예에서 이용될 수 있다. 예를 들면, 웹 서비스 설명 언어(WSDL)의 버전은 인터페이싱 요구를 지연 클라이언트들에게 공개하기 위하여 웹 서비스 종료점에 의해 이용될 수 있다. 웹 서비스 종료점은 그들 자신이 유니버셜 설명, 발견 및 집적 (UDDI) 프로토콜의 버전과 같은 디렉터리 프로토콜을 통하여 지연 클라이언트들에게 알려지게 된다. 웹 서비스 인터페이스를 통한 컴퓨팅 서비스의 공급에 관한 다수의 다른 방식의 프로토콜들이 존재할 수 있고, 어느 주어진 웹 서비스 실행은 이러한 프로토콜들의 바람직한 결합을 사용할 수 있다.

[0018] 일부의 실시예에서는, 인터페이스(10)가 웹 서비스 인터페이스 대신 또는 외에 웹 서비스 인터페이스와 다른 인터페이스들을 지지할 수 있다. 예를 들면, 기업은 웹 서비스 프로토콜을 통하여 서비스에 접근할 수 있는 기업 외부의 클라이언트들, 및 다른 방식 인터페이스(예를 들면, 기업의 인트라넷에 주문을 받아 만들어진 소유 인터페이스)를 사용할 수 있는 기업 내 사용자들에 의해 저장 서비스를 실행할 수 있다. 일부의 실시예에서는, 인터페이스(10)는 저장 서비스의 어느 사용자가 서비스를 통하여 접근할 수 있는 다양한 방식의 인터페이스 프로토콜들 각각을 지지할 수 있다. 다른 실시예에 의하면, 인터페이스(10)의 다른 예는 각 별개 인터페이스 접근을 위하여 제공될 수 있다. 일부의 실시예에서는, (예를 들면, 서비스 요청들을 수신하여 대응하는) 클라이언트들과의 상호 작용을 처리하는 것에 관한 인터페이스(10)의 양상들은 저장 서비스의 일반적인 아키텍쳐(예를 들면, 서비스의 버킷 및 객체의 계층으로의 조직화)를 실행하는 양상들로부터 개별적으로 실행될 수 있다. 이러한 실시예에 의하면, (예를 들면, 서비스 프로토콜들을 경유하는) 클라이언트 상호 작용에 관련된 인터페이스(10) 부분은 도 2의 설명과 협력하여 아래에 상세히 설명되는 바와 같이, 기업 내부 사람들과 같은 어떤 사용자들에 의해 바이패스될 수 있다.

[0019] 도 1에 도시된 바와 같이, 인터페이스(10)는 버킷들(20)로의 접근을 저장 서비스 사용자들을 제공한다. 일반적으로 말해, 버킷(20)은 저장 서비스의 사용자에 관련된 객체 이름 공간의 근원으로서 기능을 한다. 예를 들면, 버킷(20)은 파일 시스템 디렉터리 또는 폴더와 유사할 수 있다. 일부의 실시예에 의하면, 개별 버킷들(20)은 또한 저장 서비스의 사용을 설명하는 기본 원리를 형성할 수 있다. 예를 들면, 사용자는, 사용자가 버킷들(20)에 의해 확립된 이름 공간 내에 계층형으로 존재하는 저장 자원(예를 들면, 객체들(30)의 저장)의 사용에 대한 청구될 수 있는 목적들을 청구하기 위하여, 적어도 하나의 버킷(20)과 협력할 수 있다.

[0020] 설명된 실시예에 의하면, 버킷(20a~20n) 각각은 관련된 메타 데이터(21a~21n) 및 각 접근 정책(23a~23n)을 포함한다. 일반적으로 말해, 메타 데이터(21)는 주어진 버킷(20)의 양상들 또는 특징을 설명하기 위하여 사용될 수 있는 어느 적합한 메타 데이터를 포함할 수 있다. 예를 들면, 메타 데이터(21)는 버킷의 생성일, 버킷 생성자의 식별자, 버킷이 버킷과 관련된 어느 객체들(30)을 가지는 지의 여부를 확인하는 정보, 및 다른 적합한 정보를 포함할 수 있다. 일부의 실시예에 의하면, 메타 데이터(21)는 버킷(20)에 관련된 객체들(30)의 전체 크기, 버킷(20)에 대한 사용자들의 접근 히스토리 및/또는 관련 객체들(30), 버킷(20)에 관련된 과금 히스토리와 같은 버킷(20)의 사용 특징을 나타내는 정보, 및 버킷의 현재 또는 역사적인 사용에 관한 어느 다른 적합한 정보를 포함할 수 있다. 일 실시예에 의하면, 각 버킷(20)은 사용자에 의해 지정되거나 저장 서비스에 의해 자동 할당될 수 있는 각각의 유일 식별자에 연관될 수 있다. 유일 식별자는 메타 데이터 내에 또는 버킷(20)의 개별 특징 또는 필드로서 저장될 수 있다. 일부의 실시예에서는, 주어진 버킷(20)은 명백한 기준들, 포인터들, 또는 주어진 버킷(20)에 관련된 객체들(30)에 대응하는 다른 정보를 포함할 수 없다. 오히려, 아래에 더욱 상세하게 설명되는 바와 같이, 객체들(30)의 위치 및 선택은 키맵으로 언급되는 개별 매핑 설비를 이용함으로써 수행될 수 있다.

[0021] 접근 정책(23)은 버킷(20)에 관련된 객체들(30)로의 접근을 제어하기 위하여 필요한 어느 정보를 포함할 수 있다. 접근 정책(23)은 버킷(20) 및 관련 객체들(30)을 접근하기 위해 허용된 클라이언트 또는 클라이언트들을 식별하는 정보, 및 그 용량을 포함할 수 있다. 예를 들면, 접근 정책(23)은 적어도 하나의 클라이언트용 식별자 및/또는 인증 증명서들(예를 들면, 비밀번호)를 저장하고, 또한 주어진 클라이언트가 객체들(30)을 변경하거나 단지 독출하기 위해 허용되었는 지를 지정할 수 있다. 접근 정책(23)은 또한 (예를 들면, 유니버셜 독출 접근을 허용하지만 지정된 클라이언트 또는 클라이언트 그룹에 대한 객체(30)의 기입 접근을 제한함으로써) 디폴트 또는 그룹-지향 정책들 또는 다른 원하는 보안 모델을 수행할 수 있다.

[0022] 설명된 실시예에 의하면, 주어진 버킷(20)은 각각 메타 데이터(31) 및 데이터(33)를 포함하는 적어도 하나의 객체(30)에 관련된다. 일반적으로 말해, 객체(30)의 데이터(33)는 어느 비트들의 결과에 대응할 수 있다. 객체(30) 내에 저장된 비트들에 의해 표현된 종류의 데이터는 저장 서비스에 투명할 수 있다. 즉, 비트들은 텍스트 데이터, 실행 프로그램 코드, 오디오, 비디오, 또는 영상 데이터, 또는 어느 다른 종류의 디지털 데이터를 의미할 수 있고, 저장 서비스는 객체(30)를 저장하고 조작하는데 다양한 데이터 종류 사이를 구별할 필요는 없다. 일부의 실시예에 의하면, 데이터(33)의 크기는 고정된 상한(예를 들면, 1기가바이트(GB))으로 제한된다. 한편, 다른 실시예에 의하면, 객체들(30)은 저장 서비스에 유용한 물리적 저장 자원에만 지배를 받는 크기로 스케일링 되도록 허용될 수 있다.

[0023] 버킷들(21)에 관련된 메타 데이터(21)와 마찬가지로, 메타 데이터(31)는 그 대응 객체(30)에 관한 어느 원하는 설명 정보를 저장하도록 구성될 수 있다. 예를 들면, 메타 데이터(31)는 대응 객체(30)가 생성되는 날짜 및/또는 시간에 관한 정보, 객체(30)의 크기, 객체(30)에 의해 저장된 데이터(33)의 종류(예를 들면, 다목적 인터넷 메일 확장(MIME) 표준에 의해 정의된 데이터 종류), 또는 설명 정보의 어느 다른 종류를 포함할 수 있다. 일부의 실시예에서는, 메타 데이터(31)는 대응 객체(30)와의 사용자 상호 작용을 나타내는 사용 및 히스토리 정보, 접근 정책 정보(예를 들면, 객체(30)를 가질 수 있는 다양한 접근 사용자들의 종류를 나타내는 허용 정보), 객체 비용 정보(예를 들면, 객체(30)에 관련된 과금율 또는 히스토리), 또는 어느 다른 적당한 정보, 또는 객체(30)에 기인하는 정보의 종류들의 결합을 저장할 수 있다. 어느 예에서는, 클라이언트는 메타 데이터(31)로서 저장될 객체 데이터에 따라는 메타 데이터를 제공할 수 있다. 한편, 다른 경우에는, 메타 데이터(31)는 저장 서비스 특징을 관리하는 시스템(예를 들면, 도 2에 도시되고 아래에 설명되는 저장 서비스 시스템)에 의해 발생되는 메타 데이터를 포함할 수 있다. 메타 데이터(31)의 일부의 또는 전체가 메타 데이터의 종류, 클라이언트의 접근 권한의 특정 제공, 또는 다른 적합한 인자에 따라 객체(30)에의 접근 권한을 갖는 클라이언트에 접근될 수 있다. 한편, 메타 데이터(31)의 아무것도 메타 데이터의 종류, 클라이언트의 접근 권한의 특정 제공, 또는 다른 적합한 인자에 따라 객체(30)에의 접근 권한을 갖는 클라이언트에 접근될 수 없다.

[0024] 일 실시예에 의하면, 개별 객체(30)는 2개의 구별되는 정보 항목: 키 또는 위치 표시자(locator)의 하나를 이용

하여 저장 서비스 시스템 내에서 식별될 수 있다. 일반적으로 말해, 비록 키들 및 위치 표시자들이 다른 방식으로 표시되더라도, 키들 및 위치 표시자들은 각각 저장 서비스 시스템의 이름 공간의 문맥 내에 전체로서 해석될 수 있는 수 문자 스트링들 또는 다른 종류의 심벌들을 포함할 수 있다. 일 실시예에 의하면, 키는 대응 객체(30)가 (예를 들면, 요청에 응답하여 클라이언트에 의해 새로운 객체를 저장하는) 특정 버킷(20) 내에 생성되는 시간에, 클라이언트에 의해 지정될 수 있다. 만일 사용자에 의해 어떤 키도 지정되지 않으면, 키는 저장 서비스 시스템에 의해 새로운 객체(30)에 할당될 수 있다. 이러한 실시예에 의하면, 특정 버킷(20)의 객체들(30)에 관련된 각 키는 버킷(20)의 이름 공간 내에 유일하도록 요구될 수 있다. 일반적으로 말해, 키는 대응 객체가 저장 서비스 시스템 내에 존재하는 한, 클라이언트가 대응 객체(30)에 접근할 수 있는 유효한 식별자로서 지속할 수 있다.

[0025] 주어진 버킷(20) 내 키들은 종래의 동작 시스템의 파일 시스템에 공통으로 파일 디렉터리 또는 폴더 이름 공간과 동일한 계층 객체 이름 공간을 발생하도록 사용될 수 있다. 예를 들면, 클라이언트에는 유일한 식별자 050739517을 갖는 특정 버킷(20)에 대한 객체 기입 및 독출 권한이 주어질 수 있다. 일 실시예에서, 그 후, 클라이언트는 버킷 내에 키들을 발생하기 위하여 웹 서비스 호출들을 어드레스 http://storageservice.domain.com/005739517에 발행할 수 있다. 예를 들면, 클라이언트는 객체(30)가 키 "나의 서류/이-메일/메시지.text"를 이용하여 특정 버킷 내에 생성되도록 지정하여, 객체(30)가 웹 서비스 호출을 이용하여 주소:http://storageservice.domain.com/005739517/나의 서류/이-메일/메시지.text에 어스레싱될 수 있도록 한다.

[0026] 일부의 실시예에서, 키에 의해 수반된 계층 구조가 객체 저장의 기초 계층에 필수적으로 반영될 수 없다는 것이 주목된다. 예를 들면, 일 실시예에서, 비록 객체들(30)에 관련된 키들이 계층을 수반할 수 있더라도, 주어진 버킷(20)에 관련된 객체들(30)은 평탄한(flat) 비-계층 방식으로 저장 서비스 시스템 내에 저장될 수 있다. 즉, 이러한 실시예에 있어서, 버킷들(20)은 다른 버킷들(20)을 계층적으로 포함할 수는 없다. 하지만, 다른 실시예에서, 비록 이러한 버킷들의 계층이 객체 키들에 의해 수반된 계층에 직접 매핑될 필요가 없더라도, 다른 버킷들(20) 내의 버킷들(20)의 계층 포함은 지지될 수 있다.

[0027] 일 실시예에서, 요청된 객체(30)의 기초 데이터(33)가 재생되거나 변경되기 전에, 키에 의해 확인되는 객체(30)로의 클라이언트의 접근 요청은 클라이언트 인증 과정들, 접근 제어 체크, 및/또는 (아래에 상세히 설명되는 되는 바와 같은) 매핑 처리에 종속될 수 있다. 예를 들면, 클라이언트는 클라이언트의 신원을 증명하기 위하여 비밀 번호 또는 다른 증명서를 제공하도록 요청될 수 있다. 또한, 일단 클라이언트의 신원이 확인되면, 요청된 버킷(20)에 연관된 접근 제어 파라미터들은, 요청된 키에 대한 접근을 보장하도록 확인된 클라이언트에 권한이 충분히 주어졌는 지를 결정하도록 평가될 수 있다. 이와는 반대로, 저장 서비스 시스템은 키들 보다는 차라리 위치 표시자에 의하여 객체(30)를 접근하기 위한 선택적인 방법을 지지할 수 있다. 일반적으로 말해, 위치 표시자는 저장 서비스 시스템에 알려진 모든 객체들(30) 사이의 객체(30)의 유일한 식별자를 전체적으로 나타낼 수 있다. 즉, 키가 특정 버킷(20)에 관련된 이름 공간에 유일할 수 있지만, 위치 표시자는 모든 버킷들(20) 내의 모든 객체들(30)의 글로벌 이름 공간 내에 유일할 수 있다. 예를 들면, 위치 표시자는 저장 서비스 시스템에 의해 발생되는 수 문자 스트링들을 포함하고 다른 위치 표시자들 중에 유일하다. 아래에 상세하게 설명되는 바와 같이, 일부의 실시예에서, 객체(30)의 다중 예들은 저장 서비스 시스템을 실행하기 위하여, 예들 들면, 데이터 리던던시 및 폴트 허용 한계를 증가하기 위하여, 사용된 물리적 저장 장치들을 통하여 복사될 수 있다. 이러한 실시예에 의하면, 유일한 위치 표시자는 주어진 객체(30)의 각 복사 예 마다 존재할 수 있다.

[0028] 일부의 실시예에, 객체(30)가 저장 서비스 시스템 내에 존재하는 한, 키는 객체(30)에 대한 접근이 유효하도록 보장될 수 있지만, 이러한 보장은 객체(30)의 어느 주어진 위치 표시자에 적용되거나 적용되지 않을 수 있다. 예를 들면, 객체(30)의 복사 예 (또는 복사)가 (예를 들면, 기초 저장 매체의 고장 또는 대체로 인하여) 다른 물리적 저장 위치로 이동하면, 비록 새로운 위치에서 객체(30)의 이동 예에 대응하는 어느 위치 표시자가 발생하여 사용되더라도, 특정 예를 언급하는 위치 표시자는 유효성을 종료할 수 있다. 키들과 위치 표시자들 간의 관계에 대한 상세한 설명은 아래의 키맵 시스템 구성 요소의 동작에 관한 검토에서 주어진다.

[0029] 키 기반 대 위치 표시자 기반 객체 접근의 예로서, 상기에서 주어진 키에 의해 참조된 객체(30)인 :http://storageservice.domain.com/005739517/나의 서류/이-메일/메시지.text는 저장 서비스 시스템 내에 저장되고 각각 포맷:http://storageservice.domain.com/locator/3859C89A208FDB5A의 위치 표시자에 의해 확인될 수 있는 적어도 하나의 예를 가질 수 있다. 특별한 실시예에서, 객체(30)에서 참조된 키는 특정 버킷(20)에 비례하도록 표현되지만, 위치 표시자 참조는 (비록 다른 형식의 위치 표시자 인코딩 또는 포맷이 사용될 수 있더라도) 글로벌 위치 표시자 공간 내에 절대 128-비트 16 진수로서 표현된다. 일 실시예에서는, 위치 표시자로 지

향된 클라이언트-발행된 웹 서비스 요청은 키 기반 웹 서비스 요청에 적용될 수 있는 인증, 접근 권한, 번역 또는 다른 단계들의 일부 또는 모두를 바이패스할 수 있다. 처리의 작은 개수의 층들로 인하여, 이러한 실시예에서는, 위치 표시자 기반 요청은 키 기반 요청보다 더 빠르게 처리될 수 있다. 하지만. 보안 측정은 위치 표시자 기반 요청에 대하여 바이패스될 수 있으므로, 클라이언트들은 감지 객체들(30)용 위치 표시자들이 (예를 들면, 암호화 또는 위치 표시자들을 전송 및 수신하기 위한 다른 보안 수단을 이용하여) 절충되지 않는 자신의 보장을 제공할 필요가 있다. 또한, 위치 표시자들의 지속성은 (예를 들면, 상기한 객체 예 이동의 경우에) 보장되지 않을 수 있으므로, 위치 표시자 기반 객체 접근을 수행하기 위해 선택하는 클라이언트는 새로운 위치 표시자들을 선점으로 기초하여 얻음으로써 또는 존재하는 위치 표시자가 더 이상 유효하지 않은 것을 발견하는 것에 응답하여 사용중에 유효하지 않게 되는 위치 표시자들의 가능성에 대해 내성이 필요하다.

[0030] 상기한 클라이언트 및 캐이비애트(caveats)의 저장 필요에 따라, 위치 표시자 기반 접근은 키 기반 접근에 관한 (예를 들면, 웹 서비스 요청 처리의 지연 및 처리량에서) 향상된 처리 성능을 제공할 수 있다. 예를 들면, 클라이언트는 특히 민감하지 않은 자주 접근한 객체(30)로 참조되는 위치 표시자 기반 접근을 사용하기 위해 선택할 수 있다. 일부의 실시예에서는, 위치 표시자 기반 접근은 개별 객체(30)에 기초하여 디스에이블되어, 이러한 객체에 접근하기를 원하는 클라이언트들이 키 기반 요청들을 사용하고, 따라서 어느 인증 및 이러한 요청에 관련된 접근 권한 제어에 따르도록 한다. 하지만. 심지어 위치 표시자 접근이 인에이블되는 객체(30)에 있어서, 유효한 위치 표시자가 부족한 부당하거나 기능이 불량한 클라이언트가 어느 주어진 객체(30)에 성공적으로 접근하는 랜덤한 기회만을 가질 수 있다. 이러한 기회는 큰 위치 표시자 이름 공간의 사용, 위치 표시자(예를 들면, 객체 데이터의 안정된 기술), 또는 다른 적당한 기술을 통하여 임의로는 생기지 않게 될 수 있다.

[0031] 저장 시스템 아키텍쳐 및 실행

[0032] 도 1에 도시된 바와 같은 웹 서비스 기반 저장 서비스를 실행하도록 구성될 수 있는 저장 서비스 시스템 아키텍쳐의 일 실시예가 도 2에 도시되어 있다. 도시된 실시예에서, 다수의 저장 클라이언트(50a~50n)는 네트워크(60)를 통하여 웹 서비스 플랫폼(100)과 상호 작용하도록 구성될 수 있다. 웹 서비스 플랫폼(100)은 저장 서비스 조정자(120)(간단히 조정자(120))의 적어도 하나의 예와 상호 작용하도록 구성될 수 있고, 다음에 적어도 하나의 키맵 인스턴스들(140) 및 비트 저장 노드들(160)과 상호 작용할 수 있다. 또한, 반복자(180)는 반복자 키맵 인스턴스(190)뿐만 아니라 비트 저장 노드들(160)과 상호 작용하도록 구성될 수 있다. 조정자(들)(120) 및 반복자(180)는 모두 노드 피커(nodepicker) 서비스(130)와 상호 작용할 수 있다. 도시된 실시예에서, 노드 피커(130)의 각 예, 키맵(140), 비트 저장 노드들(160), 및 반복자 키맵(190)은 발견 및 고장 검출 데몬(discovery and failure detection daemon)(DFDD)(110)의 각 예와 결합될 수 있다. 주어진 구성 요소의 적어도 하나의 예가 존재할 수 있는 경우, 구성 요소에 대한 참조가 단일 또는 복수로 만들어질 수 있다는 것이 주목된다. 하지만, 각 형태의 사용은 다른 것을 배제하도록 의도하지 않는다.

[0033] 다양한 실시예에 의하면, 도 2에 도시된 구성 요소들은 컴퓨터 하드웨어(예를 들면, 마이크로 프로세서 또는 컴퓨터 시스템), 또는 이 기술들의 결합에 의해 직접적으로 또는 간접적으로 수행할 수 있는 명령들처럼, 컴퓨터 하드웨어 내에서 직접 실행될 수 있다. 예를 들면, 도 2의 구성 요소들은 도 29에 도시되고 아래에 설명될 컴퓨터 시스템 실시예와 같은 다수의 컴퓨팅 노드들(또는 간단히, 노드들)을 포함하는 분배형 시스템에 의해 실행될 수 있다. 다양한 실시예에서, 주어진 저장 서비스 시스템 구성 요소의 기능은 특정 노드에 의해 실행되거나 다수의 노드들을 가로질러 분배될 수 있다. 일부의 실시예에서, 주어진 노드는 2개 이상의 저장 서비스 시스템 구성 요소의 기능을 실행할 수 있다. 도 2의 구성 요소들의 일반적인 기능의 요약 및 도 3에 도시된 바와 같은 저장 서비스 시스템의 바람직한 물리적 배치 다음에, 특정 저장 시스템 구성 요소들의 일부의 실시예에 대한 상세한 설명이 도 4 내지 도 28의 설명과 결합하여 아래에 제공된다.

[0034] 일반적으로 말해, 저장 클라이언트들(50)은 웹 서비스 요청들을 네트워크(60)를 통하여 웹 서비스 플랫폼(100)에 제공하도록 구성될 수 있는 어느 방식의 클라이언트를 포함할 수 있다. 예를 들면, 주어진 저장 클라이언트(50)는 웹 브라우저의 적당한 버전, 또는 웹 브라우저에 의해 제공된 실행 환경 내의 확장부로서 실행하도록 구성된 플러그인 모듈 또는 다른 방식의 코드 모듈을 포함할 수 있다. 대안으로, 저장 클라이언트(50)는 데이터베이스 어플리케이션과 같은 어플리케이션, 미디어 어플리케이션, 오피스 어플리케이션, 또는 지속성 저장 자원의 사용을 가능하게 하는 어느 다른 어플리케이션을 포함할 수 있다. 일부의 실시예에서, 이러한 어플리케이션은 모든 방식의 웹 기반 데이터용 전체 브라우저 지지를 필수적으로 실행하지 않으면서, 웹 서비스 요청들을 발생하여 처리하기 위한 (예를 들면, HTTP의 적당한 버전을 위한) 충분한 프로토콜 지지를 포함할 수 있다. 즉, 저장 클라이언트(50)는 웹 서비스 플랫폼(100)과 직접 상호 작용하도록 구성된 어플리케이션일 수 있다. 상기한 바와 같이, 저장 클라이언트(50)는 REST-스타일 웹 서비스 아키텍쳐, 서류- 또는 메시지-기반 웹 서비스 아키텍

쳐, 또는 다른 적당한 웹 서비스 아키텍쳐에 따라 웹 서비스 요청들을 발생하도록 구성될 수 있다.

[0035] 다른 실시예에서, 저장 클라이언트(50)는 어플리케이션들에 투명한 방법으로 웹 서비스 기반 저장에의 접근을 다른 어플리케이션들에 제공하도록 구성될 수 있다. 예를 들면, 저장 클라이언트(50)는 상기한 저장 모델의 변형에 따라 동작 시스템 또는 파일 시스템과 상호 작용하여 저장 장치를 제공하도록 구성될 수 있다. 하지만, 동작 시스템 또는 파일 시스템은 종래의 파일들, 디렉터리들, 및/또는 폴더들의 파일 시스템 계층과 같은 어플리케이션들에 다른 저장 인터페이스를 표현할 수 있다. 이러한 실시예에서, 어플리케이션들은 도 1의 저장 시스템 서비스 모델을 사용하기 위해 변경될 필요가 없을 수 있다. 대신, 웹 서비스 플랫폼(100)에의 인터페이스의 상세는 동작 시스템 환경 내에서 실행되는 어플리케이션들 대신에 저장 클라이언트(50) 및 동작 시스템 또는 파일 시스템에 의해 조정될 수 있다.

[0036] 저장 클라이언트들(50)은 웹 서비스 요청들을 네트워크를 통하여 웹 서비스 플랫폼(100)으로 전송하고 상기 웹 서비스 플랫폼(100)으로부터 응답을 수신한다. 다양한 실시예에서, 네트워크(60)는 클라이언트들(50)과 플랫폼(100) 사이의 웹 기반 통신을 형성하는데 필요한 네트워크 하드웨어 및 프로토콜의 어느 적당한 결합을 포함할 수 있다. 예를 들면, 네트워크(60)는 일반적으로 인터넷을 집합적으로 실행하는 다양한 통신 네트워크 및 서비스 제공자들을 포함할 수 있다. 네트워크(60)는 또한 근거리 통신망(LAN) 또는 광역망(WAN)과 같은 개인 통신망, 공용 또는 개인 무선망을 포함할 수 있다. 예를 들면, 주어진 클라이언트(50) 및 웹 서비스 플랫폼(100)은 자신의 내부 네트워크를 갖는 기업 내에 각각 제공될 수 있다. 이러한 실시예에서, 네트워크(60)는 인터넷과 웹 서비스 플랫폼(100) 사이뿐만 아니라 주어진 클라이언트(50)와 인터넷 사이의 네트워크 연결을 생성하는데 필요한 하드웨어(예를 들면, 모뎀들, 라우터들, 스위치들, 부하 평형 장치, 프록시 서버들 등) 및 소프트웨어(예를 들면, 프로토콜 스택들, 어카운팅 소프프웨어, 방화벽/보안 소프트웨어 등)을 포함할 수 있다. 일부의 실시예에서, 저장 클라이언트(50)은 공용 인터넷보다는 차라리 개인 네트워크를 이용하여 웹 서비스 플랫폼(100)과 통신할 수 있다. 예를 들면, 클라이언트들(50)은 저장 서비스 시스템과 동일한 기업 내에 제공될 수 있다. 이러한 경우에, 클라이언트들(50)은 개인 네트워크(60)(예를 들면, 인터넷 기반 통신 프로토콜을 이용할 수 있지만 공용으로 접근할 수 없는 LAN 또는 WAN)을 통하여 플랫폼(100)과 완전히 통신할 수 있다.

[0037] 일반적으로 말해, 웹 서비스 플랫폼(100)은 저장 서비스 시스템에 의해 저장된 객체(30)에 접근하기 위한 요청들과 같은 웹 서비스 요청들을 수신하고 처리하도록 구성된 적어도 하나의 서비스 종료점을 실행하기 위해 구성될 수 있다. 예를 들면, 웹 서비스 플랫폼(100)은 이전 예들에서 사용된 종료점 http://storageservice.domain.com을 실행하기 위해 구성된 하드웨어 및/또는 소프트웨어를 포함할 수 있어서, 종료점으로 지향된 HTTP 기반 웹 서비스 요청이 적당하게 수신 및 처리된다. 일 실시예에서, 웹 서비스 플랫폼(100)은 클라이언트들(50)로부터 웹 서비스 요청들을 수신하고 웹 서비스 요청들을 조정자(들)(120) 또는 처리를 위한 저장 서비스 시스템의 다른 구성 요소들로 전달하도록 구성된 서버 시스템으로서 구현될 수 있다. 다른 실시예에서, 웹 서비스 플랫폼(100)은 부하를 처리하는 대규모 웹 서비스 요청을 동적으로 관리하기 위해 구성된 부하 균형 및 다른 요청 관리 특성을 실행하는 다수의 별도 시스템(예를 들면, 클러스터 토폴로지에서)로서 구성될 수 있다.

[0038] 다양한 실시예에서, 웹 서비스 플랫폼(100)은 상기에서 상세히 설명된 바와 같이 REST-스타일 또는 서류 기반(예를 들면, SOAP 기반) 방식의 웹 서비스 요청들을 지지하도록 구성될 수 있다. 일 실시예에서, 플랫폼(100)은 저장 서비스 시스템에 의해 관리되는 엔티티 상에서의 다양한 동작을 지지하는 특정 웹 서비스 어플리케이션 프로그래밍 인터페이스(API)를 실행하도록 구성될 수 있다. 예를 들면, 플랫폼(100)에 의해 실행된 API는 (필터 패턴 또는 기준에 따라 임의로 필터링된) 버킷들(20) 또는 객체들(30)의 리스팅을 포함하는 버킷들 또는 객체들에서의 기본 클라이언트 동작, 버킷들(20) 또는 객체들(30)의 데이터 또는 메타 데이터의 검색, 및 버킷들(20) 또는 객체들(30)의 생성 또는 삭제를 지지할 수 있다. 일부의 실시예에서, API는 다중 버킷들(20) 또는 객체들(30)에서의 동작 배치 어플리케이션과 같은 더욱 정밀한 클라이언트 동작들을 지지할 수 있다.

[0039] 일부의 실시예에서, 클라이언트들의 웹 서비스 요청들용 어드레스 가능한 종료점(endpoint)으로서 기능하는 것에 더하여, 플랫폼(100)은 다양한 클라이언트 관리 특성을 실행할 수 있다. 예를 들면, 플랫폼(100)은 요청 클라이언트들(50)의 식별자, 클라이언트 요청들의 개수의 및/또는 횟수, 클라이언트들(50) 대신 저장되거나 검색된 객체들(30)의 크기, 클라이언트들(50)에 의해 사용된 전체 저장 대역폭, 클라이언트들(50)이 요청한 저장 등급, 또는 다른 측정 가능한 클라이언트 사용 파라미터를 추적함으로써 저장 자원을 포함하는 웹 서비스의 클라이언트 사용의 측정 및 어카운팅을 조정할 수 있다. 플랫폼(100)은 또한 회계 어카운팅 및 과금 시스템을 실행하거나 클라이언트 사용 활동을 보고하고 과금하기 위해 외부 시스템에 의해 질문되고 처리될 수 있는 사용 데

이터의 데이터베이스를 유지할 수 있다.

[0040] 일부의 실시예에서, 플랫폼(100)은 클라이언트들(50)로부터 수신된 요청들의 비율 및 방식에 대한 계량(metrics) 반영, 이러한 요청들에 의해 사용된 대역 폭, 이러한 요청들용 시스템 처리 지연, 시스템 구성 요소 사용(예를 들면, 네트워크 대역 폭 및/또는 저장 서비스 시스템 내 저장 사용), 요청들로부터 얻어지는 에러 비율 및 방식, 요청된 객체(30)의 특징(예를 들면, 크기, 데이터 크기 등), 또는 다른 적합한 계량과 같은 다양한 저장 서비스 시스템 동작 계량을 수집 및/또는 모니터링하도록 구성될 수 있다. 이러한 실시예에 의하면, 플랫폼(100)은 예를 들면, 평균 초과 시간 또는 다양한 분석을 받을 수 있는 특정 데이터 포인트들과 같은 집합체에서의 계량을 수집하도록 구성될 수 있다. 다양한 실시예에서, 이러한 계량은 클라이언트들(50)에게 보일 수도 있고 보이지 않을 수도 있는 방법으로 시스템 성능을 검사 또는 모니터링하기 위해 사용될 수 있다. 예를 들면, 일 실시예에서, 이러한 계량은 시스템 구성 요소들을 조정하고 유지하기 위해 시스템 관리에 의해 사용될 수 있다. 한편, 다른 실시예에서, 이러한 계량(또는 이러한 계량의 적절한 부분들)은 클라이언트들(50)에 노출되어 이러한 클라이언트들이 저장 서비스 시스템의 사용을 모니터링할 수 있도록 한다.

[0041] 일부의 실시예에서, 플랫폼(100)은 또한 사용자 인증 및 접근 제어 과정을 실행할 수 있다. 예를 들면, 주어진 버킷(20)에 연관된 특정 객체(30)에 접근하기 위한 주어진 웹 서비스 요청에 있어서, 플랫폼(100)은 요청에 관련된 클라이언트(50)가 주어진 버킷(20) 및 특정 객체(30)를 접근하도록 허용되었는 지를 확인하도록 구성될 수 있다. 플랫폼(20)은 예를 들면 주어진 버킷(20)에 관련된 증명서에 대한 식별자, 비밀번호, 또는 다른 증명서를 평가하고, 특정 객체(30)에 허용 가능한 동작을 지정하는 접근 제어 리스트에 대한 특정 객체(30)로 요청된 접근을 평가함으로써, 이러한 인증을 결정할 수 있다. 만일 클라이언트(50)가 버킷(20)에 접근하거나 객체(30)에서의 요청된 동작을 수행하기 위해(예를 들면, 클라이언트(50) 독출 접근 권한을 가지면서 객체(30)를 기입하는 것을 시도한다), 충분한 증명서를 가지지 않는다면, 플랫폼(100)은 에러 조건을 나타내는 응답을 요청 클라이언트(50)로 반환함으로써, 대응 웹 서비스 요청을 거절할 수 있다. 일부의 실시예에서는, 각 버킷(20) 및 객체(30)는 버킷 또는 객체로의 접근을 운영하는 관련된 접근 제어 정책을 가질 수 있다는 것이 예측된다. 이러한 접근 제어 정책은 메타 데이터(21 또는 31) 내의 접근 제어 정보의 기록 또는 리스트, 또는 메타 데이터(21 또는 31)와 구별되는 데이터 구조로서 저장될 수 있다.

[0042] 일부의 실시예에서, 도 2의 시스템과 같은 저장 서비스 시스템은 임의 크기의 객체들(30)을 지지할 수 있지만, 다른 실시예에서 객체들(30)은 청크(chunk) 크기로 불리는 어떤 최소 크기로 제한될 수 있다. 이러한 실시예에서, 클라이언트가 키와 결합하여 저장될 데이터를 제공하고 데이터가 청크 크기를 초과하는 경우, 플랫폼(100)은 청크 크기에 따라 데이터를 2개 이상의 청크로 분리하도록 구성될 수 있다. 일 실시예에서, 플랫폼(100)은 관련된 키 값을 갖는 각 객체(30)로서 각 청크를 발생하도록 구성될 될 수 있다. 클라이언트-공급된 키를 참조하는 접근 요청이 수행되는 경우, 플랫폼(100)은, 원래 클라이언트 데이터가 청크들로부터 재구성될 수 있는 방법으로 클라이언트-공급된 키의 기능으로서 각 청크에 대한 키 값들을 발생할 수 있다. 예를 들면, 플랫폼(100)은 클라이언트 데이터로부터 N개 청크를 발생하도록 구성될 수 있고, N개의 별개 패턴이 N개 청크가 발생되는 동일한 순서로 사전 편집으로 지시되는 경우, 플랫폼(100)은 N개의 별개 패턴을 클라이언트-공급된 키에 부착함으로써 청크들용 N개의 대응 키를 발생할 수 있다. 그 후, N개의 청크 각각은 아래에 설명되는 기술을 이용하여 별개 객체(30)로서 관리될 수 있고, 원래 데이터는 클라이언트-공급된 키가 접두부인 키 값들을 갖는 모든 객체들(30)을 리스팅하고 리스트된 순서로 객체들을 검색함으로써 재생될 수 있다. 일부의 실시예에서, 개별 청크들은 다른 청크들을 방해하지 않으면서 접근, 변경, 또는 제거되어, 단일 대형 객체(30)로서의 관리 데이터에 비례하여 시스템 성능을 향상시킬 수 있다. 일부의 실시예에서, 클라이언트(50)는, 클라이언트가 제공하는 데이터 객체가 청크들로 분리되어야 하는 지의 여부를 지정하도록 허용될 수 있다.

[0043] 도 2에 도시된 다수의 저장 서비스 시스템 구성 요소들을 갖는 경우처럼, 다른 구성 요소들로부터 웹 서비스 플랫폼(100)의 기능의 분리는 저장 서비스 시스템의 유지 및 전체 비례 축소 가능성을 향상시킨다. 예를 들면, 추가 하드웨어 및 소프웨어 자원들은 다른 작업에 할당된 자원들과는 무관하게 부가적인 웹 서비스 처리 부하를 관리하기 위해 특별하게 제공될 수 있다. 또한, 플랫폼(100)에 관련된 어느 자원의 파손의 영향은 특정 기능 영역으로 제한되어, 파손의 분리 및 분해를 촉진할 수 있다. 하지만, 일부의 실시예에서, 플랫폼(10)의 기능은 다른 구성 요소들에 통합될 수 있다는 것이 예측된다. 예를 들면, 조정자(들)(120)는 플랫폼(100)에 관련된 작업을 포함하도록 구성될 수 있다.

[0044] 또한, 웹 서비스 플랫폼(100)은 클라이언트들(50)이 저장 서비스 시스템의 특성에 접근할 수 있는 주요 인터페이스를 제시할 수 있지만, 이러한 특성에 유일한 인터페이스를 제시할 필요가 없다는 것이 주목된다. 예를 들면, 일부의 실시예에서, 조정자(들)(120)은 웹 서비스 인터페이스와 구별될 수 있는 선택적 API를 지지하도록

구성될 수 있다. 이러한 선택적 API가 사용되어, 예를 들면 저장 서비스 시스템을 제공하는 기업 내부 클라이언트가 웹 서비스 플랫폼(100)을 바이패스할 수 있도록 한다. 어떤 경우에, 플랫폼(100)의 어카운팅 및/또는 증명서 발행 서비스는 관리 클라이언트들과 같은 내부 클라이언트들에게는 불필요할 수 있다.

[0045] 조정자들(120)은 웹 서비스 플랫폼(100)과 저장 서비스 시스템의 다른 구성 요소들 사이의 동작을 조정하도록 구성될 수 있다. 일 실시예에서, 조정자들(120)의 주요 책임은 객체들(30)에 지향된 웹 서비스 요청들에 응답하여 객체들(30)용 객체 데이터(33) 및 메타 데이터(31)의 독출 및 기입 동작을 수행하는 것을 포함할 수 있다. 예를 들면, 아래에 상세히 설명되는 바와 같이, 주어진 객체(30)의 복사가 저장되고 다음으로 요청된 데이터를 독출하기 위해 특정 비트 저장 노드(160)로의 접근을 수행하는 경우, 객체 독출 접근은 비트 저장 노드들(160)을 나타내는 조정자들을 검색하기 위한 키맵 인스턴스(140)로의 접근을 수행하는 것을 포함할 수 있다. 마찬가지로, 객체 생성 및 변경은, 필요하다면 생성되거나 변경된 복사들의 조정자들을 반영하기 위하여, 객체들(30)의 다수의 복사를 비트 저장 노드들(160) 및 업데이트 키맵 인스턴스(140)에 저장하는 것을 포함할 수 있다. 일부의 실시예에서, 조정자들(120)은 키맵 인스턴스들(140) 및 비트 저장 노드들(160)에 대한 독출 및 기입 동작을 수행하도록 구성될 수 있다. 하지만, 일부의 실시예에서, 조정자들(120)은 생성 또는 변경 시에 객체(30)의 원하는 복사들의 전체 수를 생성하도록 동작하지 않을 수 있다는 것이 주목된다. 아래에 설명되는 바와 같이, 일부의 실시예에서, 조정자들(120)은 객체들(30)의 복사들(예를 들면, 2개의 복사)의 어떤 개수에 대한 기입을 완성하는 경우, 객체(30)로의 기입 동작은 완료된 것으로 고려될 수 있다. 객체(30)의 추가 복사는 반복자(180)에 의해 아웃-오브-밴드 또는 비동기식 동작으로 완료될 수 있다. 즉, 이러한 실시예에서, 객체 생성 또는 변경 동작의 인-밴드 또는 동기 부분은 영향을 주는 객체(30)의 복사들의 원하는 전체 수보다 작은 발생을 포함할 수 있다. 조정자(120)가 키맵 인스턴스들(140), 비트 저장 노드들(160), 및 다른 시스템 구성 요소들과 구별되는 구성 요소로서 설명되었지만, 일부의 실시예에서는 조정자(120)의 예가 다른 시스템 구성 요소(예를 들면, 단일 컴퓨터 시스템에 의해 실행될 수 있는 소프트웨어 구성 요소들로서)와 함께 실행될 수 있다는 것이 주목된다. 그래서, 비록 상기 설명은 비트 저장 노드(160), 키맵 인스턴스(140), 또는 다른 구성 요소에 데이터를 저장하거나, 비트 저장 노드(160), 키맵 인스턴스(140), 또는 다른 구성 요소로부터 데이터를 검색하는 조정자를 참조하더라도, 일부의 실시예에서는 이러한 처리가 공유 컴퓨터 시스템 자원 내에 이루어진다는 것으로 이해된다.

[0046] 도 1에 대한 상기한 바와 같이, 일부의 실시예에서, 저장 서비스 시스템은 다양한 객체(30)용 키들이 관리(예를 들면, 어카운팅, 과금), 보안, 또는 다른 목적용의 버킷들로 분류될 수 있는 버킷 기반 저장 모델을 포함할 수 있다. 일 실시예에서, 조정자들(120)은 클라이언트들(50)로부터의 대응 웹 서비스 요청들에 응답하여 다양한 버킷 관련 동작들을 처리하도록 구성될 수 있다. 예를 들면, 조정자들(120)은 다음의 버킷 동작들의 일부의 또는 모두를 수행하도록 구성될 수 있다.

[0047] - 버킷 생성: 버킷용 새로운 버킷 이름을 발생하고 저장한다.

[0048] - 비-공백 버킷 삭제: 관련 메타 데이터(21)를 포함하는 주어진 버킷(20) 및 주어진 버킷(20) 내의 객체들(30)에 관련된 모든 키들 삭제한다.

[0049] - 공백 버킷 삭제: 객체들(30)의 키들이 주어진 버킷들(20)에 관련이 없는 경우에만 주어진 버킷(20) 및 관련 메타 데이터(21)를 삭제하고, 그렇지 않은 경우에는 에러 조건을 반환한다.

[0050] - 버킷 데이터 기입: 데이터(예를 들면, 메타 데이터(21))를 존재하는 버킷(20)에 기입한다.

[0051] - 버킷 키값의 목록: (패턴, 정규 표현, 임의 문자에 따라 임의로 분류되거나 필터링된) 주어진 버킷(20)에 관련된 객체들(30)의 키들에 대한 목록을 만든다.

[0052] - 버킷의 목록: 주어진 가입자(예를 들면, 사용자 또는 클라이언트(50))에 관련된 버킷들(20)의 목록을 만든다.

[0053] 일부의 실시예에서, 조정자들(120)은 저 충돌 발생 가능성을 갖는 적당한 난수 알고리즘을 이용하여 새로이 생성된 버킷들(120)용 식별자들을 발생하도록 구성될 수 있다. 다른 실시예에서, 조정자들(120)은 예를 들면, 버킷 생성에 대한 클라이언트의 요청에 있는 경우, 존재하는 버킷 식별자들에 대한 유일성에 대하여 요청된 식별자들을 점검함으로써, 클라이언트-지정된 버킷 식별자들을 지지하도록 구성될 수 있다.

[0054] 상기한 바와 같이, 객체들(30)의 예들은 다른 비트스토어(bitstore) 노드들(160)을 가로질러 복사되어, 예를 들면 객체 데이터가 어느 주어진 노드(160) 또는 관련된 인트라 구조의 파손에도 생존하는 가능성을 증가시킨다. 저장 서비스 시스템 내 객체 복사는 설명된 실시예에서 다음과 같은 노드 피커(130) 및 반복자(180)에 의해 어

드레스될 수 있는 관리 및 최적화를 위한 수 번의 기회를 제공한다.

[0055] 조정자(120)가 객체(30)를 기입하기 위한 요청을 수신한 경우, 이에 대응하여 조정자(120)는 기입의 완료를 선언하기 전에, 객체(30)를 주어진 개수의 노드들(160)에 기입할 수 있다. 하지만, 객체(30)가 기입되어야 하는 노드들(160)의 개수 및 특정 선택은 다수의 다른 저장 정책 고려에 따라 변할 수 있다. 예를 들면, 기입 동작이 완료되는 것을 고려되기 전에, 객체(30)의 어떤 최소 수(예를 들면, 2 또는 3)의 복사가 성공적으로 기입될 것을 요구하는 것은 기입된 데이터가 가능한 파손에 의하여 지속할 수 있는 순서로 신중하게 이루어질 수 있다. 하지만, 또한 복사들의 최소 수를 저장하기 위해 선택된 노드들(160)이 파손의 다른 가능한 장소들, 또는 영역들 사이에 분배되는 것을 보장하는 것이 바람직하다. 예를 들면, 동일한 데이터 중심에 위치하는 노드들(160)은 지리적으로 분리된 노드들(160)보다 (자연 재해, 파워 고장 등과 같은 큰 재해로 인한) 더욱 동시에 파손할 수 있다.

[0056] 일반적으로 저장 노드 선택 로직으로 불릴 수 있는 노드 피커(130)는, 일 실시예에서, 다양한 저장 정책을 만족하도록 객체 독출 및 기입 동작용 노드들(160)을 선택하기 위한 알고리즘을 실행할 수 있는 조정자(120) 및 반복자(180)에 의해 접근할 수 있는 서비스로서 구성될 수 있다. 예를 들면, 상기한 객체(30)를 기입하는 경우에, 노드 피커(130)는 객체(30)가 기입되어야 하는 노드들(160)의 특정 시퀀스 또는 기입 계획을 전개하기 위하여 동작할 수 있다. 특정 기입 계획을 전개할 때, 노드 피커(130)는 만일 완료되면 기입 계획이 기입 동작에 적절한 모든 저장 정책을 만족시키는 성공의 적당한 기회- 예를 들면, 기입 계획에 지정된 노드들(160)이 객체(30)를 수용하여 이용할 수 있는 충분한 저장 자원을 가지도록 사실 동작적이고 예상되는 기회-를 가지는 것을 보장하도록 구성될 수 있다. 기입 저장 정책은 예는 다음의 정책을 가질 수 있다:

[0057] - 지속성 정책: 기입 계획이 성공적으로 완성하면, 객체(30)의 예들은 적어도 N개의 다른 노드들(160)에 저장될 것이다.

[0058] - 영역 다양화 정책: 가능하면, 기입 계획은 적어도 M개의 다른 영역 사이에 분배된 노드들(160)을 포함한다.

[0059] - 지역성 정책: 가능하면, 기입 계획은 요청하는 조정자(120)의 국부 영역에서의 노드들(160)에 우선권(예를 들면, 수에 있어서)을 줄 것이다.

[0060] - 부하 균형 정책: (예를 들면, "핫 노드들"을 회피하기 위해) 노드들(160) 사이의 기입 요청 트래픽에 대한 균등화를 시도한다.

[0061] - 공간 균형 정책: 노드들(160) 사이의 저장 자원 용량 이용의 균등화를 시도한다.

[0062] - 최저가 체인 정책: 기입 계획에서 노드 기입 동작의 시퀀스의 전체 가격(예를 들면, 네트워크 지연)의 최소화를 시도한다.

[0063] 다양한 실시예에서, 주어진 기입 계획을 공식화하는 경우, 노드 피커(130)는 정책들 및 목록에 포함되지 않은 다른 정책들을 고려하도록 구성될 수 있다. 또한, 다른 정책들은 다른 우선권을 갖도록 가중치가 부여될 수 있다. 일 실시예에서, 예를 들면, 지속성 정책은 모든 기입 계획이 만족해야 하는 의무적 정책일 수 있지만, 남은 정책들은 최상의 노력 기반에 따라 만족할 수 있다. 일부의 경우에, 일부의 저장 정책들은 다른 정책과 충돌할 수 있다. 예를 들면, 다른 영역들 사이의 객체 예들의 넓은 분배에 유리한 지역성 정책은 일반적으로 특정 영역 내 객체 예들을 국부화하는 것에 유리한 지역성 정책에 적합하지 않게 된다. 만일 객체 예들의 수가 충분히 크면, 양쪽 정책들을 만족시킬 수 있다. 예를 들면, 객체(30)의 5개의 예가 생성되는 경우, 2개의 별도 영역에 2개의 예를, 요청하는 조정자(120)의 3개의 별도 국부 영역 내에 3개 예를 저장하여, 지역성 및 영역 다양화 정책을 만족시킬 수 있다. 만일 기입 계획용으로 지정된 모든 정책을 만족시킬 수 없다면, 노드 피커(130)는 만족할 정책들을 우선 순위를 매기고 최고-노력 기입 계획을 생성하는 것을 시도할 수 있거나, 객체 기입이 만족스럽게 수행될 수 없는 것을 나타내는 요청하는 조정자(120)에 에러 표시를 반환할 수 있다.

[0064] 일부의 실시예에서는, 노드 피커(130)는 또한 객체(30)를 독출할 때 조정자들(120)을 지원할 수 있다. 예를 들면, 객체 독출 동작은 요청된 객체(30)를 최초에 또는 가장 최근에 기입한 조정자와는 다른 조정자(120)에 의해 요청될 수 있다. 그래서, 기입 조정자(120)에 대하여 국부적으로 저장될 수 있는 객체(30)의 예들은 독출 조정자(120)에 대하여는 국부적이 아닐 수 있다. 노드 피커(130)는 독출 조정자(120)용 최상 독출 성능을 제공할 수 있는 노드(160)를 확인하도록 구성될 수 있다. 예를 들면, 노드 피커(130)는 (예를 들면, 지리학적 거리 또는 네트워크 토폴로지에 의하여) 독출 조정자(120)에 가장 가깝고 최고 독출 대역 폭을 제공하는 노드(160)(예를 들면, 최저 부하 노드(160) 또는 저장 하드웨어의 더 높은 성능 등급을 갖는 노드(160))를 확인하거나 객체(130)를 독출하는 노드들로부터 노드(160)을 선택하기 위한 다른 성능 기준을 이용할 수 있다. 다른 실시예에서

는, 노드 피커(130)는 독출 조정자(120)에 대하여 독출 동작의 성능을 최적화하는 것보다 차라리, 시스템의 성능을 전체적으로 최적화하기 위하여(예를 들면, 글로벌 독출 처리율을 최소화하기 위하여), 동시 독출 동작을 전체적으로 계획할 수 있다.

[0065] 기입 계획을 전개하고 객체 독출 동작에 대한 조정자들(120)을 권유하기 위하여, 노드 피커(130)는 예를 들면, 노드들(160)의 동작 상태 및 유용한 자원들에 대한 노드들(160)의 상태를 모니터링하도록 구성될 수 있다. 일 실시예에서, 노드 피커(130)는 현재 동작하는 저장 서비스 시스템 내의 노드들(160)을 확인하기 위하여, (아래에 설명되는) DFDD(110)의 예와 상호 작용하도록 구성될 수 있다. 일단 노드 피커(130)가 동작 노드(160)를 알고 있으면, 각각 하나에 이용할 수 있는 자원(예를 들면, 저장 용량)을 확인하기 위한 노드들에게 질문할 수 있다. 일부의 실시예에서, 노드들(160)의 동작 및 자원 상태들이 시간에 따라 변하므로, 노드 피커(130)는 DFDD(110)를 경유하여 동작 상태 정보를 가끔 리플레쉬하고 결과로 얻어지는 노드들(160)을 폴링하여 노드들의 자원 상태 정보를 리플레쉬할 수 있다. 일부의 예에서는, 노드 피커(130)는 노드들(160)의 상태의 완전한 동기 뷰우를 가질 수 없다는 것이 주목된다. 예를 들면, 사실 노드 피커(130)에 의해 이용될 수 있도록 믿어진 특정 노드(160)가 상태 정보의 최종 업데이트 이래로 고장날 수 있다. 이러한 예들에서, 노드 피커(130)는, 노드 피커의 독출 또는 기입 계획이 조정자(120)에 의해 완료될 수 있는 것을 보장할 수 없다. 만일 조정자(120)가 노드 피커(130)에 의해 지정된 노드(160)에 접근할 수 없다면, 관련된 동작은 실패하고 조정자에 의해 완전히 재시도될 수 있거나, 조정자(120)는 노드 피커(130)와 협의하여 요청된 계획을 수정할 수 있다. 일부의 경우에, 기입 계획에서 지정된 노드(160)의 고장이 여전히 의무적 저장 정책들을 만족할 수 있도록 하면서 동작 또는 최고-노력 저장 정책들에만 나쁜 영향을 주면, 기입 계획을 완료하는 것이 허락될 수 있다. 이러한 일부 실시예에서, 반복자(180)는 아래에 설명되는 바와 같이, 추후에 만족스럽지 못한 저장 정책들을 만족시키기 위한 시도를 하도록 구성될 수 있다.

[0066] 일부의 실시예에서, 노드 피커(130)의 다중 예들은 저장 서비스 시스템을 통하여 배치될 수 있다. 예를 들면, 노드 피커(130)의 각 예는 조정자(120)의 각 예 마다 배치될 수 있다. 노드 피커(130)가 API를 경유하여 조정자들(120) 및 (반복자(180))로부터 접근할 있는 서비스로서 배치될 수 있지만, 이 구성은 필수적인 것이 아니다. 다른 실시예들에서, 노드 피커(130)의 기능은 조정자들(120) 및 반복자(180)의 예들 내에 직접 포함될 수 있다.

[0067] 상기한 바와 같이, 객체 데이터의 신뢰성 및 이용성은 저장 서비스 시스템을 통하여 객체들(30)을 복사함으로써 증가할 수 있다. 예를 들면, 객체들(30)의 예들 또는 복사들을 지리학적으로 분산된 시스템 내에 분배하는 것은 아마도 이러한 클라이언트들에 가까운 일부 객체 예들의 위치를 정함으로써 이러한 객체들(30)에 접근을 시도하는 마찬가지로 분산된 클라이언트들(50)의 성능을 향상시킬 수 있다. (객체 복사의 문맥에서, 용어 "인스턴스(instance)" 및 "복사(replica)"는 교체 가능하게 사용될 수 있다는 것이 주목된다). 또한, 객체 복사는 일반적으로 특정 객체 예의 파괴로부터 생기는 데이터 손실의 변화를 감소시킬 수 있다. 하지만, 일부의 실시예에서, 주어진 시점에, 객체(30)의 유효한 복사의 수는 복사의 원하는 또는 타켓 수보다 작을 수 있다. 예를 들면, 저장 서비스 시스템을 통하여 실시될 복사 저장 정책은 각 객체의 특정 타켓 수(예를 들면, 3 또는 어느 다른 적당한 수)가 어느 주어진 시간에 존재해야 하는 것을 지정해야 한다. 하지만, 주어진 객체(30)에 대하여, 유효한 복사의 실제 수는 다양한 이유 때문에 타켓 수보다 작을 수 있다. 예를 들면, 이전 유효 복사는 이것이 저장되는 장치의 고장으로 인하여 접근할 수 없게 될 수 있다. 대안으로, 일부의 실시예에서, 조정자(120)에 의해 기입되는 객체(30)의 예들의 수는 객체(30)별 복사들의 타켓 수보다 작을 수 있다. 예를 들면, 상기한 바와 같이, 객체의 예들은 노드 피커(130)에 의해 지정된 기입 계획에 따라 기입될 수 있고, 타켓 수보다 더 작은 예들을 요구하는 지속성 정책을 고려할 수 있다.

[0068] 일 실시예에서, 반복자(180)는, 각 객체(30)의 유효한 복사들의 수가 타켓 수를 만족시키는 지(예를 들면, 결정이 이루어지는 시간에 복사 수가 적어도 타켓 수인 지)를 결정하기 위하여 객체들(30)을 검사하도록 동작할 수 있다. 특히, 일 실시예에서, 반복자(180)는 각 객체(30)의 예들의 수 및 위치를 지정하는 기록을 연속적으로 반복하도록 구성될 수 있다. 예를 들면, 반복자(180)는 아래에 자세히 설명되는 키맵 인스턴스들(140)과 같이 반복자 키맵(190)을 참조하고, 객체 키들 및 복사된 객체 예들을 확인하는 대응 위치 표시자들 사의 매핑을 저장하도록 구성될 수 있다. (다른 실시예에서, 반복자(180)는 키맵의 전용 예보다 차라리 키맵 인스턴스들(140)의 하나를 참고할 수 있다). 일부의 실시예에서, 반복자(180)의 다중 예들은 키맵 공간의 다른 부분들을 동시에 검사하도록 구성되고, 저장 서비스 시스템에 의해 관리된 모든 객체(30)의 상태를 검사하기 위해 요구된 전체 시간 양을 감소시킬 수 있다.

[0069] 만일 반복자(180)가 유효한 복사들의 타켓 수가 주어진 객체(30)용으로 만족하지 않는 것으로 결정하면, 기입 동작을 주어진 객체(30)에 수행하는 조정자(120)에서와 동일한 방법으로, 반복자(180)는 주어진 객체(30)의 추

가 복사들을 기입하도록 구성될 수 있다. 예를 들면, 반복자(180)는 상기한 바와 같이 노드 피커(130)와 상호작용하여 추가 복사들을 생성하기 위한 기입 계획을 얻을 수 있다. 대안으로, 반복자(180)는 객체 복사들을 발생하기 위한 정책들을 반영하는 자체 알고리즘을 실행할 수 있다. 일부의 실시예에서, 반복자(180)는 추가 복사들이 요구되는 조건에 따라 객체(30)용 복사들을 생성하기 위한 다른 우선권들을 일치시킬 수 있다. 예를 들면, 반복자 키맵(10)의 목록에 포함된 위치 표시자의 타켓 수보다 작은 수의 위치 표시자를 갖는 객체(30)는 조정자(120)에 의해 최근에 기입될 수 있다. 반대로, 위치 표시자의 타켓 수의 일부가 유효하지 않은 객체(30)는 기초 저장의 고장을 나타낼 수 있다. 정책의 문제로서, 반복자(180)는 후자 전에 전자 또는 전자 전에 후자를 정정하는 것을 시도할 수 있다. 대안으로, 반복자(180)는, 이 조건을 일으키는 특정 환경에 무관하게 이 조건이 발생할 때는 언제나 유효 복사들의 타켓 수보다 작은 복사들을 갖는 어느 객체(30)용 추가 복사들을 발생하는 시도를 할 수 있다.

[0070] 상기한 바와 같이, 객체(30)의 저장의 전체 신뢰도는 예를 들면, 다른 영역들 또는 데이터 중심들 내에 객체 데이터의 복사들을 저장함으로써 증가할 수 있다. 하지만, 일부의 실시예에서, 각 복사는 객체 데이터의 정확한 복사에 대응할 필요는 없다. 일 실시예에서, 객체(30)는 (패리티, 에러 정정 코드 또는 다른 스킴(scheme)과 같은) 리던던시 인코딩 스킴에 따라 다수의 부분들 또는 "사아드(shards)"으로 분리되어, 객체 데이터가 모든 발생한 부분보다 작은 부분으로부터 재생성될 수 있도록 한다. 예를 들면, 객체(30)로부터 N개 부분을 발생하기 위한 다양한 스킴을 이용하여, 객체 데이터는 인코딩 스킴에 따라 (N-1)개 부분, N개 부분의 단순한 대다수, 또는 부분들의 다른 결합으로부터 재생성될 수 있다. 이러한 실시예에서, 객체(30)의 복사들은 발생된 부분들 또는 부분들의 어떤 결합에 대응할 수 있다. 이러한 접근은 객체 데이터의 다중 완전 복사들을 저장하는 것과 비교하여 데이터 저장 요구를 감소시키면서, 효과적인 폴트(fault) 허용 한계를 제공할 수 있다. 하지만, 일부의 실시예에서, 리던던시 인코딩 기술은 또한 객체 데이터의 완전한 복사와 결합하여 사용될 수 있다. 예를 들면, 객체 데이터의 개별 다중 완전 복사들은 상기한 바와 같이 적합한 리던던시 인코딩 기술에 따라 결정된 다중 부분들의 각 수집과 같은 노드들(60) 사이에 저장될 수 있다. 마지막으로, 일부의 실시예에서, 어떤 객체(30)는 복사 정도 또는 폴트 허용 한계를 갖도록 저장될 필요가 전혀 없다. 예를 들면, 저장 등급의 설명과 관련하여 아래에 설명되는 바와 같이, 클라이언트는 객체(30)가 아마도 고 폴트 허용 한계 정도를 지정하는 저장 클래스용보다 낮은 가격으로 적은 또는 무 폴트 허용 한계 정도를 지정하는 저장 등급에 따라 저장되도록 요청할 수 있다.

[0071] 일반적으로 말해, 키맵 인스턴스(140)은 객체(30)의 키들과 특정 예들의 위치 표시자들 또는 객체들(30) 복사들 사이의 관계 기록을 제공할 수 있다. 이러한 기록을 저장할 때, 키맵 인스턴스들(140)은 또한 객체들(30)이 저장 시스템 내에 복사되는 (예를 들면, 얼마나 많은 객체(30)의 예들이 존재하고 상기 예들이 어떻게 참조되는지) 정도를 반영한다. 비트 저장 노드들(160)은 위치 표시자들에 의해 확인되는 바와 같이 일반적으로 객체들(30)의 개별 예들용 저장 장치를 제공할 수 있다. 하지만, 주어진 노드(160)는 어느 다른 노드들(160)에 대한 예의 상태, 및 예의 위치 표시자와 대응 객체(30)의 키 사이의 관계를 알지 못한다. 즉, 일반적으로 말해, 키맵 인스턴스들(140)에 의해 유지된 상태 정보는 비트 저장 노드들(160)에 투과적(transparent)일 수 있다. DFDD(110)는 동작하여 노드들(160) 및/또는 키맵 인스턴스들(140)(그리고 실행되는 경우 반복자 키맵(190))의 동작 상태에 관한 상태 정보를 검출하고 통신하여, 조정자들(120) 및 반복자(replicator)(180)와 같은 DFDD(110)의 클라이언트들이 검출된 상태의 아마도 가능한 지연된 뷰우를 통하여 정밀한 정보를 얻을 수 있다. 이 구성 요소들은 아래에 상세히 다루어진다.

[0072] 도 2의 저장 서비스 시스템 아키텍쳐의 어떤 구성 요소들의 물리적 배치를 도시한 일 실시예가 도 3에 도시되어 있다. 도시된 실시예에서, 데이터 중심(300)은 2개 영역(310a 및 310b)을 포함하는 것으로 도시된다. 추가적으로, 영역들(310c 및 310d)은 데이터 중심(300)의 외부에 도시되고, 영역들(310a~310d)은 네트워크(60)를 통하여 연결된다. 영역들(310a~310d) 각각은 각 조정자 예(120a~120d)를 포함한다. 영역들(310a~310d)은 또한 비트 저장 노드들(160) 및 키맵 인스턴스들(140)의 다양한 결합, 그리고 도 3에 도시되지 않은 도 2의 다른 구성 요소들을 포함할 수 있다. 예를 들면, 영역(310a)은 4개의 비트 저장 노드들(160)을 포함하고, 영역(310b)은 3개의 비트 저장 노드들(160) 및 하나의 키맵 인스턴스(140)를 포함하고, 영역(310c)은 2개의 비트 저장 노드들(160)을 포함하고, 영역(310d)은 하나의 비트 저장 노드(160) 및 하나의 키맵 인스턴스(140)를 포함한다.

[0073] 상기한 바와 같이, 일 실시예에서, 영역들(310a~310d) 각각은 독립된 또는 약하게 상관된 고장(failure)의 위치로 고려될 수 있다. 즉, 어느 주어진 영역(310)이 고장을 경험할 가능성은 어느 다른 주어진 영역(310)의 고장 가능성에 무관하거나 비상관할 수 있거나, 고장 가능성의 상관은 임계양보다 작을 수 있다. 예를 들면, 2개 영역(310)은 동시에 고장날 10% 기회보다 작게 나타날 수 있다. 고장 상관 또는 임피던스는 어느 적당한 통계 또

는 확률적 기술을 이용하여 측정되고 다양한 방법으로 실행될 수 있다. 예를 들면, 영역들(310)은, 하나의 영역(310)에 영향을 미치는 재해(catastrophe)가 다른 영역에 영향에 미치지 않게 하면서, 개별 이용 그리드들로부터 또는 개별 이용 그리드들에 물리적으로 분리되거나 연결될 수 있다. 마찬가지로, 데이터 중심(300) 내에, 별개 영역들(310)은, 하나의 영역(310)이 다른 영역(310)의 고장에도 불구하고 동작을 지속할 수 있도록 하는 기능을 하는 독립된 백업 전력 공급, 네트워크 연결 또는 다른 리던던시 자원을 가질 수 있다.

[0074] 작지만 고장의 각 가능성들 사이의 0이 아닌 상관을 갖는 2개 영역(310)은 여전히 고장의 독립된 가능성을 가지는 것으로 불릴 수 있다. 예를 들면, 주어진 데이터 중심(300) 내의 2개 영역(310)이 각각 백업 전력, 냉각 등을 위한 강고하고(robust) 독립된 시스템들을 갖더라도, 충분한 크기의 재해(예를 들면, 전체 데이터 중심(300)을 파괴하기에 충분한 폭발)의 이벤트에서의 동시 고장에 영향을 받을 수 있다. 하지만, 2개 영역(310)이 동시에 고장나기에 충분한 이벤트의 가능성은 충분히 작을 수 있지만, 실제 목적을 위하여 고장의 독립된 가능성을 가진다고 한다.

[0075] 영역들(310)은 (도시안된) 계층의 추가 레벨들을 포함할 수 있다. 예를 들면, 일 실시예에서, 비록 어느 적당한 영역 구성이 사용될 수 있더라도, 영역들(310)은 랙(rack)들로 세분되고, 비트 저장 노드들(160)과 같은 개별 노드들로 더 세분될 수 있다. 일반적으로 말해, 영역들(310)은 영역 내에 배치된 저장 서비스 시스템 구성 요소들을 실행하기에 충분한 컴퓨팅 자원들을 포함할 수 있다. 예를 들면, 각 비트 저장 노드(160)는 도 4 내지 도 9의 설명에 관련하여 아래에 설명된 바와 같이, 여러 가지의 하드웨어 및 소프트웨어 구성 요소들을 포함할 수 있다. 마찬가지로, 각 키맵 인스턴스(140)는 도 10 내지 도 22의 설명에 관련하여 아래에 설명되는 바와 같이 구성된 다수의 컴퓨터 시스템을 통하여 실행될 수 있다.

[0076] 일부의 실시예에서, 웹 서비스 플랫폼(100), 조정자들(120), 노드 피커(130), 반복자(180), 및 DFDD(110)는 구성 요소들이 배치된 각 영역(310) 내의 분산 컴퓨팅 자원들을 통하여 실행될 수 있다. 예를 들면, 구성 요소들 각각은 각 컴퓨터 시스템에 의해 실행될 수 있는 지시 및 데이터의 한 세트로서 실행될 수 있다. 대안으로, 구성 요소들의 일부 또는 모두는 적어도 하나의 컴퓨터 시스템에서 동시에 실행할 수 있는 처리로서 실행될 수 있다. 일부의 실시예에서, 구성 요소들의 일부 또는 모두를 실행하는데 사용된 컴퓨팅 자원들은 비트 저장 노드들(160) 또는 키맵 인스턴스들(140)을 실행하는데 사용된 자원들에 공유될 수 있다. 예를 들면, 컴퓨터 시스템은 키맵(140)의 기능 및 조정자(120)의 기능의 일부를 실행하도록 구성될 수 있다. 일반적으로 말해, 개별 영역들(310) 내에 배치된 컴퓨팅 자원들을 가로질러 도 2의 구성 요소들의 적당한 분할이 이용될 수 있다. 도 3에 도시된 바와 같이, 다른 영역들(310)은 저장 서비스 시스템 구성 요소들의 다른 결합을 포함할 수 있고, 도시된 실시예는 제한을 위해서 라기보다는 설명하는 것으로 의도되었다는 것이 주목된다.

[0077] 또한, 다른 저장 서비스 시스템 구성 요소들은 어느 적당한 방식의 통신 프로토콜에 따라 통신할 수 있다. 예를 들면, 도 2의 어떤 구성 요소들이 분산 어플리케이션 또는 실행가능한 처리들로서 실행되는 경우, 이 구성 요소들은 동작 시스템 또는 플랫폼(예를 들면, 원격 과정 호출, 대기열, 메일박스, 소켓 등) 또는 표준 또는 소유 플랫폼-독립 통신 프로토콜들에 의해 제공될 수 있는 표준 프로세스 간 통신 기술을 이용하여 상호 통신할 수 있다. 이러한 프로토콜들은 주고받기/확인의 임의 레벨들, 에러 검출 및 정정, 또는 통신 구성 요소들용으로 요구되거나 희망될 수 있는 바와 같은 다른 통신 특성을 지지할 수 있는 상태 또는 무 상태 프로토콜들을 포함할 수 있다. 예를 들면, 일 저장 서비스 시스템 실시예에서, 상당한 정도의 구성 요소 간 통신은 전송 제어 프로토콜(TCP), 사용자 데이터그램 프로토콜(UDP), 또는 동일한 표준 또는 소유 전송 프로토콜과 같은 적당한 인터넷 전송 층 프로토콜을 이용하여 실행될 수 있다. 하지만, 저장 서비스 시스템 구성 요소들 간의 통신은 프로토콜 추출의 고 층들에서의 프로토콜들을 이용하여 실행될 수 있다는 것이 또한 기대된다. 예를 들면, 클라이언트들(50)과 웹 서비스 인터페이스(100) 간의 통신과 같이, 저장 서비스 시스템 구성 요소들 간의 통신은 예를 들면 HTTP를 통한 웹 서비스 호출들과 같은 어플리케이션 층 프로토콜들을 이용하여 행해질 수 있다.

[0078] 비트 저장 구성

[0079] 상기한 바와 같이, 도 2에 도시된 저장 서비스 시스템 아키텍쳐 실시예에서, 비트 저장 노드들(160)은 일반적으로 동작하여 저장 서비스 시스템에 의해 관리된 다양한 객체들(30)용 저장 장치를 제공할 수 있다. 비트 저장 노드(160)의 일 바람직한 실시예가 도 4에 도시되어 있다. 설명된 실시예에서, 비트 저장 노드(160)는 저장 리패커(repacker)(163) 및 논리 파일 입력/출력(I/O) 매니저(165)와 상호 작용하도록 구성된 저장 노드 관리(SNM) 제어기(161)를 포함한다. 매니저(165)는 적어도 하나의 저장 장치(169)를 관리하도록 구성된 파일 시스템(167)과 상호 작용하도록 구성된다. 다양한 실시예에서, SNM 제어기(161), 저장 리패커(163), 논리 파일 I/O 매니저(165) 또는 파일 시스템(167) 중 어느 구성 요소는 컴퓨터 접속가능 매체에 저장되고 아래에 설명되는 기능들을

수행하기 위한 컴퓨터에 의해 실행될 수 있는 명령들에 따라 실행될 수 있다. 대안으로, 이 구성 요소들의 어느 것은 전용 하드웨어 회로들 또는 장치들에 의해 실행될 수 있다.

[0080] 일 실시예에서, SNM 제어기(161)는 객체 저장 API를 노드(160)가 클라이언트에게 제공하고 API에 따른 동작을 수행하기 위하여 노드(160)의 다른 구성 요소들의 동작을 조정하도록 구성될 수 있다. 예를 들면, 제어기(120)는 객체들(30)을 SNM 제어기(161)에 의해 제시된 API를 통해 주어진 노드(160)에 저장하고, 주어진 노드(160)로부터 객체들(30)을 검색하도록 구성될 수 있다. 여기에 API 관리가 SNM 제어기(161)의 특성으로서 설명되었지만, 노드(160)의 API 처리 기능은 SNM 제어기(161)와 별개의 모듈 또는 구성 요소에서 실행될 수 있다는 것이 기대된다.

[0081] 객체 저장 API는 객체 풋(put), 획득, 및 방출 동작을 지지할 수 있다. 이러한 일 실시예에서, 일반적으로 저장 동작 또는 기입 동작으로 불릴 수 있는 객체 풋 동작은 동작의 변수(argument) 또는 파라미터로서 객체(30)가 데이터 및/또는 메타 데이터를 지정할 수 있다. 주어진 노드(160)에서 객체 풋 동작이 완료되는 경우, 풋 동작은 요청한 클라이언트로 아래에 자세히 설명되는 바와 같이 객체 예를 저장 서비스 시스템을 통하여 저장된 다른 모든 객체들(30)에 비하여 주어진 노드(160)에 유일하게 지정할 수 있는, 저장된 객체(30)에 대응하는 조정자를 클라이언트에게 요청하는 단계로 복귀할 있다.

[0082] 반대로, 일반적으로 독출 또는 검색 동작이라고 불릴 수 있는 객체 획득 동작은 파라미터로서 객체(30)의 위치 표시자를 지정할 수 있다. 동작이 완료되면, 획득 동작은 요청한 클라이언트로 지정된 위치 표시자에 대응하는 객체 데이터 및/또는 메타 데이터를 반환할 수 있다. 일부 실시예에서, 획득 동작은 요청한 클라이언트가 객체 데이터, 메타 데이터 및/또는 둘 다가 클라이언트에 반환되는 지를 지정할 수 있도록 하는 파라미터를 지지할 수 있다.

[0083] 획득 동작과 동일하게, 일반적으로 삭제 또는 제거 동작이라고 불릴 수 있는 객체 방출(release) 동작은 파라미터로서 객체(30)의 위치 표시자를 지정할 수 있다. 하지만, 객체 방출 동작이 완료되면, 방출 동작은 참조된 객체(30)에 이전에 관련된 방출 저장 자원들을 방출하고, 그 후 이러한 자원들은 다른 객체들(30)을 저장하도록 사용될 수 있다. 일 실시예에서, 일단 위치 표시자가 방출되면, 위치 표시자에 대하여 이어지는 획득 동작은 시간 주기 동안 성공 또는 실패할 수 있다. 즉, 방출 동작은 노드(160)에 대하여 재 사용용 저장 자원들을 방출할 수 있는 신호로서 역할을 할 수 있다. 하지만, 노드(160)는 즉시 그렇게 하려고 하거나, 통지하거나, 그렇지않으면 클라이언트와의 이러한 재사용을 동기화할려고 시도하지 않을 수 있다. 그래서, 방출한 후에 객체(30)에 접근하기 위한 클라이언트에 의한 연속된 시도들이 임의 주기 동안 성공할 수 있고, 이어서, 객체(30)는 통지없이는 접근할 수 없게 된다. 다른 실시예에서, 노드(160)는 객체 데이터가 여전히 유효한 지에 무관하게 이전에 방출된 위치 표시자로의 클라이언트 접근을 방지하도록 구성될 수 있다.

[0084] 다양한 실시예에서, 풋, 획득, 및 방출 동작은 어느 적당한 프로토콜에 따라 다른 파라미터들을 이용하고/하거나 다양한 상태, 에러 또는 다른 표시를 반환할 수 있다는 것이 예측된다. 예를 들면, 노드(160)에 저장될 요청된 객체(30)용 자원들이 불충분하거나 어떤 다른 이유로 풋 동작이 완료될 수 없다면, 풋 동작은 에러 조건을 반환할 수 있다. 또한, 일부의 실시예에서, 노드(160)의 객체 저장 API는 다른 동작을 포함할 수 있다는 것이 예측된다. 예를 들면, API는 복사 동작을 지지함으로써 객체 복사들의 생성을 촉진하도록 구성될 수 있다. 일 실시예에서, 복사 동작은, 타켓 노드(160)에 저장될 객체(30)의 데이터를 공급하는 대신, 요청한 클라이언트가 객체(30)의 위치 표시자를 다른 노드(160)에 지정할 수 있다는 것을 제외하고는, 풋 동작과 유사하게 동작할 수 있다. 그 후, 타켓 노드(160)는 지정된 노드(160)와 상호 작용하여 객체 데이터 및/또는 메타 데이터를 얻고, 타켓 노드에 비례하는 객체의 위치 표시자를 클라이언트로 반환할 수 있다. 다른 실시예에서, 노드(160)는 다른 적당한 동작을 객체들(30)에 지지할 수 있다.

[0085] 상기한 바와 같이, 풋, 획득 및 방출 동작을 실행하는 일부의 실시예에서, 존재하는 객체들(30)은 적절하게 변경될 수 없다. 차라리, 변경된 데이터를 포함하는 새로운 예를 기입한 후, 존재하는 예를 방출함으로써, 객체(30)의 예는 효과적으로 변경될 수 있다. 이러한 접근은, 예를 들면, 만일 객체(30)에서의 변경이 노드를 원래 크기보다 더 작게 또는 더 크게 하면 발생할 수 있는 분열(fragmentation) 또는 객체 재배치를 감소시킴으로써 노드(160)의 기초 관리 층들의 실행을 단순화시킬 수 있다. 웹 서비스 플랫폼(100)에 대하여 아래에 더욱 상세하게 설명되는 바와 같이, 일부의 실시예에서 저장 서비스 시스템은 큰 객체들의 청크들로의 분할을 지지할 수 있고, 청크들 각각은 별개 객체(30)로서 관리될 수 있다. 이 접근은 재 기입될 필요가 있는 청크들의 범위를 제한함으로써 자주 변경될 수 있는 대형 객체들을 처리할 때 노드(160)의 성능을 향상시킬 수 있다. 하지만, 다른 실시예에서, 노드(160)는 설명된 방출-재 기입 접근을 통하는 것보다 차라리 객체들(30)의 변경을 적절히 지지

할 필요가 있는 특성들을 포함할 수 있다는 것이 예측된다.

[0086] 설명된 실시예에서, 논리 파일 I/O 매니저(165)(또는 간단히 매니저(165))는 객체들(30)이 존재할 수 있는 적어도 하나의 논리적으로 접촉하는 저장 공간들을 SNM 제어기(161) 및 리패커(163)에 제공하기 위하여, 기초 장치 또는 파일 시스템 특징을 가상화하도록 구성될 수 있다. 예를 들면, 주어진 객체(30)는 저장 공간 내의 오프셋 및 이 오프셋으로부터의 범위(예를 들면, 데이터 및 메타 데이터를 포함하여 객체 크기에 의하여)에 따라 논리 저장 공간 내에 위치할 수 있다. 이러한 논리 저장 공간을 제공함으로써, 매니저(165)는 이러한 기초 저장의 상세한 실행과 무관하게 기초 저장의 균일한 뷰우를 SNM 제어기(161)에 제공할 수 있다.

[0087] 논리 저장 공간 내 객체들(30)에 대한 접근을 촉진하기 위하여, 일 실시예에서, 매니저(165)는 (또한 객체 인덱스라고 불리는) 객체 인덱스 값을 노드(160)에 저장된 각 객체(30)에 할당하도록 구성될 수 있다. 일반적으로 말해, 어느 주어진 객체(30)의 인덱스는 특정 노드(160) 내에서 유일할 수 있다. 예를 들면, 일 실시예에서, 객체(30)가 노드(160)에 저장될 때는 언제나 객체 인덱스로서 결과로 얻어지는 카운터 값을 이용하여, 카운터를 증가시킴으로 객체 인덱스는 얻어질 수 있다. (실시예들에서, 다중 객체 기입 동작들이 동시에 처리하도록 허용되는 경우, 카운터 증가는 예를 들면, 순차화를 통하여 동기화되어 객체 인덱스 값들이 일치하고 예견 가능한 방법으로 할당되는 것을 보장한다). 예를 들면, 64-비트 서명이 없는 정수와 같은 충분히 큰 카운터 값이 실제 목적용으로 모든 객체(30)가 유일한 인덱스 값으로 할당되는 것을 보장할 수 있다. 이전에 발생한 인덱스 값들이 반복되고 2^{64} 객체들이 저장된 후, 이러한 카운터가 진행될 수 있다. 하지만, 카운터가 진행된 후, 이전에 주어진 인덱스 값으로 할당된 객체(30)가 노드(60) 내에 여전히 존재하는 것은 있을 법하지 않으므로, 충돌은 발생하지 않는다. 객체 인덱스를 할당하는 어느 다른 적당한 방법이 또한 이용될 수 있다는 것이 주목된다. 아래에 설명되는 바와 같이, 객체 인덱스 값들은 노드(160)의 유일한 식별자와 결합하여 사용되어 조정자(120) 또는 특정 객체(30)를 참조하는 노드(160)의 다른 클라이언트들에 의해 사용될 수 있는 위치 표시자 값을 결정할 수 있다.

[0088] 매니저(165)는 상기한 유일한 객체 인덱스 값들을 사용하여 객체들(30)이 객체 접근을 촉진하는 방법으로 논리 저장 공간 내에 위치하는 경우에 관한 정보를 형성하도록 구성될 수 있다. 예를 들면, 도 5의 상부에 도시된 바와 같이, 일 실시예에서, 매니저(165)는 객체 인덱스 값들을 통하여 접근 준비용으로 구성될 수 있는 표 또는 동일한 데이터 구조를 저장하도록 구성될 수 있다. 설명된 실시예에서, 인덱스 표(500)는 다수의 엔트리(510)를 포함할 수 있고, 다수의 엔트리 각각은 객체 인덱스 필드, 오프셋 필드, 객체 크기 필드, 메타 데이터 크기 필드, 순환 리던던시 검사(CRC) 필드로 이루어지는 다수의 필드를 포함한다. 도 5의 하부에 도시된 바와 같이, 수개의 바람직한 객체들(30)에 있어서, 엔트리(510)의 오프셋 필드는 논리 저장 공간 내 대응 객체(30)의 시작의 위치를 지정하고, 객체 크기 및 메타 데이터 크기 필드들은 객체 데이터 및 메타 데이터가 오프셋 포인트로부터 연장되는 정도를 지정할 수 있다. 설명된 실시예에서, 비록 이 명령이 다른 실시예에서는 반대로 되더라도, 객체 데이터는 객체 메타 데이터에 선행한다. CRC 필드는 순환 리던던시 검사 알고리즘 또는 다른 방식의 체크섬 또는 하쉬 알고리즘의 결과를 저장할 수 있다. 객체(30)가 노드(160)에 최초로 저장되는 경우, CRC 필드에 초기에 저장된 값은 계산될 수 있다. 이어서, 객체(30)가 접근되는 경우, 동일한 알고리즘이 객체 데이터 및/또는 메타 데이터, 및 저장된 CRC 필드 값에 대하여 비교되어 결과로 얻어진 값에 적용될 수 있다. 비교 결과가 미스매치로 끝나면, 저장된 데이터의 보전은 해결될 수 있다. 다른 실시예에서, 엔트리들(510)이 도시된 것과는 다르거나 추가적인 필드들을 포함할 수 있다. 예를 들면, CRC 필드는 생략되거나 다른 곳에서 실행될 수 있다. 또한, 객체 및 메타 데이터의 절대 위치는 상대 오프셋에 더하여 또는 대신에 저장될 수 있다.

[0089] 리패커(163)는 논리 객체 저장 공간에서 동작하여 객체들(30)이 방출되고 이에 관련된 저장 자원들이 회수되는 것을 나타낼 수 있는 갭을 제거하도록 구성될 수 있다. 일 실시예에서, 리패커(163)는 논리 객체 저장 공간(예를 들면, 주기적으로 또는 연속적으로)을 스캐닝하여 이전 방출 동작에 의해 방출된 것과 같이 SNM 제어기(161) 및/또는 매니저(165)에 의해 표시된 객체들(30)을 확인하도록 구성될 수 있다. 그 후, 리패커(163)은 방출된 객체(30)의 인덱스 후에 나타나는 인덱스들을 갖는 이 객체들이 방출된 객체(30)의 제거를 반사하기 위해 업데이트되도록 하고, 이것은 논리 객체 저장 공간의 원래 포인트 쪽으로 이동하는 객체들(30)을 효과적으로 야기한다. 예를 들면, 도 5의 하부에서의 객체 N이 방출되면, 리패커(163)는 동작하여 객체 N+1에 대응하는 엔트리(510)가 객체 N+1의 새로운 오프셋 필드처럼 객체 N의 오프셋 필드를 반사하기 위하여 업데이트되도록 할 수 있다. 리패커(163)는 또한 객체 N에 관련된 엔트리(510)가 삭제되도록 하고, 이동을 반사하는 객체 N+1 뒤에 오는 객체들의 오프셋을 업데이트할 수 있다. 일 실시예에서, 매니저(165)는 객체 데이터 및 메타 데이터의 대응 이동이 논리 객체 저장 공간 및/또는 저장 장치(169)의 기초가 되는 파일들 또는 구조들 내에 발생하도록 할 수 있다. 그래서, 일부의 실시예에서, 리패커(163)의 동작은 기초 저장 구조들의 파손을 감소시키고 따라서 노

드(160)의 객체 접근 성능을 향상시킬 수 있다.

[0090] 일부의 실시예에서, 매니저(165)는 상이한 방식의 하드웨어 및 소프트웨어를 갖는 다른 다중 실행 플랫폼들에서 실행하도록 구성될 수 있다. 이 실시예에서, 하나 이상의 추가 층의 추상화(abstraction)가 매니저(165)에 의해 SNM 제어기(161) 및 그 클라이언트들에게 제공된 논리 객체 저장 공간 사이에 존재할 수 있다. 예를 들면, 설명된 실시예에서, 매니저(165)는 파일 시스템(167)에 의해 관리된 적어도 하나의 물리적 파일들로서 논리 객체 저장 공간을 실행하도록 구성될 수 있다. 일반적으로 말해, 파일 시스템(167)은 물리적 파일들로서 불리는 논리 유닛들에 데이터를 저장할 수 있는 논리 저장 장치에 다양한 방식의 물리적 저장 장치들(160)에 형성하도록 구성될 수 있다. 파일 시스템(167)에 의해 관리된 논리 저장 장치들은 사실 계층적일 수 있다. 예를 들면, 파일 시스템(167)은 물리적 파일들을 저장하고 접근하기 위해 조종될 수 있는 디렉터리들 또는 폴더들의 계층을 지지할 수 있다. 일반적으로 말해, 물리적 파일의 대응 데이터 및/또는 메타 데이터가 저장되는 경우, 파일 시스템(167)은, 주어진 물리적 파일 사이의 관계 및 저장 장치들(169)의 위치들을 추적하고 관리하도록 구성될 수 있다. 그래서, 일 실시예에서, 매니저(165)는 논리 객체 저장 공간의 파일 시스템(167)에 의해 할당된 적어도 하나의 물리적 파일들로의 매핑을 관리할 수 있다. 다음에, 파일 시스템(167)은 물리적 파일들의 저장 장치들(169)의 어드레스 가능한 위치들로의 매핑을 관리할 수 있다.

[0091] 비록 어느 주어진 동작 시스템이 기초 장치들(169)의 관리를 위해 상이한 특성들을 제공하는 다양한 상이한 파일 시스템들(167)을 지지하더라도, 파일 시스템(167)은 일반적으로 동작 시스템 내에 일체로 형성될 수 있다. 예를 들면, 다양한 버전의 마이크로소프트 윈도우즈가 파일 할당 표-32(FAT 32) 및 FAT16 파일 시스템 뿐만 아니라 NT 파일 시스템(NTFS)와 같은 파일 시스템들을 지지한다. 다양한 버전의 리눅스 및 유닉스 동작 시스템들이 ext/ext2 파일 시스템들, 네트워크 파일 시스템(NFS), 레이서 파일 시스템(ReiserFS), 급속 파일 시스템(FFS), 및 다수의 다른 시스템과 같은 파일 시스템들을 지지할 수 있다. 일부의 제3자 소프웨어 벤더들은 예를 들면, VERITASⓡ 파일 시스템(VxFS)과 같은 다양한 컴퓨팅 플랫폼들과 일체화되기 위한 소유 파일 시스템들을 제공할 수 있다. 다른 파일 시스템들은 기초 저장 장치들(169)을 관리하기 위한 다양한 특성용 지지를 제공할 수 있다. 예를 들면, 일부의 파일 시스템들은 장치 미러링(mirroring), 줄무늬 형성(striping), 스냅 사진 촬영(snapshotting), 또는 다른 방식의 가상화 특성을 실행하기 위한 지지를 제공할 수 있다.

[0092] 일부의 실시예에서, 추출된 추가 층들이 매니저(165)와 저장 장치들(169) 사이에 존재할 수 있다는 것이 주목된다. 예를 들면, 일부의 실시예에서, 볼륨 매니저 층은 파일 시스템(167)과 저장 장치들(169) 사이에 제공되고, 상기한 가상화 특성의 일부 또는 모두를 수행하도록 구성될 수 있다. 대안으로, 특정 저장 장치(169)는 하드 디스크 드라이브들의 독립형 어레이 또는 가상화 제어기를 갖는 다른 장치들로서 구성될 수 있다. 비록 내부적으로 가상화 제어기가 상기한 바와 같은 볼륨 매니저에 의해 또는 파일 시스템(167) 내에 지지될 수 있는 가상화 매핑과 유사하게, 장치의 저장 어드레스 공간의 디스크 드라이브들로의 임의 복합 매핑을 지지할 수 있다. 일부의 실시예에서, 도시된 것보다 더 작은 수의 층들이 존재할 수 있다는 것이 또한 주목된다. 예를 들면, 일부의 실시예에서, 매니저(165)는 파일 시스템(167)을 이용하지 않고, 저장 장치들(169), 예를 들면 물리적 미가공 장치들과 직접 상호 작용하도록 구성될 수 있다.

[0093] 일반적으로 말해, 저장 장치들(169)은 파일 시스템(167) 및/또는 매니저(165)에 의해 지지될 수 있는 어느 적당한 방식의 저장 장치들을 포함할 수 있다. 저장 장치들(169)은 보통 소형 컴퓨터 시스템 인터페이스(SCSI) 장치 또는 (또한 집적 드라이브 전자(IDE) 장치로 알려진) 부착 프로그래밍 인터페이스(ATAPI) 장치와 같은 하드 디스크 드라이브 장치들을 포함할 수 있다. 하지만, 저장 장치들(169)은 자기- 또는 광학-매체-기반 장치들, 대용량 고체 저장 장치들(예를 들면, 비휘발성- 또는 "플래쉬-메모리-기반 장치들"), 자기 테이프 등과 같은 어느 방식의 대용량 저장 장치를 포함할 수 있다. 또한, 저장 장치들(169)은 상기한 장치들 외에, 유니버셜 직렬 버스 또는 IEEE1394/Firewireⓡ 표준의 버전에 추종하는 인터페이스들과 같은 방식의 적당한 인터페이스를 통하여 지지될 수 있다.

[0094] 상기한 바와 같이, 저장 서비스 시스템 내에 저장된 객체(30)의 어느 주어진 예를 위하여, 대응 위치 표시자가, 예가 시스템 내의 모든 노드들(160)을 가로지르는 것을 유일하게 확인할 수 있다. 일 실시예에서, 위치 표시자는 매니저(165)에 의해 객체 예에 할당될 수 있는 객체 인덱스 값의 연결, 결합 또는 다른 기능, 그리고 객체 예가 저장되는 노드(160)에 대응하는 유일한 식별자 또는 "노드 ID"로서 발생될 수 있다. 예를 들면, 상기한 바와 같이, 64-비트 객체 인덱스 값은 64-비트 노드 ID와 결합하여 128-비트 위치 표시자를 산출할 수 있다. 비록 다양한 실시예에서 더 작은 또는 더 큰 수의 비트가 사용되어 위치 표시자들을 형성하더라도, 이러한 위치 표시자는 2^{64} 개의 유일 노드들(160) 각각이 2^{64} 개의 유일 객체 예를 저장할 수 있도록 한다.

[0095] 일 실시예에서, 노드 ID는 주어진 노드(160)에 대응하는 인터넷 프로토콜(IP) 어드레스와 같은 유일한 네트워크 어드레스의 시간 스탬프 또는 데이터 스탬프와의 연결 또는 결합을 통하여 형성될 수 있다. 예를 들면, 노드(160)는 IP 어드레스(예를 들면, 노드 시작/초기화에 또는 초기화 중이 아니면 노드 ID가 할당될 때) IP 어드레스가 할당되는 시간 또는 IP 어드레스가 유효하게 알려지는 동안의 시간을 반영하는 시간 스탬프와 결합하여, 노드 ID로 할당될 수 있다. 일반적으로 말해, 동일 IP 어드레스 공간에 속하는 2개의 별개 노드(160)는 어느 주어진 시간에 동일한 IP 어드레스에 유효하게 할당되지 않는다. 그래서, 노드의 IP 어드레스 및 시간 스탬프의 결합은 이 노드에서 유일한 식별자를 산출할 수 있다. 예를 들면, 비록 다른 비트 폭들이 사용될 수 있더라도, 32-비트 IP 어드레스는 (예를 들면, 어느 공통 기준 시간 후 경과한 초의 수를 나타내는) 32-비트 시간 스탬프와 연결 또는 결합하여 위에서 언급한 64-비트 노드 ID를 산출할 수 있다. 다른 기술이 노드 IP 어드레스들에 의존하지 않는 유일한 노드 ID들을 할당하기 위하여 사용될 수 있다는 것이 예측된다. 예를 들면, 이름 서버와 같은 중앙 권한은 상기한 바와 같이 노드(160) 내 객체 인덱스 값들의 할당과 동일한, 노드 ID들의 유일성을 보장하는 방법으로 요청 시에 노드 ID들을 위임할 수 있다.

[0096] 노드 ID가 노드의 IP 어드레스로부터 유도되는 실시예에서, 노드 ID는 어느 주어진 시간에 노드(160)의 현재 IP 어드레스를 반영하지 않을 수 있다는 것이 주목된다. 예를 들면, 노드 ID는 노드(160)가 리셋될 때까지 지속할 수 있지만, 노드의 IP 어드레스는 노드 ID의 발생 후에 변하거나 재 할당될 수 있다. 또한, 일부의 실시예에서, 노드 ID는 저장 클라이언트들(50) 또는 다른 지연 고의성 엔티티들이 위치 표시자들을 디코딩하는 것을 방지하여 실제 노드 IP 어드레스들을 결정하도록 하는 결정적인 방법으로 해시(hash)되거나 암호화되거나 어지럽게 될 수 있다(obfuscated).

[0097] 도 4의 노드(160)의 예에 대한 획득, 풋, 및 방출 동작의 바람직한 실시예들이 도 6 내지 도 8에 설명된다. 먼저 도 6을 참조하면, 획득 동작이 조정자(120) 또는 다른 클라이언트로부터 노드(160)에서 수신되는 경우, 블록(600)에서 시작한다. 예를 들면, 조정자(120)는 획득 동작을 상기한 바와 같은 노드 ID 및 객체 인덱스 값을 갖는 특정 위치 표시자에 적용할 수 있다. 예를 들면, 노드 ID가 타켓 노드(160)의 현재 IP 어드레스를 반영하면, 노드 ID가 사용되어 획득 동작을 적합한 노드(160)로 직접 라우팅할 수 있다. 대안으로, 아래에 설명되는 DFDD(110)와 같은 디렉터리 서비스가, 위치 표시자의 노드 ID가 획득 동작이 적합한 노드(160)로 라우팅될 수 있는 어드레스 가능한 종료점 및 목적지로 분해하는데 사용될 수 있다.

[0098] 일단 노드(160)에 의해 수신되면, 획득 동작이 처리되어 노드(60)의 논리 객체 저장 공간 내 타켓 객체 예의 범위를 확인할 수 있게 된다(블록 602). 예를 들면, 제어기(161)는 획득 동작을 수신하여 매니저(165)로 보낸다. 다음으로, 매니저(165)는 인텍스 표(500)를 접근하기 위하여 획득 동작에 의해 참조된 위치 표시자의 객체 인덱스부를 사용하여 논리 객체 저장 공간 내 원하는 객체 예의 위치를 얻을 수 있다. 예를 들면, 객체 예가 시작하는 경우, 매니저(165)는 오프셋을 논리 객체 저장 공간에서 얻고, 오프셋으로부터 객체 예의 길이를 얻을 수 있다. 일부의 실시예에서, 획득 동작은 객체 데이터, 메타 데이터, 또는 둘 다가 요구되었는 지를 지정할 수 있다. 이러한 실시예에서, 매니저(165)는 요청된 데이터에 적절한 논리 객체 저장 범위를 결정할 수 있다. 예를 들면, 만일 객체 데이터 및 메타 데이터 둘 다 요구되면, 매니저(165)는 객체 데이터 크기 및 메타 데이터 크기 둘 다를 이용하여 검색될 객체 오프셋으로부터 범위를 결정할 수 있다. 상기한 바와 같이, 다른 실시예들에서, 객체 예들의 저장 범위는 논리 객체 저장 공간 내 상대 오프셋들보다 차라리 절대 위치들을 통하는 것과 같은 다른 방법으로 매니저(165)에 의해 저장 및 관리될 수 있다.

[0099] 논리 객체 저장 공간 내 객체 범위는 물리적 파일 저장 공간 내 적어도 하나의 대응 파일 내의 범위로 매핑될 수 있다(블록 604). 예를 들면, 매니저(165)는 논리 객체 저장 공간을 파일 시스템(167)에 의해 관리된 적어도 하나의 파일로 매핑하고, 예를 들면 적합한 파일 접근 동작을 파일 시스템(167)에 적용하고, 독출될 이름 파일들 내 위치들 또는 오프셋들뿐만 아니라 적어도 하나의 파일 이름을 참조함으로써 원하는 객체 범위에 대응하는 데이터를 얻을 수 있다. 대체 실시예에서, 제어기(161)은 매니저(165)에 의해 관리된 논리 블록 저장 공간 특성을 바이패스하도록 구성될 수 있고, 대신 파일 시스템(167)에 의해 관리된 물리적 파일들과 직접 상호 작용할 수 있다.

[0100] 그 후, 물리적 파일들에 대한 참조는 장치-상대 요청들로 매핑될 수 있다(블록 606). 예를 들면, 파일 시스템(167)은 논리 블록 어드레스들(LABs) 또는 장치 기하학(예를 들면, 실린더, 섹터 및/또는 헤드)에 특정된 어드레스들과 같은 저장 장치(들)(169)의 특정 어드레스 가능한 위치로 독출 요청을 발생하도록 구성될 수 있다. 상기한 바와 같이, 일부의 실시예에서, 매니저(165)는 파일 시스템(167)을 바이패스하고 저장 장치(들)(169)를 직접 관리하도록 구성될 수 있다.

[0101] 그 후, 요청된 객체 데이터는 저장 장치(들)(169)로부터 검색되어(블록 608), 요청한 클라이언트에 반환될 수 있다(블록 610). 예를 들면, 검색된 데이터는 도 4에 도시된 요청 계층을 통하여 되돌려 주거나 저장 장치(들)(169) 또는 파일 시스템(167)으로부터 제어기(161)로 직접 되돌려 요청한 클라이언트에 전달될 수 있다.

[0102] 도 7에 도시된 바와 같이, 일 실시예에서, 풋 동작이 조정자(120) 또는 다른 클라이언트로부터 노드(160)에 수신되는 경우, 도 6의 블록(600)용으로 상기한 것과 동일한 방법으로, 풋 동작이 블록(700)에서 시작할 수 있다. 예를 들면, 조정자(120)는 풋 동작들을 노드 피커(130)에 의해 발생한 기입 계획으로 지정된 노드들(160)에 적용할 수 있다. 획득 동작과는 반대로, 풋 동작은 저장될 객체 데이터 및/또는 메타 데이터를 포함하고, 데이터 및/또는 메타 데이터의 길이를 지정하는 추가 파라미터들을 임의로 포함할 수 있다.

[0103] 일단 노드(160)에 의해 수신되면, 풋 동작은 논리 객체 저장 공간 내 객체 예용 저장 범위를 할당하도록 처리될 수 있다(블록 702). 일 실시예에서, 매니저(165)는 객체 인덱스 값을 새로운 객체 예에 할당하고 새로운 객체 예의 오프셋을 지정하는 새로운 엔트리(510)를 인덱스 표(500)에 기록하도록 구성될 수 있다. 예를 들면, 새로운 엔트리의 오프셋은 최대 인덱스 값을 갖는 존재하는 객체 예의 저장 범위(예를 들면, 오프셋 및 길이)에 비례하여 결정될 수 있다. 새로운 객체 예의 데이터 및/또는 메타 데이터의 길이가 풋 동작에서의 파라미터들로서 지정되지 않으면, 매니저(165) 또는 제어기(161)는 새로운 엔트리(510)에 포함하기 위하여 이들을 계산하도록 구성될 수 있다.

[0104] 논리 객체 저장 공간 내에 새로이 할당된 범위는 물리적 파일 저장 공간 내의 적어도 하나의 대응 파일 내 범위로 매핑될 수 있다(블록 704). 예를 들면, 새로운 객체 예용으로 할당된 범위는 적어도 하나의 존재하는 물리적 파일의 끝에 첨부되거나, 그렇지 않으면, 존재하거나 새로이 할당된 물리적 파일들 내에 위치될 수 있다. 그 후, 물리적 파일 범위는 (예를 들면, 획득 동작용의 상기한 것과 동일한 방법으로 파일 시스템(167)에 의해 저장 장치 범위로 매핑되고(블록 706), 객체 예 데이터 및/또는 메타 데이터가 저장 장치(들)(169)로 저장될 수 있다(블록 708).

[0105] 데이터 및/또는 메타 데이터가 저장 장치(들)(169)에 성공적으로 기입되는 것을 확인한 경우, 저장된 객체 예에 대응하는 위치 표시자는 요청한 클라이언트에 반환될 수 있다(블록 710). 예를 들면, 매니저(165)는 발생한 객체 인덱스 값을 노드(160)의 노드 ID에 첨부하도록 구성될 수 있고, 파일 시스템(167)으로부터 물리적 파일 기입 동작이 성공적으로 완료된 것이 표시되는 경우, 객체 위치 표시자로서 결과로 얻어지는 값을 반환할 수 있다.

[0106] 도 8에 도시된 바와 같이, 일 실시예에서, 방출 동작이 조정자(120) 또는 다른 클라이언트로부터 노드(160)에서 수신되는 경우, 방출 동작은 도 6의 블록(600)용의 상기한 것과 동일한 방법으로, 블록(800)에서 시작할 수 있다. 비록 다른 실시예에서, 다른 변수들이 공급될 수 있더라도, 방출 동작은 방출될 객체 예의 위치 표시자를 간단히 지정할 수 있다.

[0107] 획득 동작과 같이, 일단 노드(160)에 의해 방출 동작이 수신되면, 방출 동작은 노드(160)의 논리 객체 저장 공간 내 타켓 객체 예의 범위를 확인하도록 처리될 수 있다(블록 802). 예를 들면, 제어기(161)는 방출 동작을 수신하여 매니저(165)로 전송한다. 다음에, 매니저(165)는 참조된 객체 예의 대응 엔트리(510)를 확인하기 위하여, 방출 동작에 의해 참조된 위치 표시자의 객체 인덱스부를 이용하여 인덱스 표(500)에 접근할 수 있다. 그 후, 참조된 객체는 방출됨에 따라 표시될 수 있다(블록 804). 예를 들면, 매니저(165)는 엔트리(510)의 오프셋 또는 다른 필드를 음수와 같은 불법 값으로 설정하도록 구성될 수 있고, 엔트리가 더 이상 유효하지 않다는 것을 표시할 수 있다. 그 후, 확인은 객체가 방출되는 것을 나타내는 요청한 클라이언트로 반환될 수 있다(블록 806).

[0108] 상기한 바와 같이, 객체 예는 방출되는 경우, 객체 예에 관련된 저장 자원들은 즉시 자유롭게 되거나, 고정되거나, 다른 용도로 재 할당되지 않을 수 있다. 일 실시예에서, 오히려, 방출 동작에 대하여 비동기식으로 동작하는 독립 처리가 자원들을 교정할 때까지 자원들은 지속할 수 있다. 도 9는 예를 들면, 저장 리패커(163)에 의해 실행될 수 있는 것과 같이, 이러한 처리의 일 실시예의 동작을 설명한다. 블록 900에서, 노드(160)에 저장된 특정 객체 예에 대응하는 객체 인덱스 엔트리가 선택될 수 있다. 예를 들면, 리패커(163)는 엔트리들에 저장된 객체 인덱스 값들에 따라 시퀀스 순서로 인덱스 표(500)로부터 인덱스 엔트리들(510)을 선택하도록 구성될 수 있다. 이어서, 선택된 엔트리는 대응 객체 예가 방출되었는 지를 판단하기 위해 검사될 수 있다(블록 902). 예를 들면, 리패커(163)는, 필드가 음의 값 또는 어느 다른 값과 같이 대응 객체 예가 방출되는 것을 나타내는 값으로 설정되었는 지를 확인하기 위해 오프셋 필드 또는 다른 필드를 점검할 수 있다.

[0109] 선택된 객체가 방출되지 않으면, 다른 객체가 선택될 수 있는 경우, 동작을 블록 900으로 복귀할 수 있다. 선택된 객체가 방출되면, 논리 객체 저장 공간은 리패킹되어 방출된 객체에 대응하는 저장 자원들을 교정한다(블록 904). 예를 들면, 리패커(163)는 논리 객체 저장 공간 내 방출된 객체 뒤에 오는 객체 예들의 인덱스 엔트리들(510)을 조정하여, 이러한 제1 객체 예의 오프셋은 방출된 객체의 오프셋으로 설정되도록 구성될 있고, 이러한 다음 객체 예의 오프셋은 제1 객체 예 및 기타의 데이터 크기, 메타 데이터 크기, 및 오프셋의 기능으로서 설정된다. 하지만, 일부의 실시예에서, 새로운 객체가 시험용으로 선택되기 전에, 방출된 객체 예 다음의 모든 객체 예들이 리패킹될 필요는 없다. 예를 들면, 리패킹은 객체 선택과 인터리브(interleave)되어, 마주친 각 객체가 시험용으로 선택된 경우, 각 객체가 리패킹될 수 있다.

[0110] 일부의 실시예에서, 매니저(165)는 논리 객체 저장 공간의 리패킹에 응답하여 물리적 파일 저장 공간 내에 유사한 리패킹 또는 통합을 수행할 수 있다. 예를 들면, 매니저(165)는 논리 객체 데이터 범위가 다른 물리적 파일 데이터 범위로 재매핑되도록 할 수 있다. 동일하게, 일부의 실시예에서, 파일 시스템(167)은 물리적 파일 저장 공간의 리패킹에 응답하여 저장 장치(들)(169) 사이의 유사 리패킹 또는 통합을 수행할 수 있다. 다른 실시예들에서, 물리적 파일 저장 공간 또는 저장 장치들 자체의 리패킹은 리패커(163)에 의해 초기화된 논리 객체 저장 공간 리패킹과 무관하게 발생할 수 있다. 예를 들면, 매핑된 장치 저장 범위가 저장 장치의 접근 패턴에 비례하여 대부분 또는 전체가 접촉하도록, 물리적 파일 저장 범위를 장치 저장 범위로 재 배열함으로써, 파일 시스템(167)은, 저장 장치(들)(169)에 저장된 물리적 파일들을 디프레그먼트(defragment)하도록 구성될 수 있다.

[0111] 논리 객체 저장 공간의 리패킹 후에, 방출된 객체에 대응하는 인덱스 엔트리는 삭제되고(블록 906), 다른 객체가 선택되는 경우 동작은 블록 900으로부터 계속할 수 있다. 상기한 바와 같이, 일부의 실시예에서, 객체들이 선택됨에 따라 리패킹은 "온 더 플라이(on the fly)"을 발생시키고, 객체들을 재배치하는데 필요한 동작의 개수를 최소화시키면서 논리 객체 저장 공간의 전체 사용을 향상시킬 수 있다.

[0112] 일부의 실시예에서, 획득, 풋, 방출, 또는 다른 동작의 무엇이든 요청한 클라이언트에 대한 다양한 방식의 주고받기, 확인, 또는 에러 처리 프로토콜들을 지지할 수 있는 노드(160)에 의해 지지될 수 있다는 것이 주목된다. 예를 들면, 만일 클라이언트가 동작용 기형(malformed) 요청을 요청하거나(예를 들면, 필요한 파라미터의 공급을 실패하면), 노드(160)가 동작을 만족스럽게 완성하지 못하면(예를 들면, 풋 동작을 유효로 간주하기에 불충분한 자원들을 가지면), 노드(160)는 에러 표시를 요청한 클라이언트에게 반환할 수 있다. 이러한 표시는 폴트 조건의 특징에 관한 특정 상세를 포함하거나 포함하지 않을 수 있다.

[0113] 일 실시예에서, 심지어 다중 동작들이 공통의 데이터를 가질 수 있는 경우에도, 조정자(120)는 동작들을 동작들에 의해 목표로 정해진 각 특정 노드(160)에 독립적으로 전송하도록 구성될 수 있다. 예를 들면, 풋 동작의 경우에, 객체(30)가 기입 계획에 따라 다중 노드들(160)에 기입될 때, 조정자(160)는 각 특정 노드(160)와 독립적으로 통신할 수 있다. 하지만, 대체 실시예에서, 다중 목적 노드들(160)로 의도된 공통 데이터 및/또는 파라미터들을 갖는 동작들이 제한될 수 있다. 일 실시예에서, 조정자(120) 또는 다른 클라이언트는 수용 리스트와 같은 동작이 파라미터에 각 수용자를 지정함으로써 제한된 동작을 시작할 수 있다. 동작에서 표시된 다중 수용자들은 디폴트에 의해 제한을 표시하거나, 다른 파라미터가 제한된 동작을 표시하기 위해 사용될 수 있다. 그 후, 조정자(120) 또는 다른 클라이언트는 제한된 동작을 동작에서 지정된 목적 노드들(160) 중의 제1 노드에 전송함으로써 제한된 동작을 시작할 수 있다.

[0114] 제한된 동작을 수신하면, 노드(160)는 동작을 처리하고 이것을 동작에서 지정된 목적 노드들(160) 중의 다른 하나에 전달할 수 있다. 이러한 전달 전에, 수용자 노드(160)는 동작에 포함된 목적지 리스트로부터 그 자체를 제거함으로써, 수용을 표시하고 순환성 전달을 회피할 수 있다. 이 동작은 수용자 노드의 처리와 동시에 전달될 수 있다. 대안으로, 전달은 처리에 대한 수용자 노드의 성공적인 완료의 부수적인 것일 수 있다. 일부의 실시예에서, 제한된 동작은 수용자들이 동작 내에 표시된 순서로 수용자들에게 운반될 수 있다. 다른 실시예들에서, 노드들(160)이 예를 들면, 남은 목적지들이 가장 가깝거나, 가장 작게 로딩되거나, 어느 다른 선택 기준을 만족시키는 지를 결정함으로써, 다음 수용자를 동적으로 선택할 수 있다. 일부의 실시예에서, 제한된 및 제한되지 않은 동작들의 결합은 조정자(160) 또는 다른 클라이언트에 의해 발생될 수 있다. 예를 들면, 동일한 데이터가 6개의 별개 노드들(160)로 예정된 풋 동작의 타켓이면, 조정자(120)는 6개의 목적 노드들을 지정하는 제한된 단일 동작 또는 각각 3개의 목적 노드들을 지정하는 제한된 2개의 동작들을 생성할 수 있다. 조정자(120)가 목적 노드들(160) 각각에 독립적으로 운반하는 제한되지 않은 6개의 동작들의 생성을 포함하는 다른 결합도 가능하다.

[0115] 키맵 구성

[0116] 상기한 바와 같이, 다양한 비트 저장 노드들(160)이 객체(30)의 예들을 위한 저장 장치를 제공하도록 구성될 수 있다. 노드들(160)은 리던던시용 특정 지지 또는 데이터 보안을 개별적으로 제공하지 않을 수 있다; 사실, 일부의 실시예에서, 노드들(160)은 개방형-소스 동작 시스템들(예를 들면, 리눅스)를 운전하고 값싼 상품 하드 드라이브들(예를 들면, ATAPI/IDE 하드 드라이브들)을 통하여 저장 장치를 제공하는 일반적인 컴퓨팅 플랫폼을 이용하여 실행될 수 있다. 이러한 실시예에서, 개별 시스템들은 특히 내 고장성(fault-tolerant)이 아닐 수 있다. 오히려, 상기한 바와 같이, 데이터 보안 및 리던던시가 다수의 노드들(160)을 가로질러 객체들(30)의 복사를 통하여 제공될 수 있다.

[0117] 상기한 바와 같이, 주어진 객체(30)는 저장 클라이언트에 의해 지정될 수 있는 키에 대응할 수 있다. 주어진 객체(30)의 개별 예들은 저장 서비스 시스템에 포함된 노드들(160)의 수집을 가로지른 예들을 유일하게 확인할 수 있는 각 위치 표시자에 대응할 수 있다. 일 실시예에서, 저장 서비스 시스템 내에 배치된 각 키맵 인스턴스(140)는 키, 주어진 객체(30)용으로 대응하는 모든 위치 표시자와 노드들(160) 사이에 저장된 그 복사 예들 사이의 관계 또는 매핑을 저장 및 유지하도록 구성될 수 있다. 키맵 인스턴스(140)의 다양한 실시예의 일반적인 특성 및 기능이 아래에 설명되고, 다음으로 키맵 인스턴스(140)의 특정 실시예가 어떻게 실행되는 지가 설명된다.

[0118] 일 실시예에서, 주어진 키맵 인스턴스(140)는 적어도 하나의 표 또는 어느 다른 종류의 데이터 구조 내의 키들 및 관련된 위치 표시자들 사이의 관계의 상세 내용을 저장하도록 구성될 수 있다. 예를 들면, 도 10에 도시된 바와 같은 일 실시예에서, 키맵 인스턴스(140)는 다수의 엔트리(144)를 갖는 키맵 데이터 구조(142)를 포함한다. 각 엔트리는 각 키(146) 및 관련 기록(148)을 포함한다. 일부의 실시예에서, 아래에 상세히 설명되는 바와 같이, 엔트리들(144)을 형성하기 위해 사용된 데이터 구조의 형성은 복잡할 수 있다. 하지만, 기능적인 관점에서, 키맵 인스턴스(140)는 일반적으로 주어진 키(144)와 대응하는 기록(148) 사이의 일대일 표-방식(table-like) 관계를 유지할 수 있다.

[0119] 기록(148)은 주어진 키(144)에 대응하는 일반적으로 위치 표시자(들)를 포함하지만, 또한 다른 정보를 포함할 수 있다. 예를 들면, 기록(148)의 일 실시예는 다음과 같이 구성될 수 있다:

[0120]
```
struct KeyRecord {
```

[0121]
```
int16_t version;
```

[0122]
```
int16_t storageClass;
```

[0123]
```
int64_t creationDate;
```

[0124]
```
int64_t objectSize;
```

[0125]
```
uint32_t crc32;
```

[0126]
```
int8_t numLocators;
```

[0127]
```
struct locator {
```

[0128]
```
int64_t nodeID;
```

[0129]
```
int64_t objectIndex;
```

[0130]
```
} replicas [];
```

[0131]
```
}
```

[0132] 이 예의 데이터 구조는 C 프로그래밍 언어의 구문(syntax)을 이용하여 표현되지만, 어느 적당한 언어, 표현, 및 포맷을 이용하여 실행될 수 있다. 기록(148)의 대체 실시예는 도시된 것보다 더 많은, 더 적은, 또는 상이한 필드들을 포함할 수 있다. 일부의 예에서, 기록(148)은 저장 공간을 어떤 방식의 유닉스 파일 시스템에 사용된 아이노드(inode)에 형성할 때, 기록(148)를 단순화하기 위하여 "아이노드" 도면으로 불릴 수 있다. 하지만, 본 명세서에서 용어 "아이노드"의 사용은 실행의 특정 상세 또는 파일 또는 다른 저장 환경 내의 아이노드의 사용을 유도할 의도는 없다.

[0133] 상기 실시예에서, 기록(148)은 7개의 특정 요소를 포함한다. 16-비트 버전 구성 요소가 기록(148)의 포맷에 특별한 유일의 확인 값을 저장하도록 구성될 수 있다. 예를 들면, 기록(148)의 상이한 버전들은 키맵 인스

턴스의 다른 실행들에 사용되고, 일부 실시예에서, 주어진 키맵 인스턴스(140) 내에 저장된 기록들(148)은 이종일 수 있다. 버전 구성 요소는 기록(148)의 상이한 버전들 사이를 구별하여, 기록의 다른 요소들이 적당하게 디코딩되어 사용될 수 있다.

[0134] 16-비트 저장 등급 요소가 기록(148)에 대응하는 객체(30)의 저장 등급의 표시를 저장하도록 사용될 수 있다. 저장 등급들에 대하여는 후속 섹션에 더욱 자세하게 설명된다. 일반적으로 말해, 객체의 주어진 저장 등급은 주어진 저장 등급의 다른 부재들에 공통일 수 있는 저장 특징 및/또는 정책들을 확인할 수 있도록 한다. 예를 들면, "고 신뢰성" 저장 등급 및 "저 신뢰성" 저장 등급은 저장 서비스 시스템의 주어진 실행을 위해 정의될 수 있다. 고 신뢰성 저장 등급의 부재들인 객체들(30)은 저 신뢰성 저장 등급의 부재들인 객체들(30)보다 더 큰 정도로 복사되어, 아마도 저 신뢰성 저장 등급의 부재들용으로 할당된 것보다 더 높은 사용 비용으로의 교환에서, 개별 복사의 손실에 대한 민감도를 감소시킬 수 있다. 저장 등급들의 다수의 다른 가능한 방식 및 결합이 가능하고 예측된다.

[0135] 64-비트 생성 날짜 요소는, 대응 객체(30)가 저장 서비스 시스템 내에서 생성된 날짜 및 시간의 표시를 저장하도록 사용될 수 있다. 이 요소는 어느 적당한 방법으로 포맷팅될 수 있다. 예를 들면, 날짜 및 시간은 요소 내 별개 필드들, 또는 참조의 공통 포인트 이래로 경과한 시간 단위(예를 들면, 초, 밀리초, 등)의 개수를 나타내는 단수로서 명백하게 인코딩될 수 있다. 비록 일부의 실시예에서, 생성 날짜 요소가 대응 객체(30)의 어느 양상의 최종 변경의 날짜 및 시간을 나타내도록 구성된 추가 필드들을 포함하더라도, 다른 실시예에서 최종 변경 요소는 기록(148) 내 별개 요소로서 포함될 수 있다.

[0136] 64-비트 객체 크기 요소는 대응 객체의 크기, 예를 들면 바이트 단위의 표시를 저장하도록 사용될 수 있다. 일부의 실시예에서, 이 요소는 객체 데이터 및 메타 데이터 둘 다의 크기를 반영할 수 있지만, 다른 실시예에서 이것들은 개별 필드들로서 저장될 수 있다. 32-비트 crc32 요소는 어느 적당한 체크섬(checksum) 알고리즘에 따라 객체 데이터 및/또는 메타 데이터용으로 계산된 순환 리던던시 검사(CRC) 체크섬의 표시를 저장하도록 사용될 수 있다. 예를 들면, 체크섬은 부패 또는 간섭에 대한 데이터 보전을 검증하도록 포함될 수 있다. 다른 실시예들에서, 객체 데이터 및/또는 메타 데이터로부터 계산된 어느 적당한 종류의 해시 또는 서명이 CRC 체크섬 외에 또는 대신 사용될 수 있다.

[0137] 8-비트 번호 위치 표시자(numLocators) 요소는 복사 [] 어레이 내 기록(148) 내에 포함된 위치 표시자의 수의 표시를 저장하도록 사용될 수 있다. 이 어레이 내에, 각 위치 표시자는 64-비트 객체 인덱스 값뿐만 아니라 64-비트 노드 ID 요소로서 저장되고, 비트 저장 노드들(160)의 구성에 대한 설명에서 상기한 바와 같이 유도될 수 있다. 일부의 실시예들에서, 위치 표시자는 복사 [] 어레이 내 단일 요소들로서 저장될 수 있다.

[0138] 일 실시예에서, 키맵 인스턴스(140)은 제공된 API를 지지하는데 필요한 기능들을 수행할 뿐만 아니라 키맵 API를 조정자(120)와 같은 키맵 클라이언트를 제공하도록 구성될 수 있다. 예를 들면, 제어기(120)는 키맵 인스턴스(140)에 의해 관리된 엔트리들(144)에 관련된 기록들(148)에 다른 동작을 저장, 검색, 삭제, 또는 수행하기 위한 API를 사용하도록 구성될 수 있다. 예를 들면, 상기한 바와 같이 노드들에 의해 지지될 수 있는 객체 예에서의 동작들과 유사하게, 일 실시예에서, 키맵 API는 키맵 엔트리들(144)에 대한 풋, 획득, 및 삭제 동작을 지지할 수 있다. 또한, 일반적으로 키맵 저장 동작 또는 키맵 기입 동작으로 불릴 수 있는 키맵 엔트리 풋 동작은 키맵 엔트리(144) 내에 저장될 키(146) 및 기록(148)을 지정할 수 있다. 일 실시예에서, 엔트리(144)가 이미 존재하는 키(146)를 지정하는 풋 동작은 풋 동작의 변수 또는 파라미터로서 지정된 기록으로 존재하는 엔트리(144)에 관련된 기록(148)을 대체할 수 있다. 주어진 키맵 인스턴스(140)에서 완료된 경우, 키맵 풋 동작은 예를 들면 동작이 성공 또는 실패하였는지 및 (만일 있다면) 무슨 방식이 발생하였는지와 같은 상태 표시를 요청한 클라이언트에 반환할 수 있다. 일부의 실시예에서, 키맵 풋 동작이 존재하는 엔트리(144)의 대체로 끝나면, 키맵 인스턴스(140)은 엔트리(144)의 이전 값을 요청하는 클라이언트로 반환하도록 구성될 수 있다.

[0139] 일 실시예에서, 또한 일반적으로 키맵 독출 또는 검색 동작으로 불릴 수 있는 키맵 엔트리 획득 동작은 파라미터로서 키를 지정한다. 키 지정이 완료된 경우, 이러한 엔트리가 존재하면, 키맵 획득 동작이 요청된 키에 관련된 키맵 엔트리(144)의 기록(148)을 요청한 클라이언트에게 반환할 수 있다. 만일 대응 엔트리(144)가 존재하지 않으면, 효과에 대한 표시는 요청한 클라이언트로 반환될 수 있다.

[0140] 일 실시예에서, 키맵 엔트리 삭제 동작은, 요청한 클라이언트가 엔트리에 기입될 기록을 지정할 필요가 없는 것 외에는 풋 동작과 유사하게 동작하도록 구성될 수 있다. 주어진 키맵 인스턴스(140)에서 완료된 경우, 키맵 삭제 동작은 키맵 풋 동작의 상태 표시와 유사한 상태 표시를 요청한 클라이언트로 반환할 수 있다. 풋 동작과 같이, 일부 실시예에서, 키맵 인스턴스(140)는 삭제된 엔트리(144)의 이전 값을 요청한 클라이언트로 반환

하도록 구성될 수 있다.

[0141] 키맵 API는 또한 다양한 실시예에서 다른 방식의 동작들을 지지할 수 있다. 예를 들면, 키맵 API는 키맵 엔트리들을 관리할 때 키맵 클라이언트들을 지원할 수 있는 동작들을 지지할 수 있다. 일 실시예에서, 키맵 API는 요청한 클라이언트에 의해 지정된 일부의 기준을 매칭하는 키들(146)을 갖는 엔트리들(144)을 확인하도록 구성될 수 있는 리스트 동작을 지지할 수 있다. 예를 들면, 리스트 동작은 클라이언트가 동작에서의 파라미터로서 스트링 또는 패턴을 지정할 수 있도록 한다. 주어진 키맵 인스턴스(140)에서 완료된 경우, 리스트 동작은 지정된 스트링 또는 패턴을 만족시키는 키들(146)의 리스트를 요청한 클라이언트에 반환할 수 있다. 일 실시예에서, 스트링이 키(146)의 적당한 접두부(예를 들면, 스트링의 모든 문자에 대한 키의 N-번째 문자를 매칭하는 스트링의 N-번째 문자)인 경우에만, 키(146)는 주어진 스트링을 만족시킬 수 있다. 다른 실시예들에서, 스트링이 키(146) 내의 어느 위치에서 발견될 수 있는 경우, 키(146)는 주어진 스트링을 만족시킬 수 있다.

[0142] 일부 실시예들에서 리스트 동작은 다른 파라미터들을 지지할 수 있다. 예를 들면, 리스트 동작은 요청한 클라이언트들이 반환될 매치 수에 대한 제한을 지정할 수 있도록 한다. 또한, 요청한 클라이언트는, 예를 들면 검색될 키들(146)이 차지해야 하는 개방-단부 또는 폐쇄-단부 사전 편집 범위를 지정함으로써, 억제를 검색될 키들(146)로 지정할 수 있다. 일부의 실시예들에서, 키맵 인스턴스(140)는 리스트 동작 기준을 만족시키는 키들(146) 뿐만 아니라 기록들(148)을 반환하도록 구성될 수 있다. 또한, 어느 동작에서, 키맵 API는 리스트 동작과 동일한 방식의 파라미터들 및 실행 작용을 지지할 수 있는 카운트 동작을 지지할 수 있다. 하지만, 요청한 클라이언트에 의해 제공된 기준을 만족시키는 키들(146) 및/또는 기록들(148)을 반환하는 대신에, 카운트 동작은 기준들(예를 들면, 대응 리스트 동작에 의해 반환된 키들의 수)를 만족시키는 키들의 수를 반환할 수 있다. 키맵 API는 또한 상기에 상세하게 설명되지 않은 다른 동작들을 지지할 수 있다.

[0143] 일부 환경에서, 상이한 키맵 클라이언트들은 동일한 키맵 엔트리(144)를 찾아 변경할 수 있다. 예를 들면, 다양한 클라이언트- 또는 시스템-구동된 동작들에 응답하여, 2개의 다른 조정자(120)는 주어진 기록(148)을 동시에 변화시킬 것(예를 들면, 복사들의 위치 표시자들 추가, 삭제 또는 변경시킬 것)을 시도하거나, 하나의 조정자는 기록(148)을 변경할 것을 시도하고 다른 하나는 대응 엔트리(144)를 삭제할 것을 시도할 수 있다. 일 실시예에서, 주어진 키맵 엔트리(144)에 대한 동시 요청들을 분석하기 위한 일관성 방법을 제공하기 위하여, 키맵 API는, 키맵 상태(예를 들면, 키맵 풋 및 삭제 동작들)를 업데이트하는 적어도 키맵 동작들이 파라미터로서 시퀀스 번호를 키맵 동작에 제공할 것을 요청할 수 있다. 그 후, 키맵 인스턴스(140)은 시퀀스 번호들을 (예를 들면, 수치상으로 또는 사전 편집으로) 비교하고, 상기 비교를 기초로 한 동작들의 하나를 일관성 있게 픽업함으로 모순된 업데이트를 해결하도록 구성될 수 있다. 일부의 실시예에서, 제공된 시퀀스 번호는 아래에 자세히 설명되는 바와 같이 동기화 회복을 위한 변경된 기록(148)에 따라 변경된 키맵 엔트리(144)에 저장될 수 있다.

[0144] 예를 들면, 키맵 클라이언트는 시간 스탬프에 기초하여 시퀀스 번호를 발생할 수 있다. 일 실시예에서, 이러한 시간 스탬프는 다음과 같이 포맷팅된 64-비트 수를 포함할 수 있다. 시간 스탬프의 비트 63은 (예를 들면, 시간 스탬프가 서명된 또는 서명되지 않은 수인 지에 대한 혼란을 회피하기 위하여) 0으로 설정될 수 있다. 비트들 62:32는 시간 상(예를 들면, 유닉스 및 리눅스의 많은 버전에 의해 이용된 기준 시간인 그리니치 평균시간, 즉 1970년 1월 1일 자정) 기준점 이래로 경과된 초의 수를 포함할 수 있다. 비트들 31:22는 마지막 초 이래로 경과한 밀리초의 수를 포함할 수 있다. 비트들 21:0는 본질적으로 랜덤하게 발생한 비트들을 포함할 수 있다. 다른 실시예에서, 시간 스탬프는 상이한 폭 및 종류의 필드들에 기초하여 발생될 수 있다. 대안으로, 키맵 클라이언트는 시퀀스 번호들을 발생하는 완전히 다른 기초(basis)를 이용할 수 있다. 시퀀스 번호의 해상도가 높게 제공된 경우, 동일한 키맵 엔트리(144) 마다 다른 키맵 클라이언트들에 의해 제공된 상기한 시퀀스 번호들 사이의 충돌 기회는 낮을 수 있다. 하지만, 만일 충돌이 발생하면, 키맵 인스턴스(140)는 어느 적당한 일관성 기술을 이용하여 충돌을 해결하도록 구성될 수 있다.

[0145] 많은 실시예에서, 키들을 위치 표시자들에 매핑할 때 키맵 인스턴스(140)의 이상적인 기능 동작은 상대적으로 간단하게 수행될 수 있다. 예를 들면, 상기한 바와 같이, 키맵 인스턴스의 일 실시예에 의해 지지된 기본 동작 한 세트는 기록들(148) 내에 포함된 키들(146)과 위치 표시자들 사이의 관계를 반영하는 엔트리들(144)을 조작하도록 구성된 풋, 획득, 및 삭제 동작을 포함할 수 있다. 하지만, 저장 서비스 시스템 내 키맵 기능의 실행은 다수의 도전을 유발할 수 있다. 특히, 만일 저장 서비스 시스템이 큰 수의 클라이언트를 대신하여 큰 수의 객체들(30)(예를 들면, 테라바이트(TB) 또는 페타바이트(EB) 또는 이상의 저장 장치를 합계한 백만 또는 10억 개의 객체(30))을 지지하기 위한 것이면, 따라서 키맵의 실행은 용량을 스케일링하는 것이 요구될 수 있다. 하지만, 단일 컴퓨터 시스템 내의 키맵에 포함된 전체 정보를 표현하기 위하여, 충분한 시스템 메모리 자원들을 실행할 수 없거나 경제적으로 적합하지 않을 수 있다. 또한, 키맵 클라이언트 요청들에 대한 내 고장성

및 증가된 처리량을 위하여, 키맵 데이터의 다중 복사는 저장 서비스 시스템 내 분배 방식으로 배치될 수 있다. 하지만, 예를 들면, 하나의 복사는 처리되지만 다른 복사가 변경되는 경우, 키맵 데이터의 복사는 키맵 동기화를 일으킬 수 있다.

[0146] 키맵 기능(functionality)의 스케일가능성(scalability)은 키맵 인스턴스들(140) 내의 계층화 층들을 도입함으로써 향상될 수 있다. 이러한 계층의 일 실시예는 도 11A 내지 도 11D에 도시되어 있다. 키맵 배치(1100)의 예가 도 11A에 도시되어 있다. 상기한 바와 같이, 도 3에 대한 일부 저장 서비스 시스템 실시예에서, 다중 키맵 인스턴스들은 예를 들면, 상이한 데이터 중심들(300) 또는 영역들(310)에 있는 시스템을 통하여 분배될 수 있다. 일반적으로, 키맵 인스턴스들의 수집은 배치(deployment)라고 불릴 수 있다. 일부 실시예에서, 저장 서비스 시스템은 시스템 내에 제공된 모든 키맵 인스턴스들(140)을 갖는 단일 키맵 배치(1100)를 포함할 수 있다. 하지만, 다른 실시예들에서, 시스템은 키맵 계층의 추가 레벨 하에 통합된 다중 키맵 배치들(1100)을 포함할 수 있다.

[0147] 설명된 실시예에서, 배치(1100)는 키맵 인스턴스들(140a~140c)을 포함한다. 키맵 인스턴스들(140a~140c) 각각은 예를 들면 아래에 자세히 설명되는 바와 같이 인스턴스 동기화 프로토콜에 따라 다른 키맵 인스턴스와 키맵 정보를 교환하도록 구성될 수 있다. 도시된 바와 같이, 각 키맵 인스턴스(140)는 상호 통신하도록 구성된 다수의 호스트를 포함한다. 예를 들면, 키맵 인스턴스(140a)는 호스트들(400a~400c)을 포함하고, 키맵 인스턴스(140b)는 호스트들(400d~400g)을 포함하고, 키맵 인스턴스(140c)는 호스트들(400h~400j)을 포함할 수 있다. 일반적으로 말해, 각 호스트(400)는 컴퓨터 시스템 및 관련 소프웨어를 포함할 수 있고, 프로세서, 시스템 메모리, 저장 장치들, 네트워크 인터페이스들, 또는 다른 적당한 구성 요소들과 같은 구성 요소들을 포함할 수 있다. 예를 들면, 호스트(400)로서 역할을 하도록 구성될 수 있는 컴퓨터 시스템 또는 노드의 일 실시예가 도 29의 설명에 결합하여 아래에서 설명된다.

[0148] 일반적으로, 각 키맵 인스턴스(140)는 저장 서비스 시스템 내에 저장된 모든 객체들(30)에 대한 키맵 엔트리들(144) 및 키맵 계층을 인덱스하고 관리하도록 사용된 어느 다른 데이터를 포함하는 키맵 데이터의 완전한 표시를 유지하도록 구성될 수 있다. 키맵 인스턴스(140) 내에, 키맵 데이터가 호스트들(400)을 통하여 분배되어, 개별 호스트들(400)이 키맵 데이터의 일부 (가능한 리던던시) 부분을 저장할 수 있도록 한다. 단지 소수의 호스트(400)가 도 11A에 도시되었지만, 다른 실시예에서 각 키맵 인스턴스(140)는 임의의 적당한 수의 호스트들(140)을 포함할 수 있다는 것이 주목된다. 예를 들면, 일부 큰-스케일 실행에서, 수십 또는 아마도 수백 호스트들(140)이 키맵 인스턴스(140)에 포함될 수 있다. 일부의 실시예에서 주어진 키맵 인스턴스(140)용 호스트들(140)은 주어진 영역(310) 또는 데이터 중심(300) 내에 위치될 수 있지만, 다른 실시예에서 이러한 호스트(140)는 다른 영영들(310) 또는 데이터 중심(300) 사이에 분배될 수 있다는 것이 또한 예측된다. 또한, 호스트들(400)은 일부의 실시예에서 단지 키맵-관계된 기능을 실행하도록 구성될 수 있지만, 다른 실시예에서 호스트들(400)은 저장 서비스 시스템의 다른 구성 요소들에 관련된 기능을 실행할 수 있다. 예를 들면, 일 실시예에서 호스트들(400)은 또한 비트 저장 노드들(160)로서 구성될 수 있고, 그래서 객체 데이터뿐만 아니라 키맵 데이터를 저장할 수 있다.

[0149] 도 11B는 키맵 인스턴스(140a)의 바람직한 실시예를 상세하게 도시한다. 도시된 실시예에서 키맵 인스턴스(140a) 내에 포함된 호스트들(400) 각각은 각 인덱스(410a~410c) 및 임의 수의 브(brick)릭들(415)을 포함한다. 일반적으로 말해, 브릭(415)은 키맵 인스턴스 내 중간 키맵 데이터 구조에 대응할 수 있다. 일부의 실시예에서, 도 12의 설명에 관련하여 아래에 자세히 설명되는 바와 같이, 키맵 데이터는 브릭들(415) 사이의 파티션들로 분리될 수 있고, 키맵 인스턴스들(140) 내의 파티션들의 복사가 브릭 레벨에서 발생할 수 있다. 파티션 인덱스(410)는 키맵 동작 중 처리하기 위한 적어도 하나의 특정 브릭(415)의 선택들을 촉진하도록 하는 브릭들(415)을 인덱스하도록 구성될 수 있다. 예를 들면, 파티션 인덱스(410)는 트리 또는 다른 적당한 데이터 구조로서 구성될 수 있다. 일 실시예에서, 파티션 인덱스(410) 및 키맵 인스턴스(140) 내 더 깊은 인덱스 레벨들이 다음 섹션에서 자세히 설명되는 계층화된 불균형 트리 또는 트리에(trie)로 불리는 특정 형식의 데이터 구조의 일부로서 구성될 수 있다. 설명된 실시예에서, 키맵 인스턴스(140)는 또한 키맵 조정자(412)를 포함한다. 일반적으로 말해, 키맵 조정자(412)는 아래에 자세히 설명되는 것들과 같은 키맵 접근 관리, 컨텐트 관리 및 동기화 방법들 또는 프로토콜들을 실행하도록 구성될 수 있다. 키맵 조정자(412)는 호스트들(400)과는 별개로 도시되지만, 일부의 실시예에서 적어도 하나의 호스트(400) 내의 처리 또는 모듈로서 실행될 수 있는 것이 또한 주목된다. 일부의 실시예에서 파티션 인덱스들(410)은 호스들(400) 내에 단독으로 보다는 키맵 조정자(412) 내에 실행될 수 있다는 것이 또한 주목된다.

[0150] 도 11C는 브릭들(415a~415n)을 포함하는 호스트(400a)의 바람직한 실시예를 나타낸다. 도시된 바와 같

이, 브릭들(415a~415n) 각각은 각 블록 인덱스(420a~420n) 및 임의 수의 블록들(425)을 포함한다. 일반적으로 말해, 블록(425)은 브릭(415)과 유사하지만 추출의 브릭 레벨에 종속하는 키맵 인스턴스(140) 내 중간 키맵 데이터 구조에 대응할 수 있다. 파티션 인덱스(410)와 유사하게, 블록 인덱스(420)는 브릭(415) 내 블록들(425)을 인덱스하기 위해 구성된 어느 적당한 데이터 구조일 수 있다. 예를 들면, 블록 인덱스(420)는 일 실시예에서 계층화된 불균형 트리의 일부로서 구성될 수 있다.

[0151] 도 11D에 도시된 바와 같이, 일 실시예에서 블록들(425)은 임의 수의 개별 키맵 엔트리들(144a~144n) 및 선택용 엔트리들(144a~144n)을 인덱스하기 위한 구성된 엔트리 인덱스(430)를 포함한다. 상기한 바와 같이, 엔트리들(144a~144n) 각각은 각 키(146a~146n) 및 각 기록(148a~148n)의 표시를 포함할 수 있다.

[0152] 도 11A 내지 도 11D에 도시된 실시예의 키맵 인스턴스들(140)과 키맵 엔트리들(144) 간 계층화 층들 사이의 관계는 도 12에 요약되어 있다. 다중 키맵 인스턴스들(140)을 포함하는 추출의 배치 레벨에서, 특정 키맵 인스턴스(140)는 추출의 예 레벨에서 파티션 인덱스(410)를 참조할 수 있다. 참조된 파티션 인덱스(410)는 특정 엔트드(144)에 대응하는 브릭 또는 브릭들을 확인할 수 있도록 한다. 예를 들면, 설명된 실시예에서, 모든 키맵 엔트리들은 별개의 브릭들(415)에 대응하는 3개의 별개 파티션들에 의해 복사된다. 다음, 주어진 브릭은 블록 인덱스(420)를 통하여 (도 12에는 도시안된) 특정 블록(425)을 참조하고, 참조된 블록은 엔트리 인덱스(430)를 통하여 특정 엔트리(144)를 참조할 수 있도록 한다. 키맵은 도 12에 도시된 바와 같은 계층화 실행을 이용하여 실행될 수 있지만, 다른 실행들도 가능하다는 것이 주목된다. 대체로 말하면, 키맵 인스턴스들(140)은 키들(144)을 기록들(148)과 관련시키기 위한 어느 적당한 기술을 이용하여 실행될 수 있다. 예를 들면, 일 실시예에서 키맵 인스턴스(140)는 종래의 데이터베이스 또는 다른 방식의 구성된 인덱스를 이용하여 실행될 수 있다.

[0153] 도 12의 실시예에서 계층화 층들의 일부는 리던던시(예를 들면, 배치 레벨 내 키맵 인스턴스들(140)의 복사 및 파티션 레벨에서 브릭들(415)의 복사)을 제공하도록 구성될 수 있지만 다른 층들은 분해 능력을 제공하도록 구성될 수 있다는 것이 주목된다. 예를 들면, 교차 다중 별개 레벨들(예를 들면, 파티션 인덱스(410), 블록 인덱스(420) 및 엔트리 인덱스(430))을 인덱스하는 분배가, 인덱스의 각 부분이 키맵 배치 증가 내에 인덱스될 엔트리들(144)의 수가 증가함에 따라 관리가능한 방법으로 성장할 수 있도록 함으로써 데이터 구조의 스케일링을 촉진할 수 있다. 다른 실시예에서 계층의 더 많거나 작은 레벨들 및 리던던시 및 비-리던던시 레벨의 다른 결합이 사용될 수 있다는 것이 주목된다.

[0154] 객체들(30)을 갖는 것과 같이, 키맵 계층의 층들 내의 복사의 이용은 개별 복사들의 손실의 민감도를 감소시킴으로써 내 고장성을 향상시킬 수 있다. 하지만, 변경으로서 키맵 데이터의 복사들을 동기화하기 위한 어떠한 시도도 발생하지 않으면, 키맵의 정확한(예를 들면, 가장 최근) 상태가 불명료하게 되고, 다음으로 예측할 수 없거나 에러 시스템 동작을 일으킬 수 있다. 일부의 실시예에서, 키맵 데이터의 복사된 부분은, 업데이트들이 매 복사에 대하여 지속적이고 입증할 수 있게 완성될 때까지 완전하게 기록될 수 없는 원자(atomic) 또는 트랜잭션 의미론(semantics) 예를 들면, 2 단계 완료 의미론)을 이용한 엄격한 동기식 방식으로 업데이트될 수 있다. 원자 업데이트 의미론이 불일치 상태에서 키맵 데이터를 남기는 업데이트의 가능성을 최소화하거나 심지어 제거할 수 있지만, 원자 업데이트의 성능은 큰 스케일의 분배된 환경에서 많이 저하될 수 있다. 예를 들면, 키맵 데이터의 복사들이 넓게 분배되면, 각 클라이언트로부터의 복사 접근 지연이 업데이트 동작을 완료하는데 요구된 전체 시간을 지시하는 최저 복사에 의해 많이 변할 수 있다. 또한, 하나의 복사가 실패해야 하면, 엄격한 원자 업데이트 의미론이, 상기 실패가 정정될 때까지 클라이언트가 스톨(stall)하도록 하여, 클라이언트들이 용인할 수 있는 지연을 일으킬 수 있다.

[0155] 원자 프로토콜들보다 더 우수한 클라이언트 성능을 제공할 수 있는 다른 방식의 동기화 프로토콜들이 키맵 계층 내에서 이용될 수 있다. 일부의 실시예에서, 일 방식의 동기화 프로토콜이 특정 키맵 인스턴스(140) 내의 복사들(예를 들면, 도 12에 도시된 바와 같이 파티션 레벨에서의 복사들)에 대하여 이용될 수 있는, 하이브리드 동기화 접근이, 실행될 수 있지만, 다른 방식의 프로토콜은 키맵 배치 내 상이한 키맵 인스턴스들(140)을 동기화하는데 이용될 될 수 있다. 이러한 하이브리드 접근은 동기화 오버헤드가 키맵 계층 내 상기한 레벨에서 복사들의 사용 동력(usage dynamics)으로 더욱 특히 맞추어지도록 할 수 있다.

[0156] 예를 들면, 키맵 데이터 접근들은 기준의 위치를 나타내어, 특정 엔트리들(144)에 반복된 요청들이 다른 키맵 인스턴스(140)보다 특정 키맵 인스턴스(140)(예를 들면, 지리적 또는 네트워크 토폴로지 또는 다른 적당한 기준에 의해 요청한 클라이언트에 가장 가까운 예)로 더욱 더 지향될 것 같다. 즉, 이것은 주어진 키맵 인스턴스(140) 내의 키맵 데이터의 복사들이 다른 키맵 인스턴스(140)에서의 대응 키맵 데이터보다 주어진 클라이언트에 의해 접근될 것 같은 경우일 수 있다. 따라서, 일부의 실시예에서, 주어진 키맵 인스턴스(140) 내의 복

사들은 별개의 키맵 인스턴스들(140)을 동기화시키는데 사용된 프로토콜보다 더욱 빠르게 집중하도록(예를 들면, 복사들 사의 변화를 전파하도록) 구성될 수 있는 프로토콜을 이용하여 동기화될 수 있다.

[0157] 일 실시예에서, 주어진 키맵 인스턴스(140) 내 키맵 데이터 복사들의 동기화는 적당한 버전의 정족수(quorum) 프로토콜을 이용하여 수행될 수 있다. 일반적으로 말해, 정족수 프로토콜에 따라 수행된 (키맵 엔트리 풋 및 삭제 동작을 포함한) 키맵 데이터의 복사들의 업데이트 또는 변경은, 변경이 적어도 정족수의 복사들에 대하여 지속적으로(예를 들면, 완전하고 계속적으로) 수행된 경우, 요청한 클라이언트에 대하여 완료될 것으로 간주될 수 있다. 마찬가지로, 동일한 데이터가 적어도 정족수의 복사들로부터 독출되는 경우, 정족수 프로토콜에 따라 수행된 키맵 엔트리 획득 동작이 완료될 것으로 간주될 수 있다. 일부의 실시예에서, 정족수는 존재하는 복사들의 수의 단순한 대부분으로서 정의되고, 다른 실시예에서 초 대다수의 임의 정도가 사용될 수 있다. 정족수 요구가 충족되면, 정족수 프로토콜 동작은 완료할 수 없다는 것이 주목된다. 하지만, 만일 복사의 정족수가 복사의 전체수보다 작으면, 주어진 정족수 프로토콜 동작의 실패 가능성은 원자 프로토콜 동작의 실패 가능성보다 더 낮아 정족수보다 오히려 복사들 사이의 일치를 효과적으로 요구한다. 여기에 설명된 하나의 정족수 프로토콜과 다른 정족수 프로토콜이 키맵 인스턴스들(140)에 의해 이용될 수 있다는 것이 주목된다. 예를 들면, 파소스(Paxos) 또는 2 단계 완료 와 같은 다단계 완료 프로토콜은 정족수 방식 키맵 의미론을 실행하는데 이용될 수 있다.

[0158] 정족수 프로토콜에 따른 독출 및 업데이트 동작의 정상 동작 중에, 업데이트는 예를 들면, 복사를 기초로 하는 자원들의 통신 고장들 또는 고장으로 인하여 모든 복사에 전파되지 않을 수 있다. 일 실시예에서, 복사들 사이의 불일치는 독출 동작 중에 검출 및 수리될 수 있다. 특히, 만일 키맵 엔트리 획득 동작 중에 특정 엔트리(144)의 상기한 복사들 사이에 상이한 값들이 검출되면, 키맵 획득 동작은 차이를 조정하도록 수행될 수 있다. 일 실시예에서 풋 동작용 기초로서 사용된 엔트리(144)는 독출된 상이한 값 중 가장 최근(예를 들면, 수치적으로 또는 사전 편집으로 가장 크게) 관련된 시간 스탬프를 갖는 엔트리일 수 있다. 그래서, 복사들 간의 불일치는, 키맵 엔트리 획득이 불일치를 수리하기 위한 별도의 처리 또는 동작을 요구하지 않으면서 처리됨에 따라 "온 더 플라이"로 해결될 수 있다.

[0159] 정족수 프로토콜을 실행하기 위해 구성된 키맵 인스턴스(140)의 일 실시예에 대한 키맵 엔트리 풋, 획득, 삭제, 및 리스트 동작의 바람직한 실시예의 동작이 도 13 및 도 14에 설명되어 있다. 다양한 실시예에서 이 방법들은 예를 들면, 키맵 인스턴스(140) 내에 포함된 적어도 하나의 호스트(400) 내에, 또는 도 11B에 도시된 키맵 조정자(412)와 같은 키맵 인스턴스(140) 내 독립 처리 또는 시스템으로서 구성될 수 있는 키맵 조정자 처리 내에서 실행될 수 있다. 먼저, 도 13을 참조하면, 키맵 엔트리 풋 동작이 조정자(120) 또는 다른 키맵 클라이언트로부터 키맵 인스턴스(140)에 수신된 경우, 키맵 엔트리 동작을 블록 1300에서 시작할 수 있다. 예를 들면, 특정 객체(30)에 대응하는 객체 예가 특정 비트 저장 노드(160)에 저장하는 것에 응답하여, 조정자(160)는 저장된 객체 예의 조정자를 반영하도록 객체(30)의 엔트리(144)를 업데이트하기 위하여, 키맵 엔트리 풋 동작을 발생할 수 있다.

[0160] 그 후, 키맵 인스턴스(140)의 계층은 키맵 엔트리 풋 동작에 대응하는 복사들을 확인하도록 조정(navigated)될 수 있다(블록 1302). 예를 들면, 도 12의 실시예에서 파티션 인덱스(410)는 브릭들(415)이 중요한 객체(30)에 대응하는 엔트리(144)를 복사하는 지를 결정하기 위해 참고될 수 있다. 이어서, 개별 풋 동작들이 확인된 복사들로 지향될 수 있다(블록 1304). 각 풋 동작에 있어서, 남은 키맵 인스턴스(140)의 계층은 대응 엔트리(144)를 접근 및 변경하도록 조정될 수 있다(블록 1306). 예를 들면, 주어진 브릭(415) 내의 블록 인덱스(420) 및 엔트리 인덱스(430)가 측정되어 지정된 엔트리(144)를 접근할 수 있게 된다. 일단 엔트리(144)의 주어진 복사가 성공적으로 기입되면, 대응 풋 동작이 성공을 표시할 수 있다(블록 1308). 엔트리(144)의 각 복사를 타켓팅하는 개별 풋 동작들이 동시에 실행되는 것이 주목된다. 따라서, 블록들(1306~1308)의 다중 예들이 병렬로 도시된다.

[0161] 개별 복사 풋 동작의 성공 표시들은, 정족수의 복사들이 성공적으로 업데이트되었는 지를 결정하기 위해 모니터링될 수 있다(블록 1310). 예를 들면, 3개의 복사를 갖는 실시예에서 키맵 엔트리 풋 동작을 완료하기 위한 복사들의 정족수는 2일 수 있다. 만일 복사들의 정족수가 성공적으로 업데이트되었으면, 요청된 키맵 엔트리 풋 동작이 완료된 것의 표시는 요청한 클라이언트로 반환될 수 있다(블록 1312). 만일 복사들의 정족수가 성공적으로 업데이트되지 않았으면, 모니터링을 계속할 수 있다. 일부의 실시예에서 타임 아웃이 실시되어, 만일 키맵 엔트리 풋 동작이 처리의 시작 후 지정된 시간 주기 내에 완료하지 못하면, 이 동작은 종료되고 에러 표시가 요청한 클라이언트로 반환될 수 있다. 다른 실시예에서, 키맵 엔트리 풋 동작은 완료될 때까지 무기한으로

미정 상태로 남을 수 있다.

[0162] 일 실시예에서, 키맵 엔트리 삭제 동작은 풋 동작의 특별 경우로 실행될 수 있다. 이러한 실시예에서, 키맵 엔트리(144)는 삭제 표지(sentinel) 또는 플래그(flag) 필드로서 구성된 추가 필드를 포함할 수 있고, 삭제 동작은 삭제 필드가 어써트(assert)◎상태로 설정하도록(예를 들면, 필드를 '1'과 같은 특정 값으로 설정함으로써) 구성된 풋 동작으로서 실행할 수 있다. 단정된 삭제 필드들을 갖는 엔트리들(144)는 추후 키맵 동작들 중 무시될 수 있다. 이러한 일부의 실시예에서, 단독 처리가 키맵 인스턴스(144)를 통하여 독립적으로 반복되어 어써트 삭제 필드들을 갖는 엔트리들(144)을 퍼지하도록 구성될 수 있다. 다른 실시예들에서, 이러한 엔트리들(144)은 이력 키맵 행동의 로그로서 무기한으로 유지될 수 있다.

[0163] 키맵 엔트리 획득 동작의 동작 방법의 일 실시예가 도 14에 설명되어 있다. 풋 동작이 조정자(120) 또는 다른 키맵 클라이언트로부터 키맵 인스턴스(140)에서 수신되는 경우, 동작은 블록(1400)에서 시작할 수 있다. 예를 들면, 저장 클라이언트(50)로부터의 특정 키에 대응하는 객체 데이터의 요청에 응답하여, 노드 피커(130) 또는 조정자(120)는 특정 키에 대응하는 위치 표시자를 얻기 위해 키맵 엔트리 획득 동작을 생성함으로써, 비트 저장 노드(160)가 이전 섹션에서 설명된 바와 같이 객체 데이터를 검색하도록 접근할 수 있도록 한다.

[0164] 키맵 엔트리 풋 동작에 의한 것과 같이, 키맵 인스턴스(140)의 계층은 키맵 엔트리 획득 동작에 대응하는 복사들을 확인하도록 조정될 수 있다(블록 1402). 이어서, 개별 획득 동작들은 지정된 복사들로 지향될 수 있다(블록 1404). 각 획득 동작을 위하여, 남은 키맵 인스턴스(140)의 계층은 대응 엔트리(144)를 접근하고 검색하도록 조정될 수 있다(블록 1406). 일단 엔트리(144)의 주어진 복사가 성공적으로 검색되면, 대응 획득 동작이 성공을 나타낼 수 있다(블록 1408). 위에서 설명되었고 도 13에 도시된 개별 풋 동작들에 의한 것과 같이, 엔트리(144)의 각 복사를 타켓팅하는 개별 획득 동작들이 동시에 실행되고, 따라서 블록들(1406~1408)이 병렬로 도시된다.

[0165] 개별 복사 풋 동작의 성공 표시들은, 정족수의 복사들이 성공적으로 업데이트되었는 지를 결정하기 위해 모니터링될 수 있다(블록 1410). 정족수의 복사들이 성공적으로 업데이트되지 않으면, 추가 복사가 독출될 때까지 모니터링이 연속할 수 있다. 상기한 키맵 엔트리 풋 동작을 위한 것과 같이, 일부의 실시예에서 정족수의 복사들이 성공적으로 판독될 때까지 키맵 엔트리 획득 동작이 무한적으로 기다릴 수 있다. 다른 실시예들에서, 키맵 엔트리 획득 동작은, 에러 표시 및/또는 이 시간에 이용할 수 있는 최상 데이터(예를 들면, 최근 시간 스탬프를 갖는 복사 데이터)가 요청한 클라이언트로 변환된 후, 소정 주기 후에 키맵 엔트리 획득 동작이 종료될 수 있다.

[0166] 정족수의 복사가 성공적으로 독출되는 경우, 검색된 복사들의 내용이 다른 지의 여부를 결정될 수 있다(블록 1412). 예를 들면, 요청된 엔트리(144)의 각 복사의 전체가 각 다른 검색된 복사에 대하여 비교되거나 엔트리(144)의 어떤 필드들(예를 들면, 기록(148)의 어떤 필드들) 만 비교될 수 있다. 비교에서 사용된 기준에 따라 검색된 복사들 사이에 차이가 없으면, 검색된 데이터는, 키맵 엔트리 획득이 완료된 것의 표시에 따라 요청한 클라이언트로 반환될 수 있다(블록 1414).

[0167] 복사들 간에 차이가 있으면, 복사 중의 하나가 선택 기준에 따라 선택될 수 있다(블록 1416). 예를 들면, 최고 시간 스탬프 값을 갖는 복사가 선택될 수 있는 경우, 기준은 각 복사의 시간 스탬프 값을 고려하는 것을 포함할 수 있다. 그 후,키맵 엔트리 풋 동작은 선택된 복사용 데이터를 이용하여 시작된다(블록 1418). 예를 들면, 풋 동작은 상기한 바와 같이 도 13에 따라 수행될 수 있다. 풋 동작의 결과로서, 원래 요청된 엔트리(144)의 정족수의 복사들이 선택된 복사의 컨텐츠를 갖도록 기입되어 추후 획득 동작이 복사들 사이의 불일치를 만날 가능성을 감소시킬 수 있다. 풋 동작 다음에, 선택된 복사용 데이터가, 키맵 엔트리 획득 동작이 완료되는 것에 대한 표시에 따라 요청한 클라이언트에 반환될 수 있다(블록 1414). 일부의 실시예에서, 복사들 사이에 검출된 불일치의 경우 풋 동작이 불일치를 해결하도록 시작된 풋 동작이 완료됨에 따라 완료될 수 있지만, 다른 실시예에서, 획득 동작은 후속 풋 동작이 완료되었는 지와는 무관하게 요청한 클라이언트에게 완전한것으로 나타낼 수 있다.

[0168] 상기한 바와 같이, 일부의 실시예에서 키맵 API는 검색 패턴과 같은 일부 기준을 만족시키는 키맵 엔트리들(144)의 키들(146)을 나타내도록 구성된 키맵 엔트리 리스트 또는 카운트 동작들을 지지할 수 있다. 일 실시예에서, 주어진 리스트 또는 카운트 동작의 기준을 만족시키는 각 엔트리(144)를 위하여, 대응 키맵 엔트리 획득 동작이 수행되는 경우, 리스트 및/또는 카운트 동작은 키맵 엔트리 획득 동작들 중의 특성 경우로서 실행될 수 있다. 하지만, 정족수 프로토콜에 따라 다중 복사들로부터 실제로 검출하는 엔트리 데이터(예를 들면, 기

록들(148))의 추가 오버헤드는 키맵 엔트리 리스트 또는 카운트 동작들용으로는 불필요할 수 있다. 즉, 일부의 실시예에서, 정족수 프로토콜에 관한 키맵 엔트리 획득 동작의 단계들은 키맵 엔트리 리스트 또는 카운트 동작으로부터 생략될 수 있다. 예를 들면, 블록들(1402~1404)에서와 같이 주어진 엔트리의 모든 복사들을 확인하고 각 복사에 대한 개별 획득 동작들을 생성하는 것보다 차라리, 리스트 또는 카운트 동작을 위하여, 단일 복사(예를 들면, 브릭(415))이 임의로 선택되고 그 대응 계층이 조정되어, 리스트 또는 카운트 동작을 만족시키는 각 엔트리(144)를 확인할 수 있게 된다. 기준을 만족시키는 결과로 얻어진 엔트리들(144), 대응 키들(146) 또는 결과로 얻어진 엔트리(144)의 카운트가 요청한 클라이언트로 반환되어, 도 14의 정족수-관련 처리 부분들(예를 들면, 블록들(1410~1418))을 바이패스한다.

[0169] 일부의 실시예에서, 키맵 인스턴스(140)는 엔트리들(144)을 인덱스하는데 사용된 다양한 데이터 구조들 외에 캐시(cach)를 실행할 수 있다. 예를 들면, 캐시는 종종 사용된 엔트리들(144)의 키들로 지향된 키맵 동작들이 대응 엔트리들(144)에 직접 접근하기 위해 인덱스 데이터 구조들의 조정을 바이스패스할 수 있도록 하고, 키맵 엔트리 획득 동작의 성능을 향상시킬 수 있다. 또한, 캐시는 인기있는 종종 접근한 키들에 관련된 호스트들(400)이 키맵 요청 트래픽(traffic)에 의해 과부하가 걸리게되는 것을 방지할 수 있다. 예를 들면, 키맵 캐시가 호스트들(400) 사이에 분배되는 일 실시예에서, 키의 복사는 키를 위한 인덱스 데이터 구조들을 유지하는 호스트(400)와는 다른 호스트(400)에 캐시될 수 있다. 이러한 호스트들(400) 사이의 키 캐시의 분배를 통하여, 키 처리 작업 부하는 호스트들(400) 사이에서 더욱 균일하게 공유될 수 있다.

[0170] 일 실시예에서, 키맵 캐시는 키들 자체가 아니라 차라리 키들(148)의 해쉬들에 의해 저장하고 인덱스되도록 구성될 수 있다. 불균형 인덱스 데이터 구조들의 설명에 관련한 아래에 자세히 설명된 데이터 해싱은 고정 길이 데이터 구조에서 키(148)와 같은 가변 길이 데이터를 표시하기 위한 효과적인 기술을 구성하고, 키맵 캐시 내에서 관리하기가 더 용이할 수 있다. 또한, 다양한 해시 알고리즘은 초기에 균일하게 분배될 수 없는 균일하게 분배된 데이터용 해시 값(예를 들면, 데이터의 상당 부분을 공통으로 갖는 키(148) 한 세트)을 발생하고, 호스트들(400) 간에 키맵 캐시 데이터의 균일한 분배를 촉진할 수 있다. 일부의 실시예에서, 엔트리(144)의 컨텐츠는 대응 키(148)가 해시된 값에 따라 키맵 캐시에 저장될 수 있다. 다른 실시예에서, 엔트리(144)의 컨텐츠보다 오히려 엔트리(144)용 포인터 또는 다른 기준 정보가 저장될 수 있다.

[0171] 일반적으로 말해, 키맵 캐시들을 갖는 키맵 실시예에서, 키맵 엔트리 풋 및 획득 동작은 상기한 설명에 작은 변경으로 행해질 수 있다. 일 실시예에서, 키맵 엔트리 획득 동작은 먼저 획득 동작이 캐시에 존재하는 데이터로부터 서비스될 수 있는 지를 결정하기 위하여 캐시를 참고할 수 있다. 풋 동작은 독출용 정족수 프로토콜에 의한 진행 전에 고정된 시간 동안 캐시로부터의 응답을 기다릴 수 있다. 만일 캐시가 정족수 프로토콜 독출이 시작된 후의 값을 반환하면, 캐시로부터 독출된 값은 처리되고 대응 엔트리(144)가 반환되고, 정족수 프로토콜 독출이 종료될 수 있다. 만일 캐시로부터 어떤 값도 반환되지 않으면, 정족수 프로토콜 독출 동작으로부터 독출된 엔트리(144) 또는 이러한 엔트리(144)로의 포인터가 대응 키 정보에 따라 키맵 캐시에 설치될 수 있다.

[0172] 일반적으로 말해, 캐시들을 갖는 키맵 실시예에서 키맵 엔트리 풋 동작은, 로킹(locking) 또는 일관성 프로토콜이 다중 풋 동작이 동일한 캐시 엔트리를 변경하는 동시 시도를 하는 것을 방지하기 위해 이용되는 것을 제외하고는 거의 상기한 바와 같이 동작할 수 있다. 일 실시예에서, 키맵 엔트리 풋 동작은 기입용 정족수 프로토콜을 시작하기 전에 키(148)에 대응하는 캐시 엔트리를 로킹하는 것을 시도하도록 구성될 수 있다. 로크 요청이 계속되는 캐시로부터 응답을 받은 경우(예를 들면, 엔트리에 대한 다른 로크가 존재하지 않으므로 또는 캐시에 대응하는 엔트리가 없으므로), 정족수 프로토콜은 진행할 수 있다. 정족수 프로토콜에 따라 풋 동작이 완료된 후, 로크가 방출되고 새로운 엔트리 데이터가 캐시 내에 설치될 수 있다.

[0173] 일부의 실시예에서, 상기한 바와 같이 키맵 엔트리 풋 및 획득 동작용 정족수 프로토콜들은 키맵 엔트리 상태를 업데이트하기 위한 강한 일관성 모델을 실행할 수 있다. 즉, 정족수 프로토콜들은, 특정 키에 대한 풋 동작이 완전한 것으로 클라이언트에게 확인되면, 심지어 모든 복사가 획득 동작이 처리될 때 업데이트되었더라도, 후속 획득 동작이 최근에 풋된 데이터를 반환하는 것을 보장할 수 있다.

[0174] 풋 및 삭제 동작과 같은 키맵 동작들이 특정 키맵 인스턴스(140)로 지향됨에 따라, 특정 키맵 인스턴스(140) 내 엔트리들(144)의 상태는 시간이 지남에 따라 변할 수 있다. 그래서, 이들을 조정하는 어떠한 시도도 없는 경우, 배치 내의 상이한 키맵 인스턴스들(140)이 시간이 지남에 따라 분기되거나 불일치되기 쉬울 수 있다. 만일 하나의 저장 서비스 클라이언트(50)가 주어진 객체(30)를 참조하고 동일한 키맵 인스턴스(140)를 통하여 그렇게 하면, 이러한 분기는 실제적인 효과를 가질 수 없다. 하지만, 만일 다중 저장 서비스 클라이언트들(50)이 다른 키맵 인스턴스들(140)을 통하여 동일한 키를 참조하면, 이러한 불일치는 클라이언트들(50)이 동일

한 시점에 객체 데이터의 다른 키맵 상태 및/또는 다른 버전을 관찰하도록 할 수 있다.

[0175] 상기한 바와 같이, 클라이언트들이 복사 불일치를 관찰하는 것을 효과적으로 방지하고 어쨌든 이러한 불일치가 발생하는 것을 방지하기 위해 복사들을 업데이트할 때, 원자 또는 정족수 프로토콜과 같은 강한 일관성 프로토콜이 이용될 수 있다. 하지만, 상이한 복사들이 접근 지연이 변할 수 있는 분배된 문맥(context)에서, 가끔 상당히 강한 일관성 프로토콜들이 높은 성능 비용을 가질 수 있다. 예를 들면, 원자 또는 정족수 프로토콜에 대한, 동작 완료용으로 요구된 시간은 모든 복사들 또는 정족수의 복사들 중의 가장 느린 것에 관한 동작을 각각 완료하도록 요구된 시간 함수일 수 있다. 또한, 강한 일관성 프로토콜들이 없는 경우, 클라이언트에 보일 수 있는 복사 불일치의 가능성(예를 들면, 저장 서비스 클라이언트(50)가 실효된 키맵 또는 객체 데이터를 얻을 가능성)은, 일반적으로 접근된 복사가 업데이트를 반영하지 않았을 때의 주기 동안 복사에 접근할 클라이언트의 가능성의 함수일 수 있다.

[0176] 많은 객체(30)에 있어서, 후자의 가능성은 낮을 수 있다. 예를 들면, 일부의 예에서, 저장 서비스 시스템에 의해 관리된 대다수의 객체(30)가, 경우 불일치가 클라이언트 견지에서 논의할 여지가 있을 수 있다. 특정 키맵 인스턴스(140)를 통하여 단일 클라이언트(50)에 의해 접근될 수 있다. 다중 클라이언트(50)에 의해 접근될 수 있는 객체들(50)에 대하여 관찰 가능한 불일치는 여전히 바람직하지 않을 수 있다. 예를 들면, 2개의 키맵 인스턴스(140)는 10초의 주기 동안 특정 키에 대하여 불 일치한다. 하지만, 만일 불일치 주기 동안 특정 키에 대하여 접근이 이루어지지 않거나(예를 들면, 대응 객체(30)의 접근들 간의 지속 기간이 불일치 주기보다 크거나), 수행된 접근이 더욱 최근에 업데이트된 키맵 인스턴스(140)로 지향되면(예를 들면, 키의 상태를 마지막에 업데이트한 클라이언트(50)가 동일한 키맵 인스턴스(140)를 통한 키를 참조한 다음이면), 불일치는 클라이언트들(50)에 관찰가능한 영향을 전혀 미치지 않을 수 있다. 따라서, 일부의 실시예에서 키맵 인스턴스들(140)은 키맵 인스턴스들(140)이 일치 상태로 분기되도록 노력하는 완화된 동기화 프로토콜을 이용할 수 있지만, 어느 주어진 시간에 키맵 인스턴스들(140) 간에 어느 정도의 불일치는 허용할 수 있다. 이러한 동기화 프로토콜은 더욱 엄격한 동기화가 불필요할 수 있는 대다수의 클라이언트들에게 더 우수한 전체 성능을 제공할 수 있다. 일부의 실시예에서, 공유된 객체들(30)을 위한 키맵 데이터의 더욱 엄격한 접근 동기화를 요구하는 클라이언트들(50)은, 모든 클라이언트들(50)이 더욱 엄격한 동기화의 부담이 발생하는 것을 요구하지 않으면서, 이들 사이의 추가 프로토콜들을 실행할 수 있다. 예를 들면, 특정 객체(30) 세트에 대한 접근을 공유하는 클라이언트(50) 세트는 세마포어(semaphore) 또는 다른 분배된 로킹 기술을 이용하여 키맵 데이터로의 접근을 조정할 수 있다.

[0177] 일부의 실시예에서, 키맵 인스턴스들(140) 사이에 완화된 동기화 프로토콜은 동기화 처리의 상이한 양상들을 독립적으로 수행할 수 있는 상기 동기화 작업들의 결합을 포함할 수 있다. 도 15A 및 도 15B는 2개의 별개 동기화 작업: 도 15A에 도시된 업데이트 전파 작업 및 도 15B에 도시된 앤티-엔트로피(anti-entropy) 또는 설정 조정 작업을 포함하는 완화된 동기화 프로토콜의 동작 방법의 일 실시예를 나타낸다. 먼저, 도 15A를 참조하면, 키맵 인스턴스들(140) 중의 하나에 대한 업데이트가 검출되는 경우, 동작이 블록 1500에서 시작된다. 예를 들면, 키맵 인스턴스(140)는 상기한 바와 같이 정족수 프로토콜에 따라 키맵 엔트리 풋 또는 삭제 동작을 수신하여 완료할 수 있다.

[0178] 그 후, 키맵 업데이트를 처리한 키맵 인스턴스(140)가 저장 서비스 시스템 내에 제공된 키맵 인스턴스(140)에게 상호 업데이트 동작을 전달할 수 있다(블록 1502). 예를 들면, 키맵 인스턴스(140a)가 키맵 엔트리 풋 동작을 처리하면, 키맵 인스턴스(140a)는 변수들, 파라미터들 등을 포함하는 동작을 키맵 인스턴스들(140b 및 140c)로 전달할 수 있다. 일 실시예에서, 이 전달은 인증 또는 확인 없이 수행될 수 있다. 예를 들면, 키맵 업데이트 동작을 처리한 키맵 인스턴스는 비동기식 메시징(fire and forget) 프로토콜을 이용하여 이 동작을 전달하여, 하나의 시도가 시도없이 상호 간의 동작을 키맵 인스턴스로 전달하도록 하여 전달된 동작이 목적지에서 수신되었는 지를 인증하거나 수신되지 않으면 이 동작을 재전송하도록 한다. 이러한 전달은 시작 키맵 인스턴스(140)로부터 다중 키맵 인스턴스들(140)로의 동시 방송, 시작 키맵 인스턴스(140)로부터 다른 예들로의 순차 전달, 또는 트리 기반 전략 등과 같은 어느 적당한 전달 전략을 이용하여 이루어질 수 있다.

[0179] 전달된 동작을 수신한 관련 호스들(400)은 업데이트 동작을 국부적으로 수행할 수 있다(블록 1504). 예를 들면, 호스트(400f)가 호스트(400a)로부터 전달된 키맵 엔트리 풋 동작을 성공적으로 수신한 경우, 호스트(400f)는 어느 키맵 클라이언트로부터의 동작을 수신했을 때처럼 이 동작을 수행할 수 있다. 풋 동작이 호스트(400f)에서 성공적으로 완료되면, 결과로서, 키맵 인스턴스들(140a 및 140b)이 풋 동작에 대하여 동기화될 수 있다.

[0180] 일반적으로 말해, 호스트들(400) 사이의 전달 키맵 업데이트 동작들은 대부분 시간 동안 계속된 것으로

예상될 수 있다. 따라서, 이러한 동작들을 전달할 때 포함된 오버헤드의 최소화는 대부분의 경우에 키맵 인스턴스들(140) 사이의 동기화를 달성하는데 요구된 시간 및/또는 대역폭을 감소시킬 수 있다. 예를 들면, 전달 처리로부터의 확인 응답, 또는 다른 방식의 프로토콜 인증 또는 주고받기의 제거는 더큰 정도의 동기화 교환을 포함한 키맵 실행의 더 큰 스케일을 지지하는 것과 같은 다른 용도용 대역폭을 자유롭게 통신할 수 있게 한다. 많은 예에서, (일반적으로 주어진 키맵 엔트리(144)의 복사들의 지연적 불일치의 윈도우에 대응할 수 있는) 키맵 배치를 통하여 키맵 업데이트를 전파하는데 요구된 시간은 이 동작을 관련된 호스트들(400)로 전달하는데 요구된 통신 지연 및 호스트들(400)이 전달된 동작을 적용하는데 요구된 처리 지연으로 제한될 수 있다. 종종, 전체 시간은 초단위이거나 분수 초단위 일 수 있다.

[0181] 하지만, 일부의 실시예에서, 호스트들(400) 간의 키맵 업데이트의 전달은 실패할 수 있다. 예를 들면, 통신 링크 실패는 하나의 호스트(400)가 다른 호스트로부터 도달되지 않도록 하거나, 전달된 동작이 분실되거나, 생략되거나 그렇지 않으면 이동 중에 손상되게 할 수 있다. 대안으로, 목적지 호스트(400)는 예를 들면, 일시적인 하드웨어 또는 소프트웨어 문제(issues)로 인하여, 적당히 전달된 업데이트 동작을 수신하거나 정확히 처리하지 못할 수 있다. 일 실시 예에서처럼, 만일 전달된 키맵 업데이트 동작이 타켓 호스트들(400)에 의해 성공적으로 수신되어 처리되는 것을 인증하거나 보장하기 위해 시작한 호스트(400)의 일부에 어떠한 시도가 이루어지지 않으면, 개별 동작의 전달 실패는 어떤 엔트리들(144)에 대한 키맵 인스턴스들(140) 사이의 불일치를 일으킬 수 있다.

[0182] 따라서, 일 실시예에서, 키맵 인스턴스들(140) 사이의 완화된 동기화 프로토콜은 상기에서 설명되고 도 15B에 도시된 앤티-엔트로피 또는 설정 조정 작업을 포함할 수 있다. 이 작업은 동작이 다른 키맵 인스턴스들(140) 사이의 차이를 감소시키고 유사성을 증가시키고, 예들을 적당히 동기화하기 위한 업데이트 전파의 랜덤하거나 체계적인 고장에 의해 도입될 수 있는 키맵 인스턴스들 사이의 전체 엔트로피를 감소시키는 기능을 하는 점에서 "앤티-엔트로피"로 불릴 수 있다. 설명된 실시예에서, 시작 키맵 인스턴스(140)가 특정 파티션의 조정을 수행하고, 상기한 바와 같이 다른 호스트들(400)에 존재하는 복사된 다수의 브릭들(415)을 가질 수 있는 다른 키맵 인스턴스(140)를 랜덤하게 선택하는 동작을 블록 1510에서 시작한다.

[0183] 그 후, 시작 키맵 인스턴스(140)는 선택된 키맵 인스턴스(140)를 갖는 예 내의 파티션들에 관한 정보를 교환할 수 있다(블록 1512). 예를 들면, 2개의 키맵 인스턴스들(140) 내의 특정 호스트들(400)은 각 예 내에 유지된 파티션 인덱스(410)의 복사들을 교환한 후, 각 예 내에 정의된 브릭들(415)을 확인하도록 구성될 수 있다.

[0184] 그 후, 교환된 파티션 정보에 기초하여, 시작 키맵 인스턴스(140)는 2개의 예 내의 파티션들 사이의 상응을 확인하고(블록 1514), 선택된 키맵 인스턴스(140) 내에 포함된 대응 파티션을 갖는 시작 키맵 인스턴스 내의 각 파티션을 조정할 수 있다(블록 1516). 예를 들면, 상기한 바와 같이, 주어진 키맵 인스턴스(140) 내의 각 파티션은 다수의 브릭들(415)을 통하여 복사될 수 있다. 일 실시예에서, 시작 키맵 인스턴스(140)는 선택된 키맵 인스턴스(140) 내의 대응 파티션의 대응 또는 "피어(peer)" 브릭(415)과 통신하는 ("리드 브릭(lead brick)"으로 불릴 수 있는) 파티션 내의 특정 브릭(415)을 지시하여, 파티션들 사이의 차이를 조정하도록 구성될 수 있다. 일 실시예에서, 2개 브릭들(415)에 대한 조정은 각 브릭(415)에 포함된 키맵 엔트리들(144)에서의 차이에 관한 정보를 교환하고, 각 키맵 인스턴스(140) 내의 가장 최근 정보를 전파하는 브릭들을 포함할 수 있다. 예를 들면, 하나의 브릭(415)이, 시간 스탬프 정보에 기초하여 하나의 브릭(415)의 엔트리(144)의 버전이 피어 브릭(415)의 엔트리의 버전 보다 현재 더 많은 것을 결정하면, 하나의 브릭은 엔트리 데이터를 피어 브릭(415)에 전달할 수 있다. 이어서, 피어 브릭(415)은 (예를 들면, 상기한 바와 같은 정족수 프로토콜에 따라) 키맵 엔트리 풋 동작을 수행하여 엔트리(144)의 피어 브릭의 복사를 업데이트할 수 있다.

[0185] 일단, 2개 키맵 인스턴스들(140) 간의 파티션 조정이 완료되면, 조정 처리가 다른 무작위 키맵 인스턴스(140)에 관하여 다시 시작하는 경우, 동작은 블록 1510으로부터 계속할 수 있다. 다양한 실시예에서, 각 키맵 인스턴스(140)는 소정 또는 동적으로 결정된 간격으로 이 처리를 수행하도록 구성될 수 있다. 예를 들면, 조정(reconciliation)은 분당 한번 정지 비율로 또는 랜덤 또는 다른 통계 확률 분배에 따라 결정된 간격으로 이루어질 수 있다. 일부의 실시예에서, 어느 회수의 키맵 접근이 이루어진 후 또는 키맵 엔트리들의 어떤 개별적인 것들, 방식들, 또는 그룹에 대한 접근이 검출된 후에 조정이 수행될 수 있다.

[0186] 일반적으로 말해, 도 15A 및 도 15B에 도시된 업데이트 전파 및 설정 조정 또는 앤티-엔트로피 방법이 보상 방식으로 동작할 수 있다. 대부분의 환경 하에, 업데이트 전파는 배치 내의 상이한 키맵 인스턴스들(140)을 만족스럽게 동기화시킬 수 있다. 키맵 불일치가 업데이트의 실패로 인하여 발생하는 예들에서, 앤티-엔트로

피 작업은 일반적으로 이러한 불일치를 조정하도록 동작할 수 있다. 일부의 실시예에서, 앤티-엔트로피 작업의 실행은, 2개의 키맵 인스턴스들(140)이 전체적으로 정밀하게 동기화되는 것을 보장할 수 없다. 하지만, 일 실시예에서, 앤티-엔트로피 작업은 그 동작이 2개의 키맵 인스턴스들(140) 사이의 불일치 정도를 증가시키지 않는 것을 보장하도록 실행될 수 있다. 그래서, 반복된 어플케이션들에 의하여, 앤티-엔트로피 작업은 키맵 인스턴스(140)의 집중을 촉진할 수 있다. 앤티-엔트로피 작업이 특별한 실시예가 키맵 인스턴스(140)가 실행될 수 있는 데이터 구조의 특정 실시예의 설명에 관련하여 아래에 더욱 자세히 설명된다.

[0187] 도 2에 도시되고 상기한 바와 같이, 일부의 실시예에서, 저장 서비스 시스템은 반복자 키맵 인스턴스(190) 및 다른 키맵 인스턴스들(140)을 포함할 수 있다. 일 실시예에서, 반복자 키맵 인스턴스(190)는 상기한 키맵 인스턴스들(140)에 본질적으로 동일하게 구성될 수 있고, 상기한 프로토콜들을 이용하여 키맵 동기화에 참여할 수 있다. 하지만, 이러한 실시예에서, 반복자 키맵 인스턴스(190)은 조정자들(120) 또는 다른 키맵 클라이언트들 보다는 차라리 반복자(180)를 서비스하도록 구성될 수 있다. 일부의 실시예에서, 다른 키맵 인스턴스들(140)로부터 반복자 키맵 인스턴스(190)를 분리하는 것은 일반적으로 키맵 성능을 향상시킬 수 있다. 예를 들면, 반복자(180)가 객체들(30)의 건강 및 복사 개수를 점검하기 위한 키맵을 통하여 되풀이됨에 따라, 반복자(180)는 본질적인 키맵 요청 소통 양을 발생할 수 있다. 반복자 키맵 소통이 저장 서비스 클라이언트들(50)의 요청을 위하여 키맵 소통에 혼합되면, 반복자 키맵 소통이 응답 시간 또는 클라이언트들(50)에 적절한 다른 서비스 품질(quality-of-service) 측정에 나쁜 영향을 줄 수 있다. 이와는 반대로, 전용 키맵 인스턴스(190)를 이용하는 반복자(180)를 구성함으로써, 클라이언트-발생한 소통으로부터 내부에서 발생한 키맵 소통을 분리할 수 있다. 또한, 이러한 분리는 각 방식의 키맵 인스턴스의 실행이 주요 클라이언트의 요구에 따라 더욱 잘 스케일링될 수 있도록 한다. 예를 들면, 반복자 키맵 인스턴스(190)에 대한 실행은 어느 주어진 키맵 동작의 지연을 최소화시키는 것보다 차라리 큰 수의 동시 키맵 동작의 처리를 촉진하도록 구성될 수 있는 반면에, 키맵 인스턴스들(140)은 서비스 품질 기준의 다른 결합용으로 최적화될 수 있다. 하지만, 이 방식으로의 키맵 인스턴스들의 분리가 요구되지 않고, 일부의 실시예에서 반복자(180)가 전용 반복자 키맵 인스턴스(190)보다는 차라리 키맵 인스턴스들(140)의 클라이언트일 수 있다는 것이 주목된다.

[0188] 일 실시예에서, 반복자 키맵 인스턴스는 또한 클라이언트들(50)에 의한 저장 서비스 시스템 자원들의 사용의 어카운팅을 촉진하도록 구성될 수 있다. 특히, 반복자 키맵 인스턴스(190)는 대응 객체들(30)에 대한 과금 또는 다른 재정적 책임을 지는 각 엔티티를 나타내는 부가 데이터와 함께 키맵 인스턴스들(140)에 의해 저장된 엔트리들(144)을 증가시키도록 구성될 수 있다. 예를 들면, 도 16에 도시된 실시예에, 반복자 키맵 엔트리(194)가 설명되어 있다. 반복자 키맵 인스턴스(10) 내 엔트리(194)는 키맵 인스턴스(140)의 구조 및 계층에 관한 엔트리들(144)과 동일하게 기능을 할 수 있다. 하지만, 설명된 실시예에서, 엔트리(194)는 추가 필드, 버킷 ID(196)를 포함한다. 일반적으로 말해, 버킷 ID(196)은 키(146)에 대응하는 객체(30)를 포함하는 버킷(20)의 식별자의 표시를 포함할 수 있다. 이러한 식별자는, 상기한 바와 같이 클라이언트(50)로부터의 요청에 응답하여 웹 서비스 인터페이스(100) 또는 조정자(120)에 의해 정의되어 객체들(30)을 저장하기 위한 버킷(20)을 생성할 수 있다. 다른 실시예에서, 어카운팅 정보는 반복자 키맵 인스턴스(190)의 엔트리들 내에 단독으로 반영될 필요가 없다. 예를 들면, 일 실시예에서 키맵 인스턴스들(140)의 일부 또는 모두의 키맵 엔트리들(144)는 예를 들면, 기록(148) 또는 키(146) 내의 추가 필드로서 버킷 ID(196)의 표시를 저장하도록 구성될 수 있다.

[0189] 상기한 바와 같이, 객체들(30) 및 버킷들(20) 사이의 관계는 키맵 인스턴스들(140)의 일반적인 동작에 투명할 수 있다. 하지만, 이 관계가 통상적으로 고정되는 것으로 주어지면, 반복자 키맵 엔트리들(194)을 통한 버킷들(20) 및 객체들(30)을 명백히 관련시키는 것은 클라이언트들(50)의 어카운팅 및 과금을 촉진할 수 있다. 예를 들면, 상호 결합된 버킷(20)용 웹 서비스 인터페이스(100)를 명백히 조회하는 것보다 차라리 (반복자(180) 또는 다른 모듈 내에 포함되거나 시스템 내의 별개 모듈로서 실행될 수 있는) 어카운팅 처리가 버킷 ID(196)에 따라 반복자 키맵 엔트리들(194)을 분류하도록 구성될 있다. 이러한 분류를 완료한 경우, 특정 버킷 ID(196)에 관련된 모든 키들(146)이 명백해진다. 그 후, 기록들(148) 내에 표시된 바와 같이 대응 객체들(30)의 크기들은 버킷 ID(196)에 관련된 전체 저장 자원 사용을 결정하도록 모여질 수 있다. 또한, 객체들(30)의 다른 특징은 특정 객체(30)에 관련된 저장 등급과 같이 고려될 수 있다. 그 후, 자원 사용은 적당한 과금 모델에 따라 통화로 정해질 수 있다.

[0190] 다양한 실시예에서, 반복자 키맵 엔트리들(194)은 다양한 내부 시스템 유지 또는 어카운팅 작업을 촉진할 수 있는 버킷 ID(196) 대신 또는 외에 다른 필드들을 포함할 수 있다. 반복자 키맵 인스턴스(190)이 다른 키맵 인스턴스들(140)과는 다른 실시예에서, 이러한 추가 필드들의 저장 비용은 반복자 키맵 인스턴스(190)를 배제하도록 구성될 수 있다는 것이 주목된다. 하지만, 전용 반복자 키맵 인스턴스(190)가 없는 실시예에서, 키맵

인스턴스들(140)의 엔트리들(144)은 이러한 추가 필드들을 포함하도록 증가할 수 있다.

[0191] 계층화된 불균형 데이터 구조

[0192] 상기한 바와 같이, 일부의 실시예에서 저장 서비스 시스템은 매우 많은 수 예를 들면, 약 10억 이상의 객체들(30)을 지지하도록 크기가 조정될 수 있다. 그래서, 이러한 실시예에서, 각 키맵 인스턴스(140)는 관리할 유사한 수의 엔트리들(144)을 가질 것이다. 일부의 실시예에서, 키맵 인스턴스들(140)은 이전 섹션에 설명된 키맵 엔트리 리스트 및 카운트 동작과 같은 다양한 방식의 분류 및/또는 그룹핑 동작을 지지할 수 있다. 또한, 일치 키맵 동작을 지지하기 위하여, 각 키맵 인스턴스(140)에 의해 관리된 많은 키들이 상기한 바와 같은 다른 키맵 인스턴스들 사이에 동기화될 필요가 있을 수 있다.

[0193] 많은 환경에서, 키맵 인스턴스들(140)에 의해 제공된 키맵 기능은 전체 저장 서비스 시스템의 동작의 중심이다. 예를 들면, 만일 클라이언트들(50)이 객체들(30)의 특정 예들에 대한 로케이터(locator) 기반 접근을 수행하지 않도록 선택하면, 키맵 인스턴스들(140)은 클라이언트들(50)에 의해 수행된 모든 키 기반 객체 접근 요청을 중재할 수 있다. 그래서, 클라이언트들(50)에 의해 보여진 바와 같은 저장 서비스 시스템의 성능은, 키맵 인스턴스들(140)이 키맵 엔트리들(144)을 접근하여 처리할 경우의 효과 및 속도에 직접 의존할 수 있다. 다음에, 키맵 인스턴스들(140)의 성능은 도 12의 실시예에서 파티션 인덱스들(410), 블록 인덱스들(420), 및 엔트리 인덱스들(430)을 실행하는데 이용된 데이터 구조들과 같은, 엔트리들(410)을 인덱스하여 구성하는데 사용된 데이터 구조들에 직접 의존할 수 있다.

[0194] 큰-스케일 키맵 실행에서의 분류 및 동기화 동작을 지지하기 위한 인덱스 데이터 구조들을 디자인하는 것은 상당한 도전을 제시할 수 있다. 데이터베이스들과 같은 대량의 데이터의 인덱싱을 요구하는 종래의 어플리케이션들은 종종 B-트리들 또는 다른 방식의 균형 트리들과 같은 종래의 균형 트리들을 사용한다. 일반적으로 말해, 균형 데이터 구조 알고리즘들이 키맵 엔트리들(144)과 같은 주어진 양의 데이터 아이템들을 인덱스하는데 이용된 경우, 이 균형 데이터 구조 알고리즘들은 관리할 아이템들의 양에 따른 균형 데이터 구조를 가로질러 데이터 아이템을 분배할 것을 시도한다. 예를 들면, 10,000개의 키맵 엔트리(144)가 인덱스되는 경우, 균형 데이터 구조 알고리즘은 엔트리들(144) 사이의 정지 지점들을 선택하는 것을 시도할 수 있다. 균형 데이터 구조 알고리즘이 예를 들면, 대략 1,000개의 엔트리를 대략 200개 엔트리의 5개의 서브 그룹으로 세분하는 각 그룹 내에 균형 계층의 레벨들을 더 생성할 수 있다. 데이터 아이템들이 균형 데이터 구조에 및 균형 데이터 구조로부터 각각 첨가되고 삭제됨에 따라, 데이터 구조 내의 그룹들 및/또는 서브그룹들은 불균형하게 될 수 있다. 그래서, 종래의 균형 데이터 구조 알고리즘들은 그룹들 사이에 데이터 아이템들을 재 할당함으로 데이터 구조를 균형화하여 추가 그룹들 및/또는 추가 계층의 레벨들을 생성한다. 이러한 재 균형화는, 데이터 아이템들이 추가되거나 삭제됨에 따라 "온 더 플라이"로 발생하거나, 몇 회의 데이터 아이템 변경이 발생한 후 또는 최종 재 균형화 이래로 어느 정도의 시간이 경과한 후 발생할 수 있다.

[0195] 균형화 방식으로 데이터 아이템들을 분리함으로써, 균형 데이터 구조들은 데이터 구조 내 어느 주어진 데이터 아이템에 대하여 예측 가능하고 대략 균일한 접근 지연을 나타낼 수 있고, 큰 수의 데이터 아이템들이 인덱스될 필요가 있는 경우 큰-스케일 실행에 바람직할 수 있다. 하지만, 예를 들면, 상기한 바와 같은 완화된 동기화 모델을 이용하여 균형 데이터 구조들의 분배된 예들을 효과적으로 조정하거나 동기화시키기는 특히 어려울 수 있다. 특히, 균형 데이터 구조들의 예들이 독립적으로 변경됨에 따라, 데이터 아이템들을 각 예 내의 그룹들로 분리하는 정지 지점들은 분기될 수 있다. 결과로서, 상이한 균형 데이터 구조 예들의 그룹들 또는 서브그룹들 사이의 데이터 아이템 멤버쉽에 의한 직접적인 상응은 없을 수 있다. 그 후, 2개의 이러한 예들을 조정하기 위하여, 2개의 예들 전체를 철저하게 비교할 수 있고, 각 예가 큰 수의 데이터 아이템을 인덱스하는 경우에 큰 시간 소모가 있을 수 있다.

[0196] 양에 따라 그룹들 사이의 데이터 아이템들을 분배하는 균형 데이터 구조들에 대한 다른 방법으로서, 일부 실시예에서 키맵 인스턴스들의 인덱스 데이터 구조들이 각 그룹 내 데이터 아이템들을 분배하는 (또한 시도들로 불릴 수 있는) 불균형 데이터 구조들을 실행하도록 구성될 수 있다. 특히, 키맵 인스턴스들(140)은 대응키들(146)의 접두부들에 따라 엔트리들(144)을 인덱스하도록 구성될 수 있다. 일 예로서, 대응하는 경우-민감 수문자식 키들(146)을 갖는 600개의 키맵 엔트리들(144)이 존재하는 경우를 고려한다. 600개의 키맵 엔트리들(144)의 균형 인덱스는 엔트리들을 각각 200개의 엔트리들의 3개의 균형 그룹들을 분리할 수 있다. 대조적으로, 일 실시예에서 불균형 인덱스는 3개의 수문자식 그룹들을 정의하여, 문자들 a 내지 l로 시작하는 엔트리들은 제1 그룹에 할당되고, 문자들 m 내지 x로 시작하는 엔트리들은 제2 그룹에 할당되고, 문자 y 또는 z 혹은 숫자들 0~9로 시작하는 엔트리들은 제3 그룹에 할당될 수 있도록 한다.

[0197] 엔트리들(144)은 불균형 인덱스의 그룹들을 통하여 균일하게 분배될 수 있다. 예를 들면, 제1 그룹에는 300개의 엔트리, 제2 그룹에는 250개의 엔트리, 및 제3 그룹에는 단지 50개의 엔트리가 포함될 수 있다. 하지만, 어느 주어진 엔트리(144)에 있어서, 불균형 인덱스의 특정 그룹에 속하는 주어진 엔트리(144)의 멤버쉽은 어느 특정 그룹에 있는 엔트리들(144)의 수에 의존하지 않는 대응 키(146)의 함수일 수 있다. 그래서, 불균형 인덱스의 2개의 예가 동일한 그룹 정의를 유지하면, 각 그룹은 다른 그룹들에 의존하지 않으면서 독립적으로 동기화될 수 있다. 예를 들면, 2개 예 사이의 a-l 그룹들은 m-x 그룹들 및 y-9 그룹들과 무관하게 동기화될 수 있다. 대조적으로, 상기한 바와 같이, 동일한 엔트리(144) 세트의 균형 인덱스의 2개의 예에 대한 동기화는 고려될 모든 그룹을 통한 모든 엔트리를 요구할 수 있다.

[0198] 다수의 데이터 아이템들을 인덱스하기 위한 불균형 데이터 구조의 이용을 설명하는 일 예가 도 17에 도시되어 있다. 도시된 실시예에서, 균형 인덱스(200)(단순히 인덱스(200))는 접두부 "al"로 시작하는 다수의 스트링 값들을 인덱스하기 위한 계층화 방식으로 배열된 다수의 노드들(210)을 포함한다. 예를 들면, 인덱스된 값들은 키맵 인스턴스(140)의 다양한 엔트리들(144)의 키들(146)에 대응할 수 있다. 인덱스(200) 내의 각 노드(210)는 인덱스할 데이터 아이템에 직접 대응하거나 대응하지 않을 수 있는 관련 태그 값을 포함한다. 설명된 실시예에서, 타원형으로 표시된 노드들은 대응하는 데이터 아이템들을 가지지 않는 인덱스(200)의 내부 노드들에 대응할 수 있지만, 사각형으로 표시된 노드들은 인덱스된 데이터 아이템들에 대응할 수 있다. 그래서, 예를 들면, 노드(210a)는 스트링 "al"에 대응하고, 인덱스(200) 내의 다수의 다른 노드들에 관련이 있지만, 스트링 "al"에 대응하는 실제 키(146)가 존재하지 않을 수 있다. 대조적으로, 태그 "alicia"를 갖는 노드들(210n)은 동일한 스트링을 지정하는 키(146)에 대응할 수 있다. 내부와 비-내부 노드들(210) 사이의 구별은 노드(210)의 상태에서 명백하게 반영될 수 있거나 되지 않을 수 있다.

[0199] 상기한 바와 같이, 일부의 실시예에서 균형 데이터 구조는 다른 인덱스들의 인덱스로서 구성될 수 있다. 이러한 일부 실시예에서, 인덱스(200)의 제1 예 내에 인덱스된 데이터 아이템은 다른 인덱스의 기본 노드(210)일 수 있고, 제1 인덱스(200) 내의 대응 노드(210)는 비-내부 노드로 고려될 수 있다. 즉, 일부의 실시예에서 주어진 인덱스(200)의 비-내부 노드(210)는 일반적으로 엔트리(144)와 같은 데이터 값에 관련된 어느 노드(210) 또는 주어진 인덱스(200) 외부에 있는 다른 인덱스(200)의 기본 노드로서 정의될 수 있다. 마찬가지로, 주어진 인덱스(200)의 내부 노드는 주어진 인덱스(200) 내의 다른 노드들(210) 만을 참조할 수 있고, 엔트리(144)에 관련된 것 또는 주어진 인덱스(200)와는 구별되는 다른 인덱스(200)를 지지하지 않을 수 있다. 도 17에 도시된 바와 같이, 비-내부 노드들(210)은 필수적으로 남겨진 노드들(예를 들면, 저 계층 레벨로 다른 노드들을 참조하지 않는 노드들)은 아니다.

[0200] 다양한 실시예에서, 각 노드(210)는 다양한 정보를 인코딩할 수 있다. 노드 내에서 인코딩될 수 있는 다양한 데이터 필드를 설명하는 일반(generic) 코드(210)의 일 실시예가 도 18에 도시되어 있다. 도시된 실시예에서, 노드(210)는 태그 필드(212), 카운트 필드(214), 지문 필드(216), 및 적어도 하나의 포인트 필드(218)를 포함한다. 일반적으로, 태그(212)는 아래에 상세히 설명되는 바와 같이, 인덱스(200)를 측정하거나 조작하는 과정에 사용될 수 있는 주어진 노드(210)에 대응하는 값을 저장하도록 구성될 수 있다. 일부의 실시예에서, 태그(212)는 인덱스(200) 내의 모든 노드들 사이의 노드(210)를 유일하게 확인할 수 있도록 한다. 또한, 일부의 실시예에서, 주어진 노드(210)의 태그(212)는 접두부들로서 인덱스(200) 내의 주어진 노드(210)의 모든 직접적인 조상들의 태그들(212)을 포함할 수 있다. 즉, 어느 주어진 노드(210)의 태그(212)는 어떤 값을 주어진 노드의 직접 부모 노드(210)의 태그에 첨부함으로써 결정될 수 있다. 예를 들면, 태그 "alicia"를 갖는 도 17의 노드(210n)를 고려한다. 노드들(210n)의 직접적인 조상 노드들(2101, 210k, 및 210a) 각각은 노드(210n)의 태그의 적당한 접수부를 형성하는 태그("alic", "ali", 및 "al")를 갖는다.

[0201] 도 17에 도시된 바와 같이, 어떤 노드들(210)은 인덱스(200)의 계층에서 더욱 아래에 적어도 하나의 자식 또는 후손 노드들(210)을 언급한다. 일 실시예에서, 포인터 필드(들)(218)은 주어진 노드(1210)로부터 다른 노드(210)까지의 포인터들 또는 참조들을 반영하는 데이터를 저장하도록 구성될 수 있다. 예를 들면, 주어진 포인터 필드(218)는 메모리 어드레스 공간과 같은 어드레스 공간 내에 참조된 노드(210)의 위치를 확인하도록 하는 어드레스를 포함할 수 있다. 주어진 포인터 필드(218)는 또한 참조된 노드(210)에 관한 추가 태그 정보를 포함할 수 있다. 예를 들면, 도 17에 도시된 바와 같이, 주어진 노드(210)로부터 후손 노드(210)까지의 각 아크는 주어진 노드(210)의 태그(212)에 의해 형성된 접두부와는 다른 후손 노드(210)의 태그(212)의 제1 문자로 라벨이 붙여진다. 일 실시예에서, 이 추가 태그 정보는 참조된 노드(210)에서의 포인터에 따라 대응하는 포인터 필드(218) 내에 저장될 수 있다. 예를 들면, 노드들(210a)에 포함된 포인터 필드들(218)은 각각 노드들(210b, 210g, 210j, 210k, 및 210t)에 대한 참조들 및 대응하는 태그 데이터 "a", "e", "f", "i", 및 "z"를 포함할

수 있다.

[0202] 도 12에 대하여 상기한 바와 같이, 인덱스(200)와 같은 인덱스는 선택용 키맵 엔트리들(144)과 같은 데이터 아이템들을 구성하도록 사용될 수 있다. 일부의 실시예에서, 비-내부 노드(210)(즉, 인덱스할 데이터 아이템에 직접 매핑하는 노드(210))의 포인터 필드(218)는 또한 키맵 엔트리(144), 블록(425), 및 브릭(415)과 같은 대응하는 데이터 아이템에 포인트를 포함할 수 있다. 일부의 실시예에서, 아래에 자세히 설명되는 바와 같이, 인덱스(200)와 같은 불균형 인덱스들은 계층적으로 실행되어, 하나의 인덱스(200)의 비-내부 노드(210)가 다른 인덱스(200)를 참조할 수 있도록 한다. 인덱스된 데이터 아이템을 참조하는 포인터 필드(218)는 상이한 방식의 포인터 필드들(218)에 대한 별개 인코딩을 이용하는 것과 같은 어느 적당한 기술을 통하여 다른 노드(210)를 참조하는 포인터 필드(218)와 구별될 수 있다. 예를 들면, 이전 단락에 설명된 바와 같이 후손 노드들(210)에서의 자식들에 관련된 태그 정보가 포인터 필드들(218) 내에 인코딩되는 실시예에서, 널 태그는 인덱스된 데이터 아이템에 대한 참조를 후손 노드들(210)에서의 참조와 구별하도록 사용될 수 있다.

[0203] 주어진 노드(210)에 있어서, 카운트 필드(214) 및 지문 필드(216)는 주어진 노드(210) 아래의 노드들(210)의 상태를 반영하도록 구성될 수 있다. 일 실시예에서, 카운트 필드(214)는 주어진 노드(210)의 (예를 들면, 아래에 계층화된) 후손들인 모든 노드들의 카운트를 저장하도록 구성될 수 있다. 예를 들면, 도 17의 노드(210k)는 인덱스(200) 내 노드 아래에 8개의 다른 노드들(210)을 갖는다.

[0204] 다양한 실시예에서, 주어진 노드(210)의 지문 필드(216)는 주어진 노드(210) 아래에 계층화된 노드들(210)의 데이터의 일부에서 수행된 해시(예를 들면, 적당한 해시 알고리즘의 결과)를 나타내는 값을 저장하도록 구성될 수 있다. 예를 들면, 주어진 노드(210)의 지문 필드(216)는 주어진 노드(210)의 자손들인 모든 노드들(210)의 태그들(212)의 해시들의 접합(concatenation)을 반영할 수 있다. 대안으로, 지문 필드(216)는 순회(traversal)(예를 들면, 폭우선 또는 깊이우선 순회)의 특정의 일관된 순서에 따라 후손 노드들(210)의 태그들(212)의 연결의 해시를 반영할 수 있다. 다른 실시예에서, 태그(212) 외에 노드(210)의 다른 필드들은 해싱에 참여할 수 있다. 일부의 실시예에서, 주어진 노드(210)에 관련된 데이터는 그 자체 지문 필드(216) 내에 반영될 수 있는 반면, 다른 실시예에서 주어진 노드(210)의 지문 필드(216)는 그 후손 노드들에 기초하여 엄격히 결정될 수 있다. 설명의 일치를 위해, 여기에 설명된 바와 같이, 주어진 노드(210)의 지문은 주어진 노드(210)의 적어도 일부 후손 노드들의 함수인 해시 값을 참조할 수 있지만, 주어진 노드(210)의 해시는 단지 주어진 노드(210)에 관련되고 후손들이 아닌 데이터의 함수인 해시 값을 참조할 수 있다.

[0205] 일반적으로 말해, 해시 알고리즘은 가능한 임의 길이의 주어진 소스 데이터 값을 통상적으로 작은 고정된 길이 해시 값으로 매핑하도록 구성될 수 있어, 만일 2개의 해시 값이 다르면, 2개의 해시 값이 발생된 원래 소스 데이터 값도 일부 방법에서 달라지도록 해야 한다. 해시 알고리즘들은 통상적으로 1대 1 함수가 아니므로, 2개의 해시 값들 사이의 식별자는 원래 소스 데이터 값들 사이의 식별자를 필수적으로 포함하지 않는다. 하지만, 해시 알고리즘들의 일부 등급에 있어서, 주어진 동일한 해시 값들의 원래 소스 데이터 값들 사이의 식별자는 특히 통계적으로 어느 리던던시 정도를 나타내는 소스 데이터 값들에 대한 성질상 가능성 또는 신뢰도 내에 있을 것 같다. 다른 방식의 해시 알고리즘들은 또한 서명, 지문, 또는 체크섬 알고리즘이라고 불릴 수 있다. 어느 적당한 방식의 해시 알고리즘들은, 비-제한적 목적의 예시로서, 어느 적당한 버전의 메시지 다이제스트 5(MD5) 또는 SHA-1, SHA-256, SHA-512 등과 같은 보안 해시 알고리즘(SHA)을 포함하는 지문 필드들(216)에 저장된 해시 값을 발생하도록 사용될 수 있다는 것이 예측된다.

[0206] 이전 섹션에서 설명된 바와 같이, 키맵 인스턴스(140)에서 수행될 있는 기본 동작들은 파라미터로서 동작에 지정된 키에 대응하는 엔트리(144)를 각각 저장하고 검색할 수 있는 풋 및 획득 동작들을 포함할 수 있다. 일부의 실시예에서, 키맵 인스턴스들(140) 내의 다양한 인덱스들이 인덱스(200)와 같은 불균형 인덱스들로서 실행될 수 있다.

[0207] 다수의 데이터 아이템들이 인덱스될 필요가 있는 경우, 키맵 인스턴스들(140)에 보통 존재할 수 있는 바와 같이, 모든 데이터 아이템들에 대한 인덱스(200)의 단일 예를 사용하는 것은 비현실적일 수 있다. 예를 들면, 단일 대형 인덱스는 인덱스를 처리하는 시스템의 메모리에 완전히 적합하지 않을 수 있고, 인덱스에 의존하는 동작들의 성능에 나쁜 영향을 미칠 수 있다. 일부의 실시예에서, 대형 인덱스는 계층화된 불균형 데이터 구조 또는 계층화된 인덱스를 이용하여 실행될 수 있다. 일반적으로 말해, 계층화된 인덱스에서, 계층에서 더 높은 인스턴스들이 다른 인덱스들(200)을 인덱스하고 계층에 더 낮은 인덱스들이 특정 엔트리들(144) 또는 다른 엔트리들(예를 들면, 블록들(425) 또는 브릭들(415))를 인덱스할 수 있는 경우, 인덱스(200)의 다중 인스턴스들은 계층적으로 정의될 수 있다.

[0208] 계층화된 인덱스의 일 실시예가 도 19에 도시되어 있다. 도시된 실시예에서, 계층화된 인덱스(220)는 5개의 인덱스들(200a~200e)을 포함한다. 인덱스(200a)는 노드들(210u~210x)을 포함하고, 노드들(210u~210x) 각각은 인덱스들(200b-200e)의 하나의 각 기본 노드를 참조하는 비-내부 노드이다. 다음에, 인덱스들(200b-200e) 각각은 도 17에 도시된 다양한 노드들(210a~200t)을 포함한다. 계층화된 인덱스들(220)의 일부의 실시예에서, 인덱스(200a)와 같은 고 레벨 인덱스들은 인덱스를 처리하는 메모리, 캐시, 또는 시스템의 다른 고 레벨의 메모리 계층에 존재하도록 구성될 수 있다. 이러한 실시예에서, 저 레벨 인덱스들은 예를 들면 페이지 방식 기술을 이용하여 필요에 따라 저 레벨로부터 고 레벨의 메모리 계층으로 다시 배치될 수 있다. 다수의 데이터 아이템들의 인덱스들의 계층 파티셔닝을 지지함으로써, 계층화된 인덱스들(220)은 시스템 자원들을 더 효과적으로 이용할 수 있다.

[0209] 예를 들면, 상기한 페이징 기술을 이용하여, 계층화된 인덱스(220) 중 종종 사용된 인덱스들(200)이 메모리 계층의 고 레벨로서 유지될 수 있고, 통상적으로 접근은 더 빠르지만 용량으로는 제한된다. 하지만, 덜 종종 사용된 인덱스들(200)은 저 메모리 계층 레벨로 저장될 수 있고, 통상적으로 접근은 더 느리지만 고 레벨 보다 더 큰 저장 용량을 갖는다. 일부의 실시예에서, 노드들(210)은 계층화된 인덱스(220) 내의 인덱스들(200)에 부가됨에 따라, 개별 인덱스들(200)은 (인덱스들을 실행하는 시스템에서 페이징하는 디스크 블록 또는 메모리의 크기와 같은) 타켓 크기를 초과하여 성장할 수 있다는 것이 예측된다. 이러한 실시예에서, 만일 주어진 인덱스(200)가 타켓 크기를 초과하도록 성장하면, 이것은 2개 이상의 인덱스 예들로 분할될 수 있다. 이러한 분할을 수행하는 과정에서, 노드들(210)은 새로운 인덱스 인스턴스들을 어카운팅하는데 필요한 것과 같이 고 레벨 인덱스(200)에 부가될 수 있다.

[0210] 키맵 엔트리 풋 또는 획득 동작에 응답하여, 계층화되거나 비-계층화된 균형 인덱스들이, 지정된 키가 인덱스(200) 내의 노드(210)에 대응하는 지를 결정하도록 순회될 수 있다. 불균형 인덱스 순회 방법의 일 실시예가 도 20에 도시되어 있다. 도시된 실시예에서, 인덱스 내에서 (또한 검색 값으로 불리는) 검색될 키 값이 예를 들면, 적절한 키맵 동작을 통하여 지정되는 경우, 동작이 블록 2000에서 시작한다. 이어서, 인덱스의 기본 노드(210)(예를 들면, 부모 노드가 없는 노드(210))가 선택된다(블록 2002).

[0211] 선택된 노드(210)에서, 노드의 대응 태그 값(212)이 검색 값에 대하여 비교되어, 태그 값이 검색 값에 정확히 매칭하는 지, 검색 값의 접두부인 지, 또는 둘 다 아닌 지를 결정한다(블록 2004). 만일 선택된 노드(210)의 태그 값(212)이 검색 값에 매칭하면, 선택된 노드(210)가, 이것이 내부 또는 비-내부 노드인 지를 결정하도록 검사된다(블록들 2006~2008). 예를 들면, 포인터들(218) 또는 선택된 노드(210)의 다른 내용이 검사되어, 노드가 엔트리(144) 또는 인덱스(200)의 다른 예와 같은 인덱스(200)에 의해 인덱스된 데이터 값을 참조하는 지를 결정하도록 할 수 있다. 만일 선택된 노드(210)가 내부 노드이면, 인덱스 미스(miss)가 상기한 바와 같이 발생할 수 있다(블록 2022).

[0212] 만일 선택된 노드(210)가 비-내부 노드이면, 선택된 노드(210)에 의해 참조된 데이터 값이 검색된다(블록 2010). 계층화된 불균형 데이터 구조들을 지지하는 실시예들에서, 일부 데이터 구조 예들은 다른 데이터 구조 예들을 인덱스할 수 있는 경우, 검색된 데이터 값은 엔트리(144) 또는 인덱스(200)의 다른 예의 기본 노드에 대응할 수 있다. 만일 검색된 데이터 값이 엔트리(144)이면, 인덱스 순회는 완료될 수 있고 검색된 엔트리(144)는 순회를 시작한 키맵 동작에 따라 처리될 수 있다(블록들 2012~2014). 예를 들면, 만일 시작 키맵 동작이 획득 동작이면, 검색된 엔트리(144)는 획득 동작의 결과로서 반환될 수 있다. 만일 시작 키맵 동작이 풋 동작이면, 검색된 엔트리(144)는 풋 동작에서 지정된 파라미터들에 따라 변경될 수 있다.

[0213] 만일 검색된 데이터 값이 엔트리(144)에 대응하지 않으면, 설명된 실시예에서, 이것은 다른 인덱스(200)의 기본 노드(210)에 대응할 수 있다. 따라서, 기본 노드(210)는 선택될 수 있고(블록들 2012, 2106), 새로이 선택된 인덱스(200)의 순회가 진행될 수 있는 경우 동작이 블록 2004로부터 진행할 수 있다. 그래서, 일 실시예에서, 도 20의 방법의 실행은, 검색 값에 대응하는 노드(210)의 존재 또는 부재가 명확하게 결정될 때까지 진행할 수 있다.

[0214] 블록 2006로 돌아가서, 선택된 노드(210)의 태그(212)가 검색 값에 매칭하지 않지만 검색 값의 접두부이면, 선택된 노드(210)의 후손들은 검사되어, 어느 후손이 검색 값에 대응하는 지를 결정한다(블록 2018). 검색 값에 대응하는 어느 후손이 있는 경우, 대응하는 후손 노드(210)가 선택되고(블록 2020), 동작은 블록 2004로 진행한다. 일 실시예에서, 선택된 노드(210)의 포인터(들)(218)이 검사되고 특정 포인터(218)에 관련된 추가 태그 정보가 선택된 노드(210)의 태그(212)에 관련했을 때, 검색 값의 접두부를 형성하는 지(또는 검색값에 완전히 매칭하는 지)를 또한 결정한다. 예를 들면, 도 17을 참조하면, 노드(210a)의 태그 "a1"은 "alibaba"의 검

색 값의 접두부인 것으로 결정될 수 있다. 또한, 대응하는 포인터(218)에 의해 표현될 수 있는 노드(210a)로부터 노드(210k)까지의 아크는 추가 태그 정보 "i"에 관련된다. 이 태그 정보가 노드(210a)의 태그 "a1"에 첨부되는 경우, 이는 또한 검색 값의 접두부인 값 "ali"을 형성한다. 따라서, 노드(210k)는 추가 순회용으로 선택될 수 있다.

[0215] 블록 2018로 돌아가서, 만일 선택된 노드(210)의 어떠한 후손도 검색 값에 대응하지 않으면, 검색 값은 인덱스(200) 내의 대응하는 엔트리(144)를 가지지 않고, 이는 또한 인덱스 미스라고 불릴 수 있다(블록 2022). 그 후, 인덱스 미스는 인덱스 순회를 시작한 키 동작의 방식에 따라 처리될 수 있다(블록 2024). 예를 들면, 키 맵 엔트리 획득 동작은 요청한 클라이언트에서의 미스를 나타내는 적당한 상태 표시를 반환함으로써 인덱스 미스를 처리할 수 있다. 이와 대조로, 키맵 엔트리 획득 동작은 선택된 노드(210)의 후손으로서 인덱스에 저장될 엔트리(144)에 대응하는 새로운 노드(210)를 삽입함으로써 인덱스 미스를 처리할 수 있다. 예를 들면, 새로운 노드(210)는 생성되고 이것의 다양한 필드들은 저장될 엔트리(144)용으로 적당히 설정되고, 새로운 노드(210)에서의 포인터(218)는 선택된 노드(210) 내에 저장될 수 있다. 만일 새로운 노드(210)가 인덱스(200)에 부가되거나 존재하는 노드(210)가 변경되면, 부가되거나 변경된 노드(210)의 모든 조상 노드들(210)의 카운트 필드들(214) 및 지문 필드들(216)이 변화를 반영하도록 업데이트될 수 있다.

[0216] 블록 2006으로 복귀하여, 만일 선택된 노드(210)의 태그(212)가 검색 값에 매칭되지 않고 검색 값의 접두부도 아니면, 인덱스 미스가 또한 발생하고 처리는 블록 2022로 진행할 수 있다. 일부의 실시예에서, 선택된 노드(210)가 인덱스(200)의 기본 노드인 경우, 이 경우가 발생할 수 있다. 따라서, 일 실시예에서, 이것에 응답하여 인덱스(200)에 새로운 노드(210)를 부가하면, 미스 경우는 존재하는 기본 노드(210)(이 경우에, 선택 노드(210))의 검색 값 및 태그(212) 둘 다의 공통 접두부인 태그(212)를 갖는 새로운 기본 노드(210)를 생성하는 단계를 포함한다. (일부의 예들에서, 새로운 기본 노드(210)의 공통 접두부는 공백일 수 있고, 어느 값을 위한 유효한 접두부로서 해석될 수 있다). 그 후, 새로운 기본 노드(210)는 후손으로서 선택된 노드(210)를 언급하도록 구성될 수 있다. 만일 필요하면, 추가 노드(210)는 검색 값에 대응하도록 생성되고 새로운 기본 노드(210)의 추가 후손으로서 구성될 수 있다.

[0217] 일부의 실시예에서, 만일 선택된 노드(210)의 태그(212)가 검색 값에 매칭되지 않고 검색 값의 접두부가 아니면, 계층화된 불균형 인덱스들(200)을 순회하는 중에, 인덱스 미스가 즉시 발생하지 않을 수 있다. 일 실시예에서, 만일 이 경우를 만나면, 선택된 노드(210)가 부모인 경우, 부모 노드(210)가 선택된다. 만일 부모 노드(210)가 다른 인덱스(200)를 참조하는 비-내부 노드이면, 참조된 인덱스(200)의 기본 노드(210)가 선택되고, 처리는 블록 2004로 진행할 수 있다. 그러하지 않으면, 인덱스 미스가 발생할 수 있다. (하지만, 이 경우는 다른 인덱스들(200)을 인덱스하지 않는 비-계층화된 자체-포함된 인덱스들(200)에서 발생하지 않을 수 있다는 것이 주목된다). 이 경우의 일 예로서, 검색 값이 "alice"인 도 19의 계층화된 인덱스를 고려한다. 인덱스(200a)의 순회는 태그 "ali"를 갖는 노드(210w)로 진행할 수 있다. 노드(210w)는 검색 값의 접두부를 형성하는 "ali"와 함께, 관련 태그 정보 "c"를 갖는 후손 노드(210x)에서의 포인터를 가지므로, 노드(210x)는 선택될 수 있다. 하지만, 노드(210x)의 태그는 "alicia"이고, 검색 값의 접두부에 매칭되지 않고 검색 값의 접두부가 아니다. 그래서, 순회는 노드(210w)(노드(210x)의 부모)로 반환할 수 있고, 인덱스(200c)를 참조하는 비-내부 노드이다. 따라서, 순회는 노드(210k)에서 계속하고 결국 노드(210m)에서 계속하고, 검색 값에 매칭되는 태그(212)를 갖는다.

[0218] 다양한 실시예에서, 불균형 인덱스들(200) 또는 계층화된 불균형 인덱스들(220)은 키맵 인스턴스들(140) 내의 키맵 엔트리들(144)을 인덱스하도록 사용될 수 있다. 예를 들면, 계층화된 인덱스들(220)은 파티션 인덱스(410), 블록 인덱스(420), 또는 엔트리 인덱스(430), 또는 키맵 인스턴스들(140) 내에서 실행될 수 있는 인덱싱의 어느 다른 레벨들을 실행하는데 사용될 수 있다. 상기한 바와 같이, 완화된 동기화 프로토콜이 이용되는 경우, 상이한 키맵 인스턴스들(140)은, 보통 동작 중에 분기되거나 불일치될 수 있다. 일부의 실시예에서, 키맵 인스턴스들(140)은 일관된 순서(예를 들면, 깊이우선 또는 폭우선 검색 순서)로 각 인덱스 데이터 구조들의 각 노드를 순회하는 철저한 프로토콜을 이용하여 동기화되어 인덱스 구조 또는 인덱스된 내용에서의 불일치를 확인할 수 있다. 하지만, 키들의 개수 및 인덱스 데이터 구조 내의 카운트 및/또는 누적 해시 정보의 포함보다는 차라리 키 정보에 따라 데이터의 분배와 같은, 상기한 불균형 인덱스들의 다양한 특성이 더욱 효과적으로 계산 동기화 알고리즘의 실행을 촉진할 수 있다.

[0219] 상기한 앤티-엔트로피 설정 조정 프로토콜의 다수의 가능한 버전은 키맵 인스턴스들(140)에 의해 실행된 가능한 불균형 계층화된 인덱스들을 가져 사용하는 것이 예측된다. 비록 예를 들면, 다른 경우들을 통하여 어떤 경우들을 최적화시키거나 프로토콜의 일반적인 단계를 수행하기 위한 알고리즘의 하나 또는 다른 특정 방

식 또는 등급을 이용하는 것을 선택할 때, 일반적인 프로토콜에서 예측된 변화는 상이한 실행 우선권을 나타내더라도, 이러한 프로토콜의 일 실시예에 대한 설명은 다음과 같다. 그래서, 설명된 실시예가 한정을 위한 것이 아니라 차라리 설명하기 위한 것으로 간주되는 것으로 예측된다.

[0220] 일 실시예에서, 불균형 인덱스(200) 또는 계층화된 불균형 인덱스(220)의 상기한 인스턴스들을 조정하도록 구성된 앤티-엔트로피 프로토콜은 다양한 방식의 메시지들의 예들 사이의 교환을 포함할 수 있다. 앤티-엔트로피 프로토콜의 일 실시예가 기초가 될 수 있는 바람직한 메시지 한 세트는 데이터 메시지, 요청 메시지, 해시 메시지, 필터 메시지, 및 지문 메시지를 포함한다. 이 메시지들 각각의 실시예의 일반적인 기능은 아래에 설명되고, 어떻게 메시지들이 앤티-엔트로피 프로토콜의 일 실시예를 실행하도록 사용될 수 있는지에 대한 설명이 다음에 주어진다. 다음 설명에서, 비록 이러한 키맵 인스턴스들이 상기한 특성의 일부를 포함하는 불균형 인덱스들(200) 또는 계층화된 불균형 인덱스들(220)의 적어도 하나의 인스턴스를 실행할 수 있는 것이 이해되더라도, 키맵 인스턴스들(140) 사이의 데이터 교환에 대한 참조가 이루어질 수 있다.

[0221] 데이터 메시지는 적어도 하나의 인덱스 노드(210)에 관한 데이터를 하나의 키맵 인스턴스(140)로부터 다른 키맵 인스턴스로 전송하도록 사용될 수 있다. 일 실시예에서, 데이터 메시지는 주어진 노드(210)에 관련된 태그(122) 만을 전송하도록 구성될 수 있고, 다른 실시예에서 데이터 메시지는 주어진 노드(210)에 관련된 다른 필드들을 전송할 수 있다. 일부의 실시예에서, 만일 주어진 노드(210)가 비-내부 노드이면, 데이터 메시지는 또한 주어진 노드에 관련된 데이터 아이템(예를 들면, 엔트리(144) 또는 다른 인덱스(200)의 기본 노드(210)에 관한 정보)의 모든 부분 또는 일부를 포함할 수 있다.

[0222] 해시 메시지는 주어진 노드(210)의 필드들 또는 주어진 노드(210)에 관련된데이터 아이템을 명백하게 전송하지 않으면서, 적어도 하나의 인덱스 노드(210)에 관한 정보를 하나의 키맵 인스턴스(140)로부터 다른 키맵 인스턴스로 전송하도록 사용될 수 있다. 일 실시예에서, 해시 메시지는 적당한 해시 알고리즘에 따라 계산된 주어진 노드(120)의 해시뿐만 아니라 주어진 노드(120)에 관련된 태그를 전송하도록 구성될 수 있다. 일부의 실시예에서, 주어진 노드(120)의 해시는 또한 주어진 노드(120)에 관련된 데이터 아이템(예를 들면, 키맵 엔트리(144))를 반영할 수 있지만, 주어진 노드(120)의 후손들 어느 것도 배제할 수 있다.

[0223] 요청 메시지는 적어도 하나의 노드(210)에 관련된 정보 요청을 전송하는데 사용될 수 있다. 일 실시예에서, 요청 메시지는 적어도 하나의 태그 접두부 값을 전송하도록 구성될 수 있다. 응답에서, 요청한 인스턴스는, 사실 전송된 태그 접두부 값이 접두부인 태그들(212)을 갖는 노드들(210)에 관한 정보를 수신할 것을 예측할 수 있다. 주어진 노드(210)에 있어서, 수신된 정보는 주어진 노드(210)의 대응 필드들의 컨텐츠 및/또는 데이터 아이템(예를 들면, 주어진 노드(210)에 대응하는 키맵 엔트리(144))를 포함할 수 있다. 일부의 실시예에서, 요청 메시지는, 특정 태그 접두부 값에 의해 정의된 결과 공간 내 값 또는 값들의 범위가 태그 접두부 값용으로 반환된 결과들로부터 배제되어야 하는 것을 지정하는 것과 같이, 요청된 태그 접두부 값들의 자격을 더 지지할 수 있다. 예를 들면, 요청 메시지는, 태그 접두부 값 "alex"에 매칭되는 모든 노드들(210)에 관한 정보가 접두부 "alexe" 또는 "alexj"에 매칭되는 노드들(210)을 제외하고 반환해야 하는 것을 지정할 수 있다.

[0224] 방금 설명한 메시지들은 일반적으로 개별 노드들(210)의 입상(granularity)의 레벨로 동작할 수 있다. 하지만, 만일 키맵 인스턴스들(140) 사이의 차이가 일반적으로 작으면(예를 들면, 대다수 노드(210)에 제한된), 이것은 즉시 다중 노드(210)의 상태를 빠르게 확인하기 위한 동기화 처리를 촉진할 수 있다. 일 실시예에서, 지문 및 필터 메시지들은 노드들(210)의 집합에 관한 정보를 전달하도록 구성될 수 있다. 특히, 일 실시예에서, 지문 메시지는 노드(210)의 지문 필드(216)를 노드의 태그(212)를 따라 하나의 키맵 인스턴스(140)로부터 다른 키맵 인스턴스로 전송하도록 구성될 수 있다. 상기한 바와 같이, 주어진 노드(210)의 지문 필드(216)는 주어진 노드(210)의 후손들의 기능으로서 결정되는 해시 값을 저장하도록 구성될 수 있다. 그래서, 만일 상기한 키맵 인스턴스들(140)에서 각 노드들(210)의 지문 필드들(216)이 동일하면, 각 노드들(210)의 후손들의 배치 및 내용이 동일할 가능성이 매우 높을 수 있다. 즉, 각 노드들(210)로부터 하강하는 키맵 인스턴스들(140)의 일부들이 동기화될 가능성이 매우 높을 수 있다.

[0225] 지문들의 사용은, 본질적인 수의 노드들(210)을 갖는 키맵 인스턴스들(140)의 일부들이 동기화되는 지의 여부에 관한 빠른 결정을 허용할 수 있다. 하지만, 대응하는 일부들이 일반적으로 동기화되지 않는다는 것을 나타내는 지문들은 이 일부들이 어떻게 상이한 지에 관한 상세한 설명을 더 제공할 수는 없다. 일 실시예에서, 필터 메시지는 특정 접두부 값에 대응하는 다수의 노드들(210)을 인코딩하는필터 값을 제1 키맵 인스턴스(140)로부터 제2 키맵 인스턴스(140)로 전송하도록 구성될 수 있다. 그 후, 제2 예는 수신된 필터 값을 사용하여 접

두부 값에 대응하는 자신의 노드들을 검사하고, 어떤 노드들(210)이 제2 예에는 있더라도 제1 예에는 존재하지 않는 것이 있는 지를 확인할 수 있다.

[0226] 일 실시예에서, 비록 데이터 값 한 세트를 필터 값으로 회복 가능하게 인코딩하는 어느 적당한 필터링 기술이 이용될 수 있다는 것이 예측되더라도, 필터 메시지에 의해 전송된 필터 값은 블룸(Bloom) 필터일 수 있다. 일반적으로 말해, 데이터 값 한 세트(예를 들면, 노드들(210))는 M-비트 이진수 값에 대응할 수 있다. 여기서, M은 정수이다. 어느 값이 블룸 필터로 인코딩되기 전에, 초기 값은 0일 수 있다. 즉, 필터의 모든 비트는 결정되지 않은 상태일 수 있다. 블룸 필터는 k 독립 해시 해시 함수 한 세트의 각각을 통하여 필터 내에 인코딩될 각 값을 통과시킴으로써 채워질 수 있다(populated). k 독립 해시 해시 함수 한 세트 각각은 이 값이 [0, M-1] 범위에서의 값으로 인코딩되도록 매핑한다. k 결과로 얻어지는 해시 값 각각에 있어서, 블룸 필터 내 대응하는 필터는 결정된다(예를 들면, 논리 1 값으로 설정된다). M 및 k는 블룸 필터 내에서 인코딩될 값들의 수 및 방식 및 (아래에 설명되는) 잘못된 긍정의 원하는 가능성에 따라 디자인 파라미터들로서 선택될 수 있다. 예를 들면, 8개의 해시 함수를 이용한 1,024-비트 필터에서, 각 해시 함수는 결정될 필터의 1,024 비트의 특정 하나를 지정하는 대응 10-비트 해시 값을 생성할 수 있다.

[0227] 주어진 값이 블룸 필터로 인코딩되었는 지를 검사하기 위하여, 이값은 필터를 인코딩하기 위해 사용된 동일한 k 독립 해시 함수 세트를 통과하고, 필터 값의 결과로 얻어지는 k 비트들이 검사된다. 만일 필터의 결과로 얻어지는 k 비트들이 결정되지 않으면, 검사 값은 필터에서 명확하게 인코딩되지 않는다. 즉, 검사 값이 필터에서 원래 인코딩될 수 있거나 될 수 없거나, 잘못된 긍정일 수 있다. 일부의 실시예에서, 해시 함수들은, 블룸 필터가 발생되는 각 개별 경우에 무작위로 또는 임의로 발생한(예를 들면, 현재 시스템 시간의 함수로서) 솔트(salt) 또는 시드(seed) 값에 의해 파라미터화되어, 주어진 값 한 세트가 필터로 성공적으로 인코딩되는 경우에 동일한 잘못된 긍정 값이 발생할 가능성을 감소시킬 수 있다.

[0228] 그래서, 예를 들면, 제1 키맵 인스턴스(140)는 접두부 P에 대응하는 노드 {A, B, C, D, E} 한 세트를 블룸 필터로 인코딩하고, 필터 메시지를 이용하여 필터를 제2 키맵 인스턴스(140)로 전송할 수 있다. 제2 키맵 인스턴스(140)에서, 노드 {A, B, X, Y, Z} 한 세트는 접두부 P에 대응할 수 있다. 제2 키맵 인스턴스(140)은 필터에 대한 각 노드들을 검사할 수 있고, 노드들(A, B, 및 X)가 필터로 인코딩될 수 있고, 노드들(X 및 Y)이 필터로 명확하게 인코딩될 수 있는 지를 결정할 수 있다. 그래서, 제2 키맵 인스턴스(140)는 노드들(Y 및 Z)이 제1 키맵 인스턴스(140)에 존재하지 않는 것을 정확히 결론지을 수 있고, X가 잘못된 긍정 값인 경우 노드들(A, B, 및 X)이 아마도 제1 키맵 인스턴스(140)에 존재하는 것으로 결론지을 수 있다. 결론으로, 제2 키맵 인스턴스(140)는 노드들(Y 및 Z)에 관한 정보를 제1 키맵 인스턴스(140)로 전송하도록 동작할 수 있다.

[0229] 데이터, 해시, 요청, 지문, 및 필터 메시지들은 어느 적당한 프로토콜 또는 API에 따라 실행되어 전송될 수 있고, 메시지를 디코딩하여 적당히 처리하는데 필요한 어느 추가 정보뿐만 아니라 상기한 정보를 전송하도록 구성된 다양한 종류의 필드들 또는 파라미터들 포함할 수 있다. 일 실시예에서, 메시지들은 메시지에 포함된 주어진 태그 값에 있어서, 보내는 키맵 인스턴스가 각각 가진-데이터 및 필요-데이터 파라미터로 불리는, 대응하는 데이터를 가지거나 대응하는 데이터를 필요로 하는 지를 나타내는 추가 파라미터를 포함할 수 있다. 예를 들면, 만일 키맵 인스턴스(140)가 태그 "a1" 및 소정 개수의 후손들을 갖는 지문 메시지를 노드(210)로 보내면, 인스턴스는, 인스턴스가 "a1"로 정의된 접두부 공간 내에 어느 노드들(210)을 갖는 것을 나타내는 가진-데이터 파라미터를 포함할 수 있다. 또한 만일 "a1"로 정의된 접두부 공간의 복사가 불완전한 것으로 믿어지면, 이 인스턴스는 필요-데이터 파라미터를 포함할 수 있다. 일부의 실시예에서, 비록 데이터 또는 해시 메시지가 필요-데이터 파라미터를 명백하게 지정할 수 있고, 필터 또는 요청 메시지가 가진-데이터 파라미터를 명백하게 지정할 수 있더라도, 가진-데이터(got-data) 파라미터는 데이터 및 해시 메시지에 포함될 수 있지만, 필요-데이터 파라미터는 필터 및 요청 메시지들에 포함될 수 있다. 일 실시예에서, 필터 메시지는 필요-데이터 또는 가진-데이터 파라미터를 지정하도록 요구될 수 있다.

[0230] 일 실시예에서, 2개의 키맵 인스턴스들(140)에 수행된 앤티-엔트로피 프로토콜은 2개의 인스턴스가 상호 접촉을 형성하는 경우에 시작할 수 있다. 각 인스턴스는 일부 데이터를 가지지만 다른 데이터는 가지지 않는다고 가정한다. 따라서, 각 인스턴스는 지문 메시지를 인스턴스의 기본 노드(210)의 태그(212) 및 지문(216)을 지정하는 다른 예로 보낼 수 있고, 가진-데이터 및 필요-데이터 파라미터들을 포함한다. 예를 들면, 계층화된 불균형 인덱스(220)를 이용하는 키맵 인스턴스(140)의 실시예에서, 기본 노드(210)는 부모 또는 선배 인덱스(200)를 갖지 않는 인덱스(200) 내에 부모 노드를 가지지 않는 노드(210)에 대응될 수 있다.

[0231] 지문 메시지를 처리하는 방법의 일 실시예가 도 21에 도시되어 있다. 도시된 실시예에서, 지문 메시지

가 메시지 송신기로부터 수신된 경우, 동작은 블록 2100에서 시작한다. 예를 들면, 제1 키맵 인스턴스(140)는 태그 값, 지문 및 가진-데이터 또는 필요-데이터 파라미터들의 적어도 하나를 갖는 지문 메시지를 제2 키맵 인스턴스(140)로 전송할 수 있다. 지문 데이터가 수신된 후, 메시지 수신기의 인덱스들이 순회되어, 수신된 태그 값이 대응하는 태그 필드(212)의 접두부인 노드(210)가 존재하는(또는 정확히 매칭되는) 지를 확인한다(블록 2102). 예를 들면, 키맵 인스턴스(140)의 인덱스들은 기본(root) 노드(210)로부터 시작해서 도 20의 방법 또는 이의 적당한 변형을 이용하여 순회될 수 있다.

[0232] 만일 수신된 태그 값이 어느 노드(210)의 태그 필드(212)의 접두부 또는 정확한 매치가 아니면, 지문 메시지에 의해 참조된 노드에 대응하는 노드(210)가 메시지 수신기에 존재하지 않을 수 있다. 따라서, 수신기는 요청 메시지를 원래 수신된 지문 메시지에 포함된 태그 값을 지정하는 메시지 송신기에 전송함으로써 대응할 수 있다(블록 2104). 일 실시예에서, 요청 메시지의 처리는 아래에 상세히 설명되는 바와 같이 진행될 수 있다. 일부의 실시예에서, 수신된 지문 메시지가 가진-데이터 파라미터를 나타내는 경우에만 요청 메시지가 전송될 수 있다.

[0233] 일부의 실시예에서, 앤티-엔트로피 프로토콜의 동작 중에 교환된 개별 메시지들의 완료는 주어진 메시지에 응답하여 발생한 추가 메시지들이 성공적으로 완성되었는 지의 여부에 의존한다. 즉, 일부의 실시예에서, 개별 메시지들의 처리는 다른 메시지에 대하여 상태를 유지하지 않는 비동기식 방식으로 발생할 수 있다. 여기에 설명된 바람직한 실시예의 설명에서는, 이러한 상태를 유지하지 않는 비동기식 모델이 사용될 것이다. 그래서, 요청 메시지가 발생한 후, 지문 메시지 자체의 처리는 완전한 것으로 고려될 수 있다(블록 2106). 하지만, 이 모델은 앤티-엔트로피 프로토콜의 일반적인 동작에 필수적인 것은 아니고, 대체 실시예에서, 어느 주어진 메시지는 종속적으로 또는 주어진 메시지에 응답하여 발생한 메시지들과의 동기화를 차단하거나, 기다리거나 유지할 수 있다. 예를 들면, 일부의 실시예에서 명백한 주고받기(handshaking), 확인(acknowledgement), 재시도(retry) 또는 다른 방식의 프로토콜들이 하나의 메시지의 완성 상태를 다른 종속 메시지로 전송하도록 이용될 수 있다.

[0234] 만일 수신된 태그 값이 메시지 수신기에서 특정 노드(210)의 태그(212)의 접두부 또는 매치로서 대응하면, 수신된 지문 값은 특정 노드(210)의 지문 필드(216)에 대하여 비교되어 2개의 지문이 매칭되는 지를 결정한다(블록 2108). 만일 2개의 지문이 매칭되면, 메시지 송신기 및 메시지 수신기가 수신된 태그 값에 대하여 동기화될 가능성이 (예를 들면, 다른 데이터로부터 발생됨에도 불구하고, 충돌하거나 동일한 값을 갖는 2개의 지문을 생성하는데 사용된 지문 알고리즘의 가능성에 따라) 매우 높을 수 있다. 예를 들면, 접두부로서 수신된 태그 값을 갖는 어느 노드들(210)이, 지문 메시지가 보내진 키맵 인스턴스(140) 및 메시지가 수신된 키맵 인스턴스(140) 내의 동일한 상태에 있을 가능성이 매우 높다. 그래서, 지문 메시지에 응답하여 어떠한 추가 메시지들도 발생하지 않고, 메시지는 완성되는 것으로 고려될 수 있다(블록 2106).

[0235] 만일 지문이 매칭되지 않으면, 메시지 송신기 및 메시지 수신기는 수신된 태그 값에 대하여 동기화되지 않고, 추가 작업은 송신기 및 수신기를 상태에 있어서 인접하게 결합시킬 필요가 있을 수 있다. 상기한 바와 같이, 필터 메시지는 송신기가 어떤 노드들(210)에 관한 특정 정보를 수신기로 전달할 수 있도록 하는데 유용할 수 있다. 하지만, 일부의 실시예에서, 합당한 가상-포지티브 비율을 유지하면서 필터 메시지로 인코딩될 노드들(210)의 수는 어떤 임계값으로 한정될 수 있다. 만일 후손 노드들(20)의 수가 수신된 태그 값에 매칭되는 메시지 수신기 노드(210)에서 이 임계값을 초과하면, 필터 메시지를 보내기 전에 추가 지문 메시지 처리를 수행하는 것이 더욱 효과적일 수 있다.

[0236] 그래서, 설명된 실시예에서, 만일 지문이 매칭되지 않으면, 메시지 수신기에서 특정 노드(210)의 카운트 필드가 검사되어, 필터 메시지 처리용 임계값을 초과하는 지를 결정할 수 있다(블록 2110). 만일 카운트 필드가 임계값을 초과하면, 메시지 수신기는 수신된 태그 값이 접두부인 특정 노드(210)의 자식들에 따라 수신된 태그 값에 대응하는 인덱스 범위의 일부를 세분하도록 구성될 수 있다(블록 2112). 각 자식 노드에 있어서, 메시지 수신기는 대응하는 지문 메시지를 원래 매시지 송신기로 다시 보내어, 각 자식 노드(210)의 태그(212) 및 지문 필드(216)를 지정하도록 구성될 수 있다(블록 2114). 또한, 만일 수신된 태그 값에 대응하는 인덱스 범위의 일부에 갭들이 있으면, 예를 들면 특정 노드(210)의 자식들에 의해 표시된 바와 같이, 메시지 수신기는 갭들에 대응하는 태그 값들용 적어도 하나의 요청 메시지를 보내도록 구성될 수 있다(블록 2116). 그 후, 수신된 지문 메시지의 처리가 완료되는 것으로 고려된다(블록 2118). 일 실시예에서, 상기한 동작 외에, 만일 수신된 태그 접두부 값이 특정 노드(210)의 적당한 매치이면, 특정 노드(210)에 대응하는 해시 메시지는 메시지 송신기로 반환될 수 있다.

[0237] 예를 들면, 도 17에 도시된 바와 같이, 메시지 수신기의 인덱스(200)의 특정 노드(210a)는 태그 "al" 및 대응하는 태그들 "alan", "alex", "alfred", "ali", 및 "alz"를 갖는 자식들을 가질 수 있다. 이것은 메시지 수신기가 "alan" 및 "alex"로 시작하는 노드들(210)에 관한 정보를 포함하고 "alb", "alc" 또는 "ald"로 시작하는 노드들(210)에 관한 정보는 포함하지 않는 것을 제안한다. 따라서, 메시지 수신기는 자식들의 태그들 사이의 갭들용 요청 메시지뿐만 아니라 노드(210a)의 자식 각각용 지문 메시지를 전송할 수 있다. 네가티브 요청 메시지가 지지되는 실시예들에서, 메시지 수신기는 특정 노드의 자식들에 대응하는 것과는 다른 태그들용 요청 메시지들을 전송할 수 있다. 메시지 수신기는 "alan", "alex", "alfred", "ali", 및 "alz"로 접두사를 붙인 것과는 다른 태그들용 요청 메시지들을 전송할 수 있다.

[0238] 특정 노드(210)의 카운트 값이 필터 메시지 처리용 임계값을 초과하지 않은 경우, 수신된 지문 메시지가 가진-데이터 파라미터를 포함하면, 메시지 송신기는 메시지 수신기에 존재하지 않는 노드들(210)에 관한 특정 정보를 포함할 수 있다. 따라서, 메시지 수신기는 특정 노드(210)의 후손인 각 노드(210)를 필터(예를 들면, 상기한 바와 같은 블룸 필터)로 인코딩하는 필터 메시지를 보내도록 구성될 수 있다(블록들 2120~2122). 예를 들면, 도 17을 참조하면, 만일 특정 노드가 노드(2101)에 대응하면, 노드들(210m~210q) 각각을 인코딩하는 블룸 필터가 필터 메시지를 통하여 발생하여 반환된다. 도시된 실시예에서, 만일 가진-데이터 파라미터가 원래 지문 메시지에 포함되지 않으면, 각 지문 메시지들은 필터 메시지 대신에 발생하여 특정 노드(210)의 각 자식용 메시지 송신기로 반환될 수 있다(블록 2124). 이 경우 필터 메시지 또는 지문 메시지의 발생 후에, 수신된 지문 메시지의 처리가 완료될 수 있다(블록 2118).

[0239] 필터 메시지를 처리하는 방법의 일 실시예가 도 22에 도시되어 있다. 도시된 실시예에서, 태그 값 및 필터 값을 갖는 필터 메시지가 예를 들면 상기한 바와 같은 지문 메시지에 응답하여 메시지 송신기로부터 수신되는 경우, 동작을 블록 2200에서 시작한다. 일단 필터 메시지가 수신되면, 메시기 수신기의 인덱스(들)이 순회되어 도 21에 대하여 상기한 바와 동일한 방법으로, 수신된 태그 값(예를 들면, 수신된 태그 값이 접두부 또는 매치)에 대응하는 특정 노드를 확인한다(블록 2202). 일부의 실시예에서, 만일 다른 메시지에 응답하여 필터 메시지가 발생하면, 수신된 태그 값에 대응하는 노드(210)가 일반적으로 존재할 것이다.

[0240] 그 후, 메시지 수신기는, 만일 있다면 인코딩되지 않는 노드들(210)을 확인하기 위하여 필터 메시지에 제공된 필터 값에 대한 특정 노드(210)의 각 후손을 검사할 수 있다(블록 2204). 필터 값으로 인코딩되지 않는 메시지 수신기에서의 각 노드(210)에 있어서, 대응하는 데이터 메시지는 메시지 송신기로 반환될 수 있다(블록 2206). 그 후, 필터 메시지의 처리는 완료된 것으로 고려될 수 있다(블록 2208). 상기한 바와 같이, 필터 메시지용으로 이용된 필터 알고리즘의 방식 및 구성에 따라, 잘못된 긍정이 발생할 수 있다. 즉, 노드들(210)의 일부가 필터 값으로 인코딩되지 않고, 그래서 메시지 송신기에서의 동일한 상태에 존재하지 않더라고, 메시지 수신기는 노드들(210)의 일부가 필터 값으로 인코딩되고, 그래서 메시지 송신기에서의 동일한 상태에 존재하는 것으로 잘못 결론 지을 수 있다. 그래서, 앤티-엔트로피 프로토콜의 단일 라운드는 매 노드(120)에 대하여 동기화되는 2개의 키맵 인스턴스(140)로 끝날 수 없다. 하지만, 많은 실시예에서, 앤티-엔트로피 프로토콜의 단일 라운드는 예들이 더욱 분기되지 않도록 할 수 있고, 프로토콜의 반복된 어플리케이션들은, 예들이 다른 정도 및 사용된 필터 알고리즘의 특징(예를 들면, 필터 인코딩용 임계값으로 주어진 잘못된 긍정의 가능성)에 의존하여, 어느 정도의 가능성을 갖는 어떤 수의 라운드 내에서 분기하도록 기대될 수 있다.

[0241] 일부의 실시예에서, 해시, 요청, 및 데이터 메시지의 처리는 필터 및 지문 필터에 비해 상당히 단순화할 수 있다. 일 실시예에서, 해시 메시지 수신기는 메시지에 포함된 태그 값에 대응하는 노드(210)를 확인하는 것을 시도할 수 있고, 그 후 확인된 노드(210)의 대응하는 해시 값을 계산할 수 있다. 만일 수신된 해시 값이 계산된 해시 값에 매칭되면, 확인된 노드(210)는 메시지 송신기에서 대응하는 노드(210)에 이미 동기화될 수 있다. 그렇지 않으면, 수신된 태그 값을 갖는 요청 메시지는 메시지 송신기로 반환되어 가장 최근 데이터 버전을 얻는다.

[0242] 일 실시예에서, 요청 메시지의 처리는 메시지에 포함된 수신된 태그 값이 대응하는 태그 필드(212)의 접두부에 매칭하거나, 접두부인 각 노드(210)를 예를 들면, 상기한 불균형 인덱스 운행 기술을 이용하여 확인하는 메시지 수신기를 포함한다. 각 확인된 노드(210)에 있어서, 상기한 바와 같이 구성된 대응하는 데이터 메시지는 메시지 송신기로 반환된다. 일 실시예에서, 수신된 데이터 메시지의 처리는 메시지에 표시된 태그 값에 대응하는 노드(210)가 메시지 수신기에 존재하는 지를 확인하는 단계를 포함할 수 있다. 만일 대응하는 노드(210)가 메시지 수신기에 존재하지 않으면, 대응하는 노드(210)는 메시지로부터 추출된 데이터에 의해 생성되어 유지될 수 있다. 만일 대응하는 노드(210)가 메시지 수신기에 존재하면, 존재하는 노드(210)에 관련된 데이터 및/또는 대응하는 데이터는 메시지로부터 추출된 데이터로 대체될 수 있다. 일부의 실시예에서, 수신된 데이터가

더 많이 유통되고 있으면, 존재하는 노드(210)의 데이터 만이 대체될 수 있다. 예를 들면, 데이터 메시지는 메시지 송신기에서 노드(210)에 대응하는 엔트리(144)의 컨텐츠를 포함할 수 있고, 엔트리(144)는, 수신된 엔트리(144)가 존재하는 엔트리(144)보다 더 많이 유통되는 지를 확인하기 위해 메시지 수신기에 대응하는 시간 스탬프 정보와 비교될 수 있는 시간 스탬프 정보를 포함할 수 있다. 만일 수신된 엔트리(144)가 존재하는 엔트리(144)보다 더 많이 유통되고 있으면, 수신된 엔트리(144)는 존재하는 엔트리(144)에 대체될 수 있다.

[0243] 도 21의 일반적인 동기화 프로토콜의 변화가 가능하며 예측될 수 있다. 예를 들면, 키맵 인스턴스들 사이의 통신이 고정된 길이를 갖는 패킷들을 이용하여 수행되는 실시예들에서, 대역폭 이용은 특정 노드(210)에 대응하는 단일 지문 메시지보다 차라리 단일 패킷 내에 다중 노드들(210)용 다중 지문 메시지들을 전송함으로써 향상될 수 있다. 그 후, 이러한 패킷을 수신한 예는 인덱스(들)(200) 중 어느 특정한 것이 송신기와 추가적인 필수적인 교환 없이 송신기에 미스 매칭하는 지를 빠르게 식별할 수 있게 된다. 예를 들면, 만일 제1 지문 메시지가 매칭되지 않으면, 수신기는 요청, 필터, 또는 다른 메시지를 패킷 송신기에 발행하기 전에 패킷 내 다른 지문 메시지들을 고려할 수 있다. 이렇게 할 때, 수신기는 데이터 구조의 특정 부분에서의 불일치를 감소시킬 수 있고, 이미 동기화된 데이터 구조의 다른 부분들에 관한 메시지를 교환하기 위한 불필요한 네트워크 소통을 감소시킬 수 있다.

[0244] 일반적으로, 앤티-엔트로피 프로토콜 및/또는 업데이트 전파 프로토콜을 이용한 키맵 인스턴스 조정을 수행하기 위한 상기한 일부의 방법들 또는 기술들이 인스턴스들 내의 키맵 인스턴스들(140) 또는 개별 호스트들(400)의 레벨에서 동작하도록 구성된 키맵 조정자 처리에 의해 실행될 수 있다는 것이 예측된다. 불균형 데이터 구조들에 대하여 앤티-엔트로피 프로토콜들을 실행하기 위한 상기한 방법들 및 기술들의 다수의 변화가 가능하고 예측되는 것이 주목되고, 이러한 검토는 제한하기보다는 실시예로서 설명된 것이다. 예를 들면, 네트워크를 통하여 정보를 분배하기 위하여, 일부 엔티티가 랜덤하게 선택된 다른 엔티티와 종종 통신하도록 하는 일반적인 등급의 프로토콜들은 가십(gossip)-기반 프로토콜이라고 불릴 수 있고, 가십-기반 프로토콜의 다른 기술들 또는 양상들이 키맵 인스턴스들(140) 사이의 앤티-엔트로피 프로토콜용으로 사용될 수 있다. 다양한 실시예에서, 상기한 동기화 메시지들(또는 다른 적당한 메시지)의 예들이 상기한 특징을 갖는 동기화 프로토콜들을 산출하기 위해 상기한 방법으로 결합될 수 있다.

[0245] 또한, 도 17 내지 도 22에 대하여 상기한 계층화된 인덱스 데이터 구조 및 동기화 기술들이 키맵 인스턴스들(140) 내에 사용된 효과적인 데이터 구조들을 실행하는 문맥에서 설명되어 있지만, 이러한 데이터 구조들 및 동기화 기술들은 대량의 데이터가 급속 접근용으로 인덱스될 수 있는 어느 어플리케이션에 이용될 수 있다는 것이 예측된다. 이러한 어플리케이션들은 도 2의 시스템과 같은 객체 저장 시스템을 필수적으로 포함할 필요는 없지만, 데이터 인덱싱이 적용될 수 있는 데이터 시스템들, 검색 시스템들, 또는 어느 다른 어플리케이션들을 포함할 수 있다.

[0246] 다양한 실시예에서, 여기에 설명된 이벤트들의 어느 방식의 랜덤 발생 또는 선택에 대한 실행이 난수 또는 이벤트 발생용의 어느 적당한 알고리즘 또는 기술을 이용할 수 있다. 많은 경우에, 랜덤한 방법들을 실행하기 위한 계산 기술들은 순수한 랜덤 결과를 생성하지 않고 의사 랜덤 결과들을 생성한다. 예를 들면, 의사 랜덤 알고리즘들은 통계적 랜덤 결과를 발생하기 위해 구성된 결정적 과정을 지정할 수 있다. 여기서 사용된, "랜덤" 또는 "본질적 랜덤"의 발생은 순수한 랜덤 데이터 소스뿐만 아니라 어느 적당한 의사 랜덤 계산 기술들을 포함하는 것으로 의도된다.

[0247] 저장 서비스 구성요소 검출 및 관리

[0248] 저장 서비스 시스템의 큰-스케일, 고 분배 실행에 있어서, 다양한 다수의 시스템 구성 요소들이 시스템을 통하여 도 2에 분배 도시되어 있다. 예를 들면, 비트 저장 노드들(160)의 인스턴스, 조정자들(120), 및 키맵 인스턴스들(140)이 수백 또는 수천 개 사용될 수 있다. 이러한 스케일의 분배된 시스템의 상태를 관리하는 것은 실제적인 도전을 제공한다. 예를 들면, 특정 시스템 구성 요소들의 상이한 예들은 계획된 유지 보수, 구성 요소들에 의해 의존된 컴퓨팅 자원들의 고장, 부가-기능 구성 요소들을 분리하는 통신 고장, 또는 다른 이유로 인하여 주어진 시간에 사용되지 않을 수 있다. 또한, 구성 요소들의 새로운 또는 사용되지 않는 예들이 임의 또는 예측할 수 없는 시간의 어느 경우에, 서비스로 복귀할 수 있다.

[0249] 일 실시예에서, 발견, 고장, 및 검출 데몬(DFDD)(110)의 예들이 저장 서비스 시스템의 다양한 관련 구성 요소들의 상태를 각각 모니터링하고, 이러한 상태에 관하여 상호 통신하고, 이러한 클라이언트들이 키맵 또는 비트 저장 동작들과 같은 시스템 동작을 수행하기 위해 사용될 수 있는 유용한 시스템 구성 요소들을 식별할 수 있도록 하는 인터페이스를 DFDD 클라이언트 어플리케이션들에게 제공하도록 구성될 수 있다. 일반적으로,

DFDD(110)는 다른 구성 요소들 대신에 저장 서비스 시스템 구성 요소들의 현재 상태의 균일하게 접근할 수 있는 뷰우를 제공하도록 구성될 수 있다. 즉, 닮지않은 상태 정보의 다른 구성 요소들과 직접 통신용으로 구성된 다중의 다른 인터페이스들로 저장 서비스 시스템의 다양한 구성 요소들을 구성하는 것보다 차라리, 이러한 정보를 제공하거나 이러한 정보에 의존하는 각 구성 소가 표준 DFDD 인터페이스를 통하여 DFDD(110)의 예와 통신하도록 구성될 수 있다. 일부의 실시예에서, DFDD(110)는 운영 시스템에 의해 관리된 환경 내에서 동작하도록 구성된 데몬 프로세스로서 실행될 수 있다. 하지만, 다른 실시예에서, DFDD(110)는 운영 시스템 또는 다른 구성 요소들에 어느 필요한 의존 또는 추종없이, 여기에 설명된 기능을 실행하도록 구성된 독립 또는 자발 하드웨어 또는 소프트웨어로서 실행될 수 있다.

[0250] 일반적으로 말해, DFDD(110)의 인스턴스에 의해 발견되어 모니터링되도록 구성되는 저장 서비스 시스템 구성 요소의 각 인스턴스는 어플리케이션 인스턴스로 불릴 수 있다. 예를 들면, 주어진 비트 저장 노드(160)의 동작 상태는 주어진 비트 저장 노드(160)에 의한 실행용으로 구성된 SNM 제어기(161)의 인스턴스에 의해 표시될 수 있다. 그래서, SNM 제어기(161)는 비트 저장 어플리케이션 인스턴스에 대응할 수 있다. 마찬가지로, 키맵 인스턴스(140)의 동작 상태는 키맵 인스턴스 내 적어도 하나의 호스트(400)에서의 실행용으로 구성된 키맵 매니저의 인스턴스들에 의해 표시될 수 있다. 각 키맵 매니저 인스턴스는 키맵 어플리케이션 인스턴스에 대응할 수 있다. 다른 방식의 어플리케이션 인스턴스들이 가능하고 예측된다. 예를 들면, 일 실시예에서, 적어도 하나의 저장 서비스 시스템 구성 요소들이 배치되는 각 컴퓨터 시스템은 프로세서, 메모리, 디스크, 입력/출력(I/O) 단자, 또는 다른 시스템 자원들의 이용과 같은 시스템-특정 동작 상태 항목들을 검출하여 보고하도록 구성된 호스트 모니터 어플리케이션 인스턴스를 포함할 수 있다. 일부의 실시예에서, DFDD(110)의 각 예는 어플리케이션 인스턴스로서 자체적으로 구성될 있다. 즉, DFDD 인스턴스들은 다른 어플리케이션 인스턴스들의 상태뿐만 아니라 자신의 동작 상태를 모니터링하도록 구성될 수 있다.

[0251] 저장 서비스 시스템 내에서, 어플리케이션 인스턴스들은 어플리케이션 이름들에 의해 일반적으로 확인되고 각 어플리케이션 인스턴스 식별자들(ID들)에 의해 유일하게 확인될 수 있다. 예를 들면, 특정 어플리케이션 이름은 "키맵-매니저", "비트 저장-매니저", "호스트-매니저", 또는 적당한 이름과 같은 일반적인 방식의 어플리케이션 인스턴스를 확인하는 스트링을 포함할 수 있고, 어플리케이션 인스턴스 ID는 어플리케이션 이름 내의 특정 예를 유일하게 확인하는 스트링을 포함할 수 있다. 일부의 실시예에서, 어플리케이션 인스턴스 ID는 "키맵-매니저-4AB8D945"와 같은 어플리케이션 이름을 명백하게 포함할 수 있다. 어플리케이션 인스턴스 ID로서 다른 적당한 포맷들도 사용될 수 있다. 일 실시예에서, DFDD(110)의 주어진 예는 다수의 어플리케이션 인스턴스들(예를 들면, 이름들 및 인스턴스 ID들)을 각 상태 정보에 관련시키도록 구성될 있다. 예를 들면, 도 23에 도시된 실시예에서, DFDD(110)은 다수의 엔트리(144)를 포함하고, 엔트리들(144) 각각은 어플리케이션 이름(112) 및 인스턴스 ID(113)를 인스턴스 상태 정보(114)에 관련시킬 수 있다. 일부의 실시예에서, DFDD(110)는 상태 정보의 상이한 종류들과 주어진 어플리케이션 이름(112) 및 인스턴스 ID(113)의 관련을 반영하기 위한 적어도 하나의 표를 사용할 수 있다. 다른 실시예에서, DFDD(110)는 트리들, 상기한 바와 같은 불균형 인덱스들, 또는 주어진 어플리케이션 인스턴스와 대응하는 상태 정보 사이의 관련을 나타내는 어느 다른 적당한 방식의 데이터 구조들을 이용할 수 있다.

[0252] 일부의 실시예에서, 어플리케이션 인스턴스 ID들은 입상의 임의 레벨들의 자신의 이름 공간들을 포함할 수 있다. 예를 들면, 일 실시예에서, 주어진 키맵 어플리케이션 인스턴스 ID는 <맵 이름>/<예>/<종료점> 형태일 수 있다. 용어 <맵이름>은 키-엔트리 연관 관계(associations)의 특정 키맵 디렉터리를 식별하도록 하고, 일반적으로 주어진 키맵 배치에 대응할 수 있다. (상기한 키맵 배치들용 키맵 어플리케이션 인스턴스들이 DFDD(110)의 동일한 인스턴스 내에서 관리되도록 할 수 있다). 용어 <인스턴스>는 예를 들면 유일한 스트링에 의해 키맵 인스턴스(140) 내 특정 호스트(400)를 확인할 수 있도록 한다. 용어 <종료점>은 확인된 호스트(400)(예를 들면, 개별 처리들로서)에서 동작하는 기능적으로 동일한 다수의 독립형 키맵 어플리케이션 인스턴스들 중의 하나를 확인할 수 있다. 어플리케이션 인스턴스 ID들 내 다른 복합 이름 공간들이 가능하고 예측된다.

[0253] DFDD(110)에 의한 어플리케이션 인스턴스에 관련된 상태 정보(114)는 다른 종류의 다양한 정보를 포함할 수 있다. 일 실시예에서, DFDD(110)는 DFDD(110)에 의해 관리된 모든 방식의 어플리케이션 인스턴스들에 공통일 수 있는 글로벌 동작 상태 정보를 상태 정보(114) 내에 저장하도록 구성될 수 있다. 예를 들면, 아래에 자세하게 설명되는 바와 같이, 일부의 실시예에서, DFDD(110)는 상태 한 세트 중에서 가능한 천이(transition)들뿐만 아니라 어플리케이션 인스턴스들의 글로벌 동작 상태(또는 단순히, 글로벌 상태) 한 세트를 정의하는 글로벌 동작 상태 머신을 실행할 수 있다. 이러한 실시예에서, DFDD(112)에 의해 관리된 각 어플리케이션 인스턴스는 주어진 시간에 글로벌 상태 한 세트의 특정 하나에 관련될 수 있고, 주어진 어플리케이션 인스턴스용 글로벌

상태는 어플리케이션 인스턴스의 상태 머신 및 동작에 의해 시간이 지남에 따라 변할 수 있다.

[0254] 일부의 실시예에서, 널리 다른 방식의 어플리케이션 인스턴스들에 공통일 수 있는 글로벌 상태 정보에 더하여, 상태 정보(114)는 특정 어플리케이션 인스턴스 또는 예의 방식에 특이하거나 맞추어질 수 있는 동작 상태 정보를 반영할 수 있다. 예를 들면, 만일 어플리케이션 인스턴스가 특정 비트 저장 노드(160)의 비트 저장 매니저에 대응하면, 어플리케이션 인스턴스의 상태 정보(114)는 특정 노드에서 유용한 저장 자원들의 양에 관한 정보, 이 자원들의 방식(예를 들면, 고 성능, 저 성능 등) 또는 비트 저장 노드의 문맥에 특이할 수 있는 어느 다른 적절한 상태 정보를 포함할 수 있다. 마찬가지로, 특정 키맵 인스턴스(140)의 키맵 매니저에 대응하는 어플리케이션 인스턴스에서, 어플리케이션의 상태 정보는 특정 키맵 인스턴스에 의해 관리된 엔트리들(144)의 수에 관한 정보, 사용된 또는 유용한 키맵 저장 자원들, 또는 다른 적절한 키맵 상태 정보를 포함할 수 있다. 일부의 실시예에서, 대응하는 DFDD 엔트리(111)내에 어떤 어플리케이션 인스턴스-특정 상태 정보를 포함할 것인지의 선택은 DFDD 클라이언트들에 따라 결정될 수 있다. 예를 들면, 수개의 선택들로부터 특정 비트 저장 또는 키맵 어플리케이션을 선택할 때, 조정자(120) 또는 노드 피커(130)를 지원하는데 사용될 수 있는 상태 정보는 어플리케이션 인스턴스들의 DFDD 엔트리들(111) 내에 포함될 수 있다.

[0255] 일부의 실시예에서, 어플리케이션 인스턴스의 상태 정보(114)는 또한 DFDD 클라이언트가 어떻게 이 예에 접근할 수 있는 지에 관한 정보를 포함할 수 있다. 예를 들면, 상태 정보(114)는 DFDD 클라이언트가 어플리케이션 인스턴스와의 연결을 수립하는 인터넷 프로토콜(IP) 어드레스 및 포트 번호를 포함할 수 있다. 일부의 어플리케이션 인스턴스들은 웹 서비스 인터페이스들, 공개/가입 인터페이스들, 또는 다른 적당한 인터페이스들과 같은 다른 방식의 인터페이스들을 지지할 수 있다. 이러한 실시예에서, 상태 정보(114)는 DFDD 클라이언트가 웹 서비스 호출을 수행하고, 공개/가입 채널에 가입하고, 어플리케이션 인스턴스와의 통신을 수립하는데 필요한 다른 방식의 작용을 수행하는데 필요한 URL 또는 다른 정보를 포함할 수 있다. 일부의 실시예에서, 어플리케이션 인스턴스 접근 정보에 더하여 또는 대신에, 상태 정보(114)는 이 인스턴스가 저장 서비스 시스템 내에 물리적으로 위치하는 지에 관한 정보를 포함할 수 있다. 예를 들면, 상태 정보(114)는 특정 어플리케이션 인스턴스가 대응하는 데이터 중심(300) 또는 영역(310)의 식별자를 포함할 수 있다.

[0256] 상기한 바와 같이, 일부의 실시예에서 DFDD(110)는, 주어진 어플리케이션 인스턴스가 정상적으로 동작하고 그래서 정상 상태에서 유용하게 사용되거나, 정상 상태에 있는 지를 일반적인 방법으로 표시할 수 있는 개별 어플리케이션 인스턴스들용 글로벌 상태 정보를 유지할 수 있다. 일 실시예에서, DFDD(110)의 예에 의해 모니터링하기 위해 구성된 각 어플리케이션 인스턴스는 소정 간격의 초 또는 분과 같은 일정한 간격으로 종종(하지만 필수적이 아닌) 그 상태를 DFDD(110)에 보고하도록 구성될 수 있다. 이러한 보고는 또한 "핵심(heartbeat)"라고 불릴 수 있다. 핵심 보고들은 어느 적당한 프로토콜에 따라(예를 들면, TCP/IP 메시지들로서, 웹 서비스 호출들로서, 또는 다른 표준 또는 소유 메시징 프로토콜들에 따라) 통신될 수 있고, 정보 내용에서 변할 수 있다. 최소 예로서, 주어진 어플리케이션 인스턴스는 핵심을 단순히 주어진 예에 대응하는 어플리케이션 이름 및 어플리케이션 인스턴스 ID를 포함하는 DFDD(110)에 제출할 수 있다. 다른 경우에, 주어진 어플리케이션 인스턴스는 국부 자원 사용의 특정 상태와 같은 핵심에 추가 상태 정보를 포함할 수 있다. 일부의 실시예에서, 어플리케이션 인스턴스들은 핵심을 보내기 전에 자신의 기능 상태를 확인하기 위해 자체-진단 또는 자체-인증의 약간의 레벨을 수행하도록 구성되고, 다른 실시예에서 어플리케이션 인스턴스는 어떤 자체-평가에 의존하지 않고 핵심을 보낼 수 있다.

[0257] 일반적으로 말해, 만일 어플리케이션 인스턴스가 예상된 바와 같이 핵심들을 DFDD(110)로 보내면, DFDD가 정상적으로 동작하는 것으로 적당히 예상된다. 만일 핵심들이 일부 시간 동안 방해되어야 하면, 어플리케이션 인스턴스에 어떤 문제가 있는 것으로 적당히 예상된다. 도 24는 핵심 작용 및/또는 다른 파라미터들의 함수로서 각 어플리케이션 인스턴스에 대하여 DFDD(110)에 의하여 유지될 수 있는 글로벌 상태 머신의 일 실시예를 나타낸다. 도시된 실시예에서, 동작을 시작하고 새로운 어플리케이션 인스턴스의 존재의 DFDD(110)의 예를 통지하고, 대응하는 엔트리(111)를 초기화하기 위하여 DFDD(110)용으로 필요한 어플리케이션 이름, 어플리케이션 인스턴스 ID, 및 어느 다른 정보를 제공한 후, 새로운 어플리케이션 인스턴스는 예를 들면 짧게 새로운(NEW) 상태에서 온라인 상에 나타난다. 일단 새로운 어플리케이션 인스턴스가 안정되고 정상 동작을 시작할 준비가 되면, OK 상태로 진입한다. 다양한 실시예에서, 새로운 상태에서 OK 상태로의 천이는 시간(예를 들면, 어플리케이션 인스턴스의 방식에 기초한 디폴트 고정 시간), 어플리케이션 인스턴스 자체-보고, 관리자 중재, 또는 이들 또는 다른 인자들의 결합의 함수일 수 있다.

[0258] 설명된 실시예에서, DFDD에 대한 예의 최종 핵심이 고장 임계값 T_{fail} 미만이므로, 어플리케이션 인스턴

스는 경과한 시간 동안은 OK 상태로 남을 수 있다. 예를 들면, DFDD(110)은 대응하는 예로부터 수신된 각 핵심에 따라 증가하는 각 어플리케이션 인스턴스의 카운트를 유지할 수 있고, 카운트 값이 T_{fail}을 경과하기 전에 변하였는 지를 확인하기 위하여 각 카운트(예를 들면, 카운트다운 타이머들)를 모니터링할 수 있다. 일부의 실시예에서, 아래에 설명되는 바와 같이 이러한 상태들 사이에 구별이 있을 수 있더라도, OK (및 아마도 새로운) 상태와는 다른 글로벌 상태들이 일반적으로 정상 동작 상태들 또는 고장 상태들로 불릴 수 있다.

[0259] 어플리케이션 인스턴스용 최종 핵심 이래로 시간 T_{fail}이 경과되면, 어플리케이션 인스턴스의 글로벌 상태는 INCOMMUNICADO로 천이한다. 설명된 실시예에서, INCOMMUNICADO는 어플리케이션 인스턴스에 이상이 있다는 것을 나타내는 순간 상태로서 기능을 할 수 있지만, 이것은 영원히 고장난 것으로 명확하게 결정되지는 않는다. 예를 들면, 어플리케이션 인스턴스는 일시적으로 정지하거나 정체하거거나, DFDD(110)로의 핵심 메시지가 지연되거나 분실되거나, 아래에 상세히 설명되는 바와 같이 DFDD(110)의 일 예가 어플리케이션 인스턴스의 현재 상태에 대한 DFDD(110)의 다른 예와의 동기화에서 벗어날 수 있다. 만일 핵심이 INCOMMUNICADO 상태에 있는 어플리케이션 인스턴스로부터 수신되면, 어플리케이션 인스턴스는 다시 OK 상태로 천이할 수 있다. 일부의 실시예에서, DFDD 클라이언트들은 자체의 위험을 무릅쓰고 INCOMMUNICADO 상태에 있는 어플리케이션 인스턴스를 사용하도록 선택할 수 있다.

[0260] 만일 어플리케이션 인스턴스가 INCOMMUNICADO 상태로부터 자발적으로 회복될 수 없으면, 이 예에 영향을 미치는 더욱 심각한 문제가 생길 수 있다. 설명된 실시예에서, 2개의 가능한 고장 시나리오가 발생할 수 있다. 고장(FAIL) 상태로 표시된 바와 같이, 개별 어플리케이션 인스턴스는 예를 들면 개별 예를 호스팅하는 기초 컴퓨터 자원들의 고장으로 인하여 분리를 할 수 없다. 대안으로, 어플리케이션 인스턴스는 네트워크 분할(SPLIT) 상태로 표시된 바와 같이, 예와 DFDD(110) 사이의 네트워크 통신의 손실로 인하여 고장날 수 있다. 예를 들면, 어플리케이션 인스턴스는 DFDD(110)의 일부 예에서 동작 가능하고 접근할 수 있지만, 저장 서비스 시스템의 일부들을 상호 분리하는 통신 고장으로 인하여 DFDD(110)의 일부 예에서 동작 불가능하고 접근할 수 없다.

[0261] 주어진 어플리케이션 인스턴스 고장이 분리되었는 지 또는 네트워크 분할 때문인 지를 확실히 결정하기는 어려울 수 있다. 일부의 실시예에서, DFDD(110)는, 어플리케이션 인스턴스가 INCOMMUNICADO 상태로부터 고장 상태 또는 네트워크 고장 상태로 전이해야하는 지에 대하여 결정하기 위한 다양한 종류의 유용한 정보를 고려하는 각 발견적 지도 기준들 H_{fail} 및 $H_{netsplit}$을 이용할 수 있다. 예를 들면, 이 기준들은 주어진 어플리케이션 인스턴스가 다른 고장 상태로 전이하기 전에 적어도 임계 시간 $T_{heuristic}$ 동안 INCOMMUNICADO 상태에 있을 것을 요구한다. 또한, 이 기준들은 주어진 어플리케이션 인스턴스로서 동일한 영역(310) 또는 데이터 센터(300)와 자원들을 공유하거나 동일한 영역(310) 또는 데이터 센터(300)에 속하는 다른 어플리케이션 인스턴스들이 또한 INCOMMUNICADO, 고장, 또는 네트워크 분할 상태에 있는 지를 고려할 수 있다. 예를 들면, 만일 주어진 어플리케이션 인스턴스로서 동일한 IP 어드레스에 또는 동일한 영역(310) 또는 데이터 센터(300) 내의 다른 어드레스들에 위치한 다른 어플리케이션 인스턴스들이 OK이면, 주어진 어플리케이션 인스턴스의 고장은 분리될 것이다. 대조적으로, 만일 다중 어플리케이션 인스턴스들이 OK가 아니면, 특히 어플리케이션 상태가 지리학 또는 네크워크 토폴로지에 따라 밀집하게 되는 경우, 네트워크 분할 시나리오가 존재할 가능성이 더욱 높아진다. 일부의 실시예에서, DFDD(110)는 고장의 성질을 결정하기 위하여, 능동적으로 수신된 상태 정보 외에 고장의 혐의를 받는 어플리케이션 인스턴스들을 질문하도록 구성될 수 있다. 일부의 실시예에서, 이 발견적 지도 기준들은, 어플리케이션 인스턴스가 (예를 들면, 50 % 확률보다 큰, 90 % 확률보다 큰) 어느 확률 임계값에 따라 전망에 근거하여 고장날 것 같은 지를 결정하도록 구성될 수 있다.

[0262] 발견적(heuristic) 기준들에 따르면, 고장난 어플리케이션 인스턴스는 고장 상태 또는 네트워크 분할 상태로 천이할 수 있다. 일 실시예에서, 만일 핵심이 수신되면, 이 예는 다시 고장 상태 또는 네트워크 분할 상태로부터 OK 상태로 천이할 수 있다. 하지만, 다른 실시예에서 고장 상태 및 네트워크 분할 상태 중의 하나 또는 모두는 회복할 수 없다. INCOMMUNICADO 상태에 있는 어플리케이션 인스턴스는 고장의 가능성으로 기능하거나 회복할 수 있는 것으로 추정될 수 있지만, 고장 상태 또는 네트워크 분할 상태에 있는 어플리케이션 인스턴스들은 (일부의 실시예에서 회복의 가능성으로) 고장날 것으로 추정될 수 있다. 일반적으로, DFDD 클라이언트들은 이 고장 상태들 중의 하나에 있는 어플리케이션 인스턴스들을 선택하는 것을 회피할 수 있다. 일부의 실시예에서, DFDD(110)는 이 고장 상태들 중의 하나에 있는 어플리케이션 인스턴스들에 관한 정보를 클라이언트들로부터 숨기도록 구성될 수 있다.

[0263] 설명된 실시예에서, 어플리케이션 인스턴스는 FORGOTTEN 상태로 진행하기 전에 각 시간 주기 T_{clean} 및 $T_{recover}$ 동안 고장 또는 네트워크 분할 상태에 남을 수 있다. 예를 들면, 고장 상태의 일부 경우에, 고장난 어플리케이션 인스턴스에 관련된 자원들은 회복 또는 분석 목적을 위하여 소정 시간 동안 유지될 수 있다. 가능하다면, 그 후, 이러한 자원들(예를 들면, 비트 저장 노드(160)의 저장 자원들)은 새로운 어플리케이션 인스턴스로서 배치용으로 초기화될 수 있다. 네트워크 분할 상태의 일부 경우에, 결정은 고장난 어플리케이션 인스턴스들 없이 시스템 동작으로 진행되는지, 그리고 그렇다면, 어떤 종류의 회복 작용이 행해져야 하는 지(예를 들면, 남은 어플리케이션 인스턴스들 사이의 객체 복사 등을 재 발생하는 것)에 관하여 이루어질 필요가 있다. 일부의 실시예에서, 이러한 회복 작용이 완료될 까지, 고장난 어플리케이션 인스턴스들은 FORGOTTEN 상태로 진행하지 않을 수 있다.

[0264] 어플리케이션 인스턴스의 FORGOTTEN 상태는 DFDD(110) 내에 명백하게 표시될 수 없다. 오히려, 일부의 실시예에서 어플리케이션 인스턴스의 FORGOTTEN 상태는 DFDD(110)로부터 어플리케이션 인스턴스의 DFDD 엔트리(111)와 같은 어플리케이션 인스턴스의 존재하는 상태 정보를 삭제함으로써 표시될 수 있다. 비록 일부의 실시예에서 어플리케이션의 새로운 예가 새로운 상태를 통하여 잊혀진 예에 할당된 동일한 자원들을 이용하여 초기화될 수 있더라도, 일반적으로, 어플리케이션 인스턴스는 FORGOTTEN 상태로부터 되찾을 수 없다. 일부의 실시예에서, 만일 어플리케이션 인스턴스가 FORGOTTEN 상태에 있는 동안 핵심들을 자발적으로 다시 보내야하면, DFDD(110)는 어플리케이션 인스턴스가 잊혀지는 것(예를 들면, 더 이상 유효한 엔트리(111)에 대응하지 않는)을 인식할 수 있고, 어플리케이션 인스턴스가 동작을 정지하거나, 자체적으로 리셋하거나 재 초기화할 것을 지시할 수 있다.

[0265] 일부의 실시예에서, 글로벌 상태 천이들을 인자로 갖는 발견적 지도법(heuristics) 및 천이 시간 파라미터들은 다른 종류의 어플리케이션 인스턴스들 마다 다를 수 있고, 이 파라미터들의 일부 또는 모두는 DFDD 클라이언트들에 의해 조정될 수 있다. 또한, DFDD 클라이언트는 일반적으로 DFDD(110)의 예가 주어진 어플리케이션 인스턴스의 현재 글로벌 상태를 확인할 것을 질의하지만, 일부의 실시예에서는 공개/가입 상태 변화 통지 모델을 지지할 수 있다. 예를 들면, DFDD 클라이언트는 특정 어플리케이션 인스턴스 또는 예 한 세트의 모든 또는 어떤 방식의 글로벌 상태 변화가 클라이언트에게 통지되는 것을 원하는 가입 처리를 통하여 DFDD(110)에게 알릴 수 있다. DFFD(110)가 이러한 상태 변화를 검출하면, DFDD는 변화를 나타내는 메시지를 가입한 DFDD 클라이언트들에게 전송할 수 있다.

[0266] 종종, 어플리케이션 인스턴스는 어플리케이션 인스턴스와 가장 가까운 DFDD(110)의 인스턴스에게 핵심 정보를 전송하도록 구성될 수 있다. 예를 들면, 일부의 실시예에서 DFDD(110)의 인스턴스는 적어도 하나의 다른 어플리케이션 인스턴스를 호스팅하도록 구성된 각 컴퓨터 시스템에 제공될 수 있어서, 어플리케이션 인스턴스들은 간단히 예들의 호스트의 국부 IP 어드레스를 참조하고 어플리케이션 인스턴스-DFDD 통신용으로 저장된 널리 알려진 IP 포트를 이용하여 DFDD(110)의 국부 예에 용이하게 접근할 수 있다. 하지만, 만일 어플리케이션 인스턴스들이 그들의 상태를 DFDD(110)의 일부 예들에게 보고하고 다른 예들에게는 보고하지 않으면, 이 상태를 동기화시키기 위한 어떤 노력이 없으므로, DFDD(110)의 배치된 예들은 분기될 수 있다.

[0267] 일부의 실시예에서, DFDD(110)의 예들 사이의 분기(divergence)는 가십 기반 프로토콜들과 같은 키맵 인스턴스들(140)에 대하여 상기한 바와 같은 동기화 프로토콜들을 이용하여 해결될 수 있다(addressed). 하지만, 많은 경우에, DFDD(110)의 예들에 의해 공동으로 관리된 DFDD 엔트리들(111)의 수는 키맵 인스턴스(140)에 의해 관리된 키맵 엔트리들(144)의 수보다 본질적으로 작을 수 있다. 이것이 상기 경우일 때, 더 간단한 조정 프로토콜들이 DFDD(110)의 예들을 동기화하도록 이용될 수 있다. 이러한 가십 기반 프로토콜의 일 실시예의 동작 방법은 도 25에 도시되어 있다. 도시된 실시예에서, 시작 인스턴스로 불리는 DFDD(110)의 일 예가 동기화를 위한 DFDD(110)의 다른 피어 예를 랜덤하게 선택하는 경우, 동작은 블록 2500에서 시작한다. 일부의 실시예에서, 시작 DFDD 예는 시작 DFDD 예의 상태 정보에 따라 현재 고장난 상태(예를 들면, 네트워크 분할)에 있는 DFDD 예들로부터 피어 DFDD 예를 가끔 신중히 선택할 수 있다. 만일 시작 DFDD 예가 명백하게 고장난 피어 예와의 계약 및 동기화를 성공하면, 명백한 고장으로부터의 회복은 촉진될 수 있다.

[0268] 그 후, 시작 인스턴스는 시작 인스턴스의 엔트리들(111)(어플리케이션 인스턴스 이름들 및 ID들, 및 아마도 종료점들 또는 다른 식별 정보를 해싱함으로써)에 반영된 어플케이션 예들의 식별 정보의 해시 값을 계산할 수 있다(블록 2502). 해시 값은 예를 들면, MD5 알고리즘과 같은 어느 적당한 해시 알고리즘에 따라 결정될 수 있다. 그 후, 시작 인스턴스는 현재 어플리케이션 인스턴스 상태 정보의 분류된 리스트(예를 들면, 핵심 카운트들, 글로벌 상태 정보 및/또는 상태 정보(114)에 포함된 어느 다른 정보)에 따라 해시 값을 피어 예로 전송

한다. 상태 정보의 리스트는 시작 및 피어 예들에서 일관된 리스트를 생성하는 어느 기준에 따라 분류될 수 있다. 예를 들면, 리스트는 어플리케이션 이름 및/또는 ID에 따라 분류될 수 있다.

[0269] 상기한 바와 같이, 일부의 실시예에서, 어플리케이션 인스턴스에 관련된 상태 정보는 핵심 메시지 내에 포함된 핵심 카운트 정보로부터 도출될 수 있다는 것이 주목된다. 따라서, 일부의 실시예에서, DFDD 예들은 어플리케이션 인스턴스들용 핵심 카운트 정보를 교환할 수 있고, 다른 DFDD 예들로부터 직접 상태 정보를 수신하는 것보다 차라리, 수신된 핵심 카운트 정보로부터 어플리케이션 인스턴스 상태 정보를 도출할 수 있다. 그래서, 일 실시예에서 주어진 DFDD 예는, 핵심 카운트 정보가 특정 어플리케이션 인스턴스로부터 직접 수신되었는 지 또는 동기화 프로토콜을 통하여 다른 DFDD 예로부터 간접적으로 수신되었는 지에 무관하게 수신된 핵심 카운트 정보를 기초로 하여 (예를 들면, 도 24의 상태 정보에 따라) 특정 어플리케이션 인스턴스의 상태를 업데이트하도록 구성될 수 있다. 이러한 실시예에서, DFDD 예들 사이의 어플리케이션 인스턴스 동작 상태 정보의 동기화는 어플리케이션 인스턴스의 특정 글로벌 동작 상태(예를 들면, OK, INCOMMUNICADO 등)를 직접 교환하지 않는 핵심 정보의 동작화를 포함하고, 동기화 프로토콜의 동작을 단순화할 수 있다.

[0270] 해시 값 및 상태 정보의 리스트의 수신에 응답하여, 피어 예는 시작 인스턴스에 의해 수행된 것과 일치하는 방법으로 자신의 어플리케이션 인스턴스들의 식별 정보의 해시값을 계산하고(블록 2506), 계산된 해시 값을 시작 인스턴스로부터 수신된 해시 값과 비교한다(블록 2508). 만일 2개 값이 일치하면, 시작 및 피어 예들 둘 다가 동일한 어플케이션 예 세트에 대응하는 엔트리들(111)을 가질 가능성이 높다. 그 후, 피어 예는 수신된 상태 정보 리스트를 스캐닝하고 수신된 리스트로부터 엔트리들(111)을 적합한 것으로서 업데이트할 수 있다(블록 2510). 예를 들면, 만일 수신된 리스트에 있는 핵심 카운트 또는 시간 스탬프가 피어의 엔트리들(111)의 하나에 저장된 것보다 더 크거나 더 최근 것이면, 피어 예는 수신된 리스트에 있는 상태 정보로부터 엔트리(111)를 업데이트할 수 있다. 일부의 실시예에서, 피어 예는 시작 인스턴스로부터 리스트를 수신함과 동시에 또는 수신한 이후에 유사한 처리를 위하여 자신의 상태 정보 리스트를 시작 인스턴스로 다시 보낼 수 있다.

[0271] 만일 해시 값들이 일치하지 않으면, 피어에게 알려진 어플리케이션 인스턴스 세트와 시작 인스턴스들이 적어도 하나의 엔트리(111)에서 다를 것이다. 따라서, 피어 예는 (엔트리들(111)의 상태 정보에 바로 반대되는 것으로서) 시작 인스턴스에 알려진 엔트리들(111)의 완전한 덤프(dump)를 요청할 수 있다(블록 2512). 그 후, 피어 예는 이전에 부족했던 일부 엔트리들(11)을 부가하고 남은 엔트리들(111)의 상태를 동기화시킬 수 있다(블록 2514). 상기한 바와 같이, 일부의 실시예에서 피어 예는 시작 인스턴스로부터 엔트리들(111)의 완전한 덤프를 수신함과 동시에 또는 수신한 이후에 이 덤프를 시작 인스턴스로 다시 보낼 수 있다.

[0272] 일부의 실시예에서, 시스템 내에 존재하는 DFDD(110)의 모든 예는 소정 시간 간격으로 상기한 동기화 프로토콜 또는 적당한 그 변형을 반복적으로 실행하도록 구성될 수 있다는 것이 예측된다. 예를 들면, 이 프로토콜은 대략 1초의 주기 또는 어느 적당한 주기로 DFDD(110)의 예들에 의해 주기적으로 실행될 수 있다. 일부의 실시에에서, DFDD(110)의 예들은 상호 동일한 주기 및 상이한 위상 오프셋을 갖는 동기화 프로토콜을 실행하여, 주어진 시간에는 DFDD(110)의 예들의 일부만이 프로토콜을 시작할 수 있도록 한다는 것이 또한 예측된다.

[0273] 일부의 실시예에서, DFDD(110)의 예들은 단지 저장 서비스 시스템 내에 정의된 어플리케이션 들이 아닌, 어느 분배된 시스템 내의 어느 방식의 어플리케이션 인스턴스들을 위한 상태 정보를 조정하여 통신하도록 사용될 수 있다는 것이 주목된다. 또한, 일부 실시예에서, DFDD 인스턴스들의 상이한 그룹들은 상이한 어플리케이션 인스턴스 상태 정보를 관리할 수 있다. 이러한 일부 실시예에서, 그룹들은 동일한 그룹의 멤버들인 DFDD(110)의 인스턴스들에 공통 ID를 할당하고 DFDD 동기화로서 식별자들을 매치에 요구함으로써 상호 구별될 수 있다. 예를 들면, 저장 서비스 시스템 어플리케이션 인스턴스들을 관리하는 DFDD 예들은 저장 서비스 시스템에 무관한 다른 어플리케이션 인스턴스들의 상태를 관리하도록 구성된 DFDD 예들과 다른 식별자를 가질 수 있고, 동일한 식별자를 갖는 DFDD 예들 만이 도 25의 동기화 프로토콜에 따라 상호 정보를 교환할 수 있다.

[0274] 일부의 실시예에서, DFDD 그룹 식별자들은 동일한 시스템에 존재하는 어플리케이션 인스턴스들의 상이한 구성들을 구별하는데 사용될 수 있다. 예를 들면, "생성" 식별자에 대응하는 DFDD(110)의 예 한 세트는 저장 서비스 시스템 또는 다른 분배형 시스템의 생성 버전을 관리하도록 배치되고, 생성 시스템에 대응하는 어플리케이션 인스턴스 한 세트를 반영할 수 있다. 반면에, "검사" 식별자에 대응하는 DFDD(110)의 다른 예 한 세트는 어플리케이션들의 다른 예 한 세트에 대응하는 시스템의 검사 버전 및 상태를 관리하도록 배치될 수 있다. 일부의 경우에, 시스템이 생성 또는 검사 버전에 대응하는 어플리케이션 인스턴스들 및/또는 DFDD 예들이 동일한 기초 시스템 자원들(예를 들면, 동일한 컴퓨터 시스템)에서 실행될 수 있지만, 이들 별개의 DFDD 그룹 식별자들에 의하여 상호 투과적일 수 있다. 예를 들면, 도 25에 도시된 프로토콜과 같은 동기화 프로토콜을 실행할 때,

DFDD 예들은, DFDD 예들이 동일한 그룹의 멤버들인 지를 (예를 들면, 그룹 식별자들을 교환함으로써) 먼저 결정하고, 이 결정에 따라 후속 동기화 단계들을 수행하여, 그룹들 사이의 어플리케이션 인스턴스 상태 정보의 분리를 촉진할 수 있다.

[0275] DFDD(110)의 예들을 동기화시키기 위한 상기한 가십 기반 프로토콜은 저장 서비스 시스템을 통하여 존재하는 어플리케이션의 동작 상태들의 분배를 지원하고,

[0276] 다른 시스템 구성 요소들에 의하여 새로운 어플리케이션 인스턴스들의 발견을 촉진할 수 있다. 예를 들면, 일단 새로운 어플리케이션 인스턴스가 초기화되어 DFDD(110)의 예(예를 들면, 새로운 어플리케이션 인스턴스가 초기화하는 시스템에서 국부적으로 동작하는 예)와 접촉하면, 새로운 엔트리(111)는 새로운 예에 대응하여 생성될 수 있다. 새로운 엔트리(111)가 생성된 DFDD(110)의 예가 DFDD(110)의 다른 다양한 예들과 그 상태를 동기화시킴에 따라, 새로운 엔트리(111)는 시스템을 통하여 전파될 수 있다. 그 후, DFDD(110)에게 다양한 목적들을 위한(예를 들면, 새로운 객체(30)를 저장하거나 키맵 엔트리(140)를 업데이트하기 위한) 어플리케이션 인스턴스들을 식별하도록 질의하는 DFDD 클라이언트들은 새로운 어플리케이션 인스턴스 및 어느 존재하는 어플케이션 예들에 관한 상태 정보로 표현될 수 있다.

[0277] 상기한 실시예들에서, 고장 검출 및 발견에 관한 어플리케이션 인스턴스 상태 변화는 어플리케이션 인스턴스들의 일부 또는 어플리케이션 인스턴스들을 참조하는 DFDD 클라이언트들의 중재 없이 시스템을 통하여 전파할 수 있다는 것이 또한 주목된다. 즉, 주어진 어플리케이션 인스턴스는 단지 어떻게 핵심 정보를 DFDD(110)의 일 예로 전송하는 지를 알 필요가 있다. 이것은 시스템 내의 DFDD(110)의 모든 예들, 다른 어플리케이션 인스턴스들, 또는 주어진 어플리케이션 인스턴스를 실시하는 다양한 클라이언트들에 대한 지식을 가질 필요가 없다. 동일한 방법으로, DFDD 클라이언트들은 다른 클라이언트들, 또는 시스템 내의 모든 어플리케이션 또는 DFDD 예들에 대한 독립적인 지식을 가질 필요가 없고, 클라이언트들은, 클라이언트들이 시스템 내에서 이용할 수 있는 자원들의 상태에서 적당한 현재 정보를 얻기 위하여 통신하는 DFDD(110)의 예에 의존할 수 있다. 어플리케이션 인스턴스들의 상태에 대한 변화를 즉시 알릴 것을 다른 어플리케이션 인스턴스들 또는 클라이언트들에게 요구하지 않는 것으로 허용함으로써, DFDD(110)은 저장 서비스 시스템의 확장 적응성(scalability)를 촉진할 수 있다.

[0278] 일부의 저장 서비스 시스템 실시예들에서, 객체들(30)은 객체들의 복사 정도, 영역들(310) 사이의 복사들의 분배, 복사들이 저장되는 저장 자원들의 방식, 및/또는 다른 시스템 특성 또는 정책들에 대하여 균일하게 처리될 수 있다. 예를 들면, 시스템은 모든 객체(30)를 동일한 회수로 동일한 수의 별개 영역들(310)에 복사하는 것을 시도할 수 있다. 하지만, 다른 클라이언트들(50)은 다른 객체들(30) 마다 상기한 저장 요구를 가질 수 있다. 예를 들면, 하나의 클라이언트(50)가, 디폴트 저장 정책이 제공할 수 있는 것보다 더 높은 신뢰성(예를 들면, 복사들의 개수 및 분배에 의하여)을 갖는 특정 객체(30)를 저장하기를 바랄 수 있다. 이에 반해, 다른 클라이언트(50)는 신뢰성의 디폴트 레벨조차도 요구할 수 없다. 대안으로, 클라이언트(50)은 가능한 신뢰성의 비용으로 객체 복사들이 분배된 영역들(310)의 수를 한정함으로써 객체 기입 성능을 증가시키는 것을 바랄 수 있다.

[0279] 따라서, 일 실시예에서, 도 2에 도시된 것과 같은 저장 서비스 시스템은 객체들(30)의 저장 등급들을 지지하도록 구성될 수 있다. 일반적으로 말해, 주어진 객체의 저장 등급은 주어진 객체(30)에 대한 서비스 레벨 협약(SLA)에 영향을 주는 어느 저장 서비스 특성 또는 특징 세트를 지정할 수 있다. 서비스 레벨 협약은 일반적으로 서비스 제공자가 서비스를 클라이언트로부터 수신된 어느 대가(consideration)(예를 들면, 비용 또는 어느 다른 종류의 대가)용 교환으로, 클라이언트에게 제공하는 하에서의 보증 또는 기대 세트를 반영할 수 있다. 예를 들면, 저장 서비스 시스템에 의해 관리된 객체들(30)용 SLA는 객체 신뢰성, 유용성, 접근 성능(예를 들면, 지연, 대역폭)의 다양한 레벨들, 서비스의 비용 또는 비율, 또는 클라이언트의 객체(30)와의 상호 작용의 어느 다른 측정가능한 양상들을 지정할 수 있다. 일부의 실시예에서, 저장 등급은 SLA 특징의 특정 서브세트(예를 들면, 상기한 바와 같은 객체 복사들의 수 및 분배) 만을 지정할 수 있다. 한편, 다른 실시예에서, 저장 등급은 주어진 객체(30)에 대한 SLA의 모든 정의된 양상들을 포함하는 광범위한 SLA에 직접 대응할 수 있다.

[0280] 일 실시예에서, 저장 서비스 시스템은 각각 특정하게 정의된 SLA 특징을 갖는 고정된 저장 등급 세트를 정의할 수 있고, 클라이언트들은 특정 객체들(30)을 특정 저장 등급들에 연관되도록 선택할 수 있다. 예를 들면, 디폴트 저장 등급은, 객체(30)가 적어도 3번 적어도 2개의 다른 영역들(310)에 복사될 것이다. 버킷 저장 등급은, 객체(30)의 단일 복사가 단일 영역(310)에 저장되는 것을 지정할 수 있다. 국부 저장 등급은, 객체(30)가 적어도 3번 단일 영역(310)에 복사되는 것을 지정할 수 있다. 다른 실시예들에서, 저장 서비스 시스템은

다른 특징들을 갖는 저장 등급들을 지정할 수 있고, 클라이언트(50)가 주어진 객체(30)에 적용될 저장 정책들의 결합(노드 피커(130)에 대하여 설명된 바와 같이)을 지정함으로써, 저장 등급이 주어진 객체(30)에 맞춰질 수 있도록 한다.

[0281] 상기한 바와 같이, SLA 특징은 복사들의 수 및 복사들이 분배되어야 하는 영역들의 수를 초과하여 확장할 수 있다. 일 실시예에서, 특정 저장 등급의 SLA 특징은 특정 저장 등급에 관련된 객체들(30)에 대응하는 기대되는 처리 지연의 표시를 포함할 수 있다. 예를 들면, 하나의 저장 등급은 주어진 가격으로 낮게 기대되는 처리 지연을 지정할 수 있는 반면, 다른 저장 등급은 저가로 높게 기대되는 처리 지연을 지정할 수 있다. 기대되는 처리 지연의 상이한 레벨들은 다양한 방법으로 실행될 수 있다. 예를 들면, 주어진 조정자(120)의 견지에서, 일부 노드들(160)은 주어진 조정자(120)에서의 노드들(160)의 근접, 노드들(160)에서 유용한 자원들의 레벨 및 방식, 노드들(160)의 처리 부하, 또는 다른 적절한 인자들과 같은 인자들로 인하여 다른 것보다 더 낮은 접근 지연을 나타낼 수 있다. 일부의 실시예에서, 주어진 저장 등급에 의해 지정된 다른 SLA 특징들에 의해 실행된 억제에 따라, 조정자(120) 및/또는 노드 피커(130)는 더 낮게 기대되는 처리 지연을 지정하는 저장 등급에서 객체들(30)에 대하여 더 낮은 접근 지연을 나타내는 노드들(160)을 선택하도록 구성될 수 있다. 다른 실시예에서, 조정자(120)는 객체들(30)에 관련된 저장 등급들의 기대되는 처리 지연들에 따라 객체(30)에 대한 클라이언트 접근 요청들의 처리를 우선 수행하도록 구성될 수 있다. 예를 들면, 조정자(120)는, 더 높게 기대되는 처리 지연을 갖는 저장 등급에 대한 요청들이 결국 완료되는 것을 보장하면서, 더 낮게 기대되는 처리 지연을 갖는 저장 등급들을 위한 처리를 바이어스하도록 구성된 별개 대기열들, 또는 다른 처리 제어 또는 데이터 구조들을 실행할 수 있다.

[0282] 저장 등급은, 객체(30)가 저장 서비스 시스템 내에 초기에 저장되는 시간에 클라이언트(50)에 의해 지정될 수 있다. 대안으로, 일부의 실시예에서, 객체(30)가 저장 서비스 시스템 내에 존재하는 어떤 시간에, 클라이언트(50)는 객체(30)에 관련된 저장 등급을 변경할 수 있다. 객체(30)가 최초로 저장되는 경우에 어떤 저장 등급도 클라이언트(50)에 의해 지정되지 않으면, 상기한 하나와 같은 디폴트 저장 등급이 이용될 수 있다. 상기한 바와 같이, 일부의 실시예에서 객체(30)의 저장 등급이 객체(30)의 키에 관련된 키맵 기록(148) 내에 저장될 수 있다. 이러한 실시예에서, 조정자(들)(120) 및/또는 반복자(180)는 객체(30)의 존재하는 복사들을 저장하고, 복사하고, 유지할 때 객체(30)의 저장 등급을 고려하도록 구성될 수 있다. 클라이언트들(50)은 상이한 저장 등급들에 관련된 객체들(30)에 대하여 상이한 사용료가 청구될 수 있다. 예를 들면, 고-신뢰성 저장 등급은 일반적으로 더 많은 시스템 자원들을 이용할 수 있고, 버킷 저장 등급은 더 작은 수의 자원들을 이용할 수 있다. 따라서, 주어진 크기의 객체(30)에 대하여, 클라이언트(50)는 전자 저장 등급를 이용하여 객체(30)를 저장하기 위해 더 비싸게 청구되고, 후자 저장 등급에 대하여 더 싸게 청구될 수 있다.

[0283] 저장 서비스 시스템 내의 저장 등급들의 동작 방법의 실시예가 도 26에 도시되어 있다. 도시된 실시예에서, 클라이언트(50)가 특정 객체(30)에 관련될 저장 등급을 지정하는 경우 동작은 블록 2600에서 시작한다. 이어서, 저장 등급은 저장 서비스 시스템 내의 특정 객체(30)에 지속적으로 관련된다(블록 2602). 예를 들면, 저장 등급의 표시는 클라이언트(50) 대신에 조정자(120)에 의해 키맵 기록(148)과 같은 특정 객체(30)에 관련된 데이터 구조에 저장될 수 있다. 그 후, 객체(30)에 관련된 객체 데이터의 상태는 지정된 저장 등급의 특징에 따라 구성된다(블록 2604). 예를 들면, 만일 저장 등급이 영역들(310) 사이의 객체 복사들의 수 및/또는 분배에 대한 확실한 요구를 지정하면, 조정자(120) 및/또는 반복자(180)가 동작하여 필요한 복사들을 발생하여 분배함으로써, 특정 객체(30)에 대한 저장 시스템의 결과로 얻어지는 상태가 저장 등급의 필요 조건들을 만족시킬 수 있도록 한다. 일부의 실시예에서, 반복자(180)는, 객체(30)에 대한 저장 등급 필요 조건들이 시간이 지남에 따라 유지되는 지를 보장하도록 구성될 수 있다. 예를 들면, 만일 복사들이 실패되면, 반복자(180)는 이 실패를 검출하여 추가 복사들을 발생하도록 구성될 수 있다.

[0284] 일부의 실시예에서, 주어진 저장 등급에 의해 지정된 저장 특징들은 비트 저장 노드들(160)을 통하여 이용할 수 있는 저장 자원들의 상이한 방식들 사이를 구별하는 것을 포함할 수 있다는 것이 예측된다. 예를 들면, 일부의 실시예에서, 일부의 비트 저장 노드들(160)은 다른 장치보다 더 높은 성능의 저장 장치들을 포함할 수 있거나, 개별 비트 저장 노드(160)는 고- 및 저-성능 장치들의 결합을 포함할 수 있다. 이러한 실시예에서, 저장 등급은, 하나 또는 다른 방식의 장치가 저장 등급에 관련된 객체들(30)용으로 사용되어야 한다.

[0285] 동적 복사

[0286] 상기한 바와 같이, 일부의 실시예에서 노드 피커(130)는 특정 객체(30)가 기입되어야 하는 특정 비트 저장 노드(160)를 식별하는 기입 계획을 발생하도록 구성될 수 있다. 일단 기입 계획이 예를 들면, 조정자(12

0)에 의해 실행되었으면, 다양한 기입 정책들이 특정 객체(30)에 대하여 만족되는 방식으로, 이러한 기입 계획이 발생될 수 있다. 예를 들면, 기입 계획에 의해 지정된 노드들(160)의 수는 특정 객체(30)에 요구된 복사들의 최소 수, 복사들이 분배되어야 하는 별개 영역들(310)의 최소 수, 또는 어느 다른 저장 정책 고려 대상에 따라 결정될 수 있다.

[0287] 일부의 실시예에서, 노드들(60)이 노드들(160)의 현재 상태를 고려하지 않으면서 예측 가능한 과정에 따라 일관성 있게 선택되는 정적 방식으로, 노드 피커(130)가 기입 계획들을 발생하도록 구성될 수 있다. 예를 들면, 노드 피커(130)는 복사들을 저장하기 위한 동일한 노드(160) 세트를 일관성 있게 선택할 수 있거나, 다수의 노드(160)를 통하여 라운드-로빈(round-robin) 방식으로 회전할 수 있다. 하지만, 큰-스케일 실행에서, 저장 서비스 시스템은 다양한 회수로 상당히 다른 상태들에서 동작할 수 있는 다수의 노드들(160)을 포함할 수 있다. 예들 들면, 일부의 노드들(160)은 동작하지 않고, 다른 노드들은 동작하지만 요청 활동으로 포화되거나 거의 자유 자원들이 존재하지 않고, 여전히 다른 노드들은 상대적으로 유휴상태(idle)이거나 상당한 자유 자원들을 가질 수 있다.

[0288] 또한, 상이한 노드들(160)은 주어진 조정자(120) 또는 노드 피커(130)의 견지에서 통신비의 상이한 레벨들을 나타낼 수 있다. 예를 들면, 조정자(120)와 동일한 영역(310) 또는 데이터 센터(300) 내에 위치하는 노드들(160)이 국부적 저-지연 네트워크 연결을 통하여 접근할 수 있다. 대조적으로, 조정자(120)와 다른 영역(310) 또는 데이터 센터(300)에 위치한 노드들(160)은 국부 노드들(160)보다 본질적으로 더 높은 접근 지연을 나타낼 수 있다. 또한, 일부의 실시예에서, 영역들(310) 또는 데이터 센터들(300) 사이의 통신은 국부 통신과는 다른 경제적 가격의 모델들을 갖는 통신 네트워크들을 통하여 이루어질 수 있다. 예를 들면, 영역(30) 내의 통신은, 데이터 전송에 대한 어떠한 사용 기반 비용이 부가되는 충분한 대역폭을 갖는 개인 전용 근거리 통신망(LAN)을 통하여 이루어질 수 있다. 대조적으로, 데이터 센터들(300) 사이의 통신은 전용 통신 설비, 공용 인터넷, 개인 전용 광대역 망(WAN) 설비, 또는 다른 장거리 통신 네트워크들과 같은 설비를 통하여 이루어질 수 있다. 이 설비들은 통상적으로 LAN 설비들보다 더욱 대역폭-한정될 수 있고, 일부의 예에서는 LAN 통신에 적용될 수 없는 제3 자에 의해 청구된 사용료(예를 들면, 피크 또는 통합 대역폭 사용에 기초하여)를 제공할 수 있다.

[0289] 다양한 노드들(160)에 대한 통신비뿐만 아니라 노드들(160)의 동작 상태가 시간이 지남에 따라 변할 수 있다. 예를 들면, 동시에 동작하거나 정지하는 노드(160)는 나중에 동작하지 않거나 사용 중이 될 수 있거나 역으로 된다. 동일한 방법으로, 지연 및/또는 경제 비용과 같은 통신비는 일부 주기 중에는 더 높고 다른 주기 중(예를 들면, 피크 주기 중 대 오프-피크 사용)에는 더 낮을 수 있다. 이 변화성 때문에, 일 시간에 효과적이고 저가인 기입 계획은 다른 시간(예를 들면, 만일 기입 계획에서 지정된 노드들(160)이 사용 중이고, 느리게 통신하거나 동작하지않는)에 효율이 상당히 낮고 고가이거나, 심지어 실행 불능일 수 있다.

[0290] 그래서, 일부의 실시예에서, 노드 피커(130)는 노드들(160)에 관련된 현재 상태 정보에 따라 주어진 객체(30)의 기입된 복사들용 주어진 기입 계획을 동적으로 결정하도록 구성될 수 있다. 일반적으로 말해, 동적으로 결정된 기입 계획은 노드들(160)에 관한 관찰가능한 동적 상태 정보를 고려할 수 있다. 즉, 동적으로 결정된 기입 계획은 시간이 지남에 따라 변할 수 있는 노드 상태 정보의 기능으로서 발생될 수 있다. 즉, 주어진 객체(30)용으로 동적으로 결정된 기입 계획 자체는 노드들(160)의 상태에 무관하게 결정될 수 있는 정적으로 생성된 기입 계획에 반하여, 노드들(160)의 기초 상태 정보에 의존하여 시간이 지남에 따라 변할 수 있다. 상기한 바와 같이, 상이한 많은 종류의 상태 정보는 기입 계획들의 동적 발생을 고려할 수 있다. 일반적으로, 노드들(160)의 상태 정보는, 주어진 노드(160)가 접근할 수 있는 통신 자원들(예를 들면, 네트워크 자원들)에 관한 상태 정보뿐만 아니라 주어진 노드(160)에 관한 상태 정보를 포함할 수 있다. 다양한 실시예에서, 주어진 노드(160)에 관한 상태 정보는, 주어진 노드(160)(또는 노드에 관련된 어플리케이션 인스턴스)가 상기한 바와 같이 DFDD(110)에 의해 표시된 OK, INCOMMUNICADO 또는 다른 동작 상태에 있는 지와 같은 주어진 노드(160)의 동작 상태를 포함할 수 있다. 주어진 노드(160)에 관한 상태 정보는 동작 상태 정보보다 주어진 노드(160)의 작용에서 더 상세하게 나타낼 수 있는 부하 상태 정보를 또한 포함할 수 있다. 예를 들면, 다양한 실시예에서 부하 상태 정보는 주어진 노드(160)에 대응하는 프로세서 이용, 메모리 이용, 저장 장치 용량 이용, 저장 장치 입력/출력 대역폭 이용, 또는 네트워크 인터페이스 대역폭 이용의 레벨, 또는 노드 작용의 어느 다른 측정가능한 양상을 나타낼 수 있다. 상기한 바와 같이, 일부의 실시예에서 부하 상태 정보는 동작 상태 정보에 더하여 DFDD(110)를 통하여 이용될 수 있다.

[0291] 또한 네트워크 비용 정보로 불릴 수 있는 통신 자원 상태 정보는 주어진 노드(160)에서의 적어도 하나의 통신 경로의 상태에 관한 어떤 적당한 정보를 포함할 수 있다. 다양한 실시예에서, 네트워크 비용 정보는 메시지를 주어진 노드(160)로 전송하고 주어진 노드(160)로부터 전송받는 것에 관련된 네트워크 통신 지연을 나타

낼 수 있고, 시간(예를 들면, 초, 밀리초, 등), 네트워크 홉(hop)의 수(예를 들면, 메시지를 전송하기 위한 라우팅 단계의 수), 또는 어느 적당한 계량에 의해 표현될 수 있다. 일 실시예에서, 네트워크 비용 정보는 주어진 노드(160)와의 통신에 이용할 수 있는 유효한 대역폭(예를 들면, 데이터 전송율)의 표시를 포함할 수 있다. 다른 실시예에서, 네트워크 비용 정보는 주어진 노드(160)와의 네트워크 통신에 관련된 경제적 비용의 표시를 포함할 수 있다. 예를 들면, 이러한 비용은 어떤 양의 데이터를 전송하거나 수신하기 위해 청구된 요금, 또는 네트워크 통신용 어느 다른적당한 비용 또는 요금 모델로 표현될 수 있다.

[0292] 노드 피커(130)는 일반적으로 객체(30)용 기입 계획을 동적으로 결정할 때 노드들(160)의 상태 정보의 어느 적당한 기능을 이용할 수 있다. 일부의 실시예에서, 노드 피커(130)에 의해 실행된 (상기한 저장 정책들에 더하여 또는 대신일 수 있는) 저장 정책들은 특정 객체(30)용 기입 계획에 포함될 적합할 수 있는 노드들(160)을 억제하는 상태 정보에 대한 지침들 또는 필요 조건들을 지정할 수 있다. 다양한 실시예들에서, 이 정책들은 특정 객체(30) 세트(예를 들면, 공통 키 접두부를 갖는 특정 저장 등급 또는 버킷에 포함된, 또는 한 세트의 멤버들로서 표시된 객체들), 또는 (예를 들면, 객체(30)에 관련될 특정 정책을 지정하는 클라이언트에 응답하는) 개별 객체들(30), 또는 이들의 어느 적당한 결합에 글로벌하게 (예를 들면, 모든 객체들(30)) 적용될 수 있다. 일 예로서, 특정 저장 등급은 복사들의 약간의 최소 수가 약간의 최소 통신 지연만을 나타낼 것을 요구하는 저장 정책을 지정할 수 있다. 따라서, 저장 등급에 객체(30)용 기입 계획을 생성하는 경우, 노드 피커(130)는, 적어도 일부의 노드들(160)이 지정된 최소 통신 지연을 만족시키는 지에 따라 적어도 일부의 노드들(160)을 선택하도록 구성될 수 있다.

[0293] 일부의 실시예에서, 노드 피커(130)는 또한 노드 상태 정보에 대한 다양한 방식의 최적화에 따라 기입 계획을 생성하도록 구성될 수 있다. 예를 들면, 기입 계획에 관련된 특정 최소 네트워크 비용 또는 다른 비용을 지정하는 것에 대한 대안으로서, 저장 정책은, 이 비용이 특정 시간에 이용할 수 있는 자원들 사이에서 최소화되는 것을 지정할 수 있다. 따라서, 노드 피커(130)는 예를 들면, 저 네트워크 통신 또는 다른 관련 비용을 갖는 노드들(160)을 선택함으로써, 기입 계획에 관련된 적어도 하나의 비용을 최소화하도록 구성될 수 있다. 일부의 실시예에서, 이러한 최소화는 다른 노드 상태 정보 필요 조건들을 지정하는 다른 저장 정책들과 같은 다른 억제에 직면하여 발생할 수 있다.

[0294] 또한, 일부의 실시예에서, 일부의 노드 상태 정보는 예측가능한 방식으로 시간이 지남에 따라 변할 수 있다는 것이 주목된다. 예를 들면, 데이터 센터들(300) 사이의 네트워크 통신에 관련된 대역폭 비용들이 잘 정의된 요금 계획표에 따라 변할 수 있다. 이러한 일부의 실시예에서 기입 계획에 관련된 비용을 최소화하는 것은 기입 계획의 모두 또는 일부가 특정 시간 주기에 관련된 비용에 따라 실행되어야 하는 동안의 시간 주기를 지정하는 것을 포함할 수 있다. 예를 들면, 노드 피커(130)는, 원격 데이터 센터(300)와 통신하기 위한 대역폭이 현재 시간에서보다 일부 미래 시간에 덜 비싼 지를 결정할 수 있고, 원격 데이터 센터(300)에서 노드들(160)을 포함한 기입 계획의 비용인 지정된 미래 시간에 원격 데이터 센터로 지향된 적어도 저장 동작들을 수행함으로써 최소화될 수 있는 지를 더 결정할 수 있다. 이 처리의 하나의 가능한 결과는, 노드 피커(130)에 의해 생성된 기입 계획이, 주어진 객체(30)의 일부 (또는 아마도 모든) 복사들의 생성이 지정된 미래 시간까지 지연되어야하는 것을 나타낼 수 있다는 것이다.

[0295] 다른 많은 저장 정책들이 특정 객체(30)에 적용될 수 있다. 또한, 일부의 경우에, 특정 객체(30)에 관련된 각 저장 정책을 만족시키는 단일 기입 계획을 생성할 수 없을 수 있다. 예를 들면, 특정 객체(30)에 관련된 저장 정책들은, 최소수의 복사들이 최소수의 별개의 영역들(310) 사이에 저장되어 분배되어야 하는 것을 지정할 수 있다. 하지만, 기입 계획이 특정 객체(30)를 위하여 생성하는 시간에, 노드 피커(30)가 실행하는 영역(310)이 일시적인 통신 고장으로 인하여 다른 영역들(310)로부터 일시적으로 분리될 수 있다. 따라서, 대응하는 저장 정책의 만족으로서 복사들을 다른 영역들(310)에 성공적으로 분배하는 것은 적어도 일시적으로 불가능할 수 있다.

[0296] 일 실시예에서, 노드 피커(130)는 기입 계획에 의해 만족할 수 있는 저장 복사들의 수를 최소화하는 것에 기초를 두어 객체들(30)용 기입 계획을 동적으로 결정하도록 구성될 수 있다. 부분 최적 조건들이 존재 하에, 이것은 저장 계획들을 만족시키는 "최상 노력"을 나타내는 기입 계획으로 끝날 수 있다. 예를 들면, 바로 이전에 설명된 특정 시나리오에서, 통신 고장으로 인하여 영역 다양화 정책을 만족하지 못할 수 있지만, 국부 영역(310) 내의 특정 객체(30)의 복사들의 요구된 최소 수를 저장함으로써 최소 복사 정책을 만족할 수 있다. 일부의 실시예에서, 저장 정책들의 최소화는 다양한 억제 하에 발생할 수 있다. 예를 들면, 일부 저장 정책들은 의무로서 확인될 수 있어서, 만일 이 일부 저장 정책들을 만족할 수 없으면, 기입 계획은 결정될 수 없고, 객체를 저장하기 위한 대응 클라이언트 요청은 실패할 수 있다. 다른 저장 정책들은 우선 순위 등급 매기기 또는 가

중치 부가를 포함하여, 더 높은 순위 저장 정책들이 최소화 공정 중에 더 낮은 순위 저장 정책들에 우선하여 선택될 수 있다. 다른 실시예에서, 저장 정책들의 선택은 결과로 생기는 저장 계획에 의해 만족하는 저장 복사들의 수에 더하여 또는 대신하여 (만족한 저장 정책들의 가중치에 기초하여 결정된) 결과로 생기는 저장 계획의 전체 가중치를 최소화함으로써 수행될 수 있다.

[0297] 객체들(30)이 원래대로 저장되는 경우, 객체들(30)용 기입 계획들을 동작으로 결정하기 위한 다양한 기술들이 단독으로 발생할 필요는 없다는 것이 주목된다. 상기한 바와 같이, 일부의 실시예에서 반복자(180)는, 객체들(30)의 복사들을 접근할 수 있는 지를 결정하기 위하여 객체들(30)에 대응하는 키맵 엔트리들(144)을 검사하도록 구성될 수 있다. 만일 특정 객체(30)의 일부 복사들을 접근할 수 없으면, 반복자(180)는 추가 복사들을 생성하기 위해 사용될 수 있는 노드 피커(130)에 새로운 기입 계획을 요청하도록 구성될 수 있다. 새로운 기입 계획은 상기한 기술의 어느 적당한 결합을 이용하여 노드 피커(130)에 의해 동적으로 결정될 수 있다. 또한, 일부의 실시예에서 반복자(180)는 다양한 저장 복사들에 대하여 객체들(30)의 호환성(compliance)을 더욱 일반적으로 모니터링하도록 구성될 수 있다. 예를 들면, 반복자(180)는, 객체(30)의 존재하는 복사 세트가 최소 복사 정책 또는 어느 다른 적당한 복사 세트뿐만 아니라 영역 다양화 정책을 만족시키는 지를 결정하도록 구성될 수 있다. 이러한 일 실시예에서, 만일 반복자(180)가 특정 객체(30)의 존재하는 복사들에 의해 만족한 정책들의 수가 어떤 임계값 미만인 것을 결정하면, 반복자(180)는 노드 피커(130)에 상기한 바와 같이 만족한 저장 정책들의 수를 최소화하기 위해 동작으로 결정될 수 있는 새로운 저장 계획을 요청할 수 있다. 대체 실시예들에서, 반복자(180)는 특정 의무 저장 정책이 더 이상 만족스럽지 못하거나 만족한 저장 정책들의 전체 가중치가 임계 가중치 이하로 떨어지는 것을 결정한 것에 기초하여 새로운 저장 계획을 요청할 수 있다.

[0298] 도 27은 저장 노드들의 현재 상태 정보에 따른 데이터 객체의 적어도 하나의 복사를 저장하기 위한 기입 계획을 동적으로 결정하는 방법의 일 실시예를 설명한다. 설명된 실시예에서, 주어진 객체(30)를 저장하기 위한 클라이언트 요청이 수신된 경우, 동작을 블록 2700에서 시작한다. 일 실시예에서, 이러한 요청은 상기에서 상세히 설명된 바와 같이, 웹 서비스 인터페이스(100)를 통한 웹 서비스 프로토콜에 따라 수신될 수 있다.

[0299] 이어서, 주어진 객체(30)의 복사들을 저장하기 위한 기입 계획은 비트 저장 노드들(160)의 현재 상태 정보에 따라 동적으로 결정된다(블록 2702). 예를 들면, 다양한 저장 정책들이 노드 동작 상태, 노드 부하 상태 정보, 네트워크 통신 비용, 또는 상기에서 상세히 설명한 바와 같은 어느 다른 적당한 상태 정보와 같은 어느 적당한 상태 정보를 고려하는 경우, 노드 피커(130)는 주어진 객체(30)에 적용될 수 있는 다양한 저장 정책들에 따라 기입 계획을 결정하도록 구성될 수 있다. 또한, 상기한 바와 같이, 일부의 실시예에서 기입 계획을 동적으로 결정하는 것은 기입 계획에 관련된 비용을 최소화하거나 기입 계획에 의해 만족한 저장 정책들의 수 또는 가중치를 최소화하는 것과 같은, 저장 정책들의 상태 정보에 대한 최적화를 포함할 수 있다.

[0300] 그 후, 주어진 객체(30)의 복사들은 동적으로 결정된 기입 계획에 대응하는 적어도 하나의 비트 저장 노드들(160)에 저장된다(블록 2704). 예를 들면, 조정자(120)는 상기한 바와 같이 기입 계획에 지정된 개별 비트 저장 노드들(160)에 지향된 비트 저장 객체 풋 동작들을 생성하도록 구성될 수 있다. 일부의 실시예에서, 기입 계획의 일부 저장 동작들은 상기한 바와 같이 다른 동작들과는 다른 시간에 실행될 수 있다.

[0301] 상기한 바와 같이, 일부의 실시예에서, 기입 계획은, 적어도 하나의 복사가 이미 비트 저장 노드들(160) 사이에 저장되는 객체(30)에 대하여 동적으로 결정될 수 있다. 도 28은 이러한 방법의 실시예를 설명한다. 설명된 실시예에서, 주어진 객체(30)에 존재하는 적어도 하나의 복사가 검사되는 경우 동작을 블록 2800에서 시작한다. 예를 들면, 상기한 바와 같이, 반복자(180)의 일 실시예는 주어진 객체(30)에 존재하는 복사들을 접근할 수 있는 지 그리고/또는 주어진 객체(30)에 존재하는 복사들이 객체에 관련된 저장 정책들을 만족시키는 범위를 결정하도록 구성될 수 있다.

[0302] 주어진 객체(30)의 복사들을 검사하는 것에 응답하여, 적어도 하나의 추가 복사가 생성될 필요가 있는 지가 결정될 수 있다(블록 2802). 예를 들면, 존재하는 복사들은 실패하거나 접근할 수 없어서, 최소수의 복사들보다 더 작게 존재하는 것으로 끝나게 된다. 대안으로, 존재하는 복사들의 상태가 적어도 하나의 복사에 대하여 부족할 수 있다. 이어서, 주어진 객체(30)의 추가 복사들을 저장하기 위한 기입 계획이 비트 저장 노드들(160)(블록 2804)의 현재 상태 정보에 따라 동적으로 결정된다. 이러한 기입 계획은 상기한 방법과 동일한 방법으로 또는 이 방법의 어떤 적당한 변경에 따라 결정될 수 있다. 일부의 실시예에서, 만일 어떤 추가 복사도 주어진 객체(30)용으로 생성할 필요가 없으면, 기입 계획은 결정될 수 없다는 것이 주목된다.

[0303] 그 후, 주어진 객체(30)의 복사들은 동적으로 결정된 기입 계획에 따라 적어도 하나의 비트 저장 노드(160)에 저장된다(블록 2806). 예를 들면, 반복자(180)는 상기한 바와 같이, 기입 계획에서 지정된 개별 비트

저장 노드들(160)로 지향된 비트 저장 객체 풋 동작들을 생성하도록 구성되거나, 주어진 객체(30)에 존재하는 복사들을 복사하기 위해 이 복사들을 저장하는 적어도 하나의 노드(160)를 기입 계획에서 지정된 적어도 하나의 노드(160)로 단순히 지향시킬 수 있다.

[0304] 바람직한 컴퓨터 시스템 실시예

[0305] 일부의 실시예에서, 상기한 일부의 방법 또는 기술은, 컴퓨터-접근 가능한 매체를 통하여 저장되거나 전송될 수 있는 프로그램 명령들 또는 데이터로서 실행될 수 있다는 것이 예측된다. 이러한 방법들 또는 기술들은 예를 들면 및 제한 없이 저장 클라이언트들(50), 웹 서비스 플랫폼(100), DFDD(110), 조정자(들)(120), 노드 피커(130), 키맵 인스턴스(들)(140), 비트 저장 노드(들)(160), 반복자(180) 및/또는 반복자 키맵(190)의 기능들을 포함할 수 있다. 이러한 방법들 또는 기술들은 도 6 내지 도 9, 도 13 내지 도 15, 도 20 내지 도 22 및 도 25 내지 도 28에 설명된 방법들의 일부, 그리고 이들의 어느 적당한 변경을 더 포함할 수 있다. 이러한 프로그램 명령들은 특정 방법 또는 상기한 방법의 일부와 같은 특정 계산 기능을 수행하고, 더 일반적인 동작 시스템 기능, 어플리케이션 기능, 및/또는 어느 다른 적당한 기능들을 제공하도록 실행될 수 있다. 일부의 실시예에서, 다른 실시예 들에서는 별개로서 상기한 구성 요소들 또는 방법들이 도시된 것들보다 더 작은 엔티티들로 집적화될 수 있고, 기능은 구성 요소들 또는 방법들을 통하여 상기한 분할과는 다르게 분할될 수 있다는 것이 주목된다.

[0306] 컴퓨터-접근 가능한 매체들을 갖는 컴퓨터 시스템의 바람직한 실시예가 도 28에 설명되어 있다. 이러한 시스템은 또한 노드로서 불릴 수 있다. 상기한 바와 같이, 일 실시예에서, 상기한 다양한 저장 시스템 구성 요소들의 일부의 기능은 다수의 노드를 통하여 분배되어, 주어진 구성 요소가 적어도 하나의 노드에 의해 실행되거나 수개의 노드들을 통하여 분할될 수 있도록 한다. 일 실시예에서, 노드는 단일 저장 서비스 시스템 구성 요소의 기능들을 독점적으로 실행할 수 있는 반면, 다른 실시예에서, 노드는 수개의 상이한 시스템 구성 요소들의 모두 또는 일부의 기능을 실행할 수 있다. 설명된 실시예에서, 컴퓨터 시스템(2900)은 입력/출력(I/O) 인터페이스(2930)를 통하여 시스템 메모리(2920)에 접속된 적어도 하나의 프로세서(2910)를 포함한다. 컴퓨터 시스템(2900)은 I/O 인터페이스(2930)에 접속된 네트워크 인터페이스(2940)를 더 포함한다.

[0307] 다양한 실시예에서, 컴퓨터 시스템(2900)은 하나의 프로세서(2910)를 포함하는 단일 프로세서 시스템이거나 다수의 프로세서(2910)(예를 들면, 2개, 4개, 8개, 또는 다른 적당한 수)를 포함하는 멀티 프로세서 시스템일 수 있다. 프로세서들(2910)은 명령들을 실행할 수 있는 어느 적당한 프로세서일 수 있다. 예를 들면, 다양한 실시예에서, 프로세서들(2910)은 x86, 파워PC, SPARC, MIPS ISA들, 또는 어떤 다른 적당한 ISA와 같은 다양한 명령 설정 아키텍쳐들(ISA들)의 일부를 실행하는 다목적 또는 내장형 프로세서일 수 있다. 프로세서들(2910) 각각은 동일한 ISA를 공통으로 실행하지는 필수적인 것은 아니다.

[0308] 시스템 메모리(2920)는 프로세서(2910)에 의해 접근 가능한 명령들 및 데이터를 저장하도록 구성될 수 있다. 다양한 실시예에서, 시스템 메모리(2920)는 에스램(SRAM), 에스디램(SDRAM), 비휘발성/플래쉬 메모리, 또는 어느 다른 방식 메모리와 같은 어느 적당한 메모리 기술을 이용하여 실행될 수 있다. 설명된 실시예에서, 상기에서 상세히 설명된 저장 서비스 시스템 구성 요소들의 일부와 같은 원하는 기능들 또는 및 다른 기능들을 실행하는 프로그램 명령들 및 데이터는, 코드(2925)로서 시스템 메모리(2920) 내에 저장되는 것으로 도시되어 있다.

[0309] 일 실시예에서, I/O 인터페이스(2930)는 프로세서(2910), 시스템 메모리(2920), 및 네트워크 인터페이스(2940) 또는 다른 주변 인터페이스들을 갖는 장치 내의 어느 주변 장치들 사이의 I/O 소통을 조정하도록 구성될 수 있다. 일부의 실시예에서, I/O 인터페이스(2930)는 어느 필요한 프로토콜, 타이밍 또는 다른 데이터 변형들을 수행하여 일 구성 요소(예를 들면, 시스템 메모리(2920))로부터의 데이터 신호들을 다른 구성 요소(예를 들면, 프로세서(2910))에 의해 사용하기에 적합한 포맷으로 변환할 수 있다. 일부의 실시예에서, I/O 인터페이스(2930)는 예를 들면, 주변 소자 상호 연결(PCI) 버스 표준 또는 유니버셜 직렬 버스(USB) 표준의 변형과 같은 다양한 방식의 주변 버스들을 통하여 부착된 장치들용 지지를 포함할 수 있다. 일부의 실시예에서, I/O 인터페이스(2930)의 기능은 예를 들면 노스 브리지(north bridge) 또는 사우스 브리지(south bridge)와 같은 2개 이상의 분리형 소자들로 분리될 수 있다. 또한, 일부의 실시예에서 시스템 메모리(2920)에서의 인터페이스와 같은 I/O 인터페이스(2930)의 기능의 일부 또는 모두는 프로세서(2910)에 직접 포함될 수 있다.

[0310] 네트워크 인터페이스(2940)는 데이터가 예를 들면, 컴퓨터 시스템(2900)과 다른 컴퓨터 시스템과 같은 네트워크에 부착된 다른 장치들 사이에 교환될 수 있도록 구성될 있다. 다양한 실시예에서, 네트워크 인터페이스(2940)는 예를 들면, 어느 적당한 방식의 이더넷 네트워크와 같은 유선 또는 무선 일반 데이터 네트워크들을

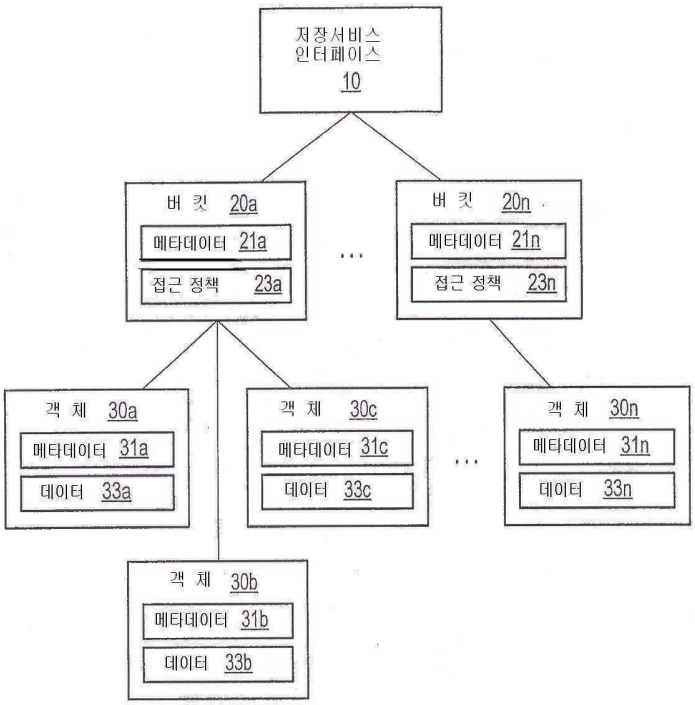
통하여, 아날로그 음성 네트워크 또는 디지털 섬유 통신 네트워크와 같은 통신/전화통신 네트워크를 통하여, 섬유 채널 SAN들과 같은 저장 영역 네트워크를 통하여, 또는 어느 적당한 방식의 네트워크 및/또는 프로토콜을 통한 통신을 지지할 수 있다.

[0311] 일부의 실시예에서, 시스템 메모리(2920)는 상기한 바와 같은 프로그램 명령들 및 데이터를 저장하도록 구성된 컴퓨터-접근 가능한 매체의 일 실시예일 수 있다. 하지만, 다른 실시예들에서, 프로그램 명령들 및/또는 데이터는 다른 방식의 컴퓨터-접근 가능한 매체들로 수신, 전송, 또는 저장된다. 일반적으로 말해, 컴퓨터-접근 가능한 매체는 I/O 인터페이스(2930)를 통하여 컴퓨터 시스템(2900)에 접속된 자기 또는 광학 매체, 예를 들면 디스크 또는 CD/DVD-ROM과 같은 저장 매체들 또는 메모리 매체들을 포함할 수 있다. 컴퓨터-접근 가능한 매체는 또한 시스템 메모리(2920) 또는 다른 방식의 메모리로서 컴퓨터 시스템(2900)의 일부 실시예에 포함될 수 있는 램(예를 들면, 에스디램, DDR 에스디램, 알디램, 에스램, 등), 롬, 등과 같은 어느 휘발성 또는 비휘발성 매체들을 포함할 수 있다. 컴퓨터-접근 가능한 매체를 통하여 저장된 프로그램 명령들 및 데이터는 전송 매체들 또는 전기, 전자기, 또는 디지털 신호들에 의해 전송될 수 있고, 네트워크 인터페이스(2940)를 통하여 실행되는 있는 것과 같이 네트워크 및/또는 무선 링크와 같은 통신 매체를 통하여 전송될 수 있다.

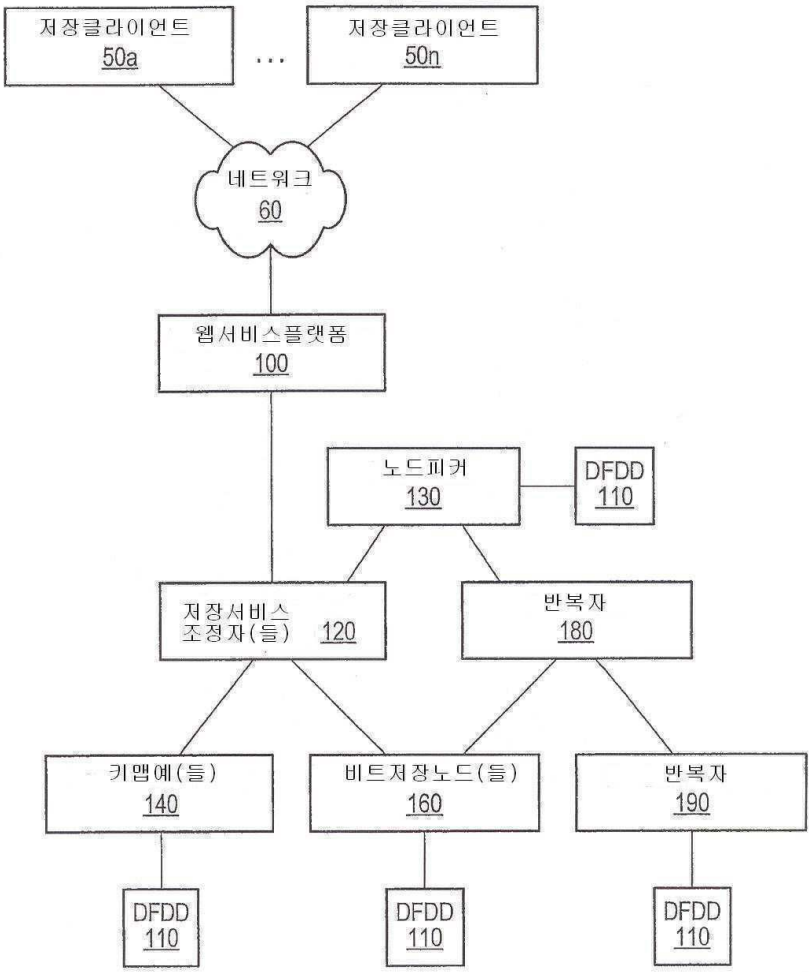
[0312] 비록 실시예들이 상당히 상세하게 설명되어 있지만, 발명의 요지를 벗어나지 않는 범위 내에서 여러 가지 개량, 변경, 대체 또는 부가하여 실시할 수 있는 것임은 본 발명이 속하는 분야에서 통상의 지식을 가진 자라면 누구든지 다양한 변형이 가능할 것이다.

도면

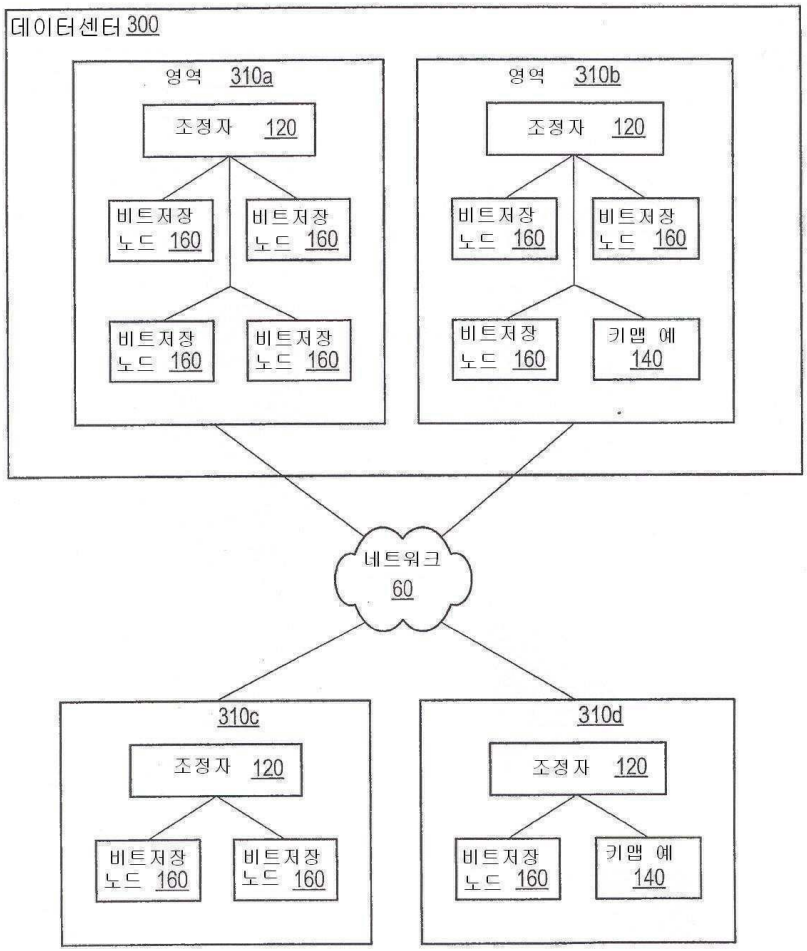
도면1



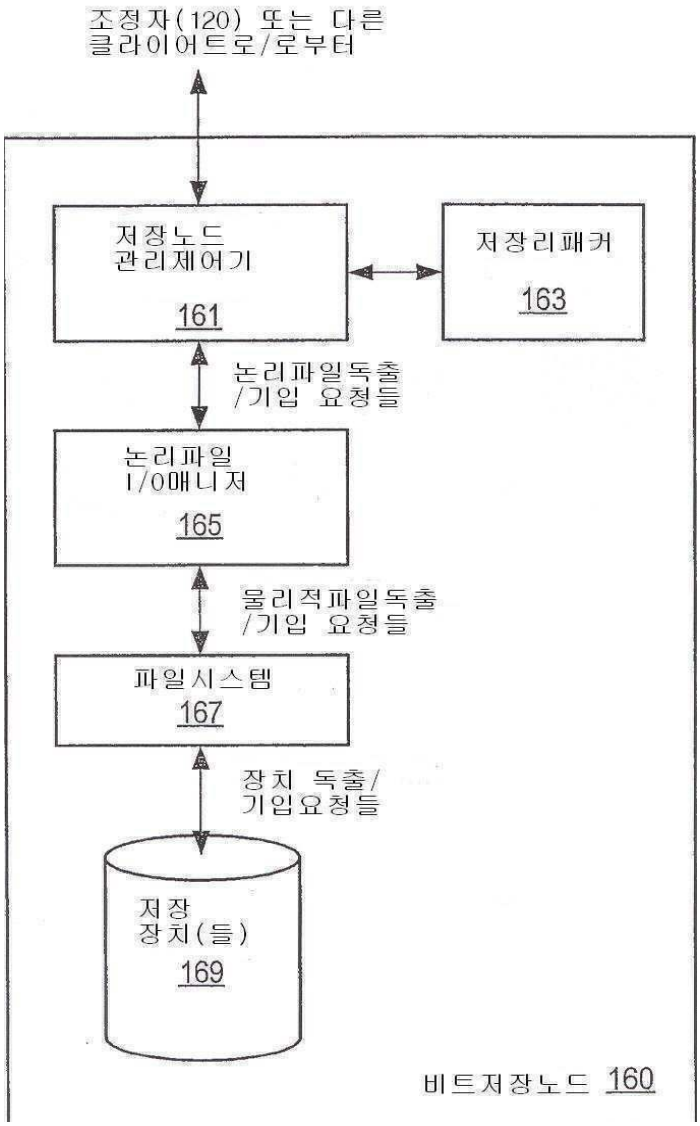
도면2


저장클라이언트
50a
...
저장클라이언트
50n
네트워크
60
웹서비스플랫폼
100
노드피커
130
DFDD
110
저장서비스
조정자(들)
120
반복자
180
키맵예(들)
140
비트저장노드(들)
160
반복자
190
DFDD
110
DFDD
110
DFDD
110

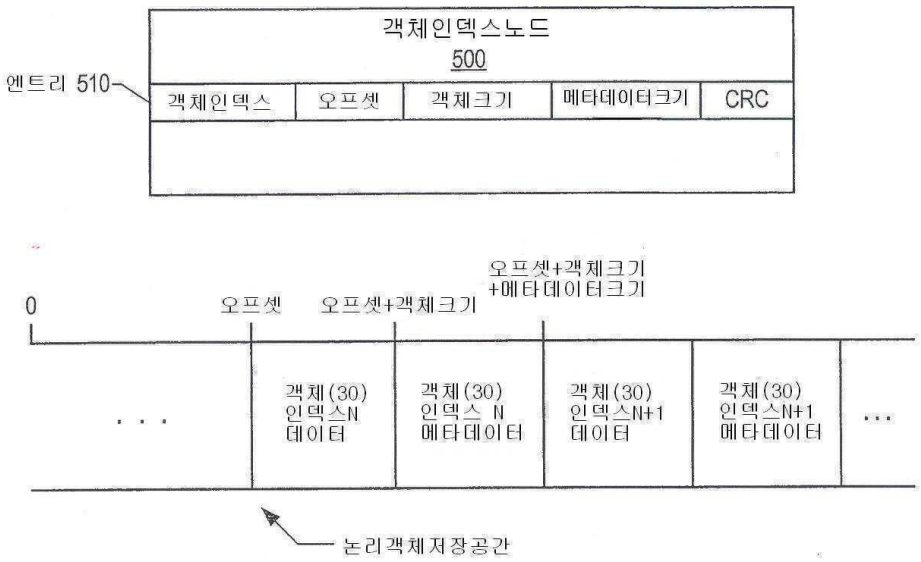
도면3


데이터센터 300
영역 310a
조정자 120
비트저장 노드 160
비트저장 노드 160
비트저장 노드 160
비트저장 노드 160
영역 310b
조정자 120
비트저장 노드 160
비트저장 노드 160
비트저장 노드 160
키맵 예 140
네트워크 60
310c
조정자 120
비트저장 노드 160
비트저장 노드 160
310d
조정자 120
비트저장 노드 160
키맵 예 140

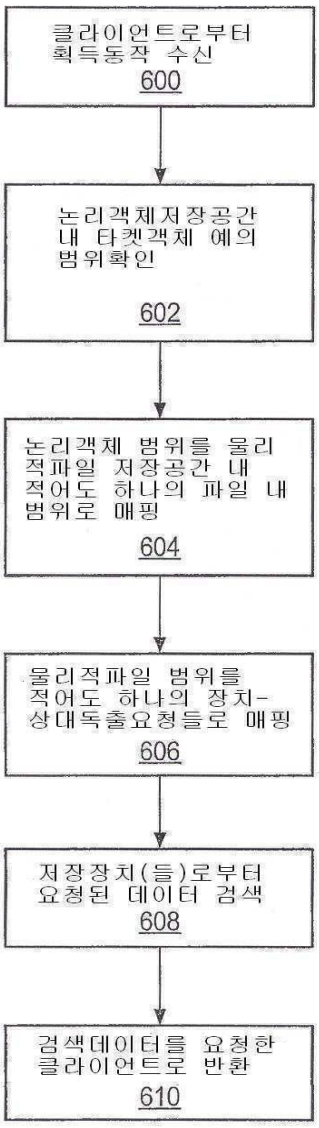
도면4


조정자(120) 또는 다른
클라이어트로/로부터
저장노드
관리제어기
161
저장리패커
163
논리파일독출
/기입 요청들
논리파일
I/O매니저
165
물리적파일독출
/기입 요청들
파일시스템
167
장치 독출/
기입요청들
저장
장치(들)
169
비트저장노드 160

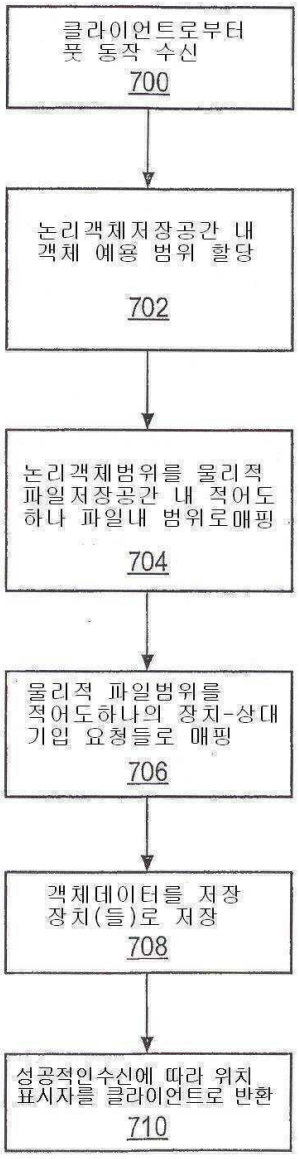
도면5


객체인덱스노드
500
엔트리 510
객체인덱스
오프셋
객체크기
메타데이터크기
CRC
0
오프셋
오프셋+객체크기
오프셋+객체크기
+메타데이터크기
객체(30)
인덱스N
데이터
객체(30)
인덱스 N
메타데이터
객체(30)
인덱스N+1
데이터
객체(30)
인덱스N+1
메타데이터
논리객체저장공간

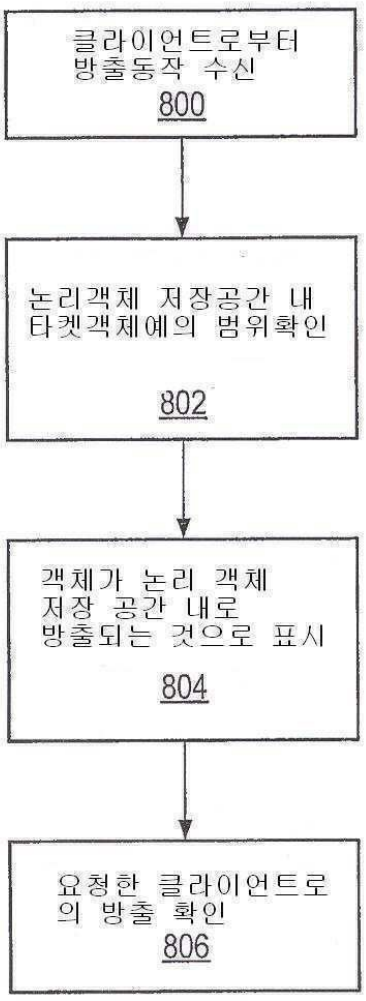
도면6


클라이언트로부터
획득동작 수신
600
논리객체저장공간
내 타겟객체 예의
범위확인
602
논리객체 범위를 물리
적파일 저장공간 내
적어도 하나의 파일 내
범위로 매핑
604
물리적파일 범위를
적어도 하나의 장치-
상대독출요청들로 매핑
606
저장장치(들)로부터
요청된 데이터 검색
608
검색데이터를 요청한
클라이언트로 반환
610

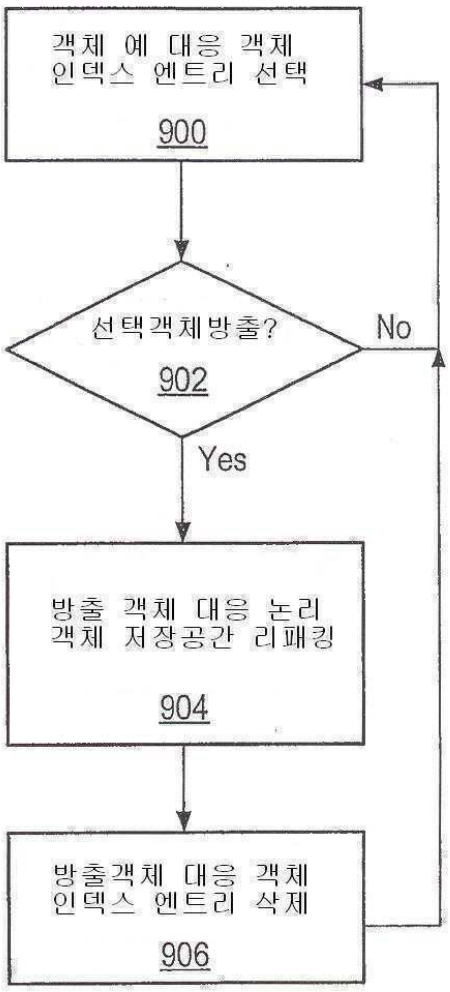
도면7


클라이언트로부터
풋 동작 수신
700
논리객체저장공간 내
객체 예용 범위 할당
702
논리객체범위를 물리적
파일저장공간 내 적어도
하나 파일내 범위로매핑
704
물리적 파일범위를
적어도하나의 장치-상대
기입 요청들로 매핑
706
객체데이터를 저장
장치(들)로 저장
708
성공적인수신에 따라 위치
표시자를 클라이언트로 반환
710

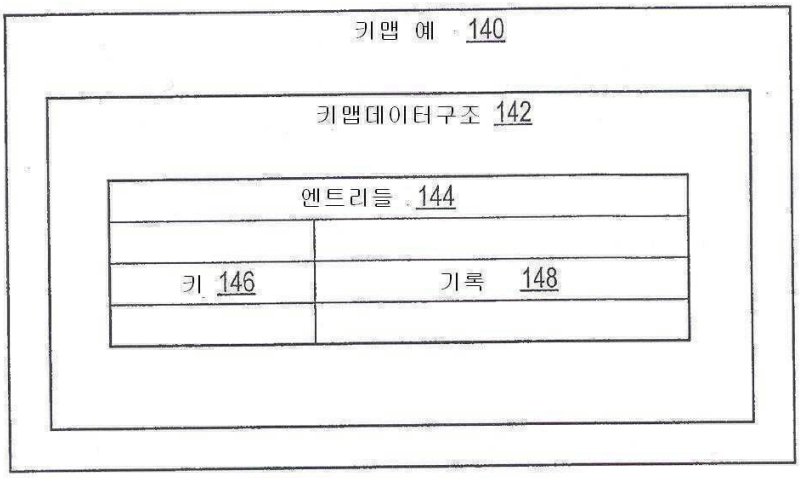
도면8


클라이언트로부터
방출동작 수신
800
논리객체 저장공간 내
타겟객체예의 범위확인
802
객체가 논리 객체
저장 공간 내로
방출되는 것으로 표시
804
요청한 클라이언트로
의 방출 확인
806

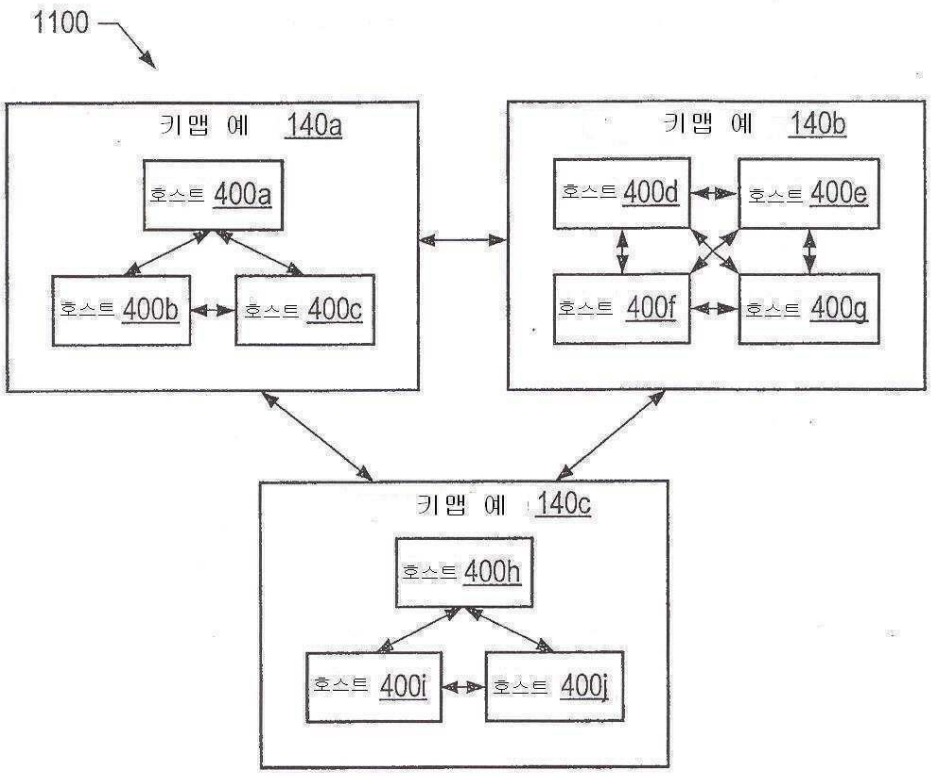
도면9


객체 예 대응 객체
인덱스 엔트리 선택
900
선택객체방출?
902
No
Yes
방출 객체 대응 논리
객체 저장공간 리패킹
904
방출객체 대응 객체
인덱스 엔트리 삭제
906

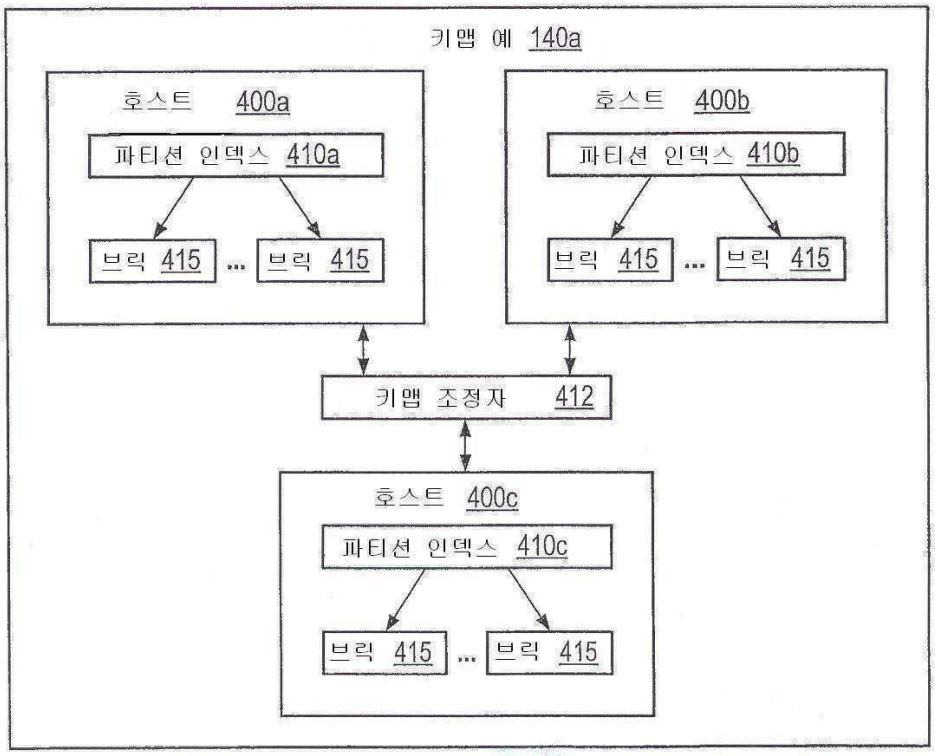
도면10


키맵 예 140
키맵데이터구조 142
엔트리들 144
키 146
기록 148

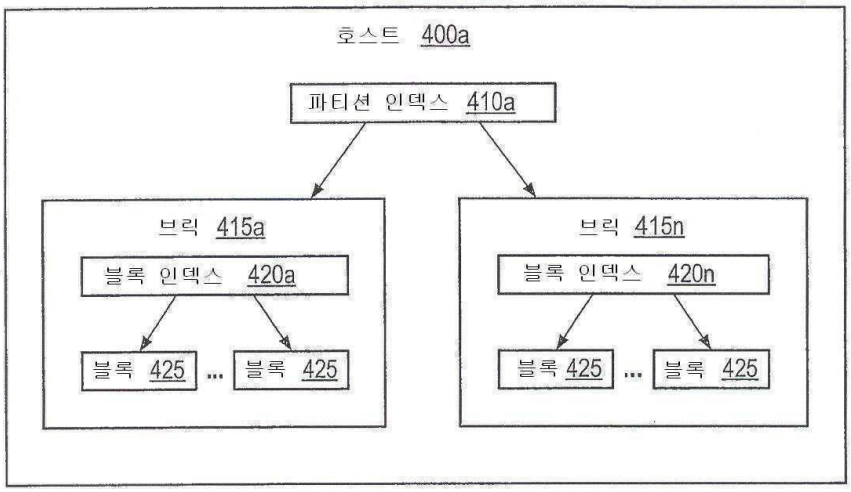
도면11a


1100
키맵 예 140a
호스트 400a
호스트 400b
호스트 400c
키맵 예 140b
호스트 400d
호스트 400e
호스트 400f
호스트 400g
키맵 예 140c
호스트 400h
호스트 400i
호스트 400j

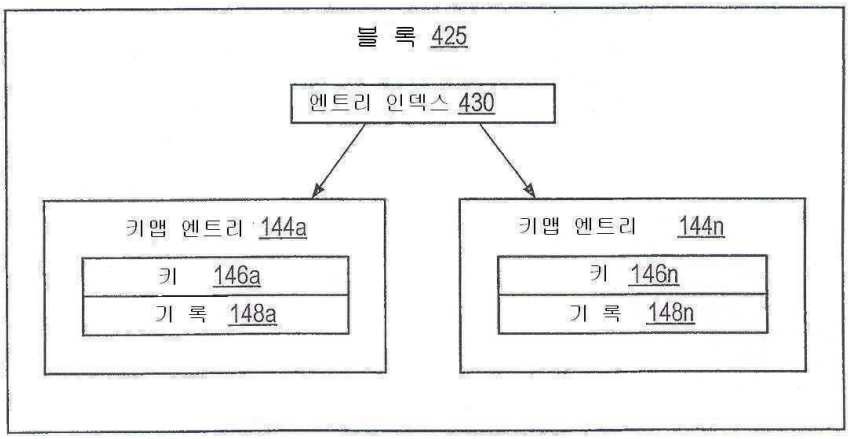
도면11b


키맵 예 140a
호스트 400a
파티션 인덱스 410a
브릭 415
...
브릭 415
호스트 400b
파티션 인덱스 410b
브릭 415
...
브릭 415
키맵 조정자 412
호스트 400c
파티션 인덱스 410c
브릭 415
...
브릭 415

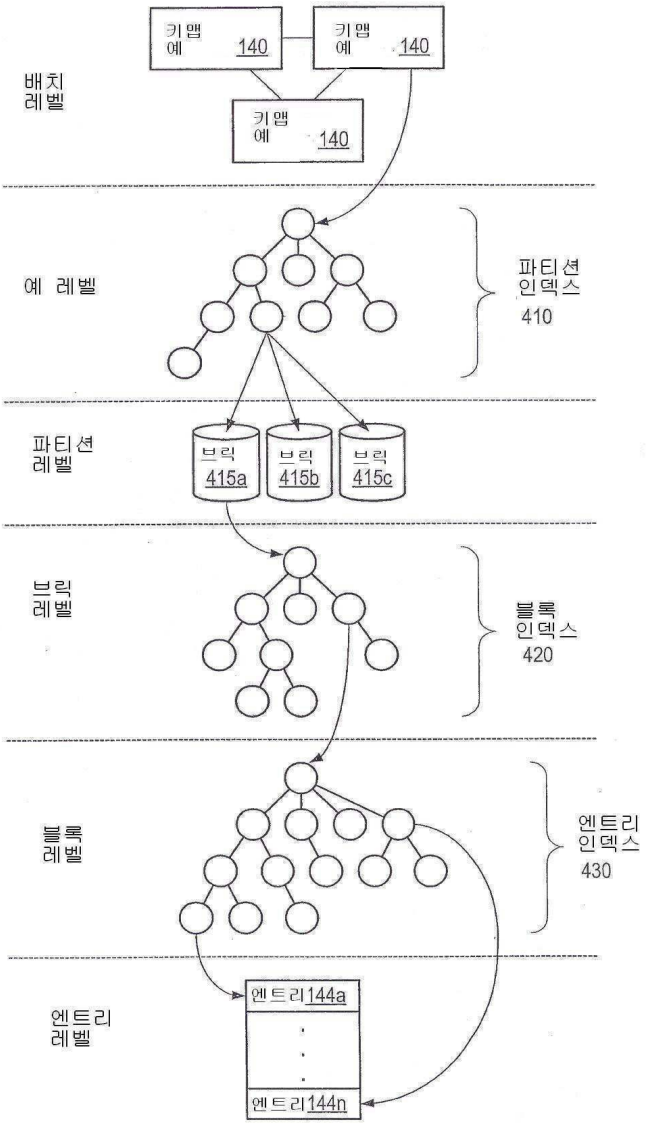
도면11c


호스트 400a
파티션 인덱스 410a
브릭 415a
블록 인덱스 420a
블록 425
...
블록 425
브릭 415n
블록 인덱스 420n
블록 425
...
블록 425

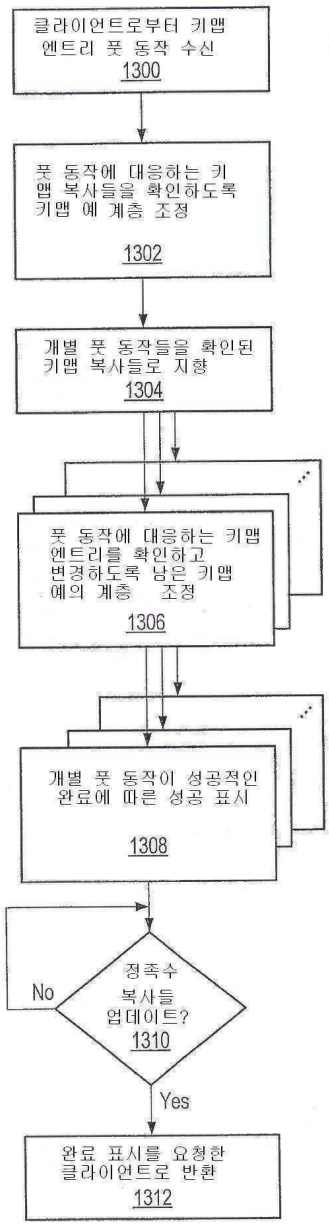
도면11d


블 록 425
엔트리 인덱스 430
키맵 엔트리 144a
키 146a
기 록 148a
키맵 엔트리 144n
키 146n
기 록 148n

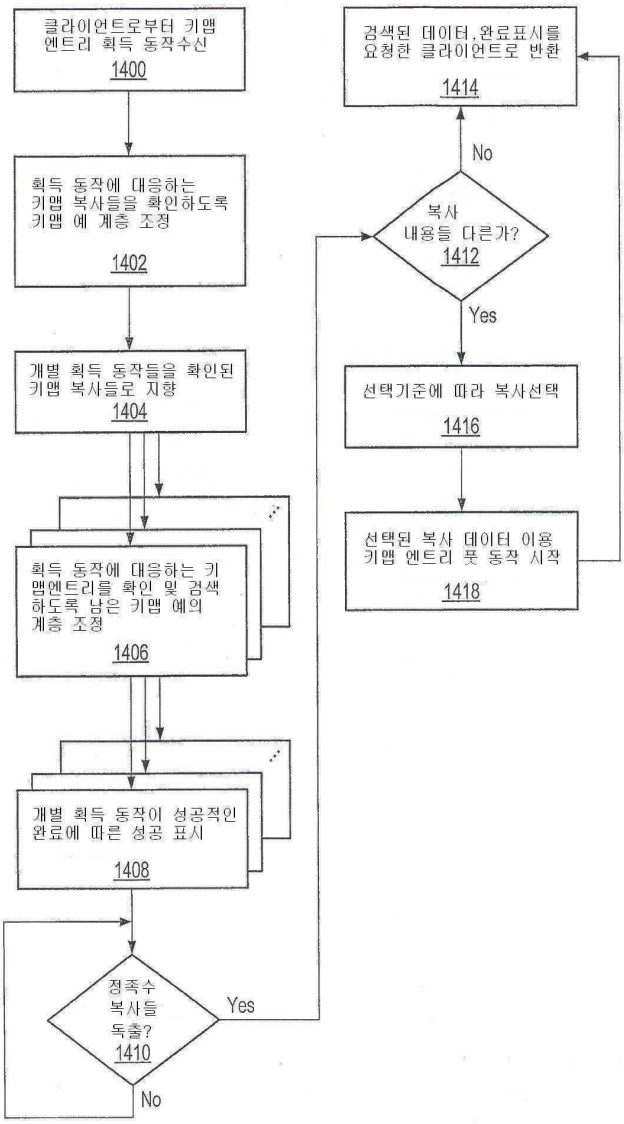
도면12


배치
레벨
카맵
예 140
카맵
예 140
카맵
예 140
예 레벨
파티션
인덱스
410
파티션
레벨
브릭
415a
브릭
415b
브릭
415c
브릭
레벨
블록
인덱스
420
블록
레벨
엔트리
인덱스
430
엔트리
레벨
엔트리144a
엔트리144n

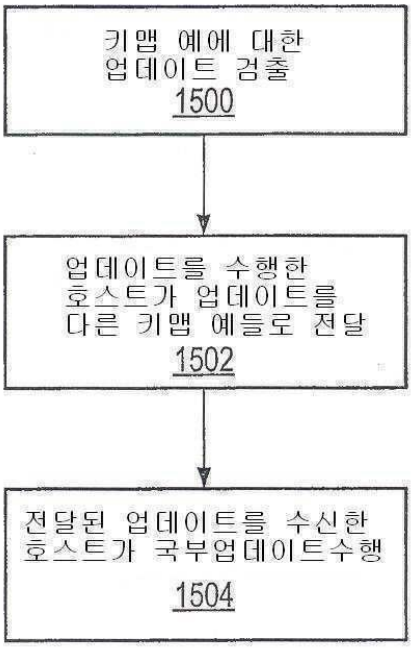
도면13


클라이언트로부터 키맵 엔트리 풋 동작 수신
1300
풋 동작에 대응하는 키맵 복사들을 확인하도록 키맵 예 계층 조정
1302
개별 풋 동작들을 확인된 키맵 복사들로 지향
1304
풋 동작에 대응하는 키맵 엔트리를 확인하고 변경하도록 남은 키맵 예의 계층 조정
1306
개별 풋 동작이 성공적인 완료에 따른 성공 표시
1308
정족수 복사들 업데이트?
1310
No
Yes
완료 표시를 요청한 클라이언트로 반환
1312

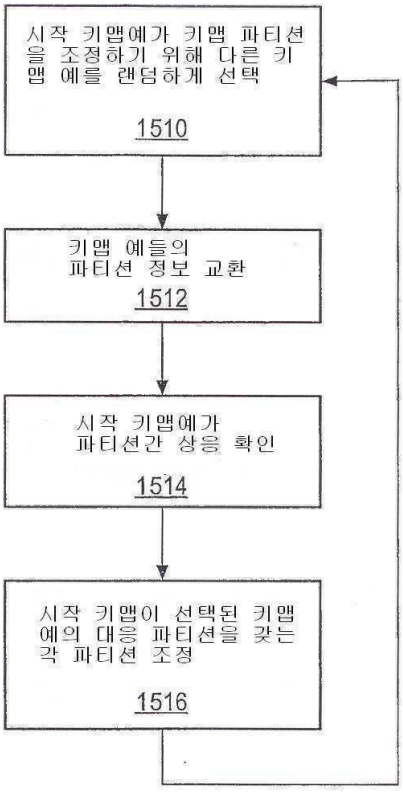
도면14


클라이언트로부터 키맵 엔트리 획득 동작수신
1400
획득 동작에 대응하는 키맵 복사들을 확인하도록 키맵 에 계층 조정
1402
개별 획득 동작들을 확인된 키맵 복사들로 지향
1404
획득 동작에 대응하는 키맵엔트리를 확인 및 검색하도록 남은 키맵 예의 계층 조정
1406
개별 획득 동작이 성공적인 완료에 따른 성공 표시
1408
정족수 복사들 독출?
1410
Yes
No
복사 내용들 다른가?
1412
No
Yes
선택기준에 따라 복사선택
1416
선택된 복사 데이터 이용 키맵 엔트리 풋 동작 시작
1418
검색된 데이터, 완료표시를 요청한 클라이언트로 반환
1414

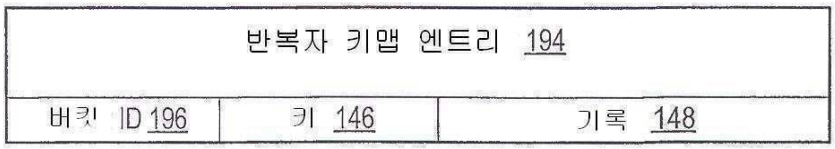
도면15a


키맵 예에 대한
업데이트 검출
1500
업데이트를 수행한
호스트가 업데이트를
다른 키맵 예들로 전달
1502
전달된 업데이트를 수신한
호스트가 국부업데이트수행
1504

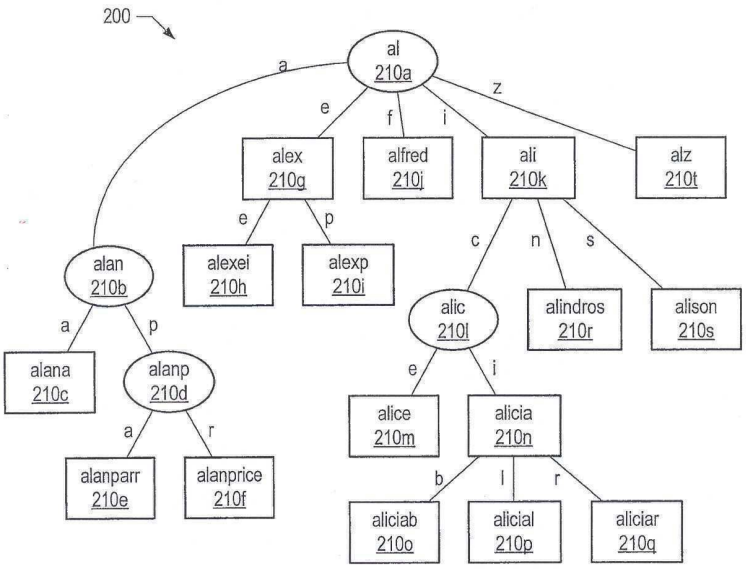
도면15b


시작 키맵예가 키맵 파티션
을 조정하기 위해 다른 키
맵 예를 랜덤하게 선택
1510
키맵 예들의
파티션 정보 교환
1512
시작 키맵예가
파티션간 상응 확인
1514
시작 키맵이 선택된 키맵
예의 대응 파티션을 갖는
각 파티션 조정
1516

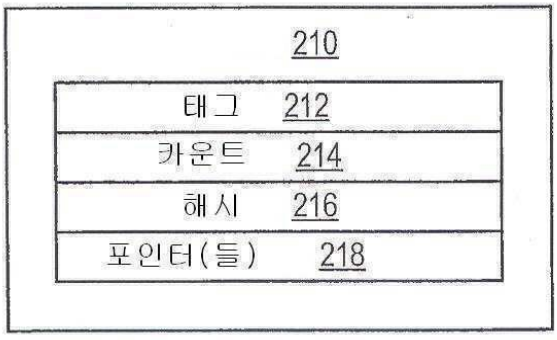
도면16



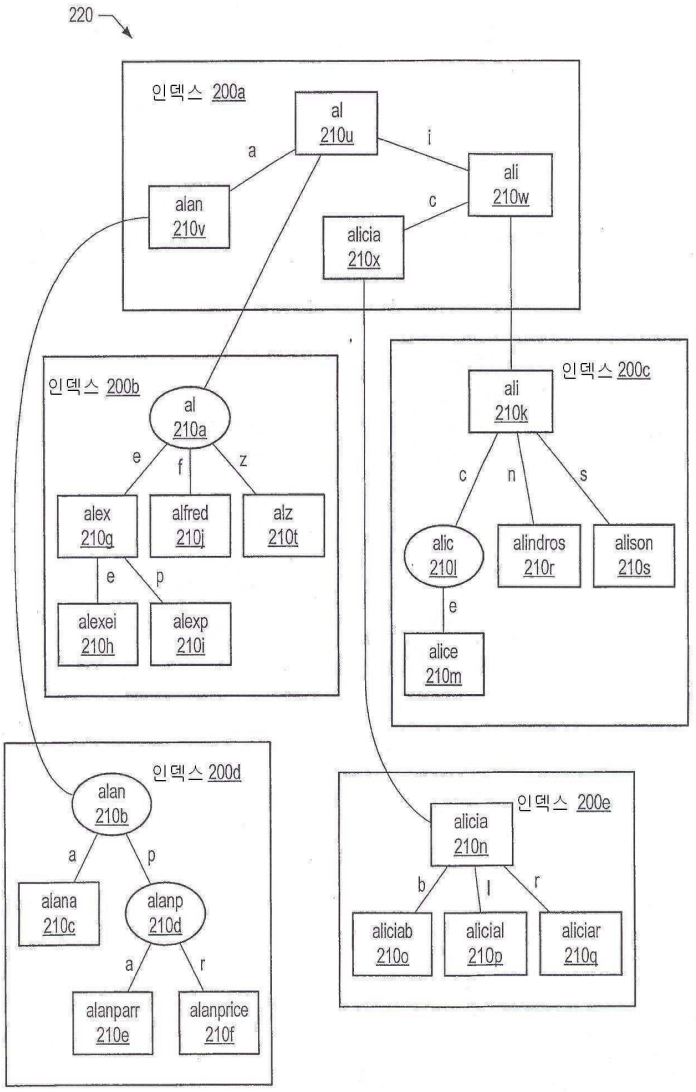
도면17



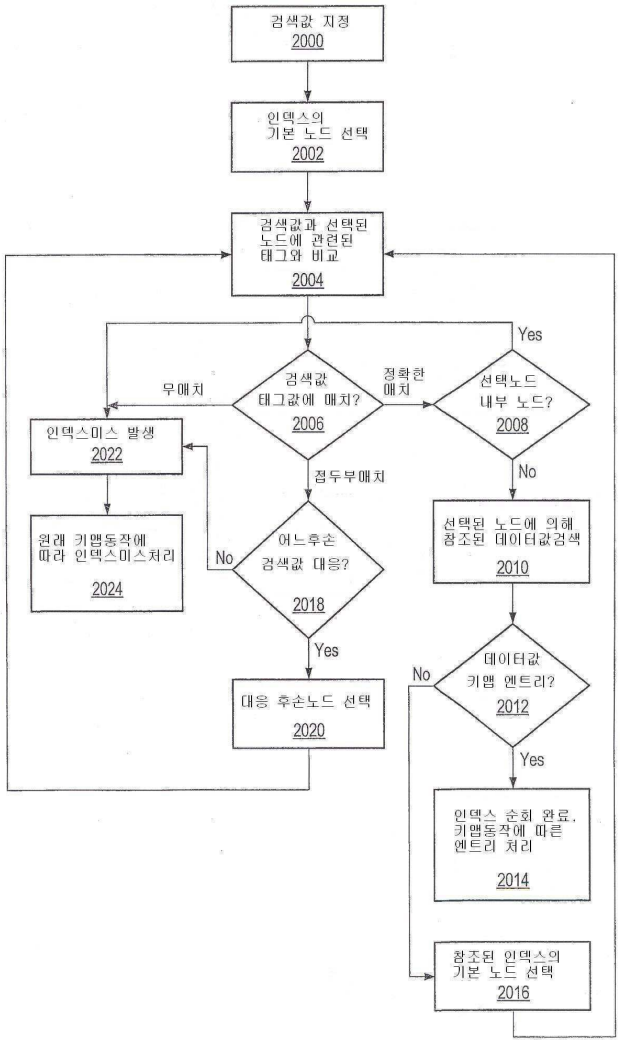
도면18



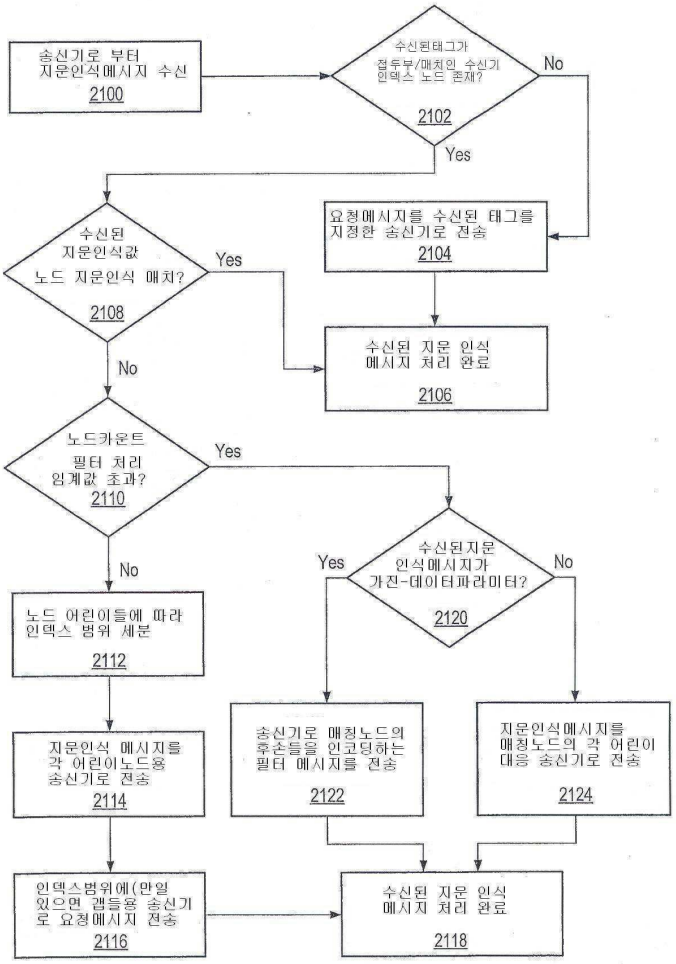
도면19


220
인덱스 200a
al
210u
a
i
ali
210w
alan
210v
c
alicia
210x
인덱스 200b
al
210a
e
f
z
alex
210g
alfred
210j
alz
210t
e
p
alexei
210h
alexp
210i
인덱스 200c
ali
210k
c
n
s
alic
210l
alindros
210r
alison
210s
e
alice
210m
인덱스 200d
alan
210b
a
p
alana
210c
alanp
210d
a
r
alanparr
210e
alanprice
210f
인덱스 200e
alicia
210n
b
l
r
aliciab
210o
alicial
210p
aliciar
210q

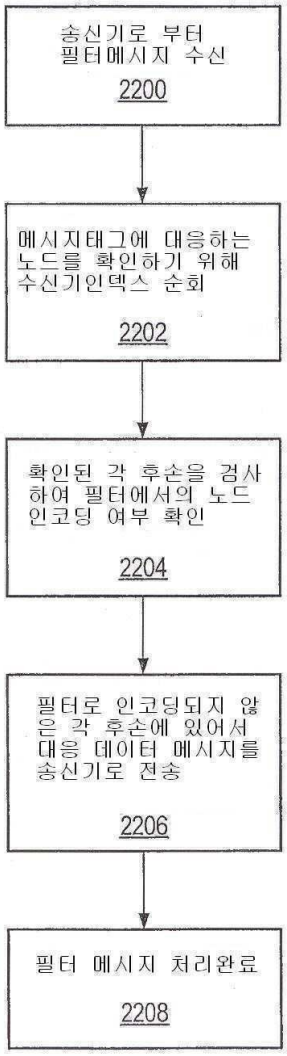
도면20


검색값 지정
2000
인덱스의
기본 노드 선택
2002
검색값과 선택된
노드에 관련된
태그와 비교
2004
검색값
태그값에 매치?
2006
무매치
정확한
매치
점두부매치
선택노드
내부 노드?
2008
Yes
No
인덱스미스 발생
2022
원래 키맵동작에
따라 인덱스미스처리
2024
어느후손
검색값 대응?
2018
No
Yes
대응 후손노드 선택
2020
선택된 노드에 의해
참조된 데이터값검색
2010
데이터값
키맵 엔트리?
2012
No
Yes
인덱스 순회 완료,
키맵동작에 따른
엔트리 처리
2014
참조된 인덱스의
기본 노드 선택
2016

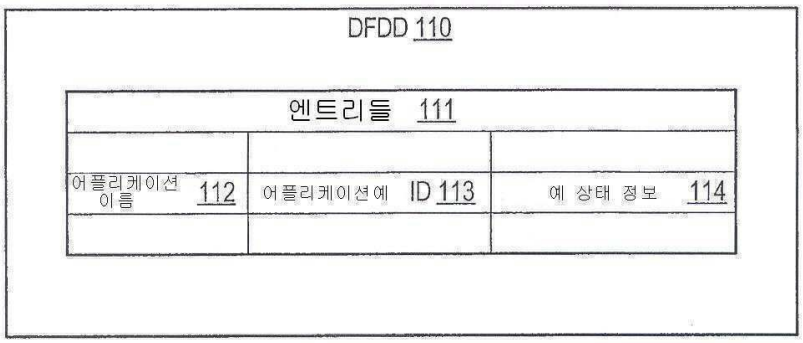
도면21


송신기로 부터
지문인식메시지 수신
2100
수신된태그가
접두부/매칭된 수신기
인덱스 노드 존재?
2102
No
Yes
요청메시지를 수신된 태그를
지정한 송신기로 전송
2104
수신된
지문인식값
노드 지문인식 매치?
2108
Yes
No
수신된 지문 인식
메시지 처리 완료
2106
노드카운트
필터 처리
임계값 초과?
2110
Yes
No
수신된지문
인식메시지가
가짜-데이터파라미터?
2120
Yes
No
노드 어린이들에 따라
인덱스 범위 세분
2112
송신기로 매칭노드의
후손들을 인코딩하는
필터 메시지를 전송
2122
지문인식메시지를
매칭노드의 각 어린이
대응 송신기로 전송
2124
지문인식 메시지를
각 어린이노드용
송신기로 전송
2114
인덱스범위에(만일
있으면 갭들용 송신기
로 요청메시지 전송
2116
수신된 지문 인식
메시지 처리 완료
2118

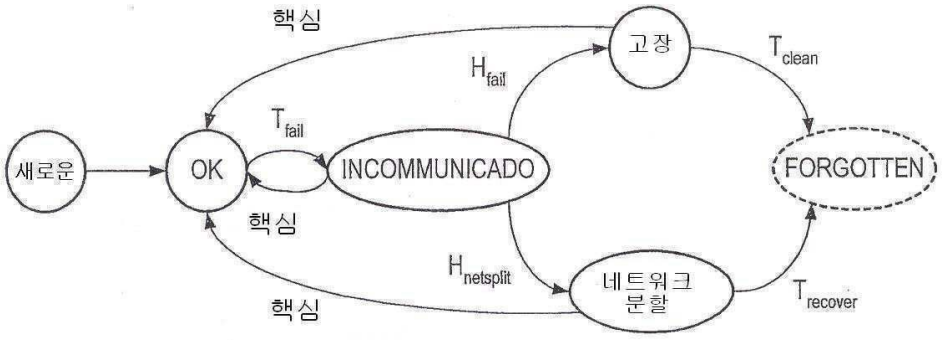
도면22


송신기로 부터
필터메시지 수신
2200
메시지태그에 대응하는
노드를 확인하기 위해
수신기인덱스 순회
2202
확인된 각 후손을 검사
하여 필터에서의 노드
인코딩 여부 확인
2204
필터로 인코딩되지 않
은 각 후손에 있어서
대응 데이터 메시지를
송신기로 전송
2206
필터 메시지 처리완료
2208

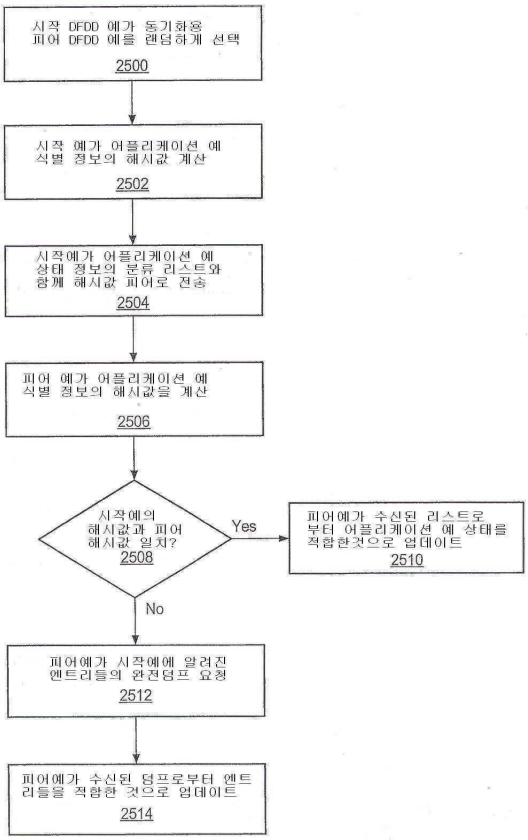
도면23


DFDD 110
엔트리들 111
어플리케이션 이름 112
어플리케이션ID 113
ID 상태 정보 114

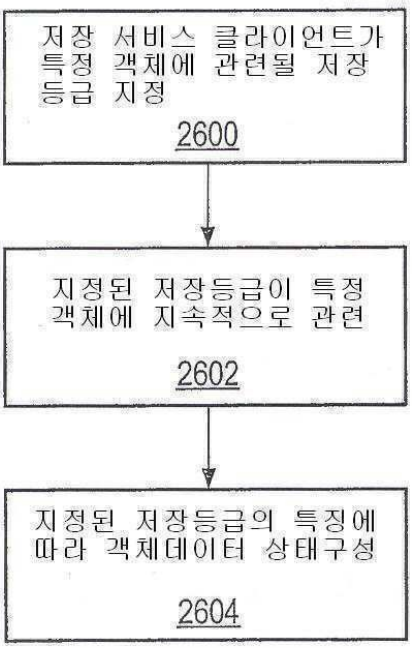
도면24


핵심
고장
T_{clean}
H_{fail}
T_{fail}
새로운
OK
INCOMMUNICADO
FORGOTTEN
핵심
$H_{netsplit}$
네트워크 분할
$T_{recover}$
핵심

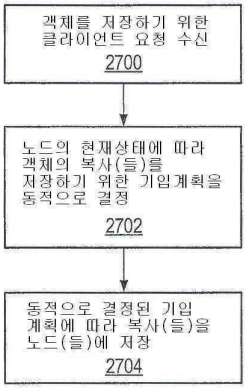
도면25


시작 DFDD 예가 동기화용 피어 DFDD 예를 랜덤하게 선택
2500
시작 예가 어플리케이션 예 식별 정보의 해시값 계산
2502
시작예가 어플리케이션 예 상태 정보의 분류 리스트와 함께 해시값 피어로 전송
2504
피어 예가 어플리케이션 예 식별 정보의 해시값을 계산
2506
시작예의 해시값과 피어 해시값 일치?
2508
Yes
피어예가 수신된 리스트로부터 어플리케이션 예 상태를 적합한것으로 업데이트
2510
No
피어예가 시작예에 알려진 엔트리들의 완전덤프 요청
2512
피어예가 수신된 덤프로부터 엔트리들을 적합한 것으로 업데이트
2514

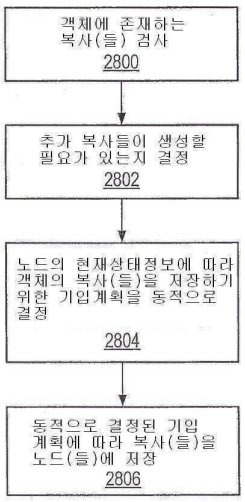
도면26


저장 서비스 클라이언트가 특정 객체에 관련될 저장 등급 지정
2600
지정된 저장등급이 특정 객체에 지속적으로 관련
2602
지정된 저장등급의 특징에 따라 객체데이터 상태구성
2604

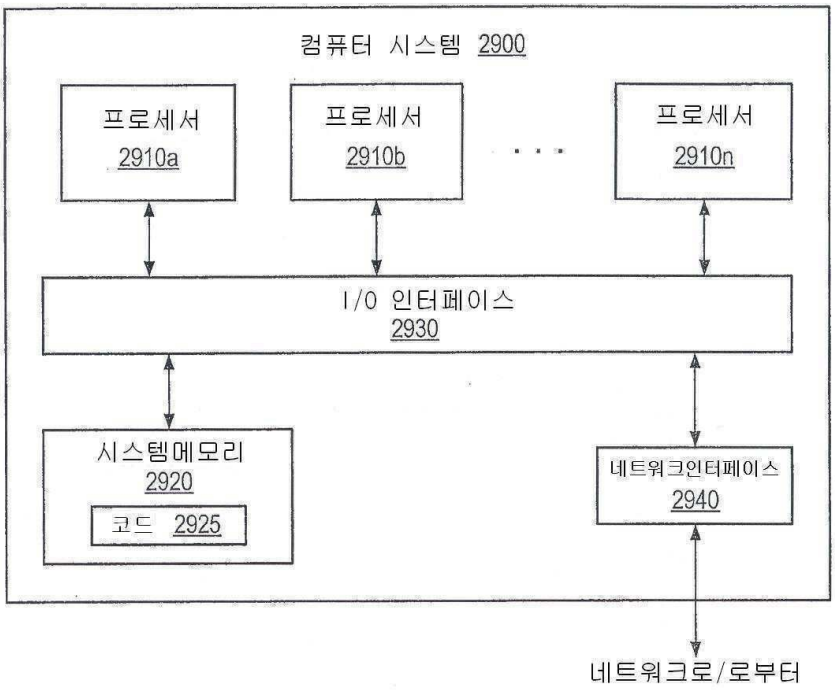
도면27


객체를 저장하기 위한 클라이언트 요청 수신
2700
노드의 현재상태에 따라 객체의 복사(들)를 저장하기 위한 기입계획을 동적으로 결정
2702
동적으로 결정된 기입 계획에 따라 복사(들)을 노드(들)에 저장
2704

도면28


객체에 존재하는
복사(들) 검사
2800
추가 복사들이 생성할
필요가 있는지 결정
2802
노드의 현재상태정보에 따라
객체의 복사(들)을 저장하기
위한 기입계획을 동적으로
결정
2804
동적으로 결정된 기입
계획에 따라 복사(들)을
노드(들)에 저장
2806

도면29


컴퓨터 시스템 2900
프로세서
2910a
프로세서
2910b
프로세서
2910n
I/O 인터페이스
2930
시스템메모리
2920
코드 2925
네트워크인터페이스
2940
네트워크로/로부터