(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0093548
(43) 공개일자 2020년08월05일

(51) 국제특허분류(Int. Cl.)
G06F 16/178 (2019.01) *G06F 16/11* (2019.01)
G06F 16/17 (2019.01) *G06F 16/176* (2019.01)
(52) CPC특허분류
G06F 16/178 (2019.01)
G06F 16/11 (2019.01)
(21) 출원번호 10-2020-7014986
(22) 출원일자(국제) 2018년12월13일
심사청구일자 2020년05월26일
(85) 번역문제출일자 2020년05월26일
(86) 국제출원번호 PCT/US2018/065347
(87) 국제공개번호 WO 2019/133269
국제공개일자 2019년07월04일
(30) 우선권주장
62/611,473 2017년12월28일 미국(US)
15/858,207 2017년12월29일 미국(US)

(71) 출원인
드롭박스, 인크.
미국 94158 캘리포니아주 샌프란시스코 스위트 200 오웬스 스트리트 1800
(72) 발명자
자야카르, 산자이
미국 94107 캘리포니아주 샌프란시스코 브라난 스트리트 333 드롭박스 인크.
(74) 대리인
양영준, 김연송, 백만기

전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **디프 값들의 효율적인 전파**

(57) 요 약

개시된 기술은 트리 데이터 구조체 내의 노드에 대한 수정을 검출하도록 구성된 시스템에 관한 것이다. 노드는 파일명뿐만 아니라 콘텐트 관리 서비스에 의해 관리되는 콘텐트 아이템과 연관된다. 시스템은 파일명 및 구분자를 파일명 어레이에 첨부하고, 파일명 어레이 내의 파일명의 위치를 결정하며, 파일명의 위치를 노드 내에 저장할 수 있다.

(52) CPC특허분류
G06F 16/1734 (2019.01)
G06F 16/1767 (2019.01)

명 세 서

청구범위

청구항 1

컴퓨터-구현형 방법으로서,

트리 데이터 구조체 내에 노드를 추가하는 단계;

상기 노드에 대한 디프(diff) 값을 계산하는 단계;

상기 노드의 상기 디프 값 및 부모 노드의 현재 디프 값에 기초하여 상기 노드의 부모에 대한 새로운 디프 값을 계산하는 단계; 및

상기 부모에 대한 상기 새로운 디프 값을 상기 부모 노드 내에 저장하는 단계를 포함하는, 컴퓨터-구현형 방법.

청구항 2

청구항 1에 있어서,

상기 부모 노드가 조상을 갖는다는 것을 결정하는 단계;

상기 조상의 오래된 디프 값 및 상기 부모에 대한 상기 새로운 디프 값에 기초하여 상기 조상에 대한 새로운 디프 값을 계산하는 단계; 및

상기 조상에 대한 상기 새로운 디프 값을 상기 조상 노드 내에 저장하는 단계를 더 포함하는, 컴퓨터-구현형 방법.

청구항 3

청구항 1에 있어서, 상기 노드의 상기 부모에 대한 상기 새로운 디프 값을 계산하는 단계는 상기 부모 노드의 상기 현재 디프 값과 상기 노드의 상기 디프 값의 배타적 논리합(XOR) 연산에 기초하는, 컴퓨터-구현형 방법.

청구항 4

청구항 1에 있어서, 상기 트리 데이터 구조체는 로컬 트리, 원격 트리, 또는 싱크(sync) 트리 중 하나인, 컴퓨터-구현형 방법.

청구항 5

청구항 1에 있어서, 상기 노드에 대한 상기 디프 값은 상기 노드의 해시(hash)를 사용하여 계산되는, 컴퓨터-구현형 방법.

청구항 6

청구항 1에 있어서,

상기 트리 데이터 구조체 내의 상기 노드의 상기 디프 값과 상이한 트리 데이터 구조체 내의 대응하는 노드에 대한 제 2 디프 값을 비교하는 단계; 및

상기 비교하는 단계에 기초하여 상기 트리 데이터 구조체와 상기 상이한 트리 데이트 구조체 사이의 차이를 식별하는 단계를 더 포함하는, 컴퓨터-구현형 방법.

청구항 7

청구항 6에 있어서,

상기 차이에 기초하여, 서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 세트를 생성하는 단계; 및

상기 동작들의 세트의 실행을 관리하는 단계를 더 포함하는, 컴퓨터-구현형 방법.

청구항 8

청구항 7에 있어서, 상기 동작들의 세트를 콘텐트 관리 시스템으로 송신하는 단계를 더 포함하는, 컴퓨터-구현형 방법.

청구항 9

청구항 1에 있어서, 상기 노드는 콘텐트 관리 서비스와 연관된 콘텐트 아이템을 나타내는, 컴퓨터-구현형 방법.

청구항 10

명령어들을 포함하는 비-일시적인 컴퓨터-판독가능 매체로서, 상기 명령어들은, 컴퓨팅 시스템에 의해 실행될 때, 상기 컴퓨팅 시스템으로 하여금:

싱크 트리 내의 노드에 대한 디프 값을 계산하되, 상기 싱크 트리는 서버 상태와 파일 시스템 상태 사이의 알려진 싱크된 상태를 나타내며;

상기 노드의 상기 디프 값 및 부모 노드의 현재 디프 값에 기초하여 상기 노드의 부모에 대한 새로운 디프 값을 계산하고; 및

상기 부모에 대한 상기 새로운 디프 값을 상기 부모 노드 내에 저장하게끔 하는, 비-일시적인 컴퓨터-판독가능 매체.

청구항 11

청구항 10에 있어서, 상기 명령어들은 추가로 상기 컴퓨팅 시스템으로 하여금 상기 노드를 상기 싱크 트리 내에 추가하게끔 하는, 비-일시적인 컴퓨터-판독가능 매체.

청구항 12

청구항 10에 있어서, 상기 명령어들은 추가로 상기 컴퓨팅 시스템으로 하여금:

상기 싱크 트리 내의 상기 노드의 상기 디프 값과 원격 트리 내의 대응하는 노드에 대한 제 2 디프 값을 비교하며;

상기 비교에 기초하여 상기 싱크 트리와 상기 원격 트리 사이의 차이를 식별하고;

상기 차이에 기초하여, 상기 서버 상태와 상기 파일 시스템 상태를 수렴시키기 위해 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대하여 동작하도록 구성된 동작들의 세트를 생성하며; 및

상기 동작들의 세트의 실행을 관리하게끔 하는, 비-일시적인 컴퓨터-판독가능 매체.

청구항 13

청구항 10에 있어서, 상기 명령어들은 추가로 상기 컴퓨팅 시스템으로 하여금:

상기 싱크 트리 내의 상기 노드의 상기 디프 값과 로컬 트리 내의 대응하는 노드에 대한 제 2 디프 값을 비교하며;

상기 비교에 기초하여 상기 싱크 트리와 상기 로컬 트리 사이의 차이를 식별하고;

상기 차이에 기초하여, 상기 서버 상태와 상기 파일 시스템 상태를 수렴시키기 위하여 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대하여 동작하도록 구성된 동작들의 세트를 생성하며; 및

상기 동작들의 세트의 실행을 관리하게끔 하는, 비-일시적인 컴퓨터-판독가능 매체.

청구항 14

시스템으로서,

프로세서; 및

명령어들을 저장하는 비-일시적인 컴퓨터-판독가능 매체를 포함하며, 상기 명령어들은 상기 프로세서에 의해 실

행될 때, 상기 프로세서로 하여금:

트리 데이터 구조체로부터 노드를 제거하되, 상기 노드는 디프 값과 연관되며;

상기 노드의 상기 디프 값 및 부모 노드의 현재 디프 값에 기초하여 상기 노드의 부모에 대한 새로운 디프 값을 계산하고; 및

상기 부모에 대한 상기 새로운 디프 값을 상기 부모 노드 내에 저장하게끔 하는, 시스템.

청구항 15

청구항 14에 있어서, 상기 명령어들은 추가로 상기 프로세서로 하여금:

상기 부모 노드가 조상을 갖는다는 것을 결정하며;

상기 조상의 오래된 디프 값 및 상기 부모에 대한 상기 새로운 디프 값에 기초하여 상기 조상에 대한 새로운 디프 값을 계산하고; 및

상기 조상에 대한 상기 새로운 디프 값을 상기 조상 노드 내에 저장하게끔 하는, 시스템.

청구항 16

청구항 14에 있어서, 상기 노드의 상기 부모에 대한 상기 새로운 디프 값은 상기 부모 노드의 상기 현재 디프 값과 상기 노드의 상기 디프 값의 배타적 논리합(XOR) 연산에 기초하는, 시스템.

청구항 17

청구항 14에 있어서, 상기 노드에 대한 상기 디프 값은 상기 노드의 해시를 사용하여 계산되는, 시스템.

청구항 18

청구항 14에 있어서, 상기 명령어들은 추가로 상기 프로세서로 하여금:

상기 트리 데이터 구조체 내의 상기 노드의 상기 디프 값과 상이한 트리 데이터 구조체 내의 대응하는 노드에 대한 제 2 디프 값을 비교하며; 및

상기 트리 데이터 구조체와 상기 상이한 트리 데이트 구조체 사이의 차이를 식별하게끔 하는, 시스템.

청구항 19

청구항 18에 있어서, 상기 명령어들은 추가로 상기 프로세서로 하여금:

상기 차이에 기초하여, 상기 서버 상태와 상기 파일 시스템 상태를 수렴시키도록 구성된 동작들의 세트를 생성하며; 및

상기 동작들의 세트의 실행을 관리하게끔 하는, 시스템.

청구항 20

청구항 19에 있어서, 상기 명령어들은 추가로 상기 프로세서로 하여금 상기 동작들의 세트를 콘텐트 관리 시스템으로 송신하게끔 하는, 시스템.

발명의 설명

기 술 분 야

배 경 기 술

[0001] 관련 출원들에 대한 상호 참조

[0002] 본 출원은, 2017년 12월 29자로 출원된 미국 정규 출원 제15/858,207호, 및 2017년 12월 28일자로 출원된 미국 가출원 제62/611,473호에 대한 우선권을 주장하며, 이들 둘 모두는 그 전체가 명백하게 본원에 참조로서 통합된

다.

[0003] 배경기술

[0004] 콘텐트 관리 시스템들은 사용자들이 네트워크를 사용하여 다수의 디바이스들에 걸쳐 콘텐트 아이템들을 액세스하고 관리하는 것을 가능하게 한다. 일부 콘텐트 관리 시스템들은 사용자들이 콘텐트 아이템들을 공유하고 콘텐트 아이템들을 사용하여 협동하는데 있어서 사용자들을 보조하는 추가적인 특징부들을 제공하는 것을 가능하게 할 수 있다. 콘텐트 관리 시스템들은 일반적으로 서버들 상에 콘텐트 아이템들을 저장하며, 사용자들이 네트워크를 통해 콘텐트 아이템들을 액세스하는 것을 가능하게 한다. 일부 콘텐트 관리 시스템들은 또한, 사용자들에게 더 자연스러운 인터페이스(예를 들어, 네이티브 애플리케이션 또는 클라이언트 디바이스의 파일 시스템 내에)에서 콘텐트 아이템들에 대한 더 빠른 액세스를 제공하기 위하여 클라이언트 디바이스 상에 로컬 카피들이 저장되는 것을 가능하게 한다. 추가적으로, 이는 사용자가 오프라인일 때 사용자가 콘텐트에 대한 액세스를 갖는 것을 가능하게 한다. 콘텐트 관리 시스템들은, 각각의 카피가 동일하도록 복수의 클라이언트 디바이스들 및 서버들에 걸쳐 콘텐트 아이템의 카피들을 동기화하려고 시도한다. 그러나, 콘텐트 아이템들의 동기화가 어려우며, 이는 다수의 기술적 장애들과 연관된다.

발명의 내용

도면의 간단한 설명

[0005] 본 기술의 이상에서 언급된 그리고 다른 장점들 및 특징들은 첨부된 도면들에 예시된 특정 구현예들을 참조함으로써 명백해질 것이다. 당업자는, 이러한 도면들이 단지 본 기술의 일부 예들만을 도시하며 본 기술의 범위를 이러한 예들로 한정하지 않는다는 것을 이해할 것이다. 추가로, 당업자는 첨부된 도면들의 사용을 통해 추가적인 특정사항 및 세부사항을 가지고 설명되고 기술되는 바와 같은 본 기술의 원리들을 인식할 것이다.

도 1은 일부 실시예들에 따른 콘텐트 관리 시스템 및 클라이언트 디바이스들의 일 예를 도시한다.

도 2는 일부 실시예들에 따른 클라이언트 동기화 서비스의 일 예를 도시한다.

도 3은 다양한 실시예들에 따른 트리 데이터 구조체들의 일 예를 도시한다.

도 4는 다양한 실시예들에 따른 트리 데이터 구조체의 일 예를 도시한다.

도 5는 본 기술의 다양한 실시예들에 따른 트리 데이터 구조체들을 사용하여 서버 상태 및 파일 시스템 상태를 동기화하기 위한 예시적인 방법을 도시한다.

도 6은 본 기술의 다양한 실시예들에 따른 트리 데이터 구조체들을 사용하여 서버 상태 및 파일 시스템 상태를 동기화할 때의 충돌들을 해결하기 위한 예시적인 방법을 도시한다.

도 7은 다양한 실시예들에 따른 추가 동작에 대한 규칙의 위반을 예시하는 트리 데이터 구조체의 일 예를 도시한다.

도 8은 본 기술의 다양한 실시예들에 따른 서버 상태 및 파일 시스템 상태를 점진적으로 수렴시키기 위한 예시적인 방법을 도시한다.

도 9는 다양한 실시예들에 따른 트리 데이터 구조체들의 일 예를 도시한다.

도 10은 예시적인 시나리오를 도시한다.

도 11은 본 기술의 다양한 실시예들에 따른 동작들의 2개의 플랜들의 예시적인 벤 다이어그램 표현을 도시한다.

도 12는 본 기술의 다양한 실시예들에 따른 동작들의 플랜들에서의 변화들을 관리하기 위한 예시적인 방법을 도시한다.

도 13은 본 기술의 다양한 실시예들에 따른 파일명 어레이 및 해시 인덱스 어레이의 일 예시를 도시한다.

도 14는 본 기술의 다양한 실시예들에 따른 파일명을 저장하기 위한 예시적인 방법을 도시한다.

도 15는 본 기술의 다양한 실시예들에 따른 주어진 파일명에 대하여 파일명의 위치를 검색하기 위한 예시적인 방법을 도시한다.

도 16a 및 도 16b는 다양한 실시예들에 따른 트리 데이터 구조체의 예들을 도시한다.

도 17은 다양한 실시예들에 따른 트리 데이터 구조체의 일 예를 도시한다.

도 18은 본 기술의 다양한 실시예들에 따른 주어진 파일명에 대한 파일명의 위치를 검색하기 위한 예시적인 방법을 도시한다.

도 19는 본 기술의 특정 측면들을 구현하기 위한 시스템의 일 예를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0006] 본 기술의 다양한 예들이 이하에서 상세하게 논의된다. 특정 구현예들이 논의되지만, 이는 오로지 예시적인 목적들을 위해 이루어진다는 것이 이해되어야 한다. 당업자는, 본 기술의 사상 및 범위로부터 벗어나지 않고 다른 컴포넌트들 및 구성들이 사용될 수 있음을 인식할 것이다.

[0007] 컴퓨팅 및 네트워킹 기술들에서의 다양한 발전들이 사용자들에게 다수의 디바이스들에 걸쳐 콘텐트 아이템들에 대한 액세스를 제공하기 위한 콘텐트 관리 시스템들을 가능하게 하였다. 콘텐트 아이템들은, 비제한적으로, 파일들, 문서들, 메시지들(예를 들어, 이메일 메시지들 또는 텍스트 메시지들), 미디어 파일들(예를 들어, 사진들, 비디오들, 및 오디오 파일들), 다른 콘텐트 아이템들을 포함하는 폴더들, 또는 콘텐트의 임의의 다른 유닛을 포함할 수 있다. 콘텐트 아이템들은 다수의 사용자들과 공유되거나, 편집되거나, 삭제되거나, 추가되거나, 재명명되거나, 또는 이동될 수 있다. 그러나, 이러한 콘텐트 아이템들을 몇몇 컴퓨팅 디바이스들(예를 들어, 서버들 및 클라이언트 디바이스들)에 걸쳐 그리고 몇몇 사용자 계정들에 걸쳐 동기화하는 것은 여전히 기술적 장애들로 인해 결점이 있는 상태로 널리 퍼져 있는 채로 남아 있다.

[0008] 기술적 장애들의 일부를 예시하기 위하여, 제 1 기계(예를 들어, 클라이언트 디바이스 또는 서버)는, 사용자가 콘텐트 관리 시스템에 의해 관리되는 콘텐트 아이템들을 어떻게 수정했는지에 대한 정보를 제공하는 통신들을 제 2 기계로 전송할 수 있다. 이러한 통신들은, 제 1 기계 상에서 콘텐트 아이템들에 대하여 수행되는 액션들이 제 2 기계 상의 콘텐트 아이템들에 반영되고 제 1 기계 상의 콘텐트 아이템들이 제 2 기계 상의 콘텐트 아이템들과 실질적으로 동일해지도록, 제 2 기계 상의 콘텐트 아이템들을 동기화하기 위하여 제 2 기계에 의해 사용될 수 있다.

[0009] 그러나, 전송되는 몇몇 통신들이 존재할 수 있으며, 통신들은 통신들을 송신하기 위해 사용되는 하나 이상의 네트워크들에 의해 사용되는 다양한 네트워크 라우팅 프로토콜들, 제 1 또는 제 2 기계의 기술적 동작들, 또는 어떤 다른 이유의 결과로서 순서가 뒤바뀌어 수신될 수 있다. 추가로, 사용자는 많은 수의 콘텐트 아이템들에 대한 많은 수의 수정들을 수행하거나, 짧은 시간 내에 이전의 수정들의 실행취소(undo)하거나, 또는 이전에 수정된 콘텐트 아이템 또는 콘텐트 아이템들의 세트에 대하여 빠르게 추가적인 수정들을 수행할 수 있다. 이는, 이러한 통신들이 순서가 뒤바뀌어 수신되거나, 특정 통신들이 유효 기간이 지났거나(out of date), 또는 제 2 기계가 최신이 아닌 콘텐트 아이템들에 대하여 동작들을 수행할 가능성을 증가시킨다. 결과적으로, 동작들 중 다수가 콘텐트 아이템들의 현재 상태와 호환되지 않을 수 있다. 실제로, 심지어 일부 동작들이 다른 동작들과 충돌하는지 여부 또는 콘텐트 아이템들의 현재 상태와 충돌하지는 여부를 검출하는 것이 어려울 수 있다.

[0010] 추가적으로, 동기화 액션들에 대하여 고유 레이턴시(latency)가 존재한다. 예를 들어, 제 1 기계 상에서 취해지는 액션들은 먼저 제 1 기계에 의해 검출되며, 통신이 생성되고 그런 다음 네트워크를 통해 송신된다. 통신은, 여전히 이전 통신들을 프로세싱하고 통신들 내에서 상세화된 액션들을 취하고 있을 수 있는 제 2 기계에 의해 수신된다. 이러한 예시적인 시나리오에 있어서, 레이턴시가 제 1 기계, 제 2 기계, 및/또는 네트워크의 한정된 컴퓨팅 자원들(예를 들어, 대역폭, 메모리, 프로세싱 시간, 프로세싱 사이클들 등)에 의해 도입되는 몇몇 포인트들이 존재한다. 레이턴시는, 어떤 이유로 인하여, 콘텐트 아이템들의 현재 상태와 충돌하는 통신들이 증가될 가능성을 증가시킨다. 추가로, 이러한 충돌되는 통신들을 프로세싱하고 충돌들을 해결하는 것이 또한 프로세싱 시간, 메모리, 에너지, 또는 대역폭과 같은 불필요한 컴퓨팅 자원들을 소비하며, 추가로 레이턴시를 증가시킨다.

[0011] 문제들을 더 복잡하게 하기 위하여, 콘텐트 아이템들에 대한 액세스를 갖는 제 2 기계 및/또는 추가적인 기계들 상에서 동일하거나 또는 상이한 사용자가 또한 콘텐트 아이템들에 대한 수정들을 수행하고 있을 수 있다. 결과적으로, 이상의 이슈들이 배가될 수 있으며, 로컬 액션들이 원격 액션들과 충돌하지는 여부 및/또는 로컬 액션들이 최신 콘텐트 아이템들에 대하여 동작하는지 여부와 관련하여 추가적인 기술적 이슈들이 발생한다.

[0012] 개시되는 기술은 이상의 기술적 문제점들뿐만 아니라 다른 문제점들에 대한 기술적 해법을 제공하는 콘텐트 관리 시스템에 대한 클라이언트 동기화 서비스에 대한 당업계의 요구를 해결한다. 클라이언트 동기화 서비스는 클

라이언트 디바이스 상에서 동작하고 클라이언트 관리 시스템의 서버 상의 콘텐트 아이템들과 클라이언트 디바이스 상의 대응하는 콘텐트 아이템들 사이의 동기화 불일치를 식별하도록 구성될 수 있다. 각각의 동기화 불일치에 대하여, 클라이언트 동기화 서비스는 콘텐트 아이템들을 동기화하기 위해 필요한 동작들을 식별하고 이러한 동작들을 개시할 수 있다.

[0013] 클라이언트 동기화 서비스는, 트리 데이터 구조체들("트리들")의 세트를 사용하여 서버 상의 콘텐트 아이템들의 상태, 클라이언트 디바이스 상의 콘텐트 아이템들의 상태, 및 그들의 동기화 상태를 추적할 수 있다. 일부 실시예들에 따르면, 3개의 트리들의 세트가 사용될 수 있다. 3개의 트리들은, 서버 상태를 나타내는 원격 트리, 클라이언트 디바이스 상의 파일 시스템 상태를 나타내는 로컬 트리, 및 로컬 트리 및 원격 트리에 대한 병합 베이스(merge base)를 나타내는 싱크(sync) 트리를 포함할 수 있다. 병합 베이스는 로컬 트리와 원격 트리의 공통 조상 또는 로컬 트리와 원격 트리 사이의 마지막으로 알려진 동기화된 상태로서 여겨질 수 있다. 따라서, 클라이언트 동기화 서비스는, 3개의 트리들(예를 들어, 원격 트리, 싱크 트리, 및 로컬 트리) 모두가 동일할 때 서버 상태와 클라이언트 디바이스 상태가 동기화되었다고 결정할 수 있다.

[0014] 콘텐트 아이템들의 서버 상태 또는 콘텐트 아이템들의 클라이언트 디바이스 파일 시스템 상태("파일 시스템 상태")에 대한 수정이 검출될 때, 클라이언트 동기화 서비스는 3인조(triumvirate) 트리들에 기초하여 적절한 트리를 업데이트하고 서버 상태 및 파일 시스템 상태가 동기화되었는지 여부를 결정한다. 트리들 중 하나에 대한 업데이트에 기초하여, 서버 상태 및 파일 시스템 상태가 동기화되거나, 비동기화(unsynchronize)되거나, 또는 더 비동기화될 수 있다. 서버 상태 및 파일 시스템 상태가 동기화되지 않은 경우, 클라이언트 동기화 서비스는 적어도 서버 상태 및 파일 시스템 상태를 수렴시키기 위해 필요한 동작들의 초기 세트를 식별하고 서버 상태 및 파일 시스템 상태를 동기화된 상태에 더 가깝게 가져갈 수 있다.

[0015] 서버 상태 및 파일 시스템 상태를 모니터링하기 위해 트리 데이터 구조체들의 세트에 의존함으로써, 다양한 기술적 문제들에 대한 컴퓨팅 기술에 뿌리를 둔 해법들 및/또는 대안예들을 제공한다. 예를 들어, 클라이언트 동기화 서비스는 파일 상태뿐만 아니라 서버 상태를 추적하고 2개의 상태들의 병합 베이스의 표현을 저장할 수 있다. 결과적으로, 본 기술의 다양한 실시예들은 사용자들이 콘텐트 아이템들을 원격적으로 어떻게 수정하는지를 명시하는 복수의 통신들을 수신하는 것 및 이러한 수정들이 로컬적으로 구현되어야만 하는 순서를 결정하는 것, 수정들이 다른 수정들과 충돌하는지 또는 유효 기간이 지났는지 여부, 및 원격 수정들이 사용자들에 의해 로컬적으로 수행된 로컬 수정들과 충돌하는지 여부와 연관된 기술적 문제점들을 회피한다. 이러한 이슈들 중 다수는 수반되는 다양한 액터(actor)(예를 들어, 서버 및 클라이언트 디바이스)의 상태를 추적할 수 없으며 상태들이 동기화되었는지 여부를 빠르게 결정할 수 없는 다른 해법들로부터 기인한다. 그 대신에, 이러한 다른 해법들은, 서버 상태 및 파일 시스템 상태가 동기화되었는지 여부의 상황정보 없이, 콘텐트 아이템을 로컬적으로 어떻게 수정할 지에 대한 명령어들을 수신하는 것에 의존한다.

[0016] 추가로, 서버 상태 및 파일 시스템 상태가 동시에 모니터링되기 때문에, 이들의 동기화되었는지 여부를 결정하는 것이 절차 복잡성뿐만 아니라 컴퓨팅 시간 및 자원들과 관련하여 훨씬 더 효율적이다. 이하에서 추가로 상세하게 설명되는 바와 같이, 클라이언트 동기화 서비스는 보다 더 결정론적(deterministic) 방식으로 서버 상태 및 파일 시스템 상태의 점진적이고 조직적인 동기화를 가능하게 한다. 결과적으로, 콘텐트 관리 시스템 특징부들의 스케일링(scaling) 및 테스팅이 또한 효율적이다.

[0017] <u>콘텐트 관리 시스템</u>

[0018] 일부 실시예들에 있어서, 개시되는 기술은, 다른 것들 중에서도 특히, 콘텐트 아이템 동기화 성능들 및 협동 특징부들을 갖는 콘텐트 관리 시스템의 맥락에서 전개된다. 예시적인 시스템 구성(100)이 도 1a에 도시되며, 이는 클라이언트 디바이스(150)와 상호작용하는 콘텐트 관리 시스템(110)을 도시한다.

[0019] 계정들

[0020] 콘텐트 관리 시스템(110)은 계정들과 연관하여 콘텐트 아이템들을 저장할 수 있을 뿐만 아니라, 다양한 콘텐트 아이템 관리 태스크(task)들, 예컨대, 콘텐트 아이템(들)의 검색, 수정, 브라우징, 및/또는 공유를 수행할 수 있다. 추가로, 콘텐트 관리 시스템(110)은 계정이 복수의 클라이언트 디바이스들로부터 콘텐트 아이템(들)을 액세스하는 것을 가능하게 할 수 있다.

[0021] 콘텐트 관리 시스템(110)은 복수의 계정들을 지원한다. 엔티티(entity)(사용자, 사용자들의 그룹, 팀, 회사, 등)는 콘텐트 관리 시스템을 가지고 계정을 생성할 수 있으며, 계정 세부사항들이 계정 데이터베이스(140) 내에 저장될 수 있다. 계정 데이터베이스(140)는 등록된 엔티티들에 대한 프로파일 정보를 저장할 수 있다. 일부 경

우들에 있어서, 등록된 엔티티들에 대한 프로파일 정보는 사용자명 및/또는 이메일 주소를 포함한다. 계정 데이터베이스(140)는 계정 관리 정보, 예컨대 계정 유형(예를 들어, 다양한 계층들의 무료 또는 유료 계정들), 할당된 저장 공간, 사용된 저장 공간, 그 위에 상주하는 등록된 콘텐트 관리 클라이언트 애플리케이션(152)을 갖는 클라이언트 디바이스들(150), 보안 세팅들, 개인용 구성 세팅들, 등을 포함할 수 있다.

[0022] 계정 데이터베이스(140)는 엔티티와 연관된 계정들의 그룹들을 저장할 수 있다. 그룹들은 그룹 정책들 및/또는 액세스 제어 리스트들에 기초하여 허가(permission)들을 가질 수 있으며, 그룹들의 멤버들은 허가들을 계승할 수 있다. 예를 들어, 마케팅 그룹은 콘텐트 아이템들 중 하나의 세트에 대한 액세스를 가질 수 있으며, 반면 엔지니어링 그룹은 콘텐트 아이템들의 다른 세트에 대한 액세스를 가질 수 있다. 관리자 그룹은 그룹들을 수정하거나, 사용자 계정들을 수정하거나 할 수 있는 등이다.

[0023] 콘텐트 아이템 저장

[0024] 콘텐트 관리 시스템(110)의 하나의 특징은, 콘텐트 저장부(142) 내에 저장될 수 있는 콘텐트 아이템들의 저장부이다. 콘텐트 아이템들은 임의의 디지털 데이터 예컨대 문서들, 협동 콘텐트 아이템들, 텍스트 파일들, 오디오 파일들, 이미지 파일들, 비디오 파일들, 웹페이지들, 실행가능 파일들, 2진 파일들 등일 수 있다. 콘텐트 아이템은 또한 콘텐트 아이템들을 폴더들, 집 파일들, 플레이리스트들, 앨범들 등과 같은 상이한 거동들과 함께 그룹화하기 위한 수집물(collection)들 또는 다른 메커니즘들을 포함할 수 있다. 수집물은, 공통 속성에 의해 관련되거나 또는 그룹화되는 복수의 콘텐트 아이템들 또는 폴더를 지칭할 수 있다. 일부 실시예들에 있어서, 콘텐트 저장부(142)는 특정 기능들을 핸들링하기 위하여 다른 유형들의 저장부들 또는 데이터베이스들과 결합된다. 콘텐트 저장부(142)는 콘텐트 아이템들을 저장할 수 있으며, 반면 콘텐트 아이템들에 관한 메타데이터는 메타데이터 데이터베이스(146) 내에 저장될 수 있다. 마찬가지로, 콘텐트 아이템이 콘텐트 저장부(142) 내에 저장되는 위치에 관한 데이터는 콘텐트 디렉토리(144) 내에 저장될 수 있다. 추가적으로, 변화들, 액세스 등에 관한 데이터는 서버 파일 저널(journal)(148) 내에 저장될 수 있다. 콘텐트 저장부(142), 콘텐트 디렉토리(144), 서버 파일 저널(148), 및 메타데이터 데이터베이스(146)와 같은 다양한 저장부들/데이터베이스들의 각각은 2개 이상의 이러한 저장부 또는 데이터베이스로 구성될 수 있으며, 다수의 디바이스들 및 위치들에 걸쳐 분산될 수 있다. 다른 구성들이 또한 가능하다. 예를 들어, 콘텐트 저장부(142), 콘텐트 디렉토리(144), 서버 파일 저널(148), 및 메타데이터 데이터베이스(146)로부터의 데이터는 하나 이상의 콘텐트 저장부들 또는 데이터베이스들 내로 결합될 수 있거나 또는 추가적인 콘텐트 저장부들 또는 데이터베이스들로 분할될 수 있다. 따라서, 콘텐트 관리 시스템(110)은 도 1에 도시된 것보다 더 많거나 또는 더 적은 저장부들 및/또는 데이터베이스들을 포함할 수 있다.

[0025] 일부 실시예들에 있어서, 콘텐트 저장부(142)는 적어도 하나의 콘텐트 저장 서비스(116)와 연관되며, 이는, 비제한적으로, 저장하기 위해 콘텐트 아이템들을 수신하는 것, 저장을 위해 콘텐트 아이템들을 준비하는 것, 콘텐트 아이템에 대한 저장 위치를 선택하는 것, 저장부로부터 콘텐트 아이템들을 검색하는 것 등을 포함하는, 콘텐트 아이템들의 저장을 관리하기 위한 소프트웨어 또는 다른 프로세서 실행가능 명령어들을 포함한다. 일부 실시예들에 있어서, 콘텐트 저장 서비스(116)는 콘텐트 저장부(142)에서의 저장을 위해 더 작은 청크(chunk)들로 콘텐트 아이템을 분할할 수 있다. 콘텐트 아이템을 구성하는 각각의 청크의 위치가 콘텐트 디렉토리(144) 내에 기록될 수 있다. 콘텐트 디렉토리(144)는 콘텐트 저장부(142) 내에 저장된 각각의 콘텐트 아이템에 대한 콘텐트 엔트리를 포함할 수 있다. 콘텐트 엔트리는, 콘텐트 아이템을 식별하는 고유 ID와 연관될 수 있다.

[0026] 일부 실시예들에 있어서, 콘텐트 디렉토리(144) 내에서 콘텐트 아이템을 식별하는 고유 ID는 결정론적 해시 함수로부터 도출될 수 있다. 콘텐트 아이템에 대한 고유 ID를 도출하는 이러한 방법은, 예컨대, 결정론적 해시 함수가 동일한 콘텐트 아이템의 매 카피에 대하여 동일한 식별자를 출력할 것이지만 상이한 콘텐트 아이템에 대하여 상이한 식별자를 출력할 것이기 때문에, 콘텐트 아이템 복제(duplicate)들이 인식되는 것을 보장할 수 있다. 이러한 방법론을 사용하면, 콘텐트 저장 서비스(116)는 각각의 콘텐트 아이템에 대하여 고유 ID를 출력할 수 있다.

[0027] 콘텐트 저장 서비스(116)는 또한 메타데이터 데이터베이스(146) 내에 콘텐트 아이템에 대한 콘텐트 경로를 지정하거나 또는 기록할 수 있다. 콘텐트 경로는 콘텐트 아이템의 명칭 및/또는 콘텐트 아이템과 연관된 폴더 계층을 포함할 수 있다. 예를 들어, 콘텐트 경로는, 그 안에 콘텐트 아이템이 클라이언트 디바이스 상의 로컬 파일 시스템 내에 저장되는 폴더들의 경로 또는 폴더를 포함할 수 있다. 콘텐트 아이템들이 블록들로 콘텐트 저장부(142) 내에 저장되고 디렉토리 구조체와 같은 트리 아래에 저장되지 않을 수 있지만, 이러한 디렉토리 구조체는 사용자들에게 편안한 네비게이션 구조체이다. 콘텐트 저장 서비스(116)는 콘텐트 아이템에 대한 콘텐트 경로를

정의하거나 또는 기록할 수 있으며, 여기에서 디렉토리 구조체의 "루트(root)" 노드는 각각의 계정에 대한 명칭공간(namespace)일 수 있다. 계정의 사용자 및/또는 콘텐트 저장 서비스(116)에 의해 정의되는 디렉토리 구조체가 명칭공간 내에 존재할 수 있다. 메타데이터 데이터베이스(146)는 콘텐트 엔트리의 부분으로서 각각의 콘텐트 아이템에 대한 콘텐트 경로를 저장할 수 있다.

[0028] 일부 실시예들에 있어서, 명칭공간은, 마치 이들이 루트 노드 내에 저장되는 것처럼 디렉토리 구조체 내에 내포되는(nested) 추가적인 명칭공간들을 포함할 수 있다. 이는, 계정이 공유된 수집물에 대한 액세스를 가질 때 발생할 수 있다. 공유된 수집물들에는 콘텐트 관리 시스템(110) 내에서 그들 자체의 명칭공간이 할당될 수 있다. 공유된 수집물들은 실제로 공유된 수집물에 대한 루트 노드일 수 있지만, 이들은 디렉토리 구조체 내에서 계정 명칭공간에 종속되어 위치되며, 계정에 대한 폴더 내의 폴더로서 나타날 수 있다. 이상에서 다루어진 바와 같이, 디렉토리 구조체는 단지 사용자들에 대한 편리한 네비게이션 구조체이며, 콘텐트 저장부(142) 내의 콘텐트 아이템들의 저장 위치와 상관되지 않는다.

[0029] 그 안에서 계정이 콘텐트 아이템들을 보는 디렉토리 구조체가 콘텐트 관리 시스템(110)에서의 저장 위치들과 상관되지 않지만, 디렉토리 구조체는 클라이언트 디바이스(150)에 의해 사용되는 파일 시스템에 의존하여 클라이언트 디바이스(150) 상의 저장 위치들과 상관될 수 있다.

[0030] 이상에서 언급된 바와 같이, 콘텐트 디렉토리(144) 내의 콘텐트 엔트리는 또한 콘텐트 아이템을 구성하는 각각의 청크의 위치를 포함할 수 있다. 보다 더 구체적으로, 콘텐트 엔트리는, 콘텐트 아이템을 구성하는 청크들의 콘텐트 저장부(142) 내의 위치를 식별하는 콘텐트 포인터(pointer)들을 포함할 수 있다.

[0031] 콘텐트 경로 및 콘텐트 포인터에 더하여, 콘텐트 디렉토리(144) 내의 콘텐트 엔트리는 또한, 콘텐트 아이템에 대한 액세스를 갖는 사용자 계정을 식별하는 사용자 계정 식별자 및/또는 콘텐트 엔트리가 속하는 명칭공간 및/또는 콘텐트 아이템에 대한 액세스를 갖는 그룹을 식별하는 그룹 식별자를 포함할 수 있다.

[0032] 콘텐트 저장 서비스(116)는, 콘텐트 아이템 또는 콘텐트 아이템의 버전들을 구성하는 복제 콘텐트 아이템들 또는 복제 블록들을 식별함으로써 요구되는 저장 공간의 양을 감소시킬 수 있다. 복수의 카피들을 저장하는 대신에, 콘텐트 저장부(142)는 콘텐트 아이템의 단일 카피 또는 콘텐트 아이템의 블록을 저장할 수 있으며, 콘텐트 디렉토리(144)는 복제들을 단일 카피에 링크하기 위한 포인터 또는 다른 메커니즘을 포함할 수 있다.

[0033] 콘텐트 저장 서비스(116)는 또한, 콘텐트 아이템의 고유 ID와 연관하여, 콘텐트 아이템들, 콘텐트 아이템 유형들, 폴더들, 파일 경로, 및/또는 다양한 계정들, 수집물들, 또는 그룹들에 대한 콘텐트 아이템들의 관계를 설명하는 메타데이터를 메타데이터 데이터베이스(146) 내에 저장할 수 있다.

[0034] 콘텐트 저장 서비스(116)는 또한 변화들, 액세스, 등에 관한 데이터의 로그(log)를 서버 파일 저널(148) 내에 저장할 수 있다. 서버 파일 저널(148)은 콘텐트 아이템의 고유 ID 및 시간 스탬프 또는 버전 번호 및 임의의 다른 관련 데이터와 함께 변화 또는 액세스 액션의 설명을 저장할 수 있다. 서버 파일 저널(148)은 또한 변화 또는 콘텐트 아이템 액세스에 의해 영향을 받은 블록들에 대한 포인터들을 포함할 수 있다. 콘텐트 저장 서비스는, 서버 파일 저널(148)로부터 획득될 수 있는, 콘텐트 아이템들에 대한 변화들, (발산(diverging) 버전 트리들을 포함하는) 콘텐트 아이템들의 상이한 버전들, 및 변화 히스토리를 추적하는 콘텐트 아이템 버전 제어를 사용함으로써, 동작들을 실행취소하기 위한 능력을 제공할 수 있다.

[0035] 콘텐트 아이템 동기화

[0036] 콘텐트 관리 시스템(110)의 다른 특징은 적어도 하나의 클라이언트 디바이스(150)와 콘텐트 아이템들의 동기화이다. 클라이언트 디바이스(들)는 상이한 형태들을 취할 수 있으며, 상이한 성능들을 가질 수 있다. 예를 들어, 클라이언트 디바이스(150_1)는 그 위에 상주하는 다수의 애플리케이션들에 의해 액세스가능한 로컬 파일 시스템을 갖는 컴퓨팅 디바이스이다. 클라이언트 디바이스(150_2)는 컴퓨팅 디바이스이며, 여기에서 콘텐트 아이템들은 오로지 특정 애플리케이션에 대해서만 또는 특정 애플리케이션에 의해 주어진 허가에 의해서만 액세스가능하며, 콘텐트 아이템들은 전형적으로 애플리케이션 특정 공간 내에 또는 클라우드 내에 저장된다. 클라이언트 디바이스(150_3)는 웹 브라우저를 통해 콘텐트 관리 시스템(110)을 액세스하며 웹 인터페이스를 통해 콘텐트 아이템들을 액세스하는 임의의 클라이언트 디바이스이다. 예시적인 클라이언트 디바이스들(150_1, 150_2, 및 150_3)은 랩탑, 모바일 디바이스, 또는 웹 브라우저와 같은 폼 팩터들로 도시되지만, 이의 설명들이 이러한 예시적인 폼 팩터들의 디바이스들로 한정되지 않는다는 것이 이해되어야만 한다. 예를 들어, 클라이언트(150_2)와 같은 모바일 디바이스는 상주하는 다수의 애플리케이션들에 의해 액세스될 수 있는 로컬 파일 시스템을 가질 수 있거나 또는 클

라이언트(150_2)는 웹 브라우저를 통해 콘텐트 관리 시스템(110)을 액세스할 수 있다. 이와 같이, 폼 팩터는 클라이언트(150)의 성능을 고려할 때 제한적으로 간주되지 않아야 한다. 클라이언트 디바이스(150)와 관련하여 본원에서 설명되는 하나 이상의 기능들은 디바이스의 특정 성능들에 의존하여 매 클라이언트 디바이스 상에서 이용가능하거나 또는 이용가능하지 않을 수 있다 - 파일 액세스 모델은 이러한 하나의 성능이다.

[0037] 다수의 실시예들에 있어서, 클라이언트 디바이스들은 콘텐트 관리 시스템(110)의 계정과 연관되지만, 일부 실시예들에 있어서, 클라이언트 디바이스들은 공유 링크들을 사용하여 콘텐트를 액세스하고 계정을 필요로 하지 않을 수 있다.

[0038] 이상에서 언급된 바와 같이, 일부 클라이언트 디바이스들은 웹 브라우저를 사용하여 콘텐트 관리 시스템(110)을 액세스할 수 있다. 그러나, 클라이언트 디바이스들은 또한 클라이언트 디바이스(150)에 저장되어 클라이언트 디바이스 상에서 구동되는 클라이언트 애플리케이션(152)을 사용하여 콘텐트 관리 시스템(110)을 액세스할 수 있다. 클라이언트 애플리케이션(152)은 클라이언트 동기화 서비스(156)를 포함할 수 있다.

[0039] 클라이언트 동기화 서비스(156)는 클라이언트 디바이스(150)와 콘텐트 관리 시스템(110) 사이에 콘텐트 아이템들에 대한 변화들을 동기화하기 위하여 서버 동기화 서비스(112)와 통신하고 있을 수 있다.

[0040] 클라이언트 디바이스(150)는 클라이언트 동기화 서비스(156)를 통해 콘텐트 관리 시스템(110)과 콘텐트를 동기화할 수 있다. 동기화는 플랫폼에 구애받지 않을 수 있다. 즉, 콘텐트는 다양한 유형, 성능, 운영 시스템 등의 다수의 클라이언트 디바이스들에 걸쳐 동기화될 수 있다. 클라이언트 동기화 서비스(156)는 클라이언트 디바이스(150)의 파일 시스템의 지정된 위치 내의 콘텐트 아이템들에 대한 임의의 변화들(신규, 삭제된, 수정된, 복사된, 또는 이동된 콘텐트 아이템들)을 동기화할 수 있다.

[0041] 콘텐트 아이템들은 클라이언트 디바이스(150)로부터 콘텐트 관리 시스템(110)으로 그리고 이의 역으로 동기화될 수 있다. 동기화가 클라이언트 디바이스(150)로부터 콘텐트 관리 시스템(110)으로 이루어지는 실시예들에 있어서, 클라이언트 동기화 서비스(156)가 모니터링되는 폴더들 내의 파일들에 대한 변화들에 대하여 클라이언트 디바이스(150) 상의 디렉토리를 모니터링할 수 있는 동안 사용자는 클라이언트 디바이스(150)의 파일 시스템으로부터 직접적으로 콘텐트 아이템들을 조작할 수 있다.

[0042] 클라이언트 동기화 서비스(156)가 그것이 모니터링하는 디렉토리 내의 콘텐트의 기입, 이동, 복사, 또는 삭제를 검출할 때, 클라이언트 동기화 서비스(156)는 변화들을 콘텐트 관리 시스템 서비스(116)로 동기화할 수 있다. 일부 실시예들에 있어서, 클라이언트 동기화 서비스(156)는, 콘텐트 아이템을 블록들로 분할하는 것, 고유 식별자를 생성하기 위하여 콘텐트 아이템을 해시하는 것 등과 같은 이상에서 다루어진 기능들을 포함하는 콘텐트 관리 시스템 서비스(116)의 일부 기능들을 수행할 수 있다. 클라이언트 동기화 서비스(156)는 클라이언트 저장 인덱스(164) 내에 콘텐트를 인덱싱(index)할 수 있으며, 결과를 저장 인덱스(164)내에 세이브(save)할 수 있다. 인덱싱은 경로들에 더하여 고유 서버 식별자를 저장하는 것을 포함할 수 있으며, 고유 클라이언트 식별자는 각각의 콘텐트 아이템에 대한 것이다. 일부 실시예들에 있어서, 클라이언트 동기화 서비스(156)는 서버 동기화 서비스(112)로부터 고유 서버 식별자를 학습하고, 클라이언트 디바이스(150)의 운영 시스템으로부터 고유 클라이언트 식별자를 학습한다.

[0043] 클라이언트 동기화 서비스(156)는 콘텐트 관리 시스템(110)의 사용자 계정과 연관된 콘텐트와 클라이언트 저장부 내의 콘텐트의 적어도 일 부분의 동기화를 가능하게 하기 위하여 저장 인덱스(164)를 사용할 수 있다. 예를 들어, 클라이언트 동기화 서비스(156)는 저장 인덱스(164)와 콘텐트 관리 시스템(110)을 비교할 수 있으며, 클라이언트 저장부 상의 콘텐트와 콘텐트 관리 시스템(110) 상의 사용자 계정과 연관된 콘텐트 사이의 차이점을 검출할 수 있다. 그러면, 클라이언트 동기화 서비스(156)는 클라이언트 저장부 상의 콘텐트를 적절하게 업로딩하거나, 다운로딩하거나, 수정하거나, 삭제함으로써 차이점들을 일치시키려고 시도할 수 있다. 콘텐트 저장 서비스(116)는 콘텐트 아이템에 대한 변화된 또는 신규 블록을 저장하고 서버 파일 저널(148), 메타데이터 데이터베이스(146), 콘텐트 디렉토리(144), 콘텐트 저장부(142), 계정 데이터베이스(140) 등을 적절하게 업데이트할 수 있다.

[0044] 콘텐트 관리 시스템(110)으로부터 클라이언트 디바이스(150)로 동기화할 때, 서버 파일 저널(148) 내에 기록된 콘텐트 아이템의 마운트(mount), 수정, 추가, 삭제, 이동은 통지 서비스(117)를 사용하여 통지가 클라이언트 디바이스(150)로 전송되도록 트리거(trigger)할 수 있다. 클라이언트 디바이스(150)가 변화를 통지받을 때, 클라이언트 디바이스에 알려진 마지막 동기화 포인트로부터 서버 파일 저널(148) 내에 리스트된 요청이 변화한다. 클라이언트 디바이스(150)가 이것이 콘텐트 관리 시스템(110)과 동기가 어긋났다(out of synchronization)고 결

정할 때, 클라이언트 동기화 서비스(156)는 변화들을 포함하는 콘텐트 아이템 블록들을 요청하고, 그것의 변화된 콘텐트 아이템들의 로컬 카피를 업데이트한다.

[0045] 일부 실시예들에 있어서, 저장 인덱스(164)는 트리 데이터 구조체들을 저장하며, 여기에서 하나의 트리는 서버 동기화 서비스(112)에 따라 디렉토리의 최신 표현을 반영하고, 반면 다른 트리는 클라이언트 동기화 서비스(156)에 따라 디렉토리의 최신 표현을 반영한다. 클라이언트 동기화 서비스는, 서버 동기화 서비스(112)로부터 데이터를 요청함으로써 또는 클라이언트 디바이스(150) 상의 변화들을 콘텐트 관리 시스템(110)으로 커밋(commit)함으로써 트리 구조체들이 일치한다는 것을 보장하도록 작동할 수 있다.

[0046] 때때로, 클라이언트 디바이스(150)가 이용가능한 네트워크 연결을 갖지 않을 수 있다. 이러한 시나리오에서, 클라이언트 동기화 서비스(156)는 콘텐트 아이템 변화들에 대하여 링크된 수집물을 모니터링하고, 이후에 네트워크 연결이 이용가능할 때 콘텐트 관리 시스템(110)에 대한 동기화를 위해 이러한 변화들을 큐잉(queue)할 수 있다. 유사하게, 사용자는 수동으로 콘텐트 관리 시스템(110)과의 동기화를 시작하거나, 중지하거나, 일시 정지하거나, 또는 재개할 수 있다.

[0047] 콘텐트 동기화 서비스(156)는 콘텐트 관리 시스템(110) 상의 특정 사용자 계정과 연관된 모든 콘텐트를 동기화할 수 있다. 대안적으로, 클라이언트 동기화 서비스(156)는 콘텐트 관리 시스템(110) 상의 특정 사용자 계정과 연관된 전체 콘텐트 중의 콘텐트의 일 부분을 선택적으로 동기화할 수 있다. 콘텐트의 일 부분만을 선택적으로 동기화하는 것은 클라이언트 디바이스(150) 상의 공간을 보존하고 대역폭을 절약할 수 있다.

[0048] 일부 실시예들에 있어서, 클라이언트 동기화 서비스(156)는 선택적으로 특정 사용자 계정과 연관된 콘텐트의 일 부분을 저장하며, 콘텐트의 나머지 부분에 대하여 클라이언트 저장부 내에 플레이스홀더(placeholder) 콘텐트 아이템들을 저장한다. 예를 들어, 클라이언트 동기화 서비스(156)는, 콘텐트 관리 시스템(110) 상의 그것의 개별적인 완전한 콘텐트 아이템의 동일한 파일명, 경로, 확장, 메타데이터를 갖지만 완전한 콘텐트 아이템의 데이터가 없는 플레이스홀더 콘텐트 아이템을 저장할 수 있다. 플레이스홀더 콘텐트 아이템은 수 바이트 또는 그 이하의 크기일 수 있으며, 반면 개별적인 완전한 콘텐트 아이템은 훨씬 더 클 수 있다. 클라이언트 디바이스(150)가 콘텐트 아이템을 액세스하려고 시도한 이후에, 클라이언트 동기화 서비스(156)는 콘텐트 관리 시스템(110)으로부터 콘텐트 아이템의 데이터를 검색하고, 액세스 중인 클라이언트 디바이스(150)에 완전한 콘텐트 아이템을 제공할 수 있다. 이러한 접근 방식은, 여전히 콘텐트 관리 시스템(110) 상의 사용자의 콘텐트에 대한 완전한 액세스를 제공하면서 상당한 공간 및 대역폭 절약을 제공할 수 있다.

[0049] 협동 특징들

[0050] 콘텐트 관리 시스템(110)의 다른 특징은 사용자들 사이에 협동을 가능하게 하는 것이다. 협동 특징들은 콘텐트 아이템 공유, 콘텐트 아이템들에 대한 코멘팅, 콘텐트 아이템들에 대한 공동-작업, 인스턴스 메시징, 콘텐트 아이템들에 관한 프레즌스(presence) 및 보여진(seen) 상태 정보를 제공하는 것 등을 포함한다.

[0051] 공유

[0052] 콘텐트 관리 시스템(110)은 공유 서비스(128)를 통해 콘텐트를 공유하는 것을 관리할 수 있다. 콘텐트에 대한 링크를 제공함으로써 콘텐트를 공유하는 것은 콘텐트 관리 시스템(110)과 통신하고 있는 네트워크 내의 임의의 컴퓨팅 디바이스로부터 콘텐트 아이템을 액세스 가능하게 만드는 것을 포함할 수 있다. 그러나, 일부 실시예들에 있어서, 링크는 콘텐트 관리 시스템(110) 및 액세스 제어 리스트(145)에 의해 집행되는 액세스 제한들과 연관될 수 있다. 콘텐트를 공유하는 것은 또한, 각각의 사용자 계정이 콘텐트 아이템에 대한 액세스를 갖도록 콘텐트 관리 시스템(110) 내의 콘텐트를 (콘텐트 아이템과 연관된 원래의 사용자 계정에 더하여) 적어도 하나의 추가적인 사용자 계정과 공유하기 위해 공유 서비스(128)를 사용하여 콘텐트를 링크하는 것을 포함할 수 있다. 추가적인 사용자 계정은 콘텐트를 수락함으로써 콘텐트에 대한 액세스를 획득할 수 있으며, 그러면 콘텐트는 웹 인터페이스 서비스(124)를 통해 또는 클라이언트 디바이스(150) 상의 그들의 계정과 연관된 디렉토리 구조체 내로부터 직접적으로 액세스될 수 있을 것이다. 공유는 플랫폼에 구애받지 않는 방식으로 수행될 수 있다. 즉, 콘텐트는 다양한 유형, 성능, 운영 시스템 등의 다수의 클라이언트 디바이스들(150)에 걸쳐 공유될 수 있다. 콘텐트는 또한 다양한 유형들의 사용자 계정들에 걸쳐 공유될 수 있다.

[0053] 콘텐트 관리 시스템(110) 내의 콘텐트 아이템을 공유하기 위하여, 공유 서비스(128)는 콘텐트 아이템과 연관된 액세스 제어 리스트 데이터베이스(145) 내의 콘텐트 엔트리에 사용자 계정 식별자 또는 다수의 사용자 계정 식별자들을 추가하고 그에 따라서 추가된 사용자 계정이 콘텐트 아이템에 액세스하는 것을 승인할 수 있다. 공유 서비스(128)는 또한 콘텐트 아이템에 대한 사용자 계정의 액세스를 제한하기 위하여 콘텐트 엔트리로부터 사용

자 계정 식별자들을 제거할 수 있다. 공유 서비스(128)는 또한 액세스 제어 리스트 데이터베이스(145) 내에 콘텐트 아이템 식별자들, 콘텐트 아이템에 대한 액세스가 주어진 사용자 계정 식별자들, 및 액세스 레벨들을 기록할 수 있다. 예를 들어, 일부 실시예들에 있어서, 단일 콘텐트 엔트리와 연관된 사용자 계정 식별자들은 연관된 콘텐트 아이템에 대하여 개별적인 사용자 계정 식별자들에 대한 상이한 허가들을 지정할 수 있다.

[0054] 콘텐트 관리 시스템(110) 외부에서 콘텐트 아이템들을 공유하기 위하여, 공유 서비스(128)는, 임의의 웹 브라우저가 임의의 인증 없이 콘텐트 관리 시스템(110) 내의 콘텐트 아이템 또는 수집물을 액세스하는 것을 가능하게 하는, 자원 위치 지정자(uniform resource locator; URL)와 같은 커스텀 네트워크 어드레스를 생성할 수 있다. 이를 달성하기 위하여, 공유 서비스(128)는 생성된 URL 내에 콘텐트 식별 데이터를 포함시킬 수 있으며, 이는 이후에 요청된 콘텐트 아이템을 적절하게 식별하고 반환하기 위하여 사용될 수 있다. 예를 들어, 공유 서비스(128)는 생성된 URL 내에 계정 식별자 및 콘텐트 경로 또는 콘텐트 아이템 식별 코드를 포함시킬 수 있다. URL의 선택 시에, URL 내에 포함된 콘텐트 식별 데이터는 콘텐트 관리 시스템(110)으로 송신될 수 있으며, 콘텐트 관리 시스템은 적절한 콘텐트 아이템을 식별하고 콘텐트 아이템을 반환하기 위하여 수신된 콘텐트 식별 데이터를 사용할 수 있다.

[0055] URL을 생성하는 것에 더하여, 공유 서비스(128)는 또한 콘텐트 아이템에 대한 URL이 생성되었다는 것을 액세스 제어 리스트 데이터베이스(145) 내에 기록하도록 구성될 수 있다. 일부 실시예들에 있어서, 콘텐트 아이템과 연관된 콘텐트 엔트리는 콘텐트 아이템에 대한 URL이 생성되었는지 여부를 나타내는 URL 플래그를 포함할 수 있다. 예를 들어, URL 플래그는 콘텐트 아이템에 대한 URL이 생성되지 않았다는 것을 나타내기 위하여 처음에 0 또는 거짓(false)으로 설정된 불(Boolean) 값일 수 있다. 공유 서비스(128)는 콘텐트 아이템에 대한 URL을 생성한 이후에 플래그의 값을 1 또는 참(true)으로 변화시킬 수 있다.

[0056] 일부 실시예들에 있어서, 공유 서비스(128)는 콘텐트 아이템에 대한 URL에 허가들의 세트를 연관시킬 수 있다. 예를 들어, 사용자가 URL을 통해 콘텐트 아이템을 액세스하려고 시도하는 경우, 공유 서비스(128)는 콘텐트 아이템에 대한 허가들의 제한된 세트를 제공할 수 있다. 제한된 허가들의 예들은, 사용자가 콘텐트 아이템을 다운로드할 수 없거나, 콘텐트 아이템을 세이브할 수 없거나, 콘텐트 아이템을 카피할 수 없거나, 콘텐트 아이템을 수정할 수 없는 등의 제한들을 포함한다. 일부 실시예들에 있어서, 제한된 허가들은, 콘텐트 아이템이 지정된 도메인으로부터만, 즉, 회사 네트워크 도메인 내로부터만 또는 지정된 도메인과 연관된 계정들에 의해서만, 예를 들어, 회사 계정과 연관된 계정들(예를 들어, @acme.com)에 의해서만 액세스되도록 허가하는 제한들을 포함한다.

[0057] 일부 실시예들에 있어서, 공유 서비스(128)는 또한 생성된 URL을 비활성화하도록 구성될 수 있다. 예를 들어, 각각의 콘텐트 엔트리는 생성된 URL로부터의 요청에 응답하여 콘텐트가 반환되어야만 하는지 여부를 나타내는 URL 활성 플래그를 포함할 수 있다. 예를 들어, 공유 서비스(128)는, URL 활성 플래그가 1 또는 참으로 설정된 경우에만 생성된 링크에 의해 요청된 콘텐트 아이템을 반환할 수 있다. 따라서, 이에 대하여 URL이 생성되었던 콘텐트 아이템에 대한 액세스는 URL 활성화 플래그의 값을 변화시킴으로써 용이하게 제한될 수 있다. 이는 사용자가, 콘텐트 아이템을 이동시키거나 또는 생성된 URL을 삭제할 필요 없이 공유된 콘텐트에 대한 액세스를 제한하는 것을 가능하게 한다. 마찬가지로, 공유 서비스(128)는, URL 활성화 플래그의 값을 다시 1 또는 참으로 변화시킴으로써 URL을 재활성화할 수 있다. 따라서, 사용자는 새로운 URL을 생성할 필요 없이 콘텐트 아이템에 대한 액세스를 용이하게 복원할 수 있다.

[0058] 일부 실시예들에 있어서, 콘텐트 관리 시스템(110)은 콘텐트 아이템을 업로딩하기 위한 URL을 지정할 수 있다. 예를 들어, 사용자 계정을 갖는 제 1 사용자가 이러한 URL을 요청하고, 그 URL을 기여(contributing) 사용자에게 제공하며, 기여 사용자는 그 URL을 사용하여 제 1 사용자의 사용자 계정으로 콘텐트 아이템을 업로드할 수 있다.

[0059] 팀 서비스

[0060] 일부 실시예들에 있어서, 콘텐트 관리 시스템(110)은 팀 서비스(130)를 포함한다. 팀 서비스(130)는 사용자 계정들의 정의된 팀들의 생성하고 관리하기 위한 기능을 제공할 수 있다. 팀들은 서브-팀들(예를 들어, 비지니스 유닛들, 또는 프로젝트 팀들 등)을 갖는 상태로 회사에 대하여 생성될 수 있으며, 팀들 및 서브-팀들에 할당된 사용자 계정들은 사용자 계정들의 임의의 정의된 그룹에 대하여 생성될 수 있다. 팀 서비스(130)는 팀에 대한 공통 공유 공간, 사적 사용자 계정 폴더들, 및 액세스 제한 공유 폴더들을 제공할 수 있다. 팀 서비스는 또한 관리자가 팀 내에서 수집물들 및 콘텐트 아이템들을 관리하기 위한 관리 인터페이스를 제공할 수 있으며, 팀과 연관된 사용자 계정들을 관리할 수 있다.

[0061] 인가 서비스

[0062] 일부 실시예들에 있어서, 콘텐트 관리 시스템(110)은 인가 서비스(132)를 포함한다. 인가 서비스(132)는, 명칭공간을 액세스하려고 시도하는 사용자 계정이 명칭공간을 액세스하기 위한 적절한 권리들을 갖는다는 것을 보장한다. 인가 서비스(132)는 명칭공간을 액세스하기 위한 요청 다음에 클라이언트 애플리케이션(152)으로부터 토큰을 수신할 수 있으며, 사용자 계정에 허가된 능력들을 반환할 수 있다. 다수의 레벨들의 액세스를 갖는 사용자 계정들(예를 들어, 사용자 권리들 및 관리자 권리들을 갖는 사용자 계정)에 대하여, 인가 서비스(132)는 또한 관리자들에 의한 비의도적 액션들을 회피하기 위해 명시적인 특권 상승을 필요로 할 수 있다.

[0063] 프레즌스 및 보여진 상태

[0064] 일부 실시예들에 있어서, 콘텐트 관리 시스템은, 콘텐트 아이템을 공유하는 사용자들이 콘텐트 아이템과 어떻게 상호작용하고 있는지 또는 상호작용하였는지에 대한 정보를 제공할 수 있다. 일부 실시예들에 있어서, 콘텐트 관리 시스템(110)은 콘텐트 아이템을 공유하는 사용자가 현재 콘텐트 아이템을 보고 있다는 것을 보고할 수 있다. 예를 들어, 클라이언트 협동 서비스(160)는, 클라이언트 디바이스(150)가 콘텐트 아이템을 액세스할 때 통지 서비스(117)에 통지할 수 있다. 그러면, 통지 서비스(117)는 콘텐트 아이템에 대하여 클라이언트 디바이스(150)의 사용자의 프레즌스의 동일한 콘텐트 아이템에 대한 액세스를 갖는 다른 사용자들의 모든 클라이언트 디바이스들로 통지할 수 있다.

[0065] 일부 실시예들에 있어서, 콘텐트 관리 시스템(110)은 공유 콘텐트 아이템과의 사용자 상호작용의 히스토리를 보고할 수 있다. 협동 서비스(126)는, 사용자가 콘텐트 아이템을 세이브했다는 것, 사용자가 아직 콘텐트 아이템을 보고 있다는 것 등을 결정하기 위하여 메타데이터 데이터베이스(146) 및 서버 파일 저널(148)과 같은 데이터 소스들에 질의할 수 있으며, 다른 사용자들이 누가 현재 콘텐트 아이템을 보고 있는지 또는 보았는지 또는 수정했는지를 알 수 있도록 통지 서비스(117)를 사용하여 다른 사용자들에게 이러한 상태 정보를 전파할 수 있다.

[0066] 협동 서비스(126)는, 심지어 콘텐트 아이템이 천성적으로 코멘팅 기능을 갖지 않는 경우에도 콘텐트와 연관된 코멘트들을 가능하게 할 수 있다. 이러한 코멘트들은 메타데이터 데이터베이스(146) 내에 저장될 수 있다.

[0067] 협동 서비스(126)는 사용자들에 대한 통지들을 발생시키고 이를 송신할 수 있다. 예를 들어, 사용자는 코멘트에서 다른 사용자를 언급할 수 있으며, 협동 서비스(126)는 통지를 코멘트에서 언급된 그 사용자에게 전송할 수 있다. 콘텐트 아이템을 삭제하는 것, 콘텐트 아이템을 공유하는 것 등을 포함하는 다양한 다른 콘텐트 아이템 이벤트들이 통지들을 트리거할 수 있다.

[0068] 협동 서비스(126)는, 이에 의해 사용자가 인스턴스 메시지들, 음성 호들, 이메일들 등을 전송하고 수신할 수 있는 메시징 플랫폼을 제공할 수 있다.

[0069] 협동 콘텐트 아이템들

[0070] 일부 실시예들에 있어서, 콘텐트 관리 서비스는 또한, 이에 의해 사용자가 동시에 협동 콘텐트 아이템들을 생성하고, 협동 콘텐트 아이템들에 코멘트하며, 협동 콘텐트 아이템들 내의 태스크들을 병합할 수 있는 상호작용식 콘텐트 아이템 협동 플랫폼을 제공할 수 있는 협동 문서 서비스(134)를 포함할 수 있다. 협동 콘텐트 아이템들은 사용자가 협동 콘텐트 아이템 에디터(editor)를 사용하여 생성하고 편집할 수 있는 파일들일 수 있으며, 협동 콘텐트 아이템 엘리먼트들을 포함할 수 있다. 협동 콘텐트 아이템 엘리먼트들은 협동 콘텐트 아이템 식별자, 하나 이상의 저자 식별자들, 협동 콘텐트 아이템 텍스트, 협동 콘텐트 아이템 속성들, 상호작용 정보, 코멘트들, 공유 사용자들 등을 포함할 수 있다. 협동 콘텐트 아이템 엘리먼트들은 데이터베이스 엔티티들로서 저장될 수 있으며, 이는 협동 콘텐트 아이템들을 탐색하고 검색하는 것을 가능하게 한다. 다수의 사용자들은 동시에 또는 상이한 시점들에 협동 콘텐트 아이템들을 액세스하며, 보고, 편집하며, 협동할 수 있다. 일부 실시예들에 있어서, 이는 2명의 사용자들이 웹 인터페이스를 통해 콘텐트 아이템을 액세스할 것을 요구함으로써 관리될 수 있으며, 여기에서 이들은 동시에 콘텐트 아이템의 동일한 카피 상에서 작업할 수 있다.

[0071] 협동 컴패니언(Companion) 인터페이스

[0072] 일부 실시예들에 있어서, 클라이언트 협동 서비스(160)는, 클라이언트 디바이스(150) 상에 제시되고 있는 콘텐트 아이템과 관련된 정보를 디스플레이하기 위한 목적을 위한 네이티브 애플리케이션 컴패니언 인터페이스를 제공할 수 있다. 콘텐트 아이템이 클라이언트 디바이스(150) 상에 저장되고 실행되는 네이티브 애플리케이션에 의해 액세스되는 실시예들에 있어서, 여기에서 콘텐트 아이템은, 콘텐트 아이템이 콘텐트 애플리케이션(152)에 의해 관리되도록 클라이언트 디바이스(150)의 파일 시스템의 지정된 위치 내에 존재하며, 네이티브 애플리케이션

은 이상에서 다루어진 협동 데이터를 디스플레이하기 위한 임의의 네이티브 방식을 제공하지 않을 수 있다. 이러한 실시예들에 있어서, 클라이언트 협동 서비스(160)는 사용자가 콘텐트 아이템을 열었다는 것을 검출할 수 있으며, 협동 데이터와 같은 콘텐트 아이템에 대한 추가적인 정보를 갖는 오버레이(overlay)를 제공할 수 있다. 예를 들어, 추가적인 정보는 콘텐트 아이템에 대한 코멘트들, 콘텐트 아이템의 상태, 이전에 또는 현재 콘텐트 아이템을 보고 있는 다른 사용자들의 활동을 포함할 수 있다. 이러한 오버레이는, 다른 사용자가 현재 콘텐트 아이템을 편집하고 있기 때문에 변화들이 상실될 수 있음을 사용자에게 경고할 수 있다.

[0073] 일부 실시예들에 있어서, 이상에서 논의된 서비스들 또는 저장부들/데이터베이스들 중 하나 이상은 공중 또는 사설 애플리케이션 프로그래밍 인터페이스들을 사용하여 액세스될 수 있다.

[0074] 특정 소프트웨어 애플리케이션들은 사용자를 대신하여 API를 통해 콘텐트 저장부(142)를 액세스할 수 있다. 예를 들어, 클라이언트 디바이스(150) 상에서 실행되는 애플리케이션과 같은 소프트웨어 패키지는, 사용자가, 콘텐트를 판독하거나, 기입하거나, 생성하거나, 삭제하거나, 공유하거나 또는 달리 조작하기 위하여 인증 증명서를 제공할 때, 콘텐트 관리 시스템(110)에 대하여 직접적으로 API 호출들을 프로그램적으로 만들 수 있다.

[0075] 사용자는, 웹 인터페이스 서비스(124)에 의해 생성되고 서비스되는 웹 인터페이스를 통해 사용자 계정에 저장된 콘텐트를 보거나 또는 조작할 수 있다. 예를 들어, 사용자는 콘텐트 관리 시스템(110)에 의해 제공되는 웹 어드레스로 웹 브라우저 내에서 네비게이션할 수 있다. 콘텐트 아이템의 새로운 버전을 업로딩하는 것과 같은, 웹 인터페이스를 통해 이루어진 콘텐트 저장부(142) 내의 콘텐트에 대한 변화들 또는 업데이트들이 다시 사용자의 계정과 연관된 다른 클라이언트 디바이스들로 전파될 수 있다. 예를 들어, 각기 그들 자체의 클라이언트 소프트웨어를 갖는 다수의 클라이언트 디바이스들은 단일 계정과 연관될 수 있으며, 계정 내의 콘텐트 아이템들은 다수의 클라이언트 디바이스들 사이에서 동기화될 수 있다.

[0076] 클라이언트 디바이스(150)는 사용자를 대신하여 콘텐트 관리 시스템(110)에 연결할 수 있다. 예를 들어, 클라이언트 디바이스(150)가 데스크탑 또는 랩탑 컴퓨터, 전화기, 텔레비전, 사물 인터넷 디바이스 등일 때, 사용자는 클라이언트 디바이스(150)와 직접적으로 상호작용할 수 있다. 대안적으로 또는 추가적으로, 클라이언트 디바이스(150)는, 예를 들어, 클라이언트 디바이스(150)가 서버일 때, 사용자가 클라이언트 디바이스(150)에 대한 물리적인 액세스를 갖지 않고, 사용자를 대신하여 행동할 수 있다.

[0077] 클라이언트 디바이스(150)의 일부 특징들은 클라이언트 디바이스(150) 상에 설치된 애플리케이션에 의해 가능해진다. 일부 실시예들에 있어서, 애플리케이션은 콘텐트 관리 시스템 특정 컴포넌트를 포함할 수 있다. 예를 들어, 콘텐트 관리 시스템 특정 컴포넌트는 독립형 애플리케이션(152), 하나 이상의 애플리케이션 플러그-인(plug-in)들, 및/또는 브라우저 확장(browser extension)일 수 있다. 그러나, 사용자는 또한, 클라이언트 디바이스(150) 상에 상주하며 콘텐트 관리 시스템(110)과 통신하도록 구성된, 웹 브라우저와 같은 제 3 자 애플리케이션을 통해 콘텐트 관리 시스템(110)과 상호작용할 수 있다. 다양한 구현예들에 있어서, 클라이언트-측 애플리케이션(152)은 사용자가 콘텐트 관리 시스템(110)과 상호작용하기 위한 사용자 인터페이스(user interface; UI)를 제공할 수 있다. 예를 들어, 사용자는 파일 시스템과 통합된 파일 시스템 익스플로러를 통해 또는 웹 브라우저 애플리케이션을 사용하여 디스플레이되는 웹페이지를 통해 콘텐트 관리 시스템(110)과 상호작용할 수 있다.

[0078] 일부 실시예들에 있어서, 클라이언트 애플리케이션(152)은 콘텐트 관리 시스템(110)의 2개 이상의 계정에 대하여 콘텐트를 관리하고 동기화하도록 구성될 수 있다. 이러한 실시예들에 있어서, 클라이언트 애플리케이션(152)은 다수의 계정들에 로그 인한 채로 남아 있고 다수의 계정들에 대하여 정상 서비스들을 제공할 수 있다. 일부 실시예들에 있어서, 각각의 계정은 파일 시스템 내에 폴더로서 나타날 수 있으며, 그 폴더 내의 모든 콘텐트 아이템들은 콘텐트 관리 시스템(110)과 동기화될 수 있다. 일부 실시예들에 있어서, 클라이언트 애플리케이션(152)은 다수의 계정들 중 하나를 주 계정 또는 기본 계정으로 선택하기 위한 선택기를 포함할 수 있다.

[0079] 콘텐트 관리 시스템(110)이 특정 컴포넌트들을 가지고 제공되지만, 시스템(100)의 아키텍처적 구성이 단지 하나의 가능한 구성이며, 더 많거나 또는 더 적은 컴포넌트들을 갖는 다른 구성들이 가능하다는 것이 당업자에 의해 이해되어야 한다. 추가로, 서비스는, 심지어 다른 서비스와 함께 설명된 기능을 포함하여 더 많거나 또는 더 적은 기능을 가질 수 있다. 또한, 일 실시예에 대하여 본원에서 설명되는 특징들은 다른 실시예에 대하여 설명된 특징들과 결합될 수 있다.

[0080] 시스템(100)이 특정 컴포넌트들을 가지고 제공되지만, 시스템(100)의 아키텍처적 구성이 단지 하나의 가능한 구성이며, 더 많거나 또는 더 적은 컴포넌트들을 갖는 다른 구성들이 가능하다는 것이 당업자에 의해 이해되어야

한다.

[0081] 클라이언트 동기화 서비스

[0082] 도 2는 일부 실시예들에 따른 클라이언트 동기화 서비스(156)의 일 예를 도시한다. 일부 실시예들에 따르면, 클라이언트 동기화 서비스(156)는 도 1의 클라이언트 디바이스 내에 구현될 수 있다. 그러나, 다른 실시예들에 있어서, 클라이언트 동기화 서비스(156)는 다른 컴퓨팅 디바이스 상에 구현될 수 있다. 클라이언트 동기화 서비스(156)는, 클라이언트 동기화 서비스(156)가 실행되는 클라이언트 디바이스와 콘텐트 관리 시스템 사이에서 콘텐트 아이템들에 대한 변화들을 동기화하도록 구성된다.

[0083] 클라이언트 동기화 서비스(156)는 파일 시스템 인터페이스(205), 서버 인터페이스(210), 트리 저장부(220), 플래너(planner)(225), 및 스케줄러(scheduler)(230)를 포함할 수 있다. 추가적인 또는 대안적인 컴포넌트들이 또한 포함될 수 있다. 클라이언트 동기화 서비스(156)의 고 레벨 설명들 및 그것의 컴포넌트들은 이하에서 도 2와 관련하여 논의된다. 그러나, 클라이언트 동기화 서비스(156) 및 그것의 컴포넌트들의 추가적인 세부사항들 및 실시예들이 전체에 걸쳐 논의된다.

[0084] 파일 시스템 인터페이스(205)는 클라이언트 디바이스의 로컬 파일시스템 상에서 콘텐트 아이템들에 대한 변화들을 프로세싱하고 로컬 트리를 업데이트하도록 구성된다. 예를 들어, 파일 시스템 인터페이스(205)는 클라이언트 디바이스의 로컬 파일시스템 상의 콘텐트 아이템들에 대한 변화들을 검출하기 위하여 도 1의 클라이언트 동기화 서비스(156)와 통신하고 있을 수 있다. 변화들은 또한 도 1의 클라이언트 애플리케이션(152)을 통해 이루어지고 검출될 수 있다. 파일 시스템 인터페이스(205)는 로컬 트리에 대한 업데이트들을 수행할 수 있다. 로컬 트리에 대한 업데이트들은 클라이언트 디바이스 상의 콘텐트 아이템들에 대한 변화들(신규, 삭제된, 수정된, 복사된, 재명명된, 또는 이동된 콘텐트 아이템들)에 기초하여 이루어질 수 있다.

[0085] 서버 인터페이스(210)는 콘텐트 관리 시스템의 원격 저장부에서의 콘텐트 아이템들에 대한 원격 변화들의 프로세싱 및 원격 트리의 업데이팅을 보조하도록 구성된다. 예를 들어, 서버 인터페이스(210)는 클라이언트 디바이스(150)와 콘텐트 관리 시스템(110) 사이에 변화들을 동기화하기 위하여 도 1의 서버 동기화 서비스(112)와 통신하고 있을 수 있다. 콘텐트 관리 시스템(110)에서의 콘텐트 아이템들에 대한 변화들(신규, 삭제된, 수정된, 복사된, 재명명된, 또는 이동된 콘텐트 아이템들)이 검출될 수 있으며, 콘텐트 관리 시스템(110)에서 변화들을 반영하기 위하여 원격 트리에 대한 업데이트들이 이루어질 수 있다.

[0086] 트리 저장부(220)는 클라이언트 동기화 서비스(156)에 의해 사용되는 트리 데이터 구조체들을 저장하고 유지하도록 구성된다. 예를 들어, 트리 저장부(220)는 로컬 트리, 싱크 트리, 및 원격 트리를 저장할 수 있다. 일부 실시예들에 따르면, 트리 저장부(220)는 레이턴시 및 반응 시간을 감소시키기 위하여 메인 메모리(예를 들어, RAM 또는 다른 1차 저장 디바이스)뿐만 아니라 영구 메모리(예를 들어, 하드 디스크 또는 다른 2차 저장 디바이스) 내에 트리 데이터 구조체들을 저장할 수 있다. 예를 들어, 클라이언트 디바이스 또는 클라이언트 동기화 서비스(156)의 기동 시에, 트리 데이터 구조체는 영구 메모리로부터 검색되어 메인 메모리 내로 로딩될 수 있다. 트리 저장부(220)는 메인 메모리 상에서 트리 데이터 구조체들을 액세스하고 업데이트할 수 있으며, 클라이언트 디바이스 또는 클라이언트 동기화 서비스(156)가 셧 다운(shut down)되기 이전에, 트리 저장부(220)는 영구 메모리 상에 업데이트된 트리 데이터 구조체들을 저장할 수 있다. 메인 메모리가 비용이 비싸고 흔히 대부분의 클라이언트 디바이스들 상에서 크기가 제한되기 때문에, 메인 메모리 상에서의 트리 데이터 구조체들의 풋프린트(footprint)를 감소시키기 위하여 추가적인 기술적 개선들이 구현된다. 이러한 기술적 해법들이 이하에서 추가로 설명된다.

[0087] 플래너(225)는 트리 데이터 구조체들의 상태에 기초하여 클라이언트 디바이스와 연관된 파일 시스템 상태와 콘텐트 관리 시스템과 연관된 서버 상태 사이의 차이점을 검출하도록 구성된다. 예를 들어, 플래너(225)는 원격 트리와 싱크 트리 사이에 차이가 존재하는지 여부를 결정할 수 있다. 원격 트리와 싱크 트리 사이의 차이는, 콘텐트 관리 시스템에서 저장된 하나 이상의 콘텐트 아이템들에 대하여 원격적으로 수행된 액션이 서버 상태 및 파일 시스템 상태의 동기가 어긋나게끔 했다는 것을 나타낸다. 유사하게, 플래너(225)는 또한 로컬 트리와 싱크 트리 사이에 차이가 존재하는지 여부를 결정할 수 있다. 로컬 트리와 싱크 트리 사이의 차이는, 클라이언트 디바이스 상에 저장된 하나 이상의 콘텐트 아이템들에 대하여 로컬적으로 수행된 액션이 서버 상태 및 파일 시스템 상태의 동기가 어긋나게끔 했다는 것을 나타낸다. 차이가 검출되는 경우, 플래너(225)는 트리 데이터 구조체들을 동기화하는 동작들의 세트를 생성한다.

[0088] 일부 시나리오들에 있어서, 원격 트리와 싱크 트리 사이의 차이에 기초하여 생성된 동작들의 세트 및 로컬 트리

와 싱크 트리 사이의 차이에 기초하여 생성된 동작들의 세트가 충돌할 수 있다. 플래너(225)는 또한 동작들의 2개의 세트들을 동작들의 단일의 병합된 플랜으로 병합하도록 구성될 수 있다.

[0089] 스케줄러(230)는 동작들의 생성된 플랜을 취하고 이러한 동작들의 실행을 관리하도록 구성된다. 일부 실시예들에 따르면, 스케줄러(230)는 동작들의 플랜 내의 각각의 동작을, 동작을 수행하기 위하여 실행될 필요가 있는 일련의 하나 이상의 태스크들로 변환한다. 일부 시나리오들에 있어서, 일부 태스크들은 유효 기간이 지나게 될 수 있거나 또는 더 이상 관련이 없게 될 수 있다. 스케줄러(230)는 이러한 태스크들을 식별하고 이들을 취소하도록 구성된다.

[0090] <u>트리 데이터 구조체들</u>

[0091] 도 3은 다양한 실시예들에 따른 트리 데이터 구조체들의 일 예를 도시한다. 트리 데이터 구조체들은 클라이언트 디바이스에 저장되고 도 2에 도시된 클라이언트 동기화 서비스(156)와 같은 클라이언트 동기화 서비스에 의해 관리될 수 있다. 도 3에서, 트리 데이터 구조체는 원격 트리(310), 싱크 트리(330), 및 로컬 트리(350)를 포함하여 도시된다.

[0092] 원격 트리(310)는 클라이언트 디바이스로부터 원격에 저장된(예를 들어, 콘텐트 관리 시스템의 서버 상에 저장된) 콘텐트 아이템들의 상태 또는 서버 상태를 나타낸다. 로컬 트리(350)는 클라이언트 디바이스 상에 로컬적으로 저장된 대응하는 콘텐트 아이템들의 상태 또는 파일 시스템 상태를 나타낸다. 싱크 트리(330)는 로컬 트리와 원격 트리에 대한 병합 베이스를 나타낸다. 병합 베이스는 로컬 트리와 원격 트리의 공통 조상 또는 로컬 트리와 원격 트리 사이의 마지막으로 알려진 동기화된 상태로서 여겨질 수 있다.

[0093] 각각의 트리 데이터 구조체(예를 들어, 원격 트리(310), 싱크 트리(330), 또는 로컬 트리(350))는 하나 이상의 노드들을 포함할 수 있다. 각각의 노드는 하나 이상의 자식 노드들을 가질 수 있으며, 부모-자식 관계가 에지에 의해 표현된다. 예를 들어, 원격 트리(310)는 노드들(312 및 314)을 포함한다. 노드(312)는 노드(314)의 부모이며, 노드(314)는 노드(312)의 자식이다. 이러한 부모-자식 관계는 에지(316)에 의해 표현된다. 루트 노드, 예컨대 루트 노드(312)는 부모 노드를 갖지 않는다. 리프(leaf) 노드, 예컨대 노드(314)는 자식 노드를 갖지 않는다.

[0094] 트리 데이터 구조체 내의 각각의 노드는 콘텐트 아이템(예를 들어, 파일, 문서, 폴더 등)을 나타낼 수 있다. 예를 들어, 루트 노드(312)는 콘텐트 관리 시스템과 연관된 루트 폴더를 나타낼 수 있으며, 노드(314)는 그 루트 폴더 내에 위치된 파일(예를 들어, "Foo.txt"라는 명칭의 텍스트 파일)을 나타낼 수 있다. 트리 데이터 구조체 내의 각각의 노드는, 예를 들어, 콘텐트 아이템의 부모 노드의 파일 식별자를 지정하는 디렉토리 파일 식별자(directory file identifier; "DirFileID"), 콘텐트 아이템에 대한 파일명, 콘텐트 아이템에 대한 파일 식별자, 및 콘텐트 아이템에 대한 메타데이터와 같은 데이터를 포함할 수 있다.

[0095] 이상에서 설명된 바와 같이, 클라이언트 동기화 서비스는, 3개의 트리들(예를 들어, 원격 트리(310), 싱크 트리(330), 및 로컬 트리(350)) 모두가 동일할 때 서버 상태와 클라이언트 디바이스의 파일 시스템 상태가 동기화되었다고 결정할 수 있다. 다시 말해서, 트리들은, 그들의 트리 구조체들 및 이들이 표현하는 관계들이 동일하고 그들의 노드들 내에 포함된 데이터가 마찬가지로 동일할 때, 동기화된다. 반대로, 3개의 트리들이 동일하지 않은 경우 트리들은 동기화되지 않는다. 도 3에 예시된 예시적인 시나리오에 있어서, 원격 트리(310), 싱크 트리(330), 및 로컬 트리(350)는 동일하고 동기화된 것으로서 도시되며, 결과적으로, 서버 상태 및 파일 시스템 상태가 동기화된다.

[0096] <u>트리 데이터 구조체들을 사용하는 변화들의 추적</u>

[0097] 도 4는 다양한 실시예들에 따른 트리 데이터 구조체의 일 예를 도시한다. 도 3에 도시된 트리 데이터 구조체들과 마찬가지로, (원격 트리(410), 싱크 트리(430), 및 로컬 트리(450)를 포함하는) 도 4에 도시된 트리 데이터 구조체들은 클라이언트 디바이스에 저장되고 도 2의 클라이언트 동기화 서비스(156)와 같은 클라이언트 동기화 서비스에 의해 관리될 수 있다. 도 4에서, 트리 데이터 구조체들이 도시된다.

[0098] 도 4는 도 3에 예시된 시나리오와 같은 이전에 동기화된 상태 이후의 시나리오를 도시하며, 콘텐트 아이템들을 수정하기 위해 트리들 내에 표현된 콘텐트 아이템들에 대하여 추가적인 액션들이 수행되었으며 그 결과 트리들이 더 이상 동기가 맞지 않는다. 싱크 트리(430)는 이전에 알려진 동기화된 상태의 표현을 유지하며, 이는, 서버 상태와 파일 시스템 상태가 동기화되도록, 서버 상태와 시스템 상태 사이의 차이들을 식별하기 위하여 그리고 또한 콘텐트 관리 시스템 및/또는 클라이언트 디바이스가 수렴하기 위하여 수행하기 위한 동작들을 생성하기

위하여 클라이언트 동기화 서비스에 의해 사용될 수 있다.

[0099] 예를 들어, 사용자(클라이언트 디바이스와 연관된 사용자와 동일한 사용자 또는 콘텐트 아이템에 대한 액세스를 갖는 상이한 사용자)는 콘텐트 관리 시스템에 의해 저장된 "foo.txt" 콘텐트 아이템에 대한 수정들을 수행할 수 있다. 이러한 콘텐트 아이템은 원격 트리(410) 내의 노드(414)에 의해 표현된다. 원격 트리(410) 내에 도시된 수정은 foo.txt 콘텐트 아이템의 제거(예를 들어, 콘텐트 관리 시스템에 의해 관리되는 공간으로부터의 콘텐트 아이템의 제거) 또는 삭제이다. 이러한 수정들은, 예를 들어, 다른 클라이언트 디바이스 상에서 수행될 수 있으며, 그런 다음 콘텐트 관리 시스템으로 동기화되거나 또는 콘텐트 관리 시스템에 연결된 웹 브라우저를 통해 수행될 수 있다.

[0100] 콘텐트 관리 시스템 상에서 변화가 이루어질 때, 콘텐트 관리 시스템은 이루어진 변화를 명시하는 수정 데이터를 생성하고 수정 데이터를 클라이언트 디바이스 상의 클라이언트 동기화 서비스로 송신한다. 예를 들어, 콘텐트 관리 시스템이 클라이언트 디바이스로 일방적으로 변화들을 송신하거나 또는 "푸시"할 수 있는 푸시 모델을 사용하는 것. 다른 구현예들에 있어서, 클라이언트 디바이스에 의한 요청에 응답하여 서버가 변환들을 전송하는 풀링(pull) 모델. 추가적으로, 클라이언트 디바이스가 요청들을 개시하지만 시간의 일정한 기간 동안 연결을 오픈된 채로 유지하며 그에 따라서 콘텐트 관리 시스템이 연결이 살아있는 동안 필요에 따라 추가적인 변화들을 푸시할 수 있는 롱 풀링(long pull)을 수반하는 하이브리드 모델. 클라이언트 동기화 서비스는 수정 데이터에 기초하여 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 서버 상태를 나타내는 원격 트리를 업데이트한다. 예를 들어, 원격 트리(410) 내에서, foo.txt 콘텐트 아이템을 나타내는 노드(414)가 삭제된 것으로서 도시된다.

[0101] 클라이언트 동기화 서비스는 원격 트리(410)와 싱크 트리(430) 사이의 차이를 식별하고, 결과적으로, 콘텐트 관리 시스템에서의 콘텐트 아이템들의 수정이 서버 상태 및 파일 시스템 상태의 동기가 더 이상은 맞지 않게끔 했다는 것을 결정할 수 있다. 클라이언트 동기화 서비스는 추가로, 이들이 동기가 맞춰지도록 서버 상태 및 파일 시스템 상태를 수렴시키도록 구성된 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대한 동작들의 세트 또는 시퀀스를 생성하고 실행할 수 있다.

[0102] 추가적으로 또는 대안적으로, 사용자(콘텐트 관리 시스템에서의 수정들과 연관된 사용자와 동일한 사용자 또는 콘텐트 아이템에 대한 액세스를 갖는 상이한 사용자)는, 콘텐트 관리 시스템과 연관된 클라이언트 디바이스 상에 로컬적으로 저장된 콘텐트 아이템들에 대하여 수정할 수 있다. 예를 들어, 사용자는 폴더 "/bar"를 "/root" 폴더에 추가하고, "Hi.doc" 문서를 "/bar" 폴더에 추가할 수 있다.

[0103] 클라이언트 디바이스 상에서 변화가 이루어질 때, 클라이언트 디바이스(예를 들어, 도 1의 클라이언트 동기화 서비스(156) 또는 클라이언트 애플리케이션(152))는 이루어진 변화를 명시하는 수정 데이터를 생성한다. 클라이언트 동기화 서비스는 수정 데이터에 기초하여 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대한 파일 시스템 상태를 나타내는 로컬 트리를 업데이트한다. 예를 들어, 로컬 트리(450) 내에, 노드(452) 및 노드(454)가 추가되는 것으로서 도시된다. 노드(452) 및 노드(454)는 각기 "/bar" 폴더 및 "Hi.doc" 문서를 나타낸다.

[0104] 클라이언트 동기화 서비스는 로컬 트리(450)와 싱크 트리(430) 사이의 차이를 식별하고, 결과적으로, 클라이언트 디바이스에서의 콘텐트 아이템들의 수정이 서버 상태 및 파일 시스템 상태의 동기가 더 이상은 맞지 않게끔 했다는 것을 결정할 수 있다. 클라이언트 동기화 서비스는 추가로, 이들이 동기가 맞춰지도록 서버 상태 및 파일 시스템 상태를 수렴시키도록 구성된 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 동작들의 세트 또는 시퀀스를 생성할 수 있다. 이러한 동작들은 실행을 위하여 콘텐트 관리 시스템으로 송신될 수 있다.

[0105] 도 4에서 보여지는 바와 같이, 클라이언트 디바이스 상에 저장된 콘텐트 아이템들 및 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 수정들은 실질적으로 동시에 또는 특정 시간 기간 이내에 발생할 수 있다. 이러한 수정들은 트리 데이터 구조체 내에 반영될 수 있으며, 병렬로 클라이언트 디바이스에 대한 그리고 콘텐트 관리 시스템에 대한 동작들을 생성하기 위하여 클라이언트 동기화 서비스에 의해 사용될 수 있다. 그러나, 다른 시나리오들에서, 수정들이 반드시 동시에 발생하지는 않을 수 있으며, 동작들은 필요에 따른 방식으로 생성될 수 있다. 추가로, 도 4가 콘텐트 아이템을 추가하고 콘텐트 아이템들을 삭제하기 위한 시나리오들을 예시하지만, 콘텐트 아이템들을 편집하는 것, 재명명하는 것, 복사하는 것, 또는 이동시키는 것과 같은 다른 유형들의 수정들이 또한 지원된다.

[0106] 다양한 실시예들에 따르면, 2개의 트리 데이터 구조체들 사이의 차이를 식별하는 것 및 동작들을 생성하는 것은, 트리 데이터 구조들 둘 모두 내의 각각의 노드를 체크하는 것 및 노드에 대하여 액션이 수행되었는지 여

부를 결정하는 것을 수반할 수 있다. 액션들은, 예를 들어, 노드의 추가, 노드의 삭제, 노드의 편집, 또는 노드의 이동을 포함할 수 있다. 그러면, 이러한 액션들은 서버 상태 및 파일 시스템 상태를 수렴시키도록 구성된 동작들을 생성하기 위해 사용될 수 있다.

[0107] 예를 들어, 2개의 트리 데이터 구조체들이 싱크 트리 및 원격 트리인 경우, 클라이언트 동기화 서비스는, 예를 들어, 싱크 트리 내의 모든 노드들에 대한 파일 식별자들을 요청함으로써 싱크 트리 내의 각각의 노드를 식별할 수 있다. 싱크 트리 내의 노드에 대한 각각의 노드 또는 파일 식별자에 대하여, 클라이언트 동기화 서비스는 노드 또는 파일 식별자가 원격 트리 내에 또한 존재하는지 여부를 결정할 수 있다. 원격 트리에서 발견되지 않는 싱크 트리 내의 노드 또는 파일 식별자는, 노드가 원격 트리에 의해 표현되는 서버 상태로부터 삭제되었다는 것을 나타낼 수 있다. 따라서, 클라이언트 동기화 서비스는, 삭제 액션이 원격 트리 상에서 발생하였다는 것을 결정할 수 있다. 노드에 대한 파일 식별자 또는 노드가 원격 트리 내에서 발견되지 않는 경우, 클라이언트 동기화 서비스는, 원격 트리 내의 노드가 편집되거나 또는 이동되었는지 여부를 체크할 수 있다.

[0108] 원격 트리 내의 노드가 싱크 트리 내의 노드에 비하여 편집되었는지 여부를 결정하기 위하여, 클라이언트 동기화 서비스는 싱크 트리 내의 노드에 대한 메타데이터를 원격 트리 내의 대응하는 노드(예를 들어, 동일한 파일 식별자를 갖는 노드)에 대한 메타데이터와 비교할 수 있다. 메타데이터는, 노드에 의해 표현되는 콘텐트 아이템이 편집되었는지 여부를 결정하기 위해 사용될 수 있는 정보를 포함할 수 있다. 예를 들어, 메타데이터는, 콘텐트 아이템 내의 데이터 또는 이의 일 부분에 기초하여 생성되는 하나 이상의 해시 값들을 포함할 수 있다. 메타데이터는 추가적으로 또는 대안적으로 콘텐트 아이템에 대한 크기 값, 마지막으로 수정된 값, 또는 다른 값을 포함할 수 있다. 싱크 트리 내의 노드에 대한 메타데이터는 원격 트리 내의 노드에 대한 메타데이터와 비교될 수 있다. 메타데이터가 일치하지 않는 경우, 콘텐트 아이템의 편집이 원격 트리에 의해 표현되는 서버 상태에서 편집되었을 수 있다. 따라서, 클라이언트 동기화 서비스는, 원격 트리 상의 노드에 대하여 편집 액션이 발생하였다는 것을 결정할 수 있다. 메타데이터가 일치하는 경우, 어떠한 편집도 발생하지 않았을 수 있다.

[0109] 원격 트리 내의 노드가 이동되었는지 여부를 결정하기 위하여, 클라이언트 동기화 서비스는 싱크 트리 내의 노드에 대한 위치와 원격 트리 내의 대응하는 노드(예를 들어, 동일한 파일 식별자를 갖는 노드)에 대한 위치를 비교할 수 있다. 위치는, 예를 들어, 노드가 위치되는 경로, 파일명, 및/또는 노드의 부모의 파일 식별자를 명시하는 디렉토리 파일 식별자(directory file identifier; "DirFileID")를 포함할 수 있다. 위치들이 일치하는 경우, 어떠한 이동도 발생하지 않았을 수 있다. 반면, 위치들이 일치하지 않는 경우, 콘텐트 아이템의 이동은 원격 트리에 의해 표현되는 서버 상태에서 발생하였을 수 있다. 따라서, 클라이언트 동기화 서비스는, 원격 트리 상의 노드에 대하여 이동 액션이 발생하였다는 것을 결정할 수 있다.

[0110] 노드가 원격 트리에 추가되었는지 여부를 결정하기 위하여, 클라이언트 동기화 서비스는, 싱크 트리 내에서 발견되지 않는 원격 트리 내의 임의의 노드들 또는 파일 식별자들을 식별할 수 있다. 노드 또는 파일 식별자가 원격 트리 내에서 발견되고 싱크 트리 내에서는 발견되지 않는 경우, 클라이언트 동기화 서비스는, 이러한 노드의 추가 액션이 서버 상태를 나타내는 원격 트리 상에서 발생하였다는 것을 결정할 수 있다.

[0111] 이상의 예가 싱크 트리 및 원격 트리에 대하여 설명되었지만, 다른 실시예들에 있어서, 싱크 트리와 로컬 트리 사이의 차이를 식별하고 파일 시스템 상태를 나타내는 로컬 트리 상에서 어떠한 액션들이 발생하였는지를 결정하기 위하여 유사한 프로세스가 싱크 트리 및 로컬 트리를 가지고 발생할 수 있다.

[0112] <u>트리 데이터 구조체들을 사용하는 동기화</u>

[0113] 도 5는 본 기술의 다양한 실시예들에 따른 트리 데이터 구조체들을 사용하여 서버 상태 및 파일 시스템 상태를 동기화하기 위한 예시적인 방법을 도시한다. 본원에서 설명되는 방법들 및 프로세스들이 특정 순서의 특정 단계들 및 동작들을 가지고 도시될 수 있지만, 달리 언급되지 않는 한, 유사하거나 또는 대안적인 순서들로, 또는 병렬로 수행되는, 추가적이거나, 더 적거나, 또는 대안적인 단계들 및 동작들이 다양한 실시예들의 범위 내에 속한다. 방법(500)은, 예를 들어, 클라이언트 디바이스 상에서 실행 중인 도 2의 클라이언트 동기화 서비스(156)와 같은 시스템에 의해 구현될 수 있다.

[0114] 시스템은, 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 서버 상태를 나타내는 원격 트리, 클라이언트 디바이스 상에 저장된 대응하는 콘텐트 아이템들에 대한 파일 시스템 상태를 나타내는 로컬 트리, 및 서버 상태와 파일 시스템 상태 사이의 알려진 싱크된 상태를 나타내는 싱크 트리 사이의 차이를 식별하도록 구성된다. 이러한 차이들에 기초하여, 실행되는 경우, 3개의 트리 데이터 구조체들이 동일하게 될 동기화된 상태를 향해 서버 상태 및 시스템 상태를 수렴시키도록 구성되는 동작들의 세트가 생성될 수 있다.

[0115] 예를 들어, 동작(505)에서, 시스템은 클라이언트 디바이스 상에 또는 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 수정 데이터를 수신할 수 있다. 수정 데이터는 동작(510)에서 원격 트리 또는 로컬 트리를 업데이트하기 위해 사용될 수 있다.

[0116] 수정 데이터는, 콘텐트 관리 서비스와 연관된 하나 이상의 콘텐트 아이템들에 대하여 어떠한 변화들이 발생하였는지를 명시한다. 따라서, 수정 데이터는 클라이언트 관리 시스템으로부터 또는 클라이언트 디바이스로부터(예를 들어, 도 1의 클라이언트 디바이스(150) 상에서 실행 중인 클라이언트 애플리케이션(152)으로부터) 수신될 수 있다. 콘텐트 관리 시스템으로부터 수신된 수정 데이터는 서버 수정 데이터로서 지칭될 수 있다. 서버 수정 데이터는 콘텐트 관리 시스템에 의해 하나 이상의 콘텐트 아이템들에 대하여 어떠한 변화들이 이루어졌는지를 명시하며, 동작(510)에서 원격 트리를 업데이트하기 위해 사용될 수 있다. 클라이언트 디바이스로부터 수신된 수정 데이터는 클라이언트 수정 데이터로서 지칭될 수 있다. 클라이언트 수정 데이터는 클라이언트 디바이스에 의해 하나 이상의 콘텐트 아이템들에 대하여 어떠한 변화들이 이루어졌는지를 명시하며, 동작(510)에서 로컬 트리를 업데이트하기 위해 사용될 수 있다.

[0117] 동작(515)에서, 시스템은, 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 서버 상태 및 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대한 파일 시스템 상태가 동기가 맞는지 여부를 결정할 수 있다. 로컬 트리 및 원격 트리가 파일 시스템 상태 및 서버 상태를 나타내며, 이들이 콘텐트 관리 시스템 및 클라이언트 디바이스에서 발생하는 변화들을 추적하기 위하여 계속적으로 업데이트되기 때문에, 서버 상태 및 시스템 상태가 동기가 맞는지 여부를 결정하는 것은, 트리들 사이의 차이들을 찾기 위하여 싱크 트리에 대하여 로컬 트리 및/또는 원격 트리를 비교함으로써 이루어질 수 있다. 트리들 사이의 차이들을 찾는 이러한 프로세스는 때때로 트리들을 "디핑(diffing)"하는 것"으로서 지칭될 수 있다.

[0118] 일부 실시예들 및 시나리오들에 따르면, 서버 상태와 파일 시스템 상태가 동기가 맞는지 여부를 결정하는 것은, 원격 트리와 싱크 트리 사이의 차이들을 식별하는 것 및/또는 로컬 트리와 싱크 트리 사이의 차이들을 식별하는 것 중 하나 이상을 포함할 수 있다. 원격 트리와 싱크 트리 사이의 차이는, 클라이언트 디바이스에서 반영되지 않았을 수 있는 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 변화들의 발생을 나타낼 수 있다. 유사하게, 로컬 트리와 싱크 트리 사이의 차이는, 콘텐트 관리 시스템에서 반영되지 않았을 수 있는 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대한 변화들의 발생을 나타낼 수 있다.

[0119] 트리들 사이에 차이들이 존재하지 않는 경우, 서버 상태 및 파일 시스템 상태가 동기가 맞으며, 어떠한 동기화 액션들도 필요하지 않다. 따라서, 방법은 동작(505)으로 복귀하고 새로운 수정 데이터를 기다릴 수 있다. 반면, 차이들이 검출되는 경우, 시스템은 동작(520)에서 서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 세트를 생성할 수 있다.

[0120] 생성되는 동작들의 세트는 검출된 하나 이상의 차이들에 의존한다. 예를 들어, 2개의 트리들 사이의 차이가 추가된 콘텐트 아이템인 경우, 생성된 동작들의 세트는 추가된 콘텐트 아이템을 검색하고 이를 추가하는 것을 포함할 수 있다. 2개의 트리들 사이의 차이가 콘텐트 아이템의 삭제인 경우, 생성된 동작들의 세트는 콘텐트 아이템을 삭제하는 것을 포함할 수 있다. 일부 실시예들에 따르면, 동작들의 세트는 또한 트리 제약들이 유지된다는 것을 보장하기 위한 복수의 체크들을 포함할 수 있다. 이하에서 추가로 설명될 바와 같이, 동작들의 세트는, 실행 계류 중인 다른 동작들, 서버 상태, 또는 파일 시스템 상태의 현재 상태와 충돌할 수 있다. 따라서, 시스템은 또한 진행하기 이전에 이러한 충돌들을 해결할 수 있다.

[0121] 이상에서 언급된 바와 같이, 원격 트리와 싱크 트리 사이에 차이들이 존재하는 경우, 클라이언트 디바이스에서 반영되지 않았을 수 있는 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 변화들이 발생하였을 수 있다. 추가적으로, 이러한 시나리오에 있어서, 시스템은, 서버 상태 및 파일 시스템 상태를 수렴시키도록 구성된 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대하여 동작하도록 구성된 동작들의 클라이언트 세트를 생성할 수 있으며, 이러한 동작들의 클라이언트 세트는 동작(525)에서 실행을 위하여 클라이언트 디바이스에 제공될 수 있다.

[0122] 유사하게, 로컬 트리와 싱크 트리 사이에 차이가 존재하는 경우, 콘텐트 관리 시스템에서 반영되지 않았을 수 있는 클라이언트 디바이스에 저장된 콘텐트 아이템들에 대한 변화들이 발생하였을 수 있다. 따라서, 이러한 시나리오에서, 시스템은, 서버 상태 및 파일 시스템 상태를 수렴시키기 위해 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대하여 동작하도록 구성된 동작들의 서버 세트를 생성할 수 있으며, 이러한 동작들의 서버 세트는 동작(525)에서 실행을 위하여 콘텐트 관리 시스템에 제공될 수 있다. 일부 경우들에 있어서, 경우들 둘 모두는 참일 수 있으며, 동작들의 클라이언트 세트 및 동작들의 서버 세트가 생성되고 동작(525)에서 그들의 의

도된 수신자들에게 제공될 수 있다.

[0123] 일단 동작들의 세트(들)가 의도된 수신자(들)에게 제공되면, 방법은 동작(505)으로 복귀하고 새로운 수정 데이터를 기다릴 수 있다. 동작들의 세트(들)는 서버 상태 및 파일 시스템 상태의 수렴을 향해 하나 이상의 단계들을 제공할 수 있거나, 또는 서버 상태와 파일 시스템 상태를 동기시키기 위해 요구되는 모든 단계들을 제공할 수 있다. 예를 들어, 콘텐트 관리 시스템은 동작들의 서버 세트를 수신할 수 있으며, 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대하여 동작들의 서버 세트를 실행할 수 있다. 동작들의 서버 세트의 이러한 실행은, 다시 시스템으로 송신되는 서버 수정 데이터 내에서 검출되고 명시된 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 변화들을 초래한다. 그러면 시스템은 원격 트리를 업데이트하고 서버 상태 및 파일 시스템 상태가 동기가 맞는지 여부를 결정할 수 있다.

[0124] 클라이언트 디바이스는 동작들의 클라이언트 세트를 수신할 수 있으며, 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대하여 동작들의 클라이언트 세트를 실행할 수 있다. 동작들의 클라이언트 세트의 실행은, 시스템으로 전달되는 클라이언트 수정 데이터 내에서 검출되고 명시된, 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대한 변화들을 초래한다. 그러면 시스템은 로컬 트리를 업데이트하고 서버 상태 및 파일 시스템 상태가 동기가 맞는지 여부를 결정할 수 있다. 방법(500)의 이러한 동작들은, 서버 상태 및 파일 시스템 상태가 동기가 맞을 때까지 계속될 수 있다.

[0125] 방법(500)의 동작들은 클라이언트 측 및 서버 측에 대하여 설명된다(예를 들어, 로컬 트리 및 원격 트리, 파일 시스템 상태 및 서버 상태, 동작들의 클라이언트 세트 및 동작들의 서버 세트, 클라이언트 수정 데이터 및 서버 수정 데이터). 다양한 실시예들에 있어서, 2개의 측들과 연관된 동작들은 병렬로, 시퀀스로, 다른 측과 별개로, 또는 조합으로 발생할 수 있다.

[0126] 더 상세하게 설명될 바와 같이, 일부 실시예들에 따르면, 동작들이 실행을 위해 제공되기 이전에, 시스템은, 이들이 불변성(invariant)들 또는 규칙들의 세트를 준수하는지 여부를 결정하기 위하여 동작들을 체크할 수 있다. 동작이 규칙을 위반하는 경우, 시스템은 규칙의 위반과 연관된 해법 프로세스를 실행한다.

[0127] 추가적으로, 일부 실시예들에 따르면, 시스템(예를 들어, 도 2의 클라이언트 동기화 서비스(156)의 스케줄러(230))은 동작들의 세트의 실행을 관리할 수 있다. 예를 들어, 동작들의 세트 내의 각각의 동작은, 태스크, 실행 스레드(thread), 일련의 단계들, 또는 명령어들과 연관될 수 있다. 시스템은, 동작들의 세트를 실행하여 서버 상태와 파일 시스템 상태를 수렴시키기 위하여, 태스크, 스레드, 단계 또는 명령어들 및 클라이언트 디바이스 및/또는 콘텐트 관리 시스템과의 인터페이스를 실행하도록 구성될 수 있다.

[0128] <u>충돌 핸들링</u>

[0129] 도 5에 대하여 이상에서 설명된 바와 같이, 싱크 트리와 원격 트리 사이의 차이들이 식별되며, 이는 서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 클라이언트 세트를 생성하기 위하여 사용된다. 그러나, 일부 경우들에 있어서, 동작들의 클라이언트 세트는 로컬 트리의 현재 상태와 충돌할 수 있다. 유사하게, 싱크 트리와 로컬 트리 사이의 차이들이 식별되며, 이는 서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 서버 세트를 생성하기 위하여 사용된다. 그러나, 동작들의 서버 세트는 원격 트리의 현재 상태와 충돌할 수 있다. 추가적으로 또는 대안적으로, 동작들의 클라이언트 세트 및 동작들의 서버 상태는 서로 충돌하거나 또는 시스템에 의해 유지되는 다른 규칙 또는 불변성을 위반할 수 있다. 따라서, 본 기술의 다양한 실시예들은 이러한 충돌들을 해결함으로써 추가적인 기술적 개선들을 제공한다.

[0130] 예를 들어, 도 2의 클라이언트 동기화 서비스(156) 내의 플래너(225)는 규칙과 충돌하는 동작들의 세트(예를 들어, 동작들의 클라이언트 세트 또는 동작들의 서버 세트) 내의 동작을 식별할 수 있다. 충돌을 식별하기 위해 사용되는 각각의 규칙은 충들에 대한 해법과 연관될 수 있다. 클라이언트 동기화 서비스는, 실행을 위해 동작들의 세트를 제공하기 이전에 충돌에 대한 해법들과 연관된 동작들을 수행함으로써 충돌을 해결하거나 또는 충돌에 대한 해법에 기초하여 동작들의 세트를 업데이트할 수 있다.

[0131] 도 6은 본 기술의 다양한 실시예들에 따른 트리 데이터 구조체들을 사용하여 서버 상태 및 파일 시스템 상태를 동기화할 때의 충돌들을 해결하기 위한 예시적인 방법(600)을 도시한다. 본원에서 설명되는 방법들 및 프로세스들이 특정 순서의 특정 단계들 및 동작들을 가지고 도시될 수 있지만, 달리 언급되지 않는 한, 유사하거나 또는 대안적인 순서들로, 또는 병렬로 수행되는, 추가적이거나, 더 적거나, 또는 대안적인 단계들 및 동작들이 다양한 실시예들의 범위 내에 속한다. 방법(600)은, 예를 들어, 클라이언트 디바이스 상에서 실행 중인 도 2의 클라이언트 동기화 서비스(156)와 같은 시스템에 의해 구현될 수 있다.

[0132] 시스템은 동작(620)에서 서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 세트를 수신할 수 있다. 동작들의 세트는, 예를 들어, 도 5의 방법(500)에 대하여 생성되고 설명된 동작들의 클라이언트 세트, 동작들의 서버 세트, 또는 동작들의 결합된 세트일 수 있다.

[0133] 동작(650)에서, 시스템은 규칙들의 세트에 기초하여 동작들의 세트 내의 하나 이상의 위반들을 식별한다. 규칙들의 세트는 도 2의 클라이언트 동기화 서비스(156)에 의해 저장될 수 있으며, 해결될 동작들에 대한 다수의 충돌들, 제약들, 또는 불변성들을 지정할 수 있다. 규칙들의 세트는 트리 데이터 구조체에 적용되고 동기화 거동을 제어하는 것을 도울 수 있다. 규칙들의 세트 내의 각각의 규칙은 또한 그 규칙의 위반에 대한 해법과 연관되거나 또는 달리 이에 링크될 수 있다. 예를 들어, 해법은 동작들의 세트 내의 하나 이상의 동작들의 변경, 하나 이상의 동작들의 제거, 하나 이상의 동작들의 추가, 서버 상태 또는 파일 시스템 상태에 대한 하나 이상의 추가적인 액션들, 또는 액션들의 조합을 포함할 수 있다.

[0134] 동작들의 세트 내의 각각의 동작에 대하여, 시스템은, 규칙들의 세트 내의 임의의 규칙이 위반되는지 여부를 결정할 수 있다. 규칙이 위반되는 경우, 시스템은, 동작(655)에서, 위반의 해법들 식별하고, 해법을 수행한다. 해법은, 동작들의 세트 내의 하나 이상의 동작들을 수정하는 것, 하나 이상의 동작들을 제거하거나 또는 추가하는 것, 또는 서버 상태 또는 파일 상태에 대한 추가적인 액션들과 같은 액션들을 포함할 수 있다.

[0135] 일단 해법 액션들이 수행되면, 시스템은, 동작(660)에서, 동작들의 세트 및 해법에 기초하여 동작들의 해결되거나 또는 리베이스된(rebased) 세트를 생성할 수 있으며, 동작(665)에서, 동작들의 해결된 세트는 실행을 위하여 적절한 엔티티로 제공할 수 있다. 예를 들어, 동작들의 해결된 세트는 관리되는 실행을 위하여 도 2의 클라이언트 동기화 서비스(156)의 스케줄러(230)에 제공될 수 있다. 대안적으로, 동작들의 세트가 동작들의 클라이언트 세트인 경우, 동작들의 해결된 세트는 클라이언트 디바이스로 제공될 수 있다. 동작들의 세트가 동작들의 서버 세트인 경우, 동작들의 해결된 세트는 콘텐트 관리 서비스로 제공될 수 있다. 추가적으로, 도 6의 방법(600)은 시퀀스로, 병렬로, 또는 다양하고 상이한 순서들로 동작들의 클라이언트 세트 및 동작들의 서버 세트에 대하여 수행될 수 있다.

[0136] 일부 실시예들에 따르면, 각각의 유형의 동작은 규칙들의 동일하거나 또는 상이한 세트와 연관될 수 있다. 예를 들어, 동작 유형들은, 예를 들어, 콘텐트 아이템을 추가하는 것, 콘텐트 아이템을 삭제하는 것, 콘텐트 아이템을 편집하는 것, 콘텐트 아이템을 이동시키는 것, 콘텐트 아이템을 재명명하는 것 등을 포함할 수 있다. 동작들의 세트는, 각기 이상의 동작 유형들 중 하나에 속하는 동작들로 구성될 수 있다. 각각의 동작 유형은 규칙들의 특정 세트와 연관될 수 있다.

[0137] 예시적인 목적들을 위하여, "추가" 동작 유형에 대한 규칙들의 세트는, 콘텐트 아이템들에 대한 파일 식별자들은 반드시 트리 내에 고유해야만 하는 것(예를 들어, 트리 내의 어떠한 2개의 노드들도 동일한 파일 식별자를 갖지 않을 수 있는 것), 콘텐트 아이템의 부모 노드의 파일 식별자를 지정하는 디렉토리 파일 식별자(directory file identifier; "DirFileID")는 반드시 반대편의 트리 데이터 구조체 내에 존재해야만 하는 것, 및 콘텐트 아이템에 대한 DirFileID 및 파일명 조합은 반대편의 트리 내에서 사용되는 않는 것과 같은 규칙들을 포함할 수 있다.

[0138] 본원에서 사용되는 바와 같은 반대편의 트리는 반대편의 엔티티의 상태를 나타내는 트리 데이터 구조체를 지칭한다. 예를 들어, 클라이언트 디바이스 상에서 동작하도록 구성된 동작들의 클라이언트 세트 및 클라이언트 디바이스 상의 파일 시스템에 대한 결과적인 변화들이 로컬 트리 내에 반영될 것이다. 따라서, 동작들의 클라이언트 세트에 대한 반대편의 트리는 원격 트리이다. 유사하게, 동작들의 서버 세트는 실행된 콘텐트 관리 시스템으로 송신되도록 구성되며, 서버 상태에 대한 결과적인 변화들이 원격 트리 내에 반영될 것이다. 따라서, 동작들의 서버 세트에 대한 반대편의 트리는 로컬 트리이다.

[0139] 도 7은 다양한 실시예들에 따른 추가 동작에 대한 규칙의 위반을 예시하는 트리 데이터 구조체의 일 예를 도시한다. 트리 데이터 구조체는 원격 트리(710), 싱크 트리(750), 및 로컬 트리(770)를 포함한다. 로컬 트리(770)를 언급할 때, 원격 트리(710)는 반대편의 트리로서 간주될 수 있다. 반면, 원격 트리(710)를 언급할 때, 로컬 트리(770)가 반대편의 트리로서 간주될 수 있다. 도 7은 원격 트리(710) 내의 노드(712)에 의해 표현되는 콘텐트 아이템을 추가하는 동작들의 세트를 예시한다. 예를 들어, 클라이언트 동기화 서비스는 원격 트리(710)를 싱크 트리(750)를 비교하고, 차이들을 식별하며, 노드(712)의 추가를 포함하는 동작들의 세트를 생성할 수 있다. 노드(712)는 4의 FileID, 3의 DirFileID(이는 노드(712)의 부모인 부모 노드(714)를 언급함), 및 "Hi"의 파일명과 연관된다. 노드(714)는 3의 FileID, 1의 DirFileID(이는 노드(714)의 부모인 루트 노드(716)를 언급함),

및 "Foo"의 파일명과 연관된다.

[0140] 클라이언트 동기화 서비스는 도 6의 방법(600)을 수행할 수 있으며, 노드(712)에 대한 추가 동작이, "추가" 동작 유형에 대하여 콘텐트 아이템의 "디렉토리 파일 식별자(directory file identifier; "DirFileID")가 반드시 반대편의 트리 데이터 구조체 내에 존재해야만 한다"는 규칙을 위반한다는 것을 결정할 수 있다. 이는, 노드(712)의 부모 노드(714)를 언급하는, 3의 파일 ID를 갖는 노드를 갖지 않는 로컬 트리(770)에 의해 도 7에 예시된다. 이는, 예를 들어, 원격 트리(710)와 싱크 트리(750) 사이의 차이들이 결정되고 동작들의 세트가 생성된 이후에, 노드(714)에 대응하는 "Foo" 노드가 반대편의 트리로부터 제거될 때, 발생할 수 있다.

[0141] 이러한 규칙과 연관된 해법은, 싱크 트리(750) 및 로컬 트리(770)를 동기화하기 위하여 로컬 트리(770)로부터 빠진 노드를 싱크 트리(750)로부터 삭제하는 것, 및 원격 트리(710)와 싱트 크리(750)를 리디핑(rediffing)하는 것(예를 들어, 이들 사이의 차이를 찾는 것)을 포함할 수 있다. 도 7에 예시된 시나리오에서, 싱크 트리(750) 내의 노드(754)는 제거될 것이며(758), 디핑 동작들이 원격 트리(710)와 싱크 트리(750) 사이의 차이들을 식별하는 것을 개시할 것이다. 이는 동작들의 세트 내에 노드(714)의 추가 동작뿐만 아니라 노드(712)에 대한 추가 동작의 포함을 야기할 것이다.

[0142] 유사하게, "추가" 동작 유형에 대한 "콘텐트 아이템에 대한 파일 식별자들은 반드시 트리 내에서 고유해야만 한다"라는 규칙의 위반은, 콘텐트 관리 시스템으로부터, 추가되는 노드에 대한 새로운 파일 ID를 요청하는 것 및 노드를 추가할 때 새로운 파일 ID를 사용하는 것을 포함하는 동작들에 의해 해결될 수 있다. "추가" 동작 유형들에 대한 "콘텐트 아이템에 대한 DirFileID 및 파일명 조합은 반대편의 트리 내에서 사용되는 않아야 한다"라는 규칙의 위반은, 콘텐트 아이템들이 동일한지 여부에 대하여 2개의 노드들과 연관된 메타데이터를 통해 체크하는 것을 포함하는 동작들에 의해 해결될 수 있다. 콘텐트 아이템들이 동일한 경우, 추가되는 콘텐트 아이템이 이미 다른 액션들에서 추가되었을 가능성이 있다. 콘텐트 아이템이 동일하지 않은 경우, 추가되는 콘텐트 아이템에 대한 파일명이 재명명될 수 있다. 예를 들어, 추가되는 콘텐트 아이템에 대한 파일명에는 텍스트 "(충돌 버전)"이 첨부될 수 있다.

[0143] <u>점진적 플래너</u>

[0144] 도 3, 도 4, 및 도 7에 도시된 다양한 트리 데이터 구조체들이 상대적으로 적은 수의 노드들을 가지며 구조적으로 상대적으로 단순하지만, 본 시스템에 의해 지원되는 트리 데이터 구조체는, 다수의 레벨들 및 각각의 레벨에서의 잠재적으로 많은 수의 노드들을 갖는 상태로 훨씬 더 크고 복잡할 수 있다. 따라서, 동작 동안 트리 데이터 구조체들을 저장하기 위해 요구되는 메모리 사용량이 상당히 클 수 있으며, 트리 데이터 구조체들에 대하여 동작하기 위해 요구되는 컴퓨팅 시간 및 자원들이 상당히 클 수 있다. 예를 들어, 원격 트리와 싱크 트리 및/또는 로컬 트리와 싱크 트리 사이의 차이들을 찾는 것 및 원격 트리와 싱크 트리 및/또는 로컬 트리와 싱크 트리를 수렴시키기 위하여 요구되는 동작들을 생성하는 것은 많은 양의 메모리, 시간 및 다른 컴퓨팅 자원들을 필요로 할 수 있다.

[0145] 불행히도, 이러한 컴퓨팅 자원들이 한정된다. 예를 들어, 클라이언트 디바이스는 제한된 양의 이용가능 메모리를 가질 수 있으며, 트리들을 디핑하고 동작들을 생성하기 위해 요구되는 시간의 길이는 클라이언트 디바이스, 클라이언트 애플리케이션, 또는 콘텐트 관리 시스템에 의해 제공되는 콘텐트 관리 서비스들의 유용성을 방해할 수 있다. 추가로, 서버 상태와 파일 시스템 상태를 수렴시키기 위해 필요한 시간이 많을수록 어느 하나의 상태에 대한 변화들을 조정(intervene)하는 것은 연산되거나 또는 실행될 동작들의 세트 및/또는 목표 싱크 상태가 유효 기간이 지나게 만들 수 있는 가능성이 더 높다. 따라서, 본 기술의 다양한 실시예들은, 이들을 표현하는 트리 데이터 구조체들에 따라 서버 상태와 파일 시스템 상태를 점진적으로 수렴시킴으로써 추가적인 기술적 개선들을 제공한다.

[0146] 도 8은 본 기술의 다양한 실시예들에 따른 서버 상태 및 파일 시스템 상태를 점진적으로 수렴시키기 위한 예시적인 방법(800)을 도시한다. 본원에서 설명되는 방법들 및 프로세스들이 특정 순서의 특정 단계들 및 동작들을 가지고 도시될 수 있지만, 달리 언급되지 않는 한, 유사하거나 또는 대안적인 순서들로, 또는 병렬로 수행되는, 추가적이거나, 더 적거나, 또는 대안적인 단계들 및 동작들이 다양한 실시예들의 범위 내에 속한다. 방법(800)은, 예를 들어, 클라이언트 디바이스 상에서 실행 중인 도 2의 클라이언트 동기화 서비스(156)와 같은 시스템에 의해 구현될 수 있다.

[0147] 동작(805)에서, 시스템은 원격 트리 또는 로컬 트리를 업데이트하기 위해 사용될 수 있는 수정 데이터를 수신할 수 있다. 예를 들어, 콘텐트 관리 시스템에 의해 저장된 하나 이상의 콘텐트 아이템들과 연관된 수정들 또는 다

른 액션들(예를 들어, 편집, 추가, 삭제, 이동, 또는 재명명)을 명시하는 서버 수정 데이터가 콘텐트 관리 시스템으로부터 수신될 수 있다. 서버 수정 데이터는, 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들의 서버 상태를 나타내는 원격 트리를 업데이트하기 위해 사용될 수 있다. 유사하게, 클라이언트 수정 데이터는 클라이언트 디바이스(예를 들어, 클라이언트 애플리케이션)으로부터 수신될 수 있으며, 클라이언트 디바이스 상에 저장된 하나 이상의 콘텐트 아이템들과 연관된 수정들 또는 다른 액션들을 명시할 수 있다. 클라이언트 수정 데이터는, 클라이언트 디바이스 상에 저장된 콘텐트 아이템들의 파일 시스템 상태를 나타내는 로컬 트리를 업데이트하기 위해 사용될 수 있다.

[0148] 콘텐트 아이템들과 연관된 수정들을 명시하는 수신된 수정 데이터에 기초하여, 시스템은, 동작(810)에서, 수정된 콘텐트 아이템들에 대응하는 노드들을 식별하고 노드를 수정된 콘텐트 아이템들의 리스트에 추가(예를 들어, 노드들과 연관된 파일 식별자를 수정된 콘텐트 아이템들의 리스트에 추가)할 수 있다. 동작들(805 및 810)은, 시스템이 방법(800)의 다음 스테이지로 진행하기 이전에 소정의 시간 동안 연속적으로 발생할 수 있다. 예를 들어, 추가적인 수정 데이터가 수신될 수 있으며, 시스템에 의해 관리되는 트리들을 업데이트하고 노드들을 수정된 콘텐트 아이템들의 리스트에 추가하기 위해 사용될 수 있다.

[0149] 서버 상태와 파일 시스템 상태를 점진적으로 수렴시키기 위하여, 동작(815)에서, 시스템은 수정된 콘텐트 아이템들의 리스트 내에서 각각의 노드를 취하고 노드가 어떻게 수정되었는지(예를 들어, 어떠한 액션들이 노드와 연관되는지)를 결정한다. 일부 실시예들에 있어서, 수정 데이터는 노드에 대한 수정을 명시할 수 있다. 그러나, 다른 실시예들에 있어서, 시스템은, 원격 트리와 싱크 트리의 비교 및/또는 로컬 트리와 싱크 트리의 비교에 기초하여 노드에 대한 수정들을 결정할 수 있다. 예를 들어, 수정들은, 노드의 추가, 노드의 삭제, 노드의 편집, 또는 노드의 이동을 포함할 수 있다.

[0150] 수정된 콘텐트 아이템들의 리스트 내의 각각의 노드 또는 노드에 대한 파일 식별자에 대하여, 시스템은, 존재하는 경우, 노드에 대하여 어떠한 수정이 수행되었는지를 결정하기 위하여 일련의 체크들을 수행할 수 있다. 예를 들어, 시스템은 파일 식별자가 싱크 트리 내에는 존재하지만 원격 트리 내에는 존재하지 않는지 여부를 결정할 수 있다. 원격 트리에서 발견되지 않는 싱크 트리 내의 파일 식별자는, 노드가 원격 트리에 의해 표현되는 서버 상태로부터 삭제되었다는 것을 나타낼 수 있다. 따라서, 클라이언트 동기화 서비스는, 노드에 대한 삭제 수정이 원격 트리 상에서 발생하였다는 것을 결정할 수 있다. 유사하게, 시스템은 또한, 파일 식별자가 싱크 트리 내에는 존재하지만 로컬 트리 내에는 존재하지 않는지 여부를 결정할 수 있다. 로컬 트리에서 발견되지 않는 싱크 트리 내의 파일 식별자는, 노드가 로컬 트리에 의해 표현되는 파일 시스템 상태로부터 삭제되었다는 것을 나타낼 수 있다. 따라서, 클라이언트 동기화 서비스는, 노드에 대한 삭제 수정이 로컬 트리 상에서 발생하였다는 것을 결정할 수 있다.

[0151] 노드에 대하여 편집 수정이 수행되었는지 여부를 결정하기 위하여, 시스템은 싱크 트리 내의 노드에 대한 메타데이터와 원격 트리 및/또는 로컬 트리 내의 대응하는 노드(예를 들어, 동일한 파일 식별자를 갖는 노드)에 대한 메타데이터를 비교할 수 있다. 메타데이터는, 노드에 의해 표현되는 콘텐트 아이템이 편집되었는지 여부를 결정하기 위해 사용될 수 있는 정보를 포함할 수 있다. 예를 들어, 메타데이터는, 콘텐트 아이템 내의 데이터 또는 이의 일 부분에 기초하여 생성되는 하나 이상의 해시 값들을 포함할 수 있다. 메타데이터는 추가적으로 또는 대안적으로 콘텐트 아이템에 대한 크기 값, 마지막으로 수정된 값, 또는 다른 값을 포함할 수 있다. 메타데이터가 일치하지 않는 경우, 콘텐트 아이템의 편집이 원격 트리에 의해 표현되는 서버 상태에서 및/또는 로컬 트리에 의해 표현되는 파일 시스템 상태에서 편집되었을 수 있다. 따라서, 시스템은, 원격 트리 및/또는 로컬 트리 상의 노드에 대하여 편집 액션이 발생하였다는 것을 결정할 수 있다.

[0152] 원격 트리 내의 노드가 이동되었는지 여부를 결정하기 위하여, 시스템은 싱크 트리 내의 노드에 대한 위치와 원격 트리 및/또는 로컬 트리 내의 대응하는 노드(예를 들어, 동일한 파일 식별자를 갖는 노드)에 대한 위치를 비교할 수 있다. 위치는, 예를 들어, 노드가 위치되는 경로, 파일명, 및/또는 노드의 부모의 파일 식별자를 명시하는 디렉토리 파일 식별자(directory file identifier; "DirFileID")를 포함할 수 있다. 위치들이 일치하는 경우, 어떠한 이동도 발생하지 않았을 수 있다. 반면, 위치들이 일치하지 않는 경우, 콘텐트 아이템의 이동이 원격 트리 또는 로컬 트리 내에서 발생하였을 수 있다. 따라서, 클라이언트 동기화 서비스는, 원격 트리 및/또는 로컬 트리 상의 노드에 대하여 이동 액션이 발생하였다는 것을 결정할 수 있다.

[0153] 노드가 원격 트리에 추가되었는지 여부를 결정하기 위하여, 시스템은, 수정된 콘텐트 아이템들의 리스트 내의 파일 식별자가 원격 트리 내에 또는 로컬 트리 내에 존재하지만 싱크 트리 내에는 존재하지 않는지 여부를 결정할 수 있다. 파일 식별자가 원격 트리 또는 로컬 트리 내에서 발견되고 싱크 트리 내에서는 발견되지 않는

경우, 시스템은, 이러한 노드에 대한 추가 수정이 발생하였다는 것을 결정할 수 있다.

[0154] 일단 수정된 콘텐트 아이템들의 리스트 내의 노드들에 대한 하나 이상의 수정들이 결정되면, 시스템은, 동작(820)에서, 이러한 수정들 중 임의의 수정이 종속성들을 갖는지 여부를 결정할 수 있다. 도 9와 관련하여 추가로 예시될 바와 같이, 노드에 대한 수정은, 예를 들어, 수정이 다른 수정들이 먼저 발생하지 않고는 실행될 수 없을 때 종속성을 갖는다.

[0155] 수정이 종속성을 갖지 않는 경우, 시스템은, 동작(825)에서, 액션들의 차단되지 않은 리스트에 수정을 추가한다. 수정이 종속성을 갖는 경우, 수정은 동작(830)에 있는 시간 동안 차단되며, 다른 수정이 먼저 프로세싱되지 않으면 실행될 수 없다. 따라서, 프로세스는 추가적인 수정들을 기다리기 위하여 동작(805)으로 복귀한다. 수정들의 각각이 프로세싱된 이후에, 시스템은 수정된 콘텐트 아이템들의 리스트로부터 수정들과 연관된 파일 식별자들을 지울 수 있다.

[0156] 도 9는 다양한 실시예들에 따른 트리 데이터 구조체들의 일 예를 도시한다. 도 9에 도시된 트리 데이터 구조체들은 클라이언트 디바이스에 저장되고 도 2에 도시된 클라이언트 동기화 서비스(156)와 같은 시스템에 의해 관리될 수 있다. 예시의 목적을 위하여, 단지 원격 트리(910) 및 싱크 트리(950)만이 도 9에 도시되고 설명된다. 유사한 동작들 및 설명이 또한 로컬 트리에도 마찬가지로 적용될 수 있다.

[0157] 원격 트리(910)는, 1의 파일 식별자를 갖는 루트 노드(912), 5의 파일 식별자 및 "Foo"의 파일명을 갖는 노드(914), 6의 파일 식별자 및 "Bar"의 파일명을 갖는 노드(916), 및 7의 파일 식별자 및 "Bye"의 파일 명을 갖는 노드(918)를 포함한다. 싱크 트리는 1의 파일 식별자를 갖는 루트 노드(952)를 포함한다.

[0158] 도 9에 도시된 트리 데이터 구조체들에 기초하여, 시스템은, 도 9에서 참조부호(980)에 의해 예시되는 바와 같이, 5, 6, 및 7의 파일 식별자들을 갖는 노드들이 동작(810)에서 수정되었으며 수정된 콘텐트 아이템들의 리스트에 추가되었다는 것을 식별했을 수 있다. 동작(815)에서, 시스템은 수정된 콘텐트 아이템들의 리스트 내의 노드들에 대한 수정들의 리스트를 결정한다. 원격 트리(910)와 싱크 트리(950)의 비교에 의해 보여지는 바와 같이, 노드들(914, 916, 및 918)이 원격 트리(910)에 추가되었다. 보다 더 구체적으로, 도 9에서 참조부호(982)에 의해 예시되는 바와 같이, 파일 식별자 6 및 명칭 "Bar"를 갖는 노드(916)가 파일 식별자 5를 갖는 노드(914)에 자식으로서 추가되었다. 이는 참조부호(982)에서 "추가(6, 5, Bar)" 엔트리에 의해 표현된다. 파일 식별자 7 및 명칭 "Bye"를 갖는 노드(918)는 파일 식별자 5를 갖는 노드(914)에 자식으로서 추가되었다. 이는 참조부호(982)에서 "추가(7, 5, Bye)" 엔트리에 의해 표현된다. 파일 식별자 5 및 명칭 "Foo"를 갖는 노드(914)가 파일 식별자 1을 갖는 루트 노드(912)에 자식으로서 추가되었다. 이는 참조부호(982)에서 "추가(5, /root, Foo)" 엔트리에 의해 표현된다.

[0159] 동작(820)에서, 시스템은, 노드(914)의 추가 수정이 종속성을 갖지 않으며, 결과적으로, 차단되지 않는다는 것을 결정한다. 따라서, 시스템은, 동작(825)에서, 노드(914)와 연관된 수정(예를 들어, 참조부호(982)에서 "추가(5, /root, Foo)"에 의해 표현되는 수정)을 액션들의 차단되지 않은 리스트에 추가한다. 이는 도 9에서 참조부호들(984)에서 보여 진다. 반면, 참조부호(982)에서 "추가(6, 5, Bar)" 및 "추가(7, 5, Bye)" 엔트리들에 의해 표현되는 노드들(916 및 918)에 대한 수정들은 먼저 발생하는 "추가(5, /root, Foo)"에 의해 표현되는 수정에 종속적이다. 다시 말해서, 노드(916) 및/또는 노드(918)는, 노드(914)가 추가될 때까지 추가될 수 없다. 따라서, 이러한 수정들은 도 9에서 참조부호(986)에 의해 예시된 액션들의 차단된 리스트 내에 포함된다.

[0160] 다시 도 8의 방법(800)을 참조하면, 동작(835)에서, 시스템은 액션들의 차단되지 않은 리스트로부터 수정들의 세트를 선택하고, 선택된 수정들의 세트에 기초하여 동작들의 세트를 생성할 수 있다. 동작들의 세트는 서버 상태 및 파일 시스템 상태를 수렴시키도록 구성된다. 생성되는 동작들의 세트는 차단되지 않은 리스트로부터 선택된 수정들의 세트에 의존한다. 예를 들어, 선택된 수정들의 세트가 도 9에서 노드(914)와 연관된 추가 수정(예를 들어, 참조부호(984)에서 "추가(5, /root, Foo)"에 의해 표현되는 수정)을 포함하는 경우, 생성된 동작들의 세트는 콘텐트 관리 시스템으로부터 추가된 콘텐트 아이템을 검색하는 것 및 이를 클라이언트 디바이스의 로컬 파일 시스템에 추가하는 것을 포함할 수 있다.

[0161] 일부 실시예들에 따르면, 시스템은 동작들의 하나 이상의 세트들을 생성하기 위하여 액션들의 차단되지 않은 리스트로부터 모든 수정들을 선택할 수 있다. 그러나, 일부 시나리오들에 있어서, 차단되지 않은 리스트 내의 수정들의 수가 상당히 높을 수가 있으며, 수정들 전부를 프로세싱하기 위해 요구되는 컴퓨팅 자원들(예를 들어, 메모리 및 프로세싱 시간)이 상당하다. 이러한 기술적인 부담들을 감소시키기 위하여, 시스템은 점진적으로 프로세싱하기 위하여 액션들의 차단되지 않은 리스트 내의 수정들의 더 작은 세트를 선택할 수 있다. 예를 들어,

시스템은 동작들을 생성하기 위하여 수정들의 제 1 또는 상단 X개 또는 퍼센트를 선택할 수 있다. 프로세스의 추가적인 반복들에서, 차단되지 않은 리스트들 내의 나머지 수정들이 프로세싱될 수 있다.

[0162] 일부 실시예들에 있어서, 차단되지 않은 리스트 내의 수정들은 프로세싱을 위하여 랭킹이 매겨질 수 있다. 수정들은, 예를 들어, 수정 유형(예를 들어, 삭제 수정들이 추가 수정들보다 더 우선순위화된다), 수정과 연관된 메타데이터(예를 들어, 더 작은 크기의 콘텐트 아이템들의 추가 수정들이 더 큰 크기의 콘텐트 아이템들의 추가 수정들보다 더 우선순위화되며, 더 큰 크기의 콘텐트 아이템들의 삭제 수정들은 더 작은 크기의 콘텐트 아이템들의 삭제 수정보다 더 우선순위화되는 등)에 기초하여 랭킹이 매겨질 수 있다.

[0163] 이러한 랭크 규칙들은 시스템에 의해 저장될 수 있으며, 콘텐트 동기화를 위한 다양한 성능 목적들을 달성하도록 설계될 수 있다. 예를 들어, 삭제 수정들은, 새로운 콘텐트 아이템들이 추가될 수 있기 이전에 사용자에 대하여 잠재적으로 제한된 저장 공간을 가능한 한 많이 비우기 위하여 추가 수정들보다 더 우선순위화될 수 있다. 더 작은 콘텐트 아이템들을 추가하는 것은, 가능한 한 빠르게 추가되는 아이템들의 수와 관련하여 가능한 한 많은 진행을 제공하기 위하여 더 큰 콘텐트 아이템들보다 더 우선순위화될 수 있다.

[0164] 동작(835)에서, 시스템은 콘텐트 관리 시스템 및/또는 클라이언트 디바이스로 동작들의 세트를 제공할 수 있다. 이상에서 언급된 바와 같이, 콘텐트 관리 시스템에 의해 수행되는 액션들과 연관된 수정들은 클라이언트 디바이스에서 반영되지 않을 수 있다. 추가적으로, 이러한 시나리오에 있어서, 시스템은 서버 상태 및 파일 시스템 상태를 수렴시키도록 구성된 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대하여 동작하도록 구성된 동작들의 클라이언트 세트를 생성할 수 있으며, 이러한 동작들의 클라이언트 세트는 동작(835)에서 실행을 위하여 클라이언트 디바이스에 제공될 수 있다.

[0165] 반면, 클라이언트 디바이스에 의해 수행되는 액션들과 연관된 수정들은 콘텐트 관리 시스템에서 반영되지 않을 수 있다. 따라서, 이러한 시나리오에서, 시스템은 서버 상태 및 파일 시스템 상태를 수렴시키기 위해 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대하여 동작하도록 구성된 동작들의 서버 세트를 생성할 수 있으며, 이러한 동작들의 서버 세트는 동작(835)에서 실행을 위하여 콘텐트 관리 시스템에 제공될 수 있다.

[0166] 일부 경우들에 있어서, 경우들 둘 모두는 참일 수 있으며, 동작들의 클라이언트 세트 및 동작들의 서버 세트가 생성되고 동작(835)에서 그들의 의도된 수신자들에게 제공될 수 있다. 동작들의 세트는 또한 트리 제약들이 유지된다는 것을 보장하기 위한 복수의 체크들을 포함할 수 있다. 예를 들어, 동작들의 세트는 도 6과 관련하여 논의된 바와 같은 다양한 충돌들 또는 제약들을 해결할 수 있다.

[0167] 일단 동작들의 세트(들)가 의도된 수신자(들)에게 제공되면, 방법은 동작(805)으로 복귀하고 새로운 수정 데이터를 기다릴 수 있다. 예를 들어, 도 9에 예시된 시나리오에 대하여, 동작들의 세트는, 콘텐트 관리 시스템으로부터 노드(914)와 연관된 콘텐트 아이템을 검색하는 것 및 이를 클라이언트 디바이스의 로컬 파일 시스템에 추가하는 것을 포함할 수 있다. 이는, 로컬 트리(도 9에는 도시되지 않음) 및 싱크 트리(950) 내에 노드(914)에 대응하는 노드의 추가를 야기할 것이다. 도 8의 프로세스(800)의 다음 반복에서, 참조부호(982)에서 "추가(6, 5, Bar)" 및 "추가(7, 5, Bye)" 엔트리들에 의해 표현되는 노드(916) 및 노드(918)의 추가 수정은 더 이상 차단되지 않으며, 이는 그들의 부모인 노드(914)가 이미 싱크 트리에 추가되었기 때문이다. 따라서, 참조부호(982)에서 "추가(6, 5, Bar)" 및 "추가(7, 5, Bye)"에 의해 표현되는 노드(916) 및 노드(918)의 추가 수정은 액션들의 차단되지 않은 리스트에 추가될 수 있으며, 서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 하나 이상의 세트들을 생성하기 위해 사용될 수 있다.

[0168] 동작들의 세트(들)는 서버 상태 및 파일 시스템 상태의 점진적인 수렴을 위한 하나 이상의 단계들을 제공할 수 있다. 점진적인 프로세스를 구현하는 것이 때때로 더 복잡할 수 있지만, 점진적인 프로세스는 요구되는 프로세싱 시간의 감소 및 메모리의 감소를 달성할 수 있다. 이러한 그리고 다른 초기의 기술적 개선들이 자연스럽게 추가적인 기술적 개선들로 이어진다. 예를 들어, 프로세싱 시간이 감소되기 때문에, 특정 수정들을 쓸모 없게 또는 유효 기간이 지나게 만드는 클라이언트 디바이스 또는 콘텐트 관리 시스템으로부터의 추가적인 변화들의 가능성이 마찬가지로 감소된다.

[0169] 도 9와 관련하여, 예시의 목적을 위한 리스트로서 콘텐트 아이템들, 수정들, 액션들, 또는 파일 식별자들의 다양한 그룹화가 설명된다. 다른 유형들의 데이터 구조체들이 또한 호환가능하다. 예를 들어, 액션들의 차단되지 않은 리스트는, 대수적 시간으로 소팅(sort)된 데이터를 유지하고, 탐색, 순차적 액세스, 삽입, 및 삭제들을 가능하게 하기 위하여 B-트리 구조체로서 구현될 수 있다.

[0170] <u>스케줄러</u>

[0171] 일부 실시예들에 있어서, 클라이언트 동기화 서비스는 서버 상태 및 파일 시스템 상태를 수렴시키도록 구성된 동작들의 세트 또는 시퀀스를 생성하고, 동작들을 실행을 위하여 콘텐트 관리 시스템 또는 클라이언트 디바이스로 제공할 수 있다. 그러나, 일부 시나리오들에 있어서, 클라이언트 디바이스의 파일 시스템 상의 또는 콘텐트 관리 시스템 상의 변화들은, 동작들의 세트가 실행 프로세스에 있는 동안 생성된 동작들의 세트가 유효 기간이 지나게 되거나 또는 쓸모 없게 되는 것을 초래할 수 있다. 다양한 실시예들은 이러한 그리고 다른 기술적 문제들에 대한 기술적 해법을 제공하는 것과 관련된다. 예를 들어, 클라이언트 동기화 서비스는 클라이언트 디바이스의 파일 시스템 상의 또는 콘텐트 관리 시스템 상의 변화들을 모니터링하고, 필요한 바와 같이 클라이언트 디바이스 및/또는 콘텐트 관리 시스템을 업데이트하도록 구성될 수 있다. 추가로, 클라이언트 동기화 서비스는, 동작들의 동시 실행을 가능하게 함으로써 성능을 개선하고 프로세싱 시간을 감소시키도록 구성될 수 있다.

[0172] 일부 실시예들에 따르면, 도 2에 도시된 클라이언트 동기화 서비스(156)의 플래너(225)는 플랜 또는 동작들의 순서화되지 않은(unordered) 세트로 구성되는 동작들의 플랜을 생성할 수 있다. 플랜 내의 모든 동작들을 종속성들을 갖지 않으며, 결과적으로, 별개의 스레드들로 동시에 또는 임의의 순서로 실행될 수 있다. 일부 실시예들에 따르면, 플랜 내의 동작들은, 상태들 및 트리 데이터 구조체들을 수렴시키기 위하여 콘텐트 관리 시스템 및/또는 클라이언트 디바이스에 의해 취해질 수 있는 추상적인 명령어들이다. 예시적인 명령어들은, 콘텐트 아이템의 원격 또는 로컬 추가, 콘텐트 아이템의 원격 또는 로컬 삭제, 콘텐트 아이템의 원격 또는 로컬 편집, 또는 콘텐트 아이템의 원격 또는 로컬 이동을 포함할 수 있다.

[0173] 도 2에 도시된 클라이언트 동기화 서비스(156)의 스케줄러(230)는, 플래너(225)로부터 동작들의 플랜을 수신하고, 플랜 내의 동작들의 실행을 관리하며, 플랜이 업데이트되었거나 또는 변화되었는지 여부를 결정하고, 업데이트되거나 또는 변화된 플랜의 실행을 관리하도록 구성될 수 있다. 예를 들어, 스케줄러(230)는 플랜 내의 동작들을 구현하기 위해 요구되는 태스크들 및 단계들을 실행하기 위하여 파일 시스템 인터페이스(205) 및 서버 인터페이스(210)를 조정(coordinate)할 수 있다. 이는, 파일 시스템 또는 콘텐트 관리 시스템으로부터 확인을 수신하는 것 또는 네트워크 연결이 존재하지 않을 때 또는 콘텐트 아이템이 어떤 다른 애플리케이션에 의해 잠겨 있을 때 재시도들을 핸들링하는 것과 같은 오류 핸들링 액티비티들을 포함할 수 있다.

[0174] 각각의 동작은 태스크로서 지칭되는 스크립트 또는 스레드에 의해 구현될 수 있다. 태스크는 연관된 동작의 애플리케이션을 조정하며, 이는 동작을 구현하기 위해 요구되는 하나 이상의 단계들을 포함할 수 있다. 예를 들어, "로컬 추가 동작"은, 콘텐트 아이템이 클라이언트 디바이스의 로컬 파일 시스템에 추가되었으며, 결과적으로, 서버 상태와 파일 시스템 상태를 동기화하기 위하여 콘텐트 아이템이 콘텐트 관리 시스템에 추가되어야만 한다는 것을 나타낼 수 있다. 따라서, 로컬 추가 동작은, 로컬 추가 동작을 구현하기 위해 요구되는 하나 이상의 단계들을 포함하는 "로컬 추가 태스크"와 연관될 수 있다. 단계들은, 새로운 콘텐트 아이템을 콘텐트 관리 시스템에 통지하는 것, 콘텐트 아이템을 데이터의 하나 이상의 블록들로 콘텐트 관리 시스템에 업로딩하는 것, 데이터의 모든 블록들이 콘텐트 관리 시스템에 의해 수신되었음을 확인하는 것, 콘텐트 아이템이 손상되지 않았음을 확인하는 것, 콘텐트 아이템에 대한 메타데이터를 콘텐트 관리 시스템에 업로딩하는 것, 및 콘텐트 아이템의 추가를 콘텐트 관리 시스템에서 적절한 위치에 커밋하는 것 중 하나 이상을 포함할 수 있다.

[0175] 태스크는 실행을 개시하고, 다른 이벤트들의 완료를 기다리면서 명확한 포인트에서 일시 중단(suspend)되며, 이벤트가 발생하였을 때 재개되고, 최종적으로 종료될 수 있다. 일부 실시예들에 따르면, 스케줄러(230)는 태스크들을 취소하거나, 재생성하거나, 또는 교체하도록 구성된다. 예를 들어, 서버 상태 또는 파일 시스템 상태에 대한 변화들에 기초하여, 태스크는 이것이 실행되기 이전에 오래된 상태가 될 수 있으며, 스케줄러(230)는 태스크가 실행되기 이전에 오래된 태스크를 취소할 수 있다.

[0176] 이상에서 설명된 바와 같이, 플래너(225)는 트리 데이터 구조체들(예를 들어, 원격 트리, 싱크 트리, 및 로컬 트리)의 세트에 기초하여 동작들의 플랜을 생성할 수 있다. 시간이 경과하면서, 플래너(225)는 트리 데이터 구조체들의 상태에 기초하여 계속해서 동작들의 플랜들을 생성한다. 트리 데이터 구조체들이 서버 상태 및 파일 시스템 상태의 상태를 반영하기 위해 변화하는 경우, 플래너(225)는 또한 이전의 플랜과는 상이한 새로운 업데이트된 플랜을 생성할 수 있다. 스케줄러(230)는 플래너(225)에 의해 생성된 동작들의 각각의 플랜을 실행한다.

[0177] 일부 시나리오들에 있어서, 후속 플랜의 동작들 내의 변화들은, 의도되지 않은 동기화 거동들이 실행 프로세스 중에 있는 이전 플랜 내의 동작과 충돌하는 것을 초래할 수 있다. 예를 들어, 제 1 플랜 내의 동작들이 실행되고 있을 때, 동작들 중 하나 이상이 제 2 플랜에서 취소된다(예를 들어, 존재하지 않는다). 예시하기 위하여, 도 10은, 시간 t1에서, 원격 트리에 의해 표현되는 서버 상태와 로컬 트리에 의해 표현되는 파일 시스템 상태는 모두 일치하는 원격 트리, 싱크 트리, 및 로컬 트리에 의해 도시되는 바와 같이 동기화되는, 예시적인 시나리오

를 도시한다. 이러한 동기화된 상태에 기초하여, 플래너(225)는 t1에서 어떠한 동작들도 갖지 않는 플랜(예를 들어, 빈 플랜)을 생성할 수 있다.

[0178] 클라이언트 디바이스 상의 사용자는 로컬 파일 시스템으로부터 콘텐트 아이템 A를 삭제하거나 또는 클라이언트 동기화 서비스(156)에 의해 관리되는 폴더 밖으로 콘텐트 아이템 A를 이동시킬 수 있으며, 이는 시간 t2에서 로컬 트리로부터의 노드 A의 제거에 의해 반영된다. 플래너(225)는, 시간 t2에서 트리 데이터 구조체들의 상태에 기초하여 동작 로컬삭제(A)를 포함하는 플랜을 생성할 수 있다. 스케줄러(230)는 로컬삭제(A) 동작을 구현하기 위해 요구되는 태스크들 또는 단계들을 개시할 수 있다. 이러한 단계들은 콘텐트 아이템 A를 삭제하기 위하여 명령어들을 콘텐트 관리 시스템으로 송신하는 것을 포함할 수 있다.

[0179] 콘텐트 아이템 A를 삭제하기 위한 명령어들이 콘텐트 관리 시스템으로 송신된 이후에, 클라이언트 디바이스 상의 사용자는 콘텐트 아이템 A의 삭제를 실행취소하거나 또는 콘텐트 아이템 A를 다시 이전 위치로 이동시킬 수 있다. 로컬 트리는 시간 t3에서 이러한 새로운 액션에 기초하여 업데이트되며, 플래너는 어떠한 동작들도 갖지 않는 상태로 비어 있는 새로운 플랜을 생성할 수 있다. 다시 한번, 시간 t3에서 트리 데이터 구조체들이 일치되며 시스템은 동기화된 상태이다.

[0180] 그러나, 콘텐트 아이템 A를 삭제하기 위한 명령어들이 콘텐트 관리 시스템으로 송신되었기 때문에, 콘텐트 관리 시스템은 서버 상태로부터 콘텐트 아이템 A를 삭제한다. 스케줄러(230)가 콘텐트 아이템 A의 삭제를 취소하려고 시도할 수 있지만, 명령어들이 이미 송신되어 콘텐트 관리 시스템에 의해 완료되었을 수 있다. 서버에서의 이러한 변화는 클라이언트 동기화 서버(156)로 통신되며, 이는 시간 t4에서 노드 A를 삭제함으로써 원격 트리를 업데이트한다. 플래너(225)는 원격 트리에서의 변화들 및 원격 트리와 싱크 트리 사이의 차이를 알아차리고, 콘텐트 아이템 A가 서버 상태에서 제거되었음을 결정할 수 있다. 따라서, 플래너(225)는 시간 t4에서 원격삭제(A) 동작을 갖는 플랜을 생성할 것이다. 서버 상태와 파일 시스템 상태를 동기화하기 위한 노력으로, 콘텐트 아이템 A는 최종적으로 클라이언트 디바이스 및 로컬 트리로부터 삭제될 것이다.

[0181] 문제는, 서버 상태로부터 콘텐트 아이템 A의 제거, 원격삭제(A) 동작의 생성, 및 파일 시스템 상태로부터의 콘텐트 아이템 A의 최종적인 제거 모두가 의도되지 않으며, 이는 사용자에 대하여 결국에는 추가적인 문제들을 초래할 수 있다. 추가로, 일부 경우들에 있어서, 애플리케이션들 또는 프로세스들이 또한 콘텐트 아이템들을 액세스할 수 있으며, 비의도적 동기화 거동이 많은 추가적인 기술적 이슈들을 초래할 수 있다. 다양한 실시예들은 서버 상태와 파일 시스템 상태 사이의 콘텐트 아이템들의 동기화에 있어서 의도되지 않은 결과들을 방지하는 것에 관한 것이다.

[0182] 일부 실시예들에 따르면, 더 이상 동작들의 플랜 내에 존재하지 않는 오래된 동작에 대한 태스크를 취소할 때, 스케줄러(230)는 다른 태스크들의 실행을 개시하도록 진행하기 이전에 취소가 완료되는 것을 기다릴 수 있다. 예를 들어, 스케줄러(230)는, 다른 태스크들을 가지고 진행하기 이전에 클라이언트 디바이스 또는 콘텐트 관리 시스템으로부터 취소의 확인을 수신하기를 기다릴 수 있다. 스케줄러(230)는 태스크가 개시되었는지 여부를 결정할 수 있으며, 태스크가 개시되지 않았던 경우, 스케줄러는 태스크를 취소하고, 태스크가 더 이상 실행을 기다리지 않는다는 것을 확인할 수 있다. 태스크가 개시되었던 경우, 확인은 클라이언트 디바이스 또는 콘텐트 관리 시스템으로부터 올 수 있으며, 취소된 태스크와 연관된 단계들 전부가 실행취소되었음을 스케줄러에 통지할 수 있다. 일부 구현예들에 따르면, 스케줄러(230)는, 일단 태스크가 개시되었으면 태스크의 취소를 허용하지 않는다. 이는, 모든 태스크들 또는, 태스크들 또는 태스크 유형들(서버 상태와의 동기화를 위해 파일 시스템 상태 상의 업데이트를 콘텐트 관리 시스템으로 전송하는 커밋 태스크)의 특정 서브세트에 대한 경우일 수 있다.

[0183] 성능을 개선하고 태스크들의 취소뿐만 아니라 태스크들의 동시 실행을 가능하게 하기 위하여, 스케줄러(230)는 또한, 동작들의 제 1 플랜과 동작들의 업데이트된 제 2 플랜 사이의 차이들에 기초하여 태스크들의 실행 및 취소를 관리하도록 구성될 수 있다. 도 11은 본 기술의 다양한 실시예들에 따른 동작들의 2개의 플랜들의 예시적인 벤 다이어그램(1100) 표현을 도시한다. 플래너(225)는 동작들의 제 1 세트를 가지고 플랜 1(1110)을 생성하고, 트리 데이터 구조체들에 대한 업데이트를 수신하며, 동자들의 제 2 세트를 가지고 업데이트된 플랜 2(1120)를 생성할 수 있다.

[0184] 플랜 1(1110) 및 플랜 2(1120)는, 벤 다이어그램(1100)의 부분(1130)에 의해 표현되는 복수의 공통 동작들을 공유할 수 있다. 플랜 1(1110) 및 플랜 2(1120)는 또한 공통되지 않는 복수의 동작들을 포함할 수 있다. 예를 들어, 플랜 2(1120) 내에 존재하지 않는 플랜 1(1110) 내의 동작들은, 플래너(225)에 의해 검출된 트리 구조체들에 대한 업데이트들에 기초하여 오래되고 더 이상 통용되지 않는다. 플랜 1(1110)의 이러한 오래된 동작들은 벤 다이어그램(1100)의 부분(1140)에 의해 표현된다. 플랜 1(1110) 내에 존재하지 않는 플랜 2(1120) 내의 새로운

동작들은 부분(1150)에 의해 표현된다. 플랜 1(1110)과 플랜 2(1120) 사이의 차이들 및 공통점들을 나타내는 부분들(1130, 1140, 및 1150)의 각각은, 트리 데이터 구조체 내에 반영된 파일 시스템 상태와 서버 상태에 대한 업데이트들에 의존하여 어떠한 동작들도 포함하지 않거나 또는 다수의 동작들을 포함할 수 있다.

[0185] 부분(1140) 내의 동작들이 더 이상 가장 최신의 플랜 내에 존재하지 않기 때문에, 스케줄러(230)는 이러한 동작들과 연관된 태스크들을 취소할 수 있다. 의도되지 않은 동기화 거동을 방지하기 위하여, 플랜 1 내에(예를 들어, 부분(1150) 내에) 존재하지 않는 플랜 2 내의 동작들과 연관된 태스크들은, 부분(1140) 내의 동작과 연관된 태스크들의 취소가 완료될 때까지 연기된다. 그러나, 각각의 플랜 내의 동작들이 동시에 실행될 수 있도록 구성되기 때문에, 부분(1130)에 의해 표현되는 플랜 1 및 플랜 2의 교차부 내의 동작들과 연관된 태스크들은, 그들의 완료를 기다릴 필요 없이 부분(1140) 내의 동작과 연관된 태스크들의 취소와 동시에 실행될 수 있다. 부분(1130)과 연관된 태스크들의 실행 및 부분(1140)과 연관된 태스크의 동시 취소를 가능하게 함으로써, 이용가능한 컴퓨팅 자원들의 더 효율적인 사용뿐만 아니라 프로세싱 시간에서의 감소가 달성될 수 있다.

[0186] 도 12는 본 기술의 다양한 실시예들에 따른 동작들의 플랜들에서의 변화들을 관리하기 위한 예시적인 방법을 도시한다. 본원에서 설명되는 방법들 및 프로세스들이 특정 순서의 특정 단계들 및 동작들을 가지고 도시될 수 있지만, 달리 언급되지 않는 한, 유사하거나 또는 대안적인 순서들로, 또는 병렬로 수행되는, 추가적이거나, 더 적거나, 또는 대안적인 단계들 및 동작들이 다양한 실시예들의 범위 내에 속한다. 방법(1200)은, 예를 들어, 클라이언트 디바이스 상에서 실행 중인 도 2의 클라이언트 동기화 서비스(156)와 같은 시스템에 의해 구현될 수 있다.

[0187] 시스템은 콘텐트 관리 서비스와 연관된 콘텐트 아이템들과 관련하여 콘텐트 관리 시스템 및/또는 클라이언트 디바이스로부터 업데이트들을 수신하도록 구성될 수 있다. 예를 들어, 시스템은 콘텐트 관리 서비스에 의해 저장된 콘텐트 아이템들에 대한 서버 수정 데이터를 수신하고, 서버 수정 데이터에 기초하여 원격 트리를 업데이트할 수 있다. 원격 트리는 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 서버 상태를 나타낸다. 시스템은 또한 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대한 클라이언트 수정 데이터를 수신하고, 클라이언트 수정 데이터에 기초하여 로컬 트리를 업데이트할 수 있다. 로컬 트리는 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대한 파일 시스템 상태를 나타낸다.

[0188] 동작(1205)에서, 시스템은 콘텐트 관리 시스템과 연관된 서버 상태와 클라이언트 디바이스와 연관된 파일 시스템 상태를 수렴시키도록 구성된 동작들의 제 1 세트를 수신할 수 있다. 예를 들어, 시스템은 싱크 트리와 원격 트리 또는 싱크 트리와 로컬 트리 사이의 차이들을 식별하고, 트리들 사이의 임의의 차이에 기초하여 동작들의 제 1 세트를 생성할 수 있다. 싱크 트리는 서버 상태와 파일 시스템 상태 사이의 알려진 싱크된 상태를 나타낸다.

[0189] 시스템은 동작들의 제 1 세트를 구현하는 것을 시작할 수 있다. 예를 들어, 일부 경우들에 있어서, 동작들은 실행을 위해 콘텐트 관리 시스템 및/또는 클라이언트 디바이스로 송신될 준비가 된 포맷으로 존재한다. 다른 경우들에 있어서, 동작들은, 시스템에 의해 관리될 수 있는 하나 이상의 태스크들, 스크립트들, 또는 실행 스레드들로 변환될 수 있다. 시스템은, 서버 상태와 파일 시스템 상태를 수렴시키기 위하여 태스크들, 스크립트들, 또는 실행 스레드들에 따라 콘텐트 관리 시스템 및/또는 클라이언트 디바이스와 인터페이스할 수 있다.

[0190] 이러한 시간 동안, 시스템은, 콘텐트 관리 서비스와 연관된 콘텐트 아이템들에 관하여 콘텐트 관리 시스템 및/또는 클라이언트 디바이스로부터 수정 데이터를 계속해서 수신할 수 있다. 수정 데이터에 기초하여, 시스템은 원격 트리 또는 로컬 트리를 업데이트하고, 트리 데이터 구조체들에 대한 업데이트들에 기초하여 동작들의 제 2 세트를 생성할 수 있다. 동작(1210)에서, 시스템은 동작들의 제 2 세트를 수신할 수 있다.

[0191] 동작(1215)에서, 시스템은, 존재하는 경우, 동작들의 제 2 세트 내에 존재하지 않는 동작들의 제 1 세트 내의 제 1 동작을 식별한다. 시스템이 동작들의 제 2 세트 내에 존재하지 않는 동작들의 제 1 세트 내의 동작을 찾는 경우, 이러한 동작은 수정 데이터 내에 명시된 변화들의 결과로서 오래되고 유효 기간이 지났을 수 있다. 따라서, 시스템은, 동작(1220)에서, 제 1 동작의 취소를 개시할 수 있다. 제 1 동작의 취소는 다수의 단계들, 단계들에 대한 다수의 확인 수령(receipt)들, 및 사소한 양의 프로세싱 시간을 포함할 수 있다.

[0192] 동작(1225)에서, 시스템은, 존재하는 경우, 동작들의 제 1 세트 및 동작들의 제 2 세트 둘 모두 내에 포함된 제 2 동작을 식별한다. 시스템이 동작들의 제 1 세트 및 동작들의 제 2 세트 둘 모두 내의 동작을 찾는 경우, 이러한 동작은 수정 데이터 내에 명시된 변화들에도 불구하고 여전히 유효할 수 있다. 추가로, 동작들의 세트들 둘 모두 내에 있는 동작들이 세트 내의 다른 동작들에 대하여 동시에 또는 임의의 순서로 실행될 수 있도록 구성되

기 때문에, 제 2 동작은, 제 1 동작이 취소되는 동안 계속해서 실행될 수 있다. 따라서, 시스템은, 제 1 동작의 취소가 완료되는 것을 기다리지 않고 동작(1230)에서 제 2 동작의 실행을 개시할 것이다.

[0193] 동작(1235)에서, 시스템은, 존재하는 경우, 동작들의 제 2 세트 내에는 존재하지만 동작들의 제 1 세트 내에는 존재하지 않는 제 3 동작을 식별한다. 시스템이 동작들의 제 1 세트 내에 존재하지 않는 동작들의 제 2 세트 내의 동작을 찾는 경우, 이러한 동작은 수정 데이터 내에 명시된 변화들의 결과로서 새로운 동작일 수 있다. 의도되지 않은 결과들을 방지하기 위하여, 시스템은 제 1 동작의 취소의 완료를 기다리는 것을 개시할 것이다. 동작(1240)에서, 시스템은 제 1 동작이 취소를 완료하였음을 결정할 수 있으며, 결과적으로, 동작(1245)에서, 제 3 동작의 실행을 개시할 수 있다.

[0194] <u>트리 데이터 구조체 저장 - 파일명 저장 공간의 감소</u>

[0195] 클라이언트 동기화 서비스(156)는, 예를 들어, 하드 디스크, 고체 상태 메모리, 또는 다른 유형들의 컴퓨터 판독가능 매체와 같은 영구 저장 디바이스 상에 트리 데이터 구조체들(예를 들어, 원격 트리, 싱크 트리, 및 로컬 트리)을 저장할 수 있다. 성능을 개선하고, 프로세싱 시간을 감소시키며, 유효 기간이 지난 동작들을 감소시키기 위하여, 클라이언트 동기화 서비스(156)는 기동 시에 트리 데이터 구조체들을 메모리(예를 들어, 랜덤 액세스 메모리 또는 고속 메모리의 캐시) 내에 트리 데이터 구조체들을 로딩하고, 메모리 내의 트리 데이터 구조체들에 대하여 동기화 기능들을 수행할 수 있다. 영구 저장 디바이스 상의 데이터 용량이 제한되며 이러한 데이터 자원들의 보존이 중요하다. 메모리에 대하여 데이터 용량이 훨씬 더 제한되고 비싸며, 이러한 데이터 자원들의 보존이 매우 중요하다.

[0196] 운영 시스템 또는 클라이언트 애플리케이션에 의존하여, 콘텐트 아이템들에 대한 파일명들은 약 1024 바이트의 크기일 수 있으며, 노드 내의 최대 데이터 컴포넌트일 수 있다. 예를 들어, 백만 개의 노드들을 가지면, 단지 노드들에 대한 파일명들의 크기만도 최소 1 기가바이트에 이를 수 있다. 이상에서 언급된 바와 같이, 클라이언트 동기화 서비스(156)는 서버 상태와 파일 시스템 상태와 로컬 트리의 동기화에 도움을 주도록 구성되며, 싱크 트리 및 원격 트리는 3개의 모든 트리들이 동등할 때의 동기화된 상태를 반영한다. 따라서, 노드들 내에 콘텐트 아이템들의 파일명들을 저장하는데 있어서 어떤 중복(redundancy)이 존재할 가능성이 있다.

[0197] 본 기술의 다양한 실시예들은, 파일명들의 중복을 감소시킴으로써 트리 데이터 구조체들에 대한 파일명들을 저장하기 위해 필요한 메모리의 양을 감소시키고 노드들의 측면을 감소시키는 것을 목적으로 한다. 노드 내에 파일명을 저장하는 대신에, 클라이언트 동기화 서비스(156)는, 트리 데이터 구조체들 내의 노드들에 대한 파일명들을 파일명 어레이 내에 저장하고 노드 내에 파일명에 대한 참조를 저장하도록 구성된다. 결과적으로, 일단 파일명이 파일명 어레이 내에 저장되면, 그 파일명을 갖는 콘텐트 아이템을 갖는 임의의 노드는 노드 내에 저장된 참조를 사용하여 파일명을 액세스할 수 있다. 일부 실시예들에 있어서, 노드 내에 저장된 파일명에 대한 참조는, 파일명 어레이 내의 파일명의 오프셋, 위치(location), 또는 포지션(position)을 나타내는 정수 값일 수 있다.

[0198] 도 13은 본 기술의 다양한 실시예들에 따른 파일명 어레이(1310)의 일 예시를 도시한다. 파일명들 "Pictures," "a.jpg," 및 "Documents"(전체적으로 도시되지는 않음)을 저장하는 파일명 어레이(1310)가 도시된다. 파일명 어레이(1310) 내에서 "Pictures" 파일명은 위치 0에 도시되며, "a.jpg" 파일명은 위치 9에 도시되고, "Documents" 파일명은 위치 16에 도시된다. 파일명 어레이(1310) 내에서, 구분자(separator)(1314)(예를 들어, 널 문자(null character))가 각각의 파일명을 분리한다.

[0199] 따라서, 노드에 대한 파일명을 조회(lookup)하기 위하여, 클라이언트 동기화 서비스(156)는 간단하게, 파일명 어레이(1310) 내의 노드에 대한 파일명의 위치를 나타내는 노드 내에 저장된 파일면에 대한 참조를 액세스할 수 있다. 클라이언트 동기화 서비스(156)는 참조에 의해 지정된 파일명 어레이(1310) 내의 위치에서 콘텐트 아이템의 파일명을 검색할 수 있다. 예를 들어, 클라이언트 동기화 서비스(156)는 참조에 지정된 위치에서 파일명을 판독하는 것을 시작하고 구분자(1314)에 도달할 때 이를 중지할 수 있다.

[0200] 일부 경우들에 있어서, 클라이언트 동기화 서비스(156)가 또한 파일명에 기초하여 참조를 조회할 필요가 있을 수 있다. 예를 들어, 새로운 노드를 추가하거나 또는 노드를 재명명할 때, 클라이언트 동기화 서비스(156)는, 노드의 파일명이 이미 파일명 어레이(1310) 내에 존재하는지 여부를 결정하기를 희망할 수 있다. 참조가 발견되는 경우, 파일명이 존재하며, 이는 참조에 기초하여 위치될 수 있다. 따라서, 클라이언트 동기화 서비스(156)는 파일명에 대한 참조를 사용할 수 있으며, 새로운 또는 재명명된 노드 내에 참조를 저장할 수 있다. 참조가 발견되지 않는 경우, 파일명 어레이(130) 내에 파일명이 존재하지 않으며, 클라이언트 동기화 서비스(156)는 파일명

을 파일명 어레이(1310)에 추가할 수 있다.

[0201] 파일명에 기초하는 참조의 조회는, 해시 인덱스 어레이의 사용에 의해 가능해진다. 도 13은 본 기술의 다양한 실시예들에 따른 해시 인덱스 어레이(1350)의 일 예시를 도시한다. 해시 인덱스 어레이(1350)는 다양한 위치들에 파일명들에 대한 참조들을 저장하도록 구성된다. 특히, 파일명에 대한 참조는 파일명의 해시에 기초하여 해시 인덱스 어레이(1350) 내의 포지션에 저장된다. 도 13에 예시된 일 예에 있어서, 사용되는 해시 함수에 기초하여, 파일명 "Pictures"의 해시는 4와 동일할 수 있다. 따라서, 파일명 "Pictures"에 대한 참조는 해시 인덱스 어레이(1350) 내에서 포지션 4에 저장된다. 이러한 참조 값은 0이며, 이는, 이상에서 언급된 바와 같이, 파일명 "Pictures"의 파일명 어레이(1310) 내의 위치를 지정한다. 유사하게, 파일명 "Documents"의 해시는 0과 동일할 수 있다. 따라서, 파일명 "Documents"에 대한 참조는 해시 인덱스 어레이(1350) 내에서 포지션 0에 저장된다. 이러한 참조 값은 16이며, 이는, 이상에서 언급된 바와 같이, 파일명 "Documents"의 파일명 어레이(1310) 내의 위치를 지정한다.

[0202] 일부 시나리오들에 있어서, 2개의 파일명들의 해시가 동일한 해시 값을 산출할 때 충돌이 발생할 수 있다. 도 13에 예시된 예에 있어서, "a.jpg"의 해시 값이 또한 4일 수 있다. 충돌이 발생하는 경우, 클라이언트 동기화 서비스(156)는 해시 인덱스 어레이(1350) 내의 다음의 이용가능 포지션을 사용할 수 있다. 예를 들어, 파일명 "Pictures"에 대한 참조가 해시 인덱스 어레이(1350) 내에서 포지션 4에 저장되기 때문에, 클라이언트 동기화 서비스(156)는 해시 인덱스 어레이(1350) 내의 다음 이용가능 포지션을 조회하고 파일명 "a.jpg"에 대한 참조(예를 들어, 9)를 저장할 수 있다.

[0203] 도 14는 본 기술의 다양한 실시예들에 따른 파일명을 저장하기 위한 예시적인 방법을 도시한다. 본원에서 설명되는 방법들 및 프로세스들이 특정 순서의 특정 단계들 및 동작들을 가지고 도시될 수 있지만, 달리 언급되지 않는 한, 유사하거나 또는 대안적인 순서들로, 또는 병렬로 수행되는, 추가적이거나, 더 적거나, 또는 대안적인 단계들 및 동작들이 다양한 실시예들의 범위 내에 속한다. 방법(1400)은, 예를 들어, 클라이언트 디바이스 상에서 실행 중인 도 2의 클라이언트 동기화 서비스(156)와 같은 시스템에 의해 구현될 수 있다.

[0204] 동작(1405)에서, 시스템은 트리 데이터 구조체 내의 노드에 대한 수정을 검출할 수 있다. 수정은, 예를 들어, 트리 데이터 구조체에 대한 노드의 추가 또는 노드의 파일명의 편집일 수 있다. 그러면, 시스템은, 동작(1410)에서 파일명이 이미 파일명 어레이 내에 존재하는지 여부를 결정할 수 있다. 시스템은, 파일명 어레이 내의 파일명의 참조 위치에 대하여, 노드와 연관된 파일명을 사용하여 질의함으로써 파일명이 파일명 어레이 내에 이미 존재하는지를 확인하기 위해 체크할 수 있다. 참조 위치가 발견되는 경우, 동작(1415)에서 파일명을 파일명 어레이에 추가해야 할 필요성이 없으며, 시스템은, 동작(1420)에서, 질의에 의해 반환되는 참조이어야만 하는 파일명 어레이 내의 파일명에 대한 위치를 결정하고 파일명의 위치를 노드 내에 저장할 수 있다.

[0205] 쿼리가 파일명 어레이 내의 파일명에 대한 참조를 반환하지 않거나 또는 파일명이 달리 파일명 어레이 내에 존재하지 않는 경우, 시스템은, 동작(1425)에서 파일명 및 구분자를 파일명 어레이에 첨부하고, 동작(1430)에서 파일명 어레이 내의 파일명의 위치를 결정하며, 동작(1435)에서 노드 내에 파일명의 위치를 저장할 수 있다.

[0206] 파일명에 기초하여 참조 위치의 후속 질의를 가능하게 하기 위하여, 시스템은 추가로 해시 인덱스 내에 파일명의 위치를 저장할 수 있다. 파일명의 위치를 저장하기 위한 해시 인덱스 내의 포지션을 결정하기 위하여, 시스템은 파일명의 해시 값을 계산할 수 있다. 이러한 해시 값은 파일명의 위치를 저장하기 위한 해시 인덱스 내의 포지션을 찾기 위해 사용될 수 있다.

[0207] 일단 파일명의 위치가 노드 내에 저장되면, 파일명을 검색하는 것은 단순히 노드 내의 파일명의 위치를 액세스하는 것 및 파일명 어레이 내에서 파일명을 조회하기 위하여 위치를 사용하는 것을 수반할 수 있다. 시스템은 노드 내에 명시된 위치로부터 시작하여 구분자에 도달할 때 중지할 수 있다.

[0208] 도 15는 본 기술의 다양한 실시예들에 따른 주어진 파일명에 대하여 파일명의 위치를 검색하기 위한 예시적인 방법을 도시한다. 본원에서 설명되는 방법들 및 프로세스들이 특정 순서의 특정 단계들 및 동작들을 가지고 도시될 수 있지만, 달리 언급되지 않는 한, 유사하거나 또는 대안적인 순서들로, 또는 병렬로 수행되는, 추가적이거나, 더 적거나, 또는 대안적인 단계들 및 동작들이 다양한 실시예들의 범위 내에 속한다. 방법(1500)은, 예를 들어, 클라이언트 디바이스 상에서 실행 중인 도 2의 클라이언트 동기화 서비스(156)와 같은 시스템에 의해 구현될 수 있다. 이상에서 설명된 바와 같이, 방법은, 파일명이 이미 파일명 어레이 내에 저장되었는지 여부를 결정하기 위해 사용될 수 있다.

[0209] 파일명들의 위치들은 해시 인덱스 또는 해시 인덱스 어레이 내에 저장된다. 따라서, 동작(1505)에서, 시스템은

파일명에 대하여 해시 함수를 실행함으로써 해시 인덱스 내의 포지션을 생성할 수 있다. 이는, 예를 들어, 재명명되는 노드에 대한 새로운 명칭 또는 새로운 노드의 명칭일 수 있다. 동작(1510)에서, 시스템은, 해시 인덱스 내의 포지션으로부터, 파일명의 위치 정보를 검색한다. 이러한 위치 정보는, 파일명이 저장되는 파일명 어레이 내의 위치에 대한 것이다.

[0210] 일부 구현예들에 있어서, 시스템은, 동작(1515)에서, 정확한 파일명이 위치 내에 저장되었음을 확인하기 위하여 체크할 수 있다. 예를 들어, 시스템은, 위치 정보에 기초하여, 파일명 어레이로부터 스트링(string)을 검색하고, 그 스트링을 파일명과 비교할 수 있다. 스트링 및 파일명이 일치하는 경우, 동작(1520)에서, 파일명의 위치가 확인되고, 파일명의 위치가 노드 내에 저장된다.

[0211] 스트링 및 파일명이 일치하지 않는 경우, 동작(1530)에서, 시스템은 해시 인덱스 내의 다음 포지션 내의 위치 정보를 검색하고, 시스템은 다음 포지션 내의 위치 정보가 정확한지 여부를 체크하기 위해 동작(1515)로 복귀할 수 있다. 시스템은 정확한 위치 정보가 발견될 때까지 이를 계속할 수 있으며, 동작(1520)에서, 노드 내에 이를 저장할 수 있다.

[0212] <u>트리들 사이의 차이들의 효율적인 식별</u>

[0213] 이상에서 언급된 바와 같이, 클라이언트 동기화 서비스(156)는, 콘텐트 관리 시스템에 의해 저장된 콘텐트 아이템들에 대한 서버 상태를 나타내는 원격 트리, 클라이언트 디바이스 상에 저장된 대응하는 콘텐트 아이템들에 대한 파일 시스템 상태를 나타내는 로컬 트리, 및 서버 상태와 파일 시스템 상태 사이의 알려진 싱크된 상태를 나타내는 싱크 트리 내의 노드들 사이의 차이들을 식별하도록 구성된다. 이러한 차이들에 기초하여, 실행되는 경우, 3개의 트리 데이터 구조체들이 동일하게 될 동기화된 상태를 향해 서버 상태 및 시스템 상태를 수렴시키도록 구성되는 동작들의 세트가 생성될 수 있다.

[0214] 다수의 노드들이 존재할 때, 트리들 사이의 차이들을 효율적으로 식별할 수 있는 것이 중요하다. 예를 들어, 트리 내에 수백만 개의 노드들이 존재할 수 있으며, 각각의 노드를 개별적으로 비교하는 것은 프로세싱 시간 및 자원 사용과 관련하여 금지될 수 있다.

[0215] 본 기술의 다양한 실시예들은 트리들 사이의 차이를 식별하기 위하여 더 효율적인 수단을 제공하는 것과 관련된다. 구체적으로, 클라이언트 동기화 서비스(156)는, 노드들이 상이한지 여부를 결정하기 위해 다른 트리들 내의 노드들과 비교하기위해 사용될 수 있는 값들을 각각의 노드에 할당하도록 구성된다. 이러한 값들이 디프(diff) 값들로서 지칭될 수 있다. 트리들 사이의 차이들을 식별하는 효율을 개선하기 위하여, 각각의 리프 노드에 디프 값이 할당될 수 있으며, 부모 노드들에 대한 디프 값들은 그들의 자식 노드들의 디프 값들에 기초하여 계산될 수 있다. 루트 노드를 포함하는 트리 데이터 구조체의 매 레벨에 대한 디프 값들이 이러한 방식으로 계산될 수 있다.

[0216] 도 16a 및 도 16b는 다양한 실시예들에 따른 트리 데이터 구조체의 예들을 도시한다. 예시적인 목적들을 위하여, 도 16a 및 도 16b는 싱크 트리 및 로컬 트리를 도시한다. 그러나, 원격 트리가 유사하게 수행할 수 있다. 도 16a 및 도 16b에서, 싱크 트리 및 로컬 트리에 대한 리프 노드들에 대한 디프 값들은 노드들의 해시 함수들을 사용하여 연산된다. 각각의 부모 노드(또는 자식들을 갖는 노드)는 그들의 자식 노드들의 디프 값들에 기초하여 연산된 디프 값을 갖는다.

[0217] 예를 들어, 도 16a의 싱크 트리(1605) 내에서, 노드 C에 대한 디프 값은 325이고, 노드 D에 대한 디프 값은 742이며, 노드들 C 및 D의 부모인 B에 대한 디프 값은 C 및 D의 디프 값들의 함수로서 계산된다. 다시 말해서, DiffValue(B) = f(DiffValue(C), DiffValue(D)) = f(325, 742)이다. 마찬가지로, 루트 노드의 디프 값은 그것의 자식 노드들인 노드 A 및 노드 B의 디프 값들의 함수이다. 바꿔 말하면, DiffValue(/root) = f(DiffValue(A), DiffValue(B)) = f(924, 789).

[0218] 트리들 사이의 차이들을 식별하기 위하여, 클라이언트 동기화 서비스(156)는, 이들이 상이한지 여부를 확인하기 위하여 대응하는 노드들의 디프 값들을 비교할 수 있다. 디프 값들이 동일한 경우, 트리들 사이에 차이가 존재하지 않는다. 디프 값들이 상이한 경우, 노드 내에 또는 노드와 연관된 경로 아래쪽에 변화가 존재한다. 결과적으로, 클라이언트 동기화 서비스(156)는, 이들이 상이한지 여부 및 자식 노드와 연관된 경로 아래쪽에 차이가 존재하는지 여부를 결정하기 위하여 자식 노드들을 자세히 살펴보고 반대편 트리 내의 대응하는 노드들을 비교할 수 있다.

[0219] 예를 들어, 도 16a에서, 클라이언트 동기화 서비스(156)는 싱크 트리(1605)의 루트 노드 및 로컬 트리(1610)의 루트 노드의 디프 값들을 비교하고, 디프 값들이 일치한다는 것을 결정할 수 있다. 따라서, 루트 노드들의 자식

들 내에 차이가 존재하지 않는다. 결과적으로, 클라이언트 동기화 서비스(156)는, 트리들 내의 각각의 그리고 매 노드를 반대편의 트리 내의 대응하는 노드와 비교할 필요 없이, 싱크 트리(1605) 및 로컬 트리(1610)가 일치하고 동기화되었다는 것을 결정할 수 있다.

[0220] 도 16b에 예시된 예에 있어서, 클라이언트 동기화 서비스(156)는 유사하게 싱크 트리(1605)의 루트 노드 및 로컬 트리(1610)의 루트 노드의 디프 값들을 비교할 수 있다. 그러나, 클라이언트 동기화 서비스(156)는, 싱크 트리(1605)의 루트 노드와 로컬 트리(1610)의 루트 노드의 디프 값들이 상이하다는 것을 결정할 수 있으며, 이는 루트 노드들의 자손들에서 차이가 존재할 수 있음을 나타낸다.

[0221] 따라서, 클라이언트 동기화 서비스(156)는 그들의 디프 값들을 비교하기 위하여 루트 노드들의 자식들의 다음 레벨로 이동할 수 있다. 노드 A에 대하여, 싱크 트리(1615) 내의 노드 A의 디프 값 및 로컬 트리(1620) 내의 노드 A의 디프 값이 일치한다. 따라서, 이들 2 사이에 차이가 존재하지 않으며, 루트 레벨에서 검출된 차이는 노드 A 또는 노드 A 아래의 임의의 경로에 의해 초래되는 것이 아니다.

[0222] 노드 B로 이동하면, 클라이언트 동기화 서비스(156)는 트리들 둘 모두 내의 노드 B의 그들의 디프 값들을 비교하며, 싱크 트리(1615) 내의 노드 A의 디프 값 및 로컬 트리(1620) 내의 노드 A의 디프 값이 상이하다는 것을 발견한다. 따라서, 루트 레벨에서 검출된 차이는 노드 B 또는 노드 B 아래의 경로에 의해 초래된다. 클라이언트 동기화 서비스(156)는, 그들의 디프 값들을 비교하기 위해 노드 B의 자식들의 다음 레벨로 이동하고, 루트 레벨에서 검출된 차이가 노드 D의 삭제에 의해 초래되었음을 발견할 수 있다. 결과적으로, 클라이언트 동기화 서비스(156)는, 트리들 내의 각각의 그리고 매 노드를 반대편의 트리 내의 대응하는 노드와 비교할 필요 없이 싱크 트리(1615)와 로컬 트리(1620) 사이의 차이점을 식별할 수 있다. 예를 들어, 트리들 둘 모두 내의 노드 A의 디프 값들이 일치되기 때문에, 노드 A는 분석될 필요가 없는 다수의 자손 노드들을 가졌을 수 있다.

[0223] 머클 트리(Merkle tree) 또는 해시 트리 메커니즘이 일부 경우들에서 작동할 수 있다. 예를 들어, 각각의 리프 노드에 대한 디프 값은 리프 노드의 해시에 기초하여 계산될 수 있으며, 비-리프 노드들의 디프 값은 자식 디프 값들의 합계의 해시에 기초하여 연산될 수 있다. 그러나, 여기에서 머클 트리 또는 해시 트리 메커니즘은 특정 상황들에서 바람직하지 않은 성능 메트릭들을 갖는다. 당신이 리프 노드를 추가하거나 또는 삭제할 때, 리프 노드의 조상 노드들의 전부의 디프 값들은 그들의 디프 값들이 재연산되어야 할 필요가 있으며, 조상 노드들의 각각을 재연산하는 것은 각각의 조상 노드에 대한 자식 노드들의 전부의 리스팅을 필요로 한다. 이는 연산적으로 비용이 비싸며, 특히, 트리 데이터 구조체들이, 각각의 노드가 형제(sibling) 노드 다음에 또는 그 근처에서 메모리 내에 저장되지 않는 방식으로 저장될 때 그러하다.

[0224] 본 기술의 다양한 실시예들은, 다른 것들 중에서도 특히, 부모 노드들의 디프 값들을 상이하게 연산함으로써, 이러한 그리고 다른 기술적 단점들을 해결한다. 각각의 리프 노드의 디프 값은 리프 노드의 해시를 계산함으로써 결정될 수 있다. 각각의 부모 노드의 디프 값은 그것의 자식들 모두의 해시들의 배타적-논리합 연산 또는 XOR 연산을 수행함으로써 계산될 수 있다.

[0225] 도 17은 다양한 실시예들에 따른 트리 데이터 구조체의 일 예를 도시한다. 도 17의 트리 데이터 구조체에서, 리프 노드들 A, C, 및 D의 디프 값들은 노드들을 해싱함으로써 계산된다. 노드 B의 디프 값은 그것의 자식 노드들인 노드 C 및 노드 D의 XOR 함수이다. 루트 노드의 디프 값은 그것의 자식 노드들인 노드 A 및 노드 B의 해시의 XOR 함수이다. XOR 함수의 중요한 특성들은 XOR이 순서에 민감하지 않다는 것을 포함하며, 이는 m XOR n이 n XOR m과 동일할 것임을 의미한다. 또한 m XOR m은 0과 동일하다.

[0226] 당신이 리프 노드를 추가하거나 또는 삭제할 때, 리프 노드의 조상 노드들의 전부의 디프 값들이 재연산된 그들의 디프 값들을 가져야 할 필요가 있다. 그러나, 조상 노드들의 디프 값들을 연산하는 것은 이들 전부의 리스팅을 요구하지 않고 이루어질 수 있다. 예를 들어, 자식 노드가 삭제될 때, 부모의 새로운 디프 값은, 삭제된 자식 노드의 디프 값 및 부모의 오래된 디프 값에 대한 XOR 연산을 수행함으로써 계산될 수 있다. 자식 노드가 추가될 때, 부모의 새로운 디프 값은, 새로운 자식 노드의 디프 값 및 부모의 오래된 디프 값에 대한 XOR 연산을 수행함으로써 계산될 수 있다.

[0227] 도 18은 본 기술의 다양한 실시예들에 따른 주어진 파일명에 대한 파일명의 위치를 검색하기 위한 예시적인 방법을 도시한다. 본원에서 설명되는 방법들 및 프로세스들이 특정 순서의 특정 단계들 및 동작들을 가지고 도시될 수 있지만, 달리 언급되지 않는 한, 유사하거나 또는 대안적인 순서들로, 또는 병렬로 수행되는, 추가적이거나, 더 적거나, 또는 대안적인 단계들 및 동작들이 다양한 실시예들의 범위 내에 속한다. 방법(1800)은, 예를 들어, 클라이언트 디바이스 상에서 실행 중인 도 2의 클라이언트 동기화 서비스(156)와 같은 시스템에 의해 구

현될 수 있다. 이상에서 설명된 바와 같이, 방법은, 파일명이 이미 파일명 어레이 내에 저장되었는지 여부를 결정하기 위해 사용될 수 있다.

[0228] 동작(1805)에서, 클라이언트 동기화 서비스(156)는, 원격 트리, 싱크 트리, 또는 로컬 트리와 같은 트리 데이터 구조체로부터 노드를 제거하거나 또는 추가할 수 있다. 노드가 삭제되는 경우, 노드는 이것과 연관된 이전에 계산된 디프 값을 가져야만 한다. 노드가 추가되는 경우, 클라이언트 동기화 서비스(156)는, 예를 들어, 노드를 해싱함으로써, 새로운 노드에 대한 디프 값을 계산할 수 있다. 트리에 대한 변화가 존재하기 때문에, 노드의 조상들의 디프 값들이 업데이트되어야만 한다.

[0229] 동작(1810)에서, 클라이언트 동기화 서비스(156)는 부모 노드의 현재 디프 값 및 그 노드의 디프 값에 기초하여 노드의 부모에 대한 새로운 디프 값을 계산할 수 있다. 동작(1815)에서, 부모에 대한 새로운 디프 값이 부모 노드 내에 저장된다.

[0230] 동작(1820)에서, 클라이언트 동기화 서비스(156)는, 부모 노드 자체가 부모를 갖는지 여부를 결정한다. 다시 말해서, 부모 노드가 루트 노드인지 또는 이에 대하여 디프 값들을 연산할 추가적인 조상들이 존재하는지 여부이다. 추가적인 조상이 존재하는 경우, 프로세스는 동작(1810)으로 복귀할 수 있으며, 여기에서 부모 노드의 부모는 계산되고 저장되는 새로운 디프 값을 가질 것이다. 추가적인 부모가 존재하지 않고 부모 노드가 루트 노드인 경우, 프로세스는 동작(1825)에서 중지될 수 있다.

[0231] 일단 루트 노드에 도달되면, 트리 데이터 구조체는 차이들을 식별하기 위하여 다른 트리 데이터 구조체들과 비교될 준비가 된다. 이상에서 언급된 바와 같이, 클라이언트 동기화 서비스(156)는, 이러한 차이들에 기초하여, 실행되는 경우, 3개의 트리 데이터 구조체들이 동일하게 될 동기화된 상태를 향해 서버 상태 및 시스템 상태를 수렴시키도록 구성되는 동작들의 세트를 생성할 수 있다.

[0232] 도 19는, 클라이언트 디바이스(150), 콘텐트 관리 시스템(110) 또는 이의 임의의 컴포넌트를 구성하는 예시적인 임의의 컴퓨팅 디바이스일 수 있는 컴퓨팅 시스템(1900)을 도시하며, 여기에서 시스템의 컴포넌트들은 연결(1905)을 사용하여 서로 통신하고 있다. 연결(1905)은 버스를 통한 물리적 연결일 수 있거나, 또는, 예컨대 칩셋 아키텍처 내의 프로세서(1910) 내로의 직접 연결일 수 있다. 연결(1905)은 또한 가상 연결, 네트워크형 연결, 또는 논리적 연결일 수도 있다.

[0233] 일부 실시예들에 있어서, 컴퓨팅 시스템(1900)은 분산형 시스템이며, 여기에서 본 개시에서 설명된 기능들은 데이터센터, 다수의 데이터센터들, 피어(peer) 네트워크 등 내에 분산될 수 있다. 일부 실시예들에 있어서, 설명되는 시스템 컴포넌트 중 하나 이상은, 각기 컴포넌트가 이러한 것에 대하여 설명되는 기능 중 일부 또는 전부를 수행하는 이러한 다수의 컴포넌트들을 나타낸다. 일부 실시예들에 있어서, 컴포넌트들은 물리적인 또는 가상 디바이스들일 수 있다.

[0234] 예시적인 시스템(1900)은 적어도 하나의 프로세싱 유닛(CPU 또는 프로세서)(1910), 및 판독 전용 메모리(read only memory; ROM)(1920) 및 랜덤 액세스 메모리(random access memory; RAM)(1925)와 같은 시스템 메모리(1915)를 포함하는 다양한 시스템 컴포넌트들을 프로세서(1910)에 결합하는 연결(1905)을 포함한다. 컴퓨팅 시스템(1900)은, 프로세서(1910)와 직접적으로 연결되거나, 또는 이에 인접하여 위치되거나 이의 부분으로서 통합되는 고속 메모리(1912)의 캐시를 포함할 수 있다.

[0235] 프로세서(1910)는, 소프트웨어 명령어들이 실제 프로세서 설계 내로 통합되는 특수-목적 프로세서뿐만 아니라 프로세서(1910)를 제어하도록 구성되는, 저장 디바이스(1930) 내에 저장된 서비스들(1932, 1934, 및 1936)과 같은 하드웨어 서비스 또는 소프트웨어 서비스 및 임의의 범용 프로세서를 포함할 수 있다. 프로세서(1910)는, 본질적으로, 다수의 코어들 또는 프로세스들, 버스, 메모리 제어기, 캐시 등을 포함하는 완전한 자급식 컴퓨팅 시스템일 수 있다. 다중-코어 프로세서는 대칭적이거나 또는 비대칭적일 수 있다.

[0236] 사용자 상호작용을 가능하게 하기 위하여, 컴퓨팅 시스템(1900)은 입력 디바이스(1945)를 포함하며, 이는 임의의 수의 입력 메커니즘들, 예컨대, 언어(speech)에 대한 마이크, 제스처 또는 그래픽 입력에 대한 터치-감지 스크린, 키보드, 마우스, 모션 입력부, 언어부 등을 나타낼 수 있다. 컴퓨팅 시스템(1900)은 또한, 당업자들에게 알려진 다수의 출력 메커니즘들 중 하나 이상일 수 있는 출력 디바이스(1935)를 포함할 수 있다. 일부 경우들에 있어서, 멀티모달(multimodal) 시스템들은 사용자가 컴퓨팅 시스템(1900)과 통신하기 위한 다수의 유형들의 입력/출력을 제공하는 것을 가능하게 할 수 있다. 컴퓨팅 시스템(1900)은, 일반적으로 사용자 입력 및 시스템 출력을 지배하고 관리할 수 있는 통신 인터페이스(1940)를 포함할 수 있다. 임의의 특정 하드웨어 배열 상에서 동작하는데 제한이 존재하지 않으며, 따라서, 본원에서 기본적인 특징부들은, 이들이 개발됨에 따라 개선된 하드

웨어 또는 펌웨어 배열들로 용이하게 대체될 수 있다.

[0237] 저장 디바이스(1930)는 비-휘발성 메모리 디바이스일 수 있으며, 이는, 컴퓨터에 의해 액세스가능한 데이터를 저장할 수 있는 하드 디스크 또는 다른 유형의 컴퓨터 판독가능 매체들, 예컨대, 자기 카세트들, 플래시 메모리 카드들, 고체 상태 메모리 디바이스들, 디지털 다용도 디스크들, 카트리지들, 랜덤 액세스 메모리(random access memory; RAM)들, 판독 전용 메모리(read only memory; ROM), 및/또는 이러한 디바이스들의 일부 조합일 수 있다.

[0238] 저장 디바이스(1930)는, 이러한 소프트웨어를 정의하는 코드가 프로세서(1910)에 의해 실행될 때, 이것이 시스템으로 하여금 기능을 수행하게끔 하는, 소프트웨어 서비스들, 서버들, 서비스들 등을 포함할 수 있다. 일부 실시예들에 있어서, 특정 기능을 수행하는 하드웨어 서비스는, 기능을 수행하기 위한 필수 하드웨어 컴포넌트들, 예컨대 프로세서(1910), 연결(1905), 출력 디바이스(1935), 등과 관련하여 컴퓨터 판독가능 매체 내에 저장된 소프트웨어 컴포넌트를 포함할 수 있다.

[0239] 설명의 명료성을 위하여, 일부 경우들에 있어서, 본 기술은, 소프트웨어로, 또는 하드웨어 또는 소프트웨어의 조합으로 구현된 방법 내의 단계들 또는 루틴들, 디바이스, 디바이스 컴포넌트들을 포함하는 기능 블록들을 포함하는 개별적인 기능 블록들을 포함하는 것으로서 제시될 수 있다.

[0240] 본원에서 설명된 단계들, 동작들, 또는 프로세스들 중 임의의 것은, 단독으로 또는 다른 디바이스들과 조합되어, 하드웨어 및 소프트웨어 서비스들 또는 서비스들의 조합에 의해 수행되거나 또는 구현될 수 있다. 일부 실시예들에 있어서, 서비스는, 콘텐트 관리 시스템의 하나 이상의 서버들 및/또는 클라이언트 디바이스의 메모리 상주하고 프로세스가 서비스와 연관된 소프트웨어를 실행할 때 하나 이상의 기능들을 수행하는 소프트웨어일 수 있다. 일부 실시예들에 있어서, 서비스는 특정 기능을 수행하는 프로그램 또는 프로그램의 수집물이다. 일부 실시예들에 있어서, 서비스는 서버로서 간주될 수 있다. 메모리는 비-일시적인 컴퓨터-판독가능 매체일 수 있다.

[0241] 일부 실시예들에 있어서, 컴퓨터-판독가능 저장 디바이스들, 매체들, 및 메모리들은 비트 스트림 및 유사한 것을 포함하는 무선 신호 또는 케이블을 포함할 수 있다. 그러나, 언급될 때, 비-일시적인 컴퓨터-판독가능 저장 매체는 명백히 그 자체로는 에너지, 반송 신호들, 전자기파들과 같은 매체들을 배제한다.

[0242] 이상에서 설명된 예들에 따른 방법들은 컴퓨터 판독가능 매체에 저장되거나 또는 달리 이로부터 이용가능한 컴퓨터-실행가능 명령어들을 사용하여 구현될 수 있다. 이러한 명령어들은, 예를 들어, 범용 컴퓨터, 특수 목적 컴퓨터, 또는 특수 목적 프로세싱 디바이스가 특정 기능 또는 기능들의 그룹을 수행하게끔 하거나 또는 달리 수행하도록 구성되는 데이터 및 명령어들을 포함할 수 있다. 사용되는 컴퓨터 자원들의 부분들은 네트워크를 통해 액세스가능할 수 있다. 컴퓨터 실행가능 명령어들은, 예를 들어, 2진수들, 중간 포맷 명령어들 예컨대 어셈블리 언어, 펌웨어, 또는 소스 코드일 수 있다. 명령어들, 설명된 예들에 따른 방법들 동안 사용되는 정보, 및/또는 생성되는 정보를 저장하기 위해 사용되는 컴퓨터-판독가능 매체들의 예들은, 자기 또는 광 디스크들, 고체 상태 메모리 디바이스들, 플래시 메모리, 비-휘발성 메모리가 구비된 USB 디바이스들, 네트워크형 저장 디바이스들 등을 포함한다.

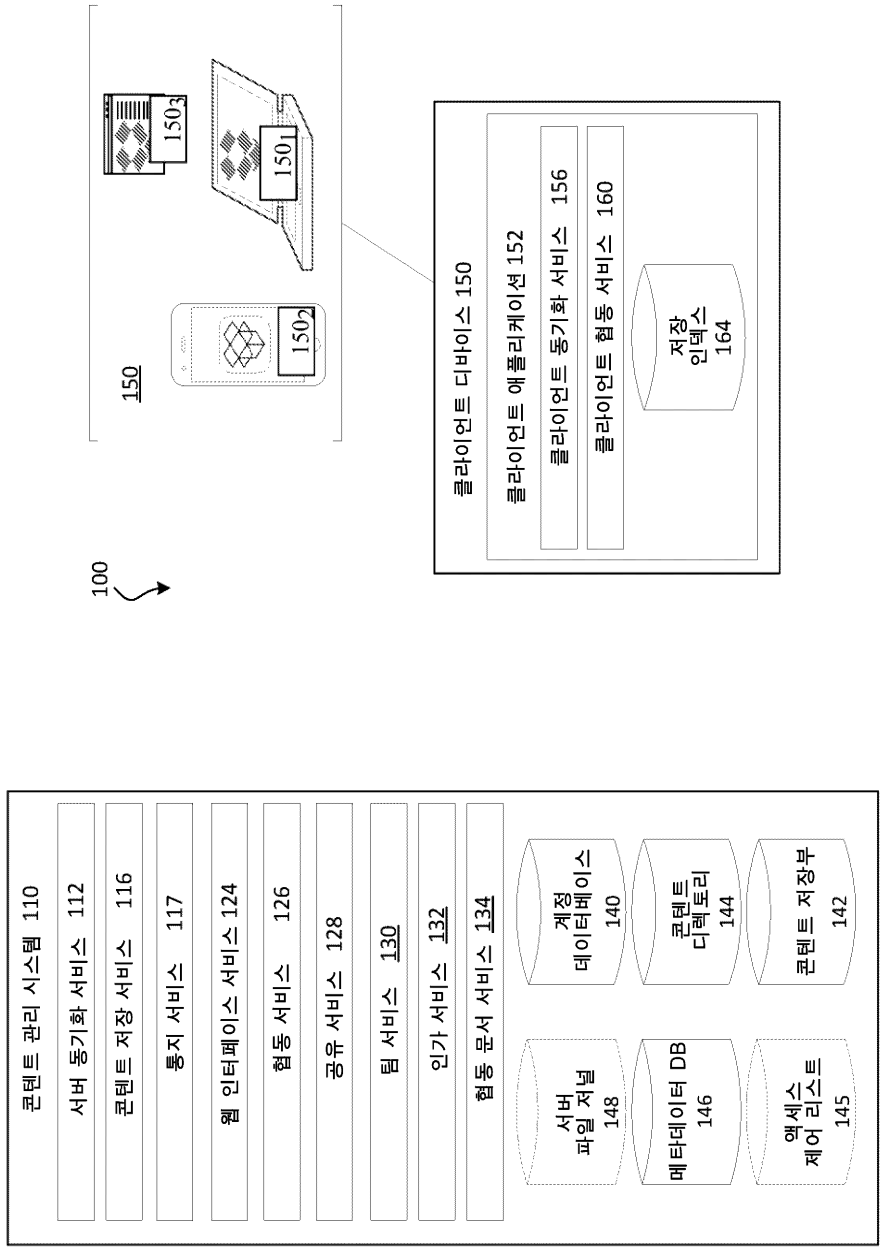
[0243] 이러한 개시들에 따른 방법들을 구현하는 디바이스들은 하드웨어, 펌웨어 및/또는 소프트웨어를 포함할 수 있으며, 다양한 폼 팩터들 중 임의의 것을 취할 수 있다. 이러한 폼 팩터들의 전형적인 예들은, 서버들, 랩탑들, 스마트 폰들, 소형 폼 팩터 개인용 컴퓨터들, 개인용 디지털 보조기기들 등을 포함한다. 본원에서 설명되는 기능은 또한 주변기기들 또는 애드-인 카드들 내에 구현될 수 있다. 이러한 기능은 또한, 추가적인 예로서, 단일 디바이스 내에서 실행되는 상이한 칩들 또는 상이한 프로세스들 중에서 회로 보드 상에 구현될 수 있다.

[0244] 명령어들, 이러한 명령어들을 운반하기 위한 매체들, 이들을 실행하기 위한 컴퓨팅 자원들, 및 이러한 컴퓨팅 자원들을 지원하기 위한 다른 구조체들은 이러한 개시들 내에서 설명되는 기능들을 제공하기 위한 수단들이다.

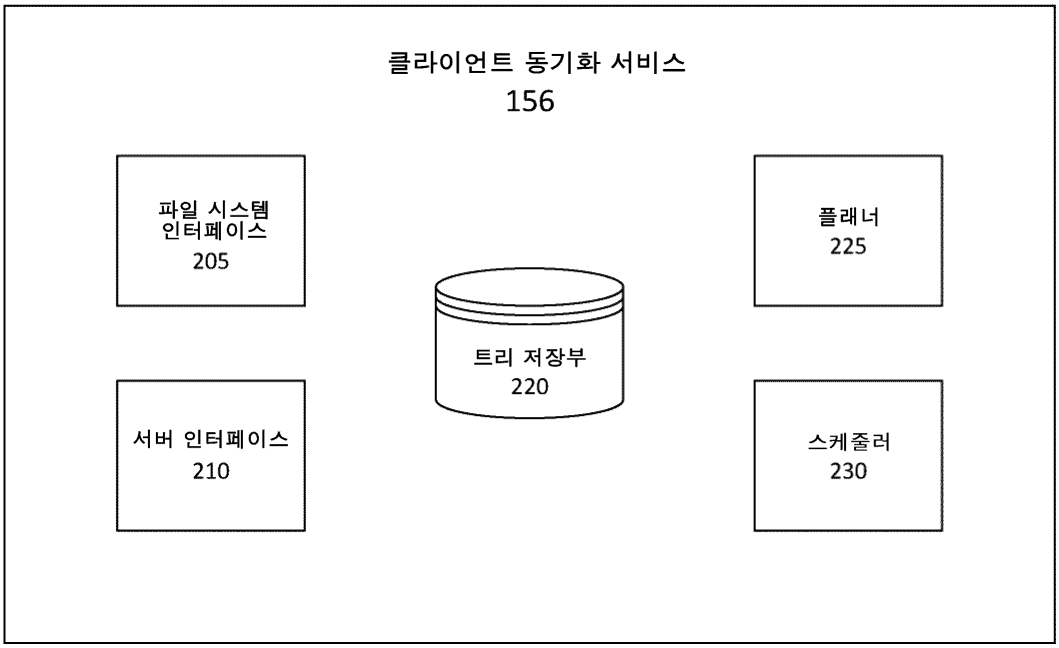
[0245] 다양한 예들 및 다른 정보가 첨부된 청구항들의 범위 내의 측면들을 설명하기 위하여 사용되었지만, 당업자가 매우 광범위한 구현예들을 도출하기 위하여 이러한 예들을 사용할 수 있는 바와 같이, 이러한 예들에서 특정 특징부들 또는 배열들에 기초하는 청구항들의 어떠한 제한도 암시되지 않는다. 추가로 그리고 내용이 구조적 특징들 및/또는 방법 단계들에 특유한 표현으로 설명되었을 수 있지만, 첨부된 청구항들에서 정의되는 내용이 반드시 이러한 특징들 또는 행위들에 한정되는 것은 아님이 이해되어야 한다. 예를 들어, 이러한 기능은 본원에서 식별되는 것과는 다른 컴포넌트들 내에서 수행되거나 또는 다르게 분산될 수 있다. 오히려, 설명된 특징들 및 단계들은 첨부된 청구항들의 범위 내의 방법들 및 시스템들의 컴포넌트들의 예들로서 개시된다.

도면

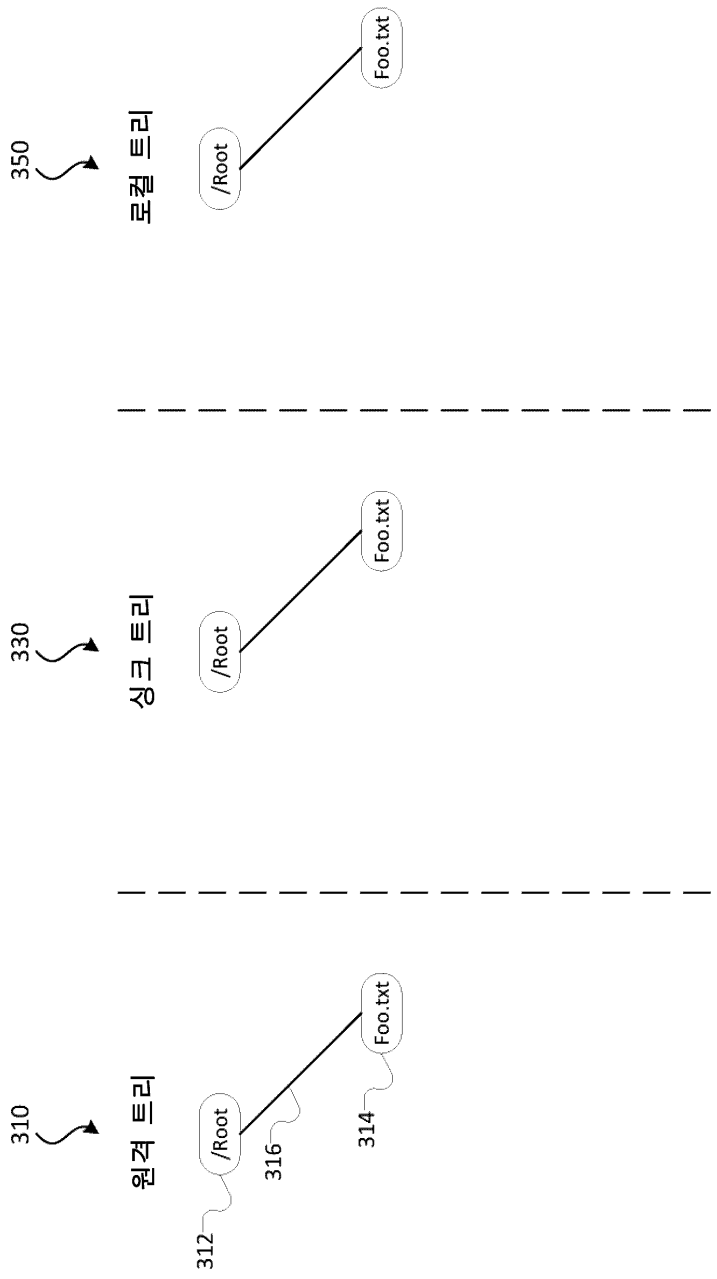
도면1


100
150
150_1
150_2
150_3
클라이언트 디바이스 150
클라이언트 애플리케이션 152
클라이언트 동기화 서비스 156
클라이언트 협동 서비스 160
저장 인덱스 164
콘텐트 관리 시스템 110
서버 동기화 서비스 112
콘텐트 저장 서비스 116
통지 서비스 117
웹 인터페이스 서비스 124
협동 서비스 126
공유 서비스 128
팀 서비스 130
인가 서비스 132
협동 문서 서비스 134
서버 파일 저널 148
메타데이터 DB 146
액세스 제어 리스트 145
계정 데이터베이스 140
콘텐트 디렉토리 144
콘텐트 저장부 142

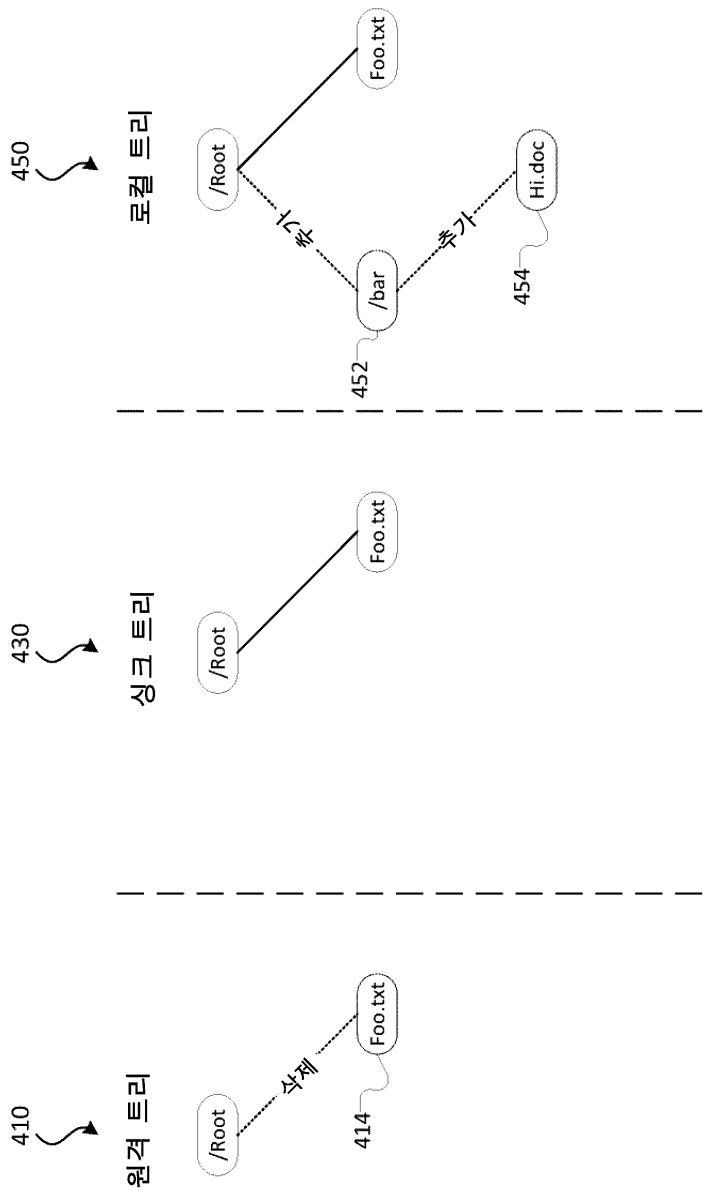
도면2


클라이언트 동기화 서비스
156
파일 시스템
인터페이스
205
플래너
225
트리 저장부
220
서버 인터페이스
210
스케줄러
230

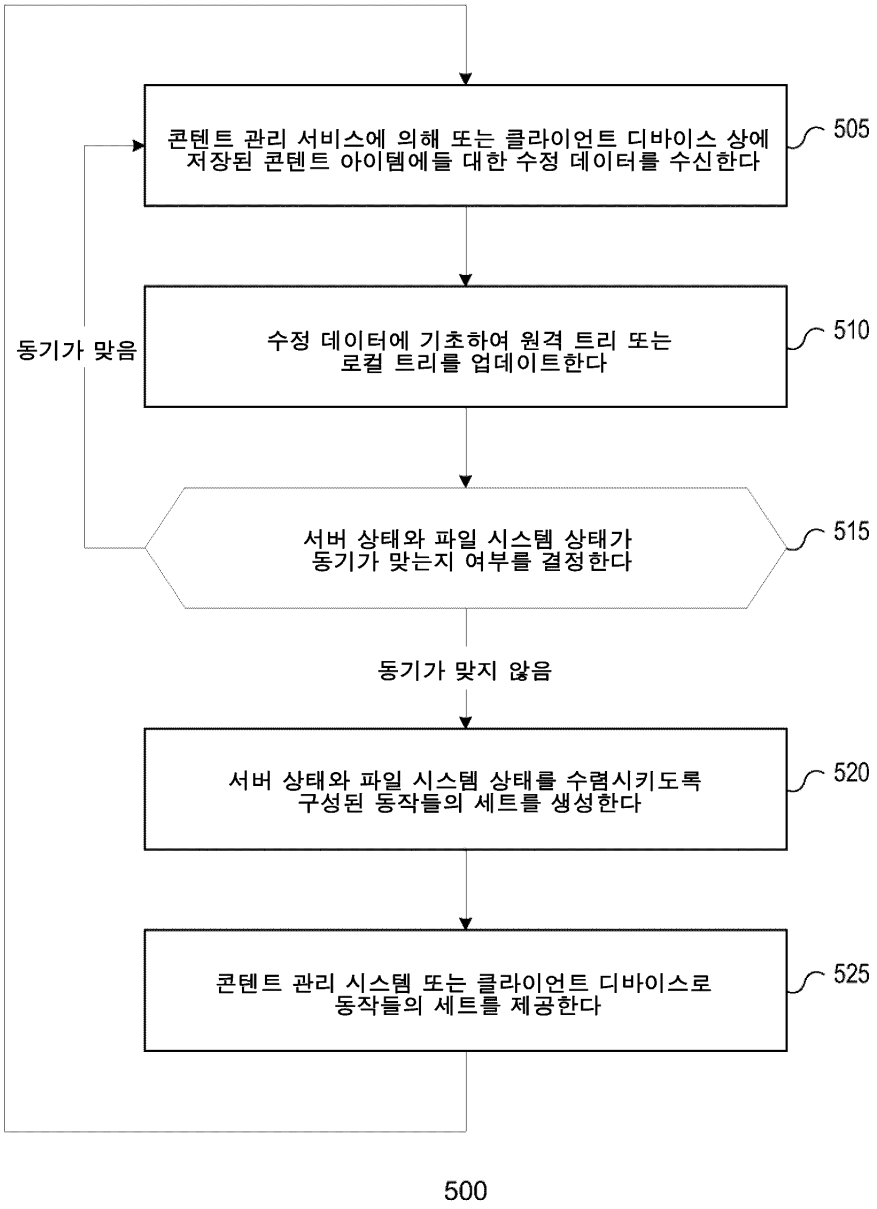
도면3


310
원격 트리
/Root
Foo.txt
312
316
314
330
싱크 트리
/Root
Foo.txt
350
로컬 트리
/Root
Foo.txt

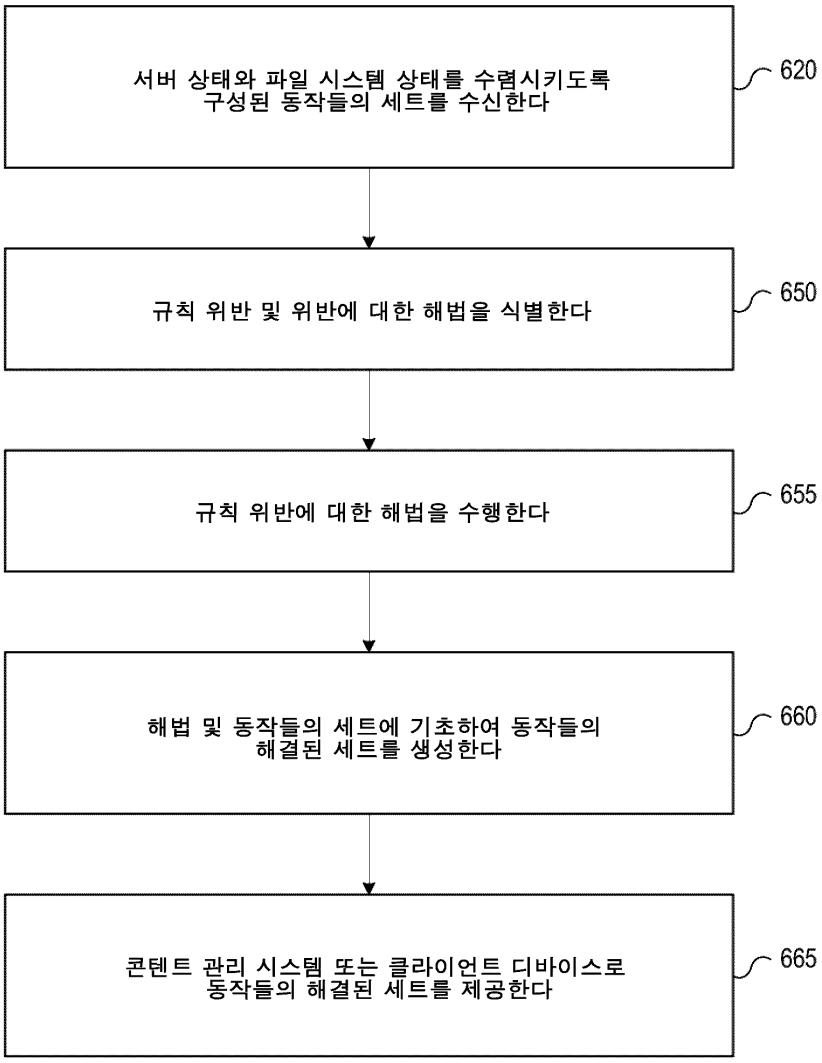
도면4


450
로컬 트리
/Root
Foo.txt
삭제
/bar
452
추가
Hi.doc
454
430
싱크 트리
/Root
Foo.txt
410
원격 트리
/Root
삭제
Foo.txt
414

도면5

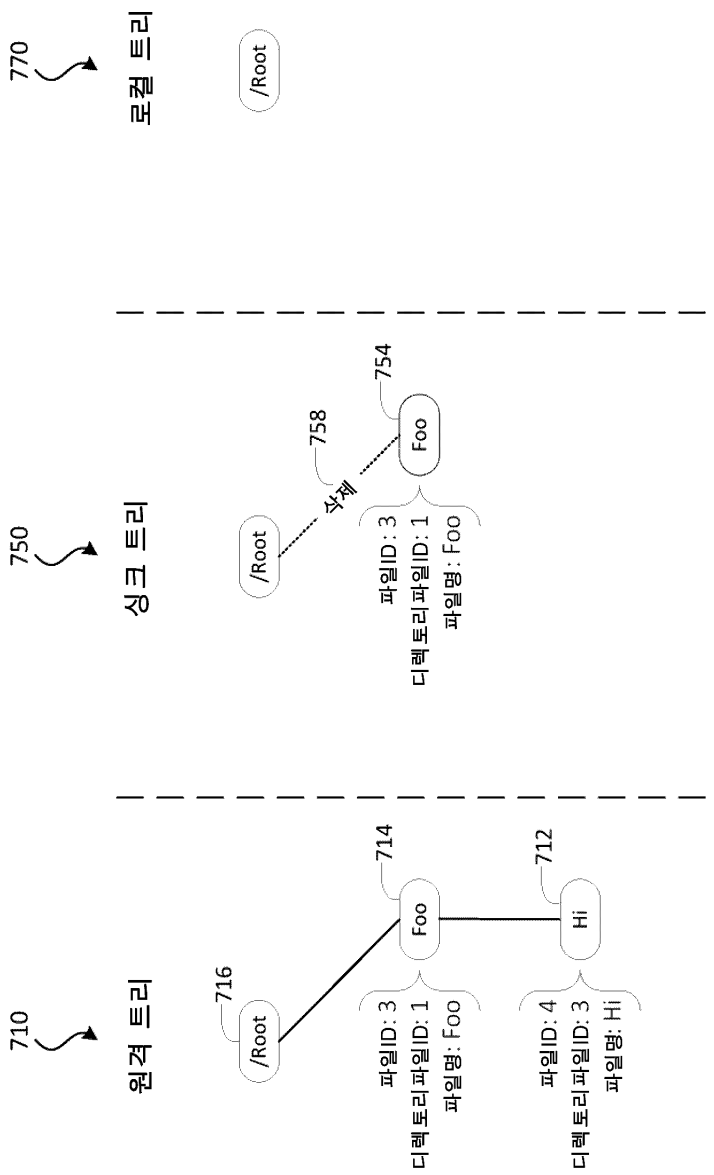

콘텐트 관리 서비스에 의해 또는 클라이언트 디바이스 상에 저장된 콘텐트 아이템에들 대한 수정 데이터를 수신한다
505
수정 데이터에 기초하여 원격 트리 또는 로컬 트리를 업데이트한다
510
동기가 맞음
서버 상태와 파일 시스템 상태가 동기가 맞는지 여부를 결정한다
515
동기가 맞지 않음
서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 세트를 생성한다
520
콘텐트 관리 시스템 또는 클라이언트 디바이스로 동작들의 세트를 제공한다
525
500

도면6

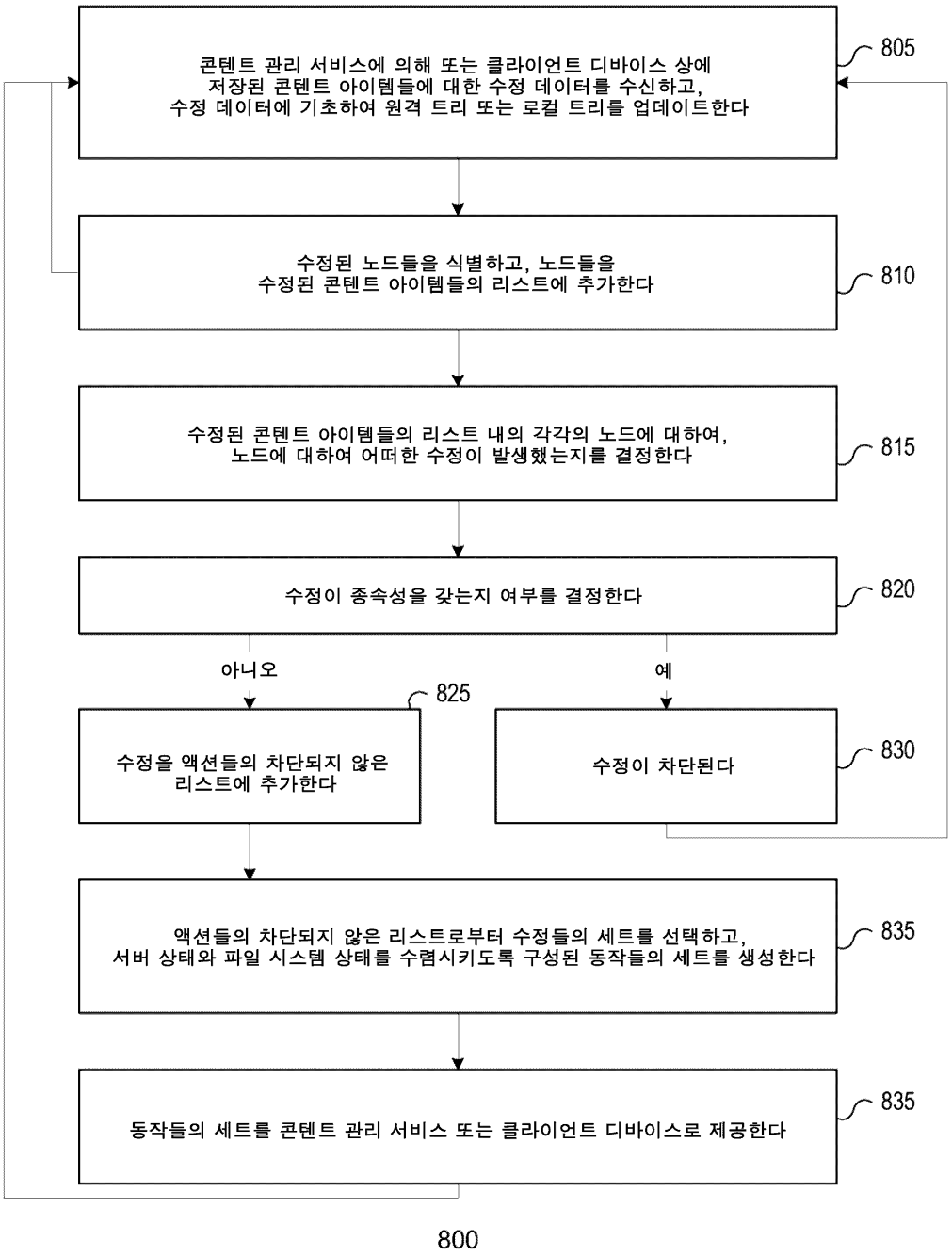

서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 세트를 수신한다
620
규칙 위반 및 위반에 대한 해법을 식별한다
650
규칙 위반에 대한 해법을 수행한다
655
해법 및 동작들의 세트에 기초하여 동작들의 해결된 세트를 생성한다
660
콘텐트 관리 시스템 또는 클라이언트 디바이스로 동작들의 해결된 세트를 제공한다
665

600

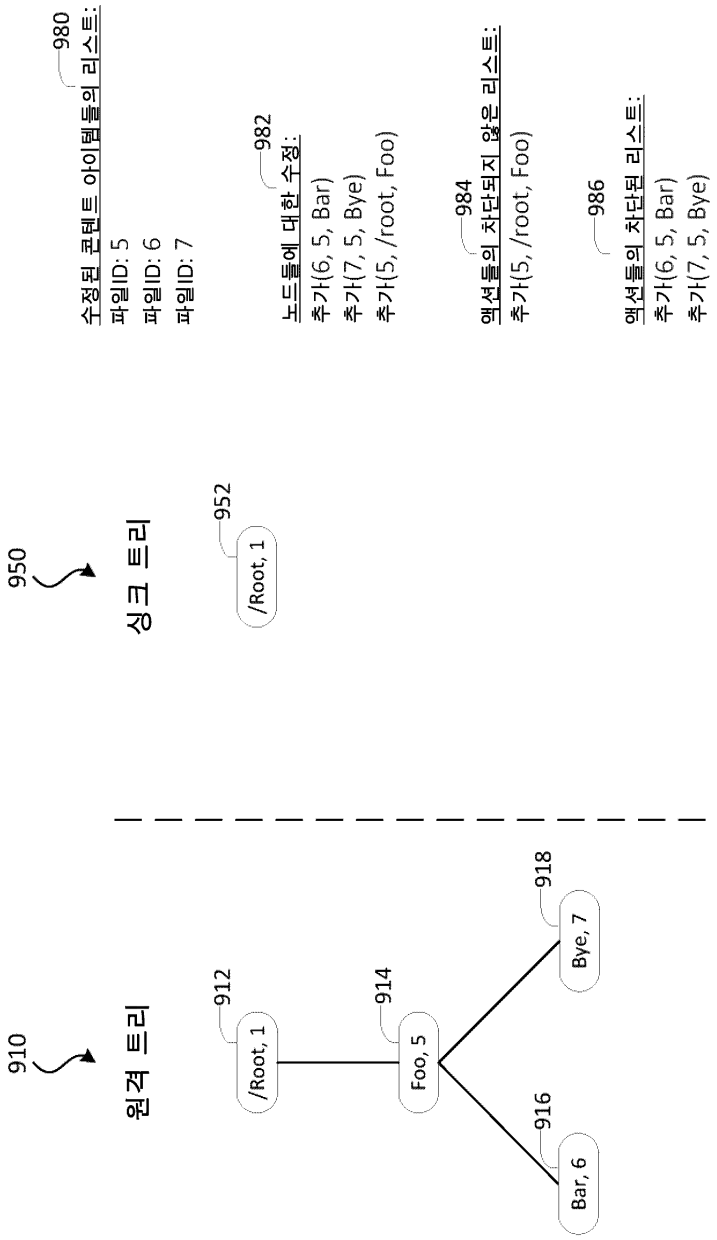
도면7


710
원격 트리
716
/Root
714
Foo
파일ID: 3
디렉토리파일ID: 1
파일명: Foo
712
Hi
파일ID: 4
디렉토리파일ID: 3
파일명: Hi
750
싱크 트리
/Root
758
삭제
754
Foo
파일ID: 3
디렉토리파일ID: 1
파일명: Foo
770
로컬 트리
/Root

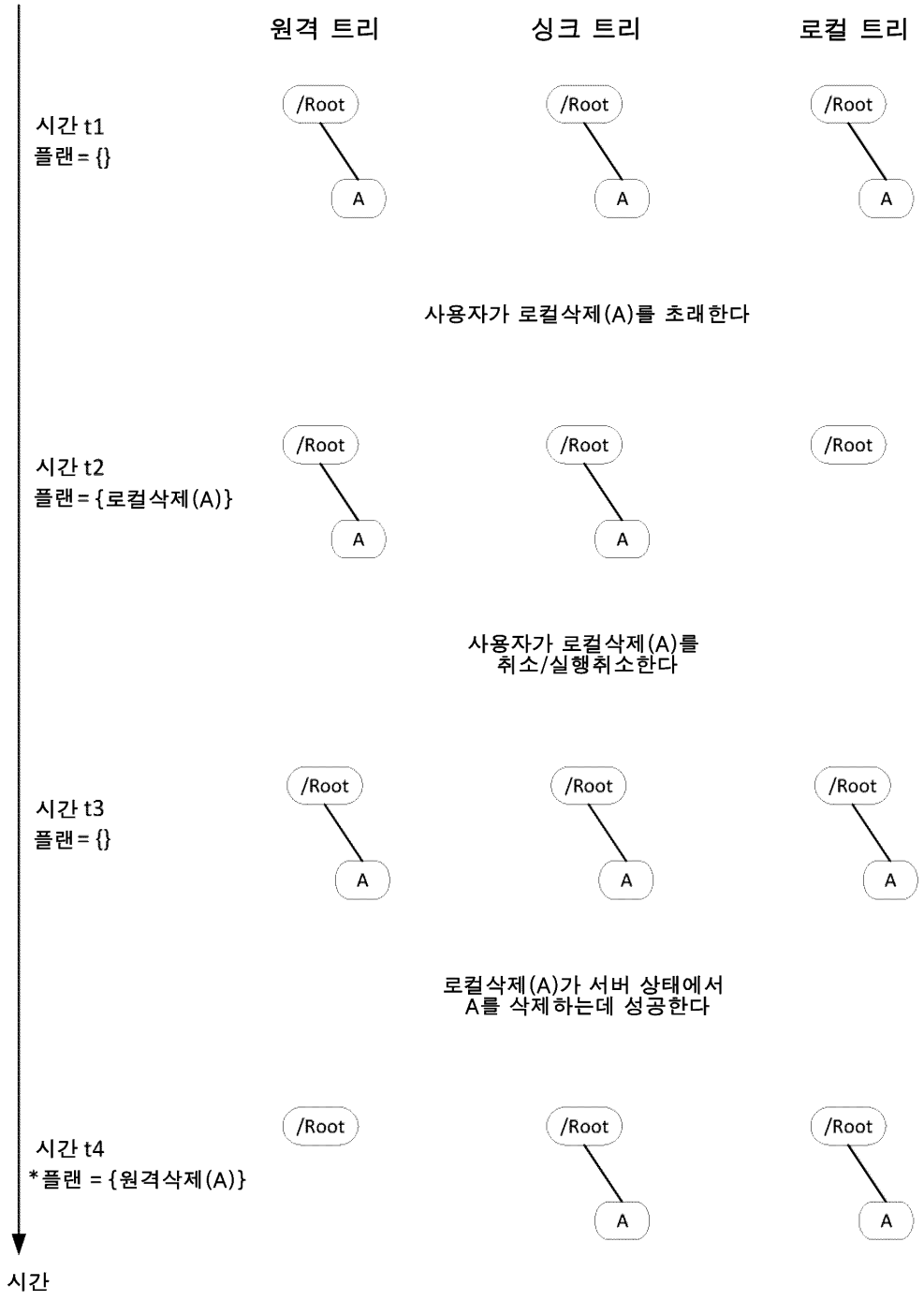
도면8


콘텐트 관리 서비스에 의해 또는 클라이언트 디바이스 상에 저장된 콘텐트 아이템들에 대한 수정 데이터를 수신하고, 수정 데이터에 기초하여 원격 트리 또는 로컬 트리를 업데이트한다
805
수정된 노드들을 식별하고, 노드들을 수정된 콘텐트 아이템들의 리스트에 추가한다
810
수정된 콘텐트 아이템들의 리스트 내의 각각의 노드에 대하여, 노드에 대하여 어떠한 수정이 발생했는지를 결정한다
815
수정이 종속성을 갖는지 여부를 결정한다
820
아니오
예
825
수정을 액션들의 차단되지 않은 리스트에 추가한다
수정이 차단된다
830
액션들의 차단되지 않은 리스트로부터 수정들의 세트를 선택하고, 서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 세트를 생성한다
835
동작들의 세트를 콘텐트 관리 서비스 또는 클라이언트 디바이스로 제공한다
835
800

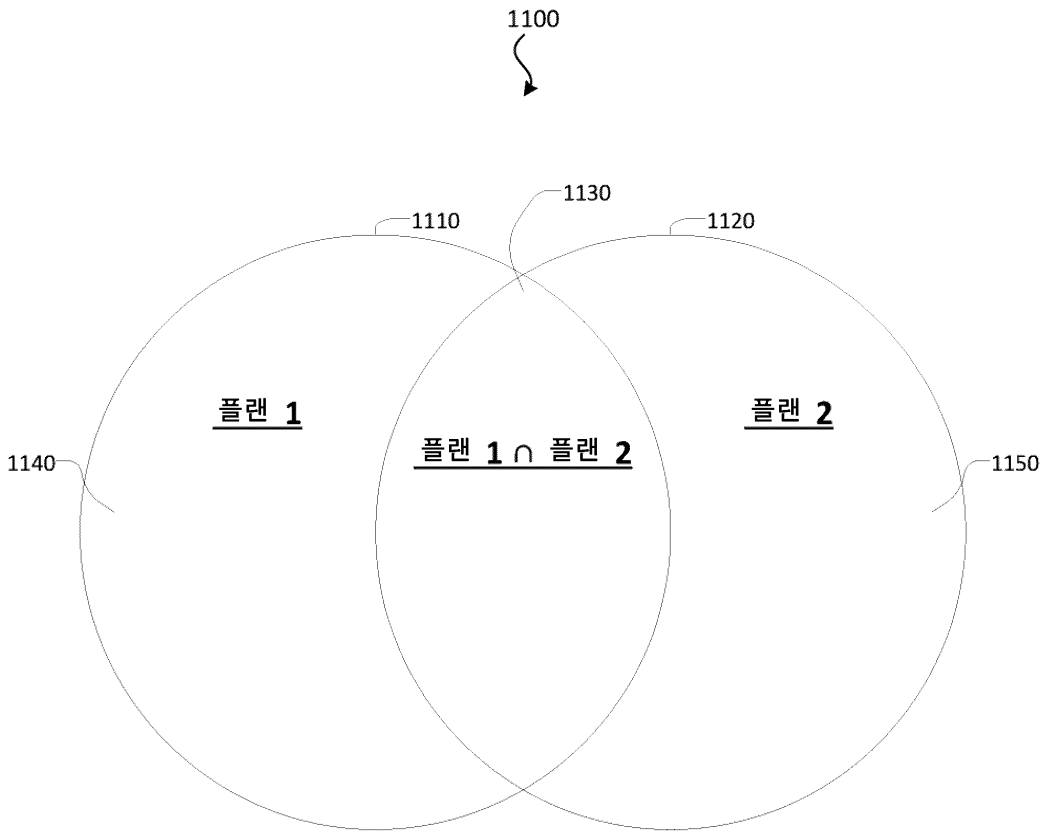
도면9


910
원격 트리
912
/Root, 1
914
Foo, 5
916
Bar, 6
918
Bye, 7
950
싱크 트리
952
/Root, 1
980
수정된 콘텐트 아이템들의 리스트:
파일ID: 5
파일ID: 6
파일ID: 7
982
노드들에 대한 수정:
추가(6, 5, Bar)
추가(7, 5, Bye)
추가(5, /root, Foo)
984
액션들의 차단되지 않은 리스트:
추가(5, /root, Foo)
986
액션들의 차단된 리스트:
추가(6, 5, Bar)
추가(7, 5, Bye)

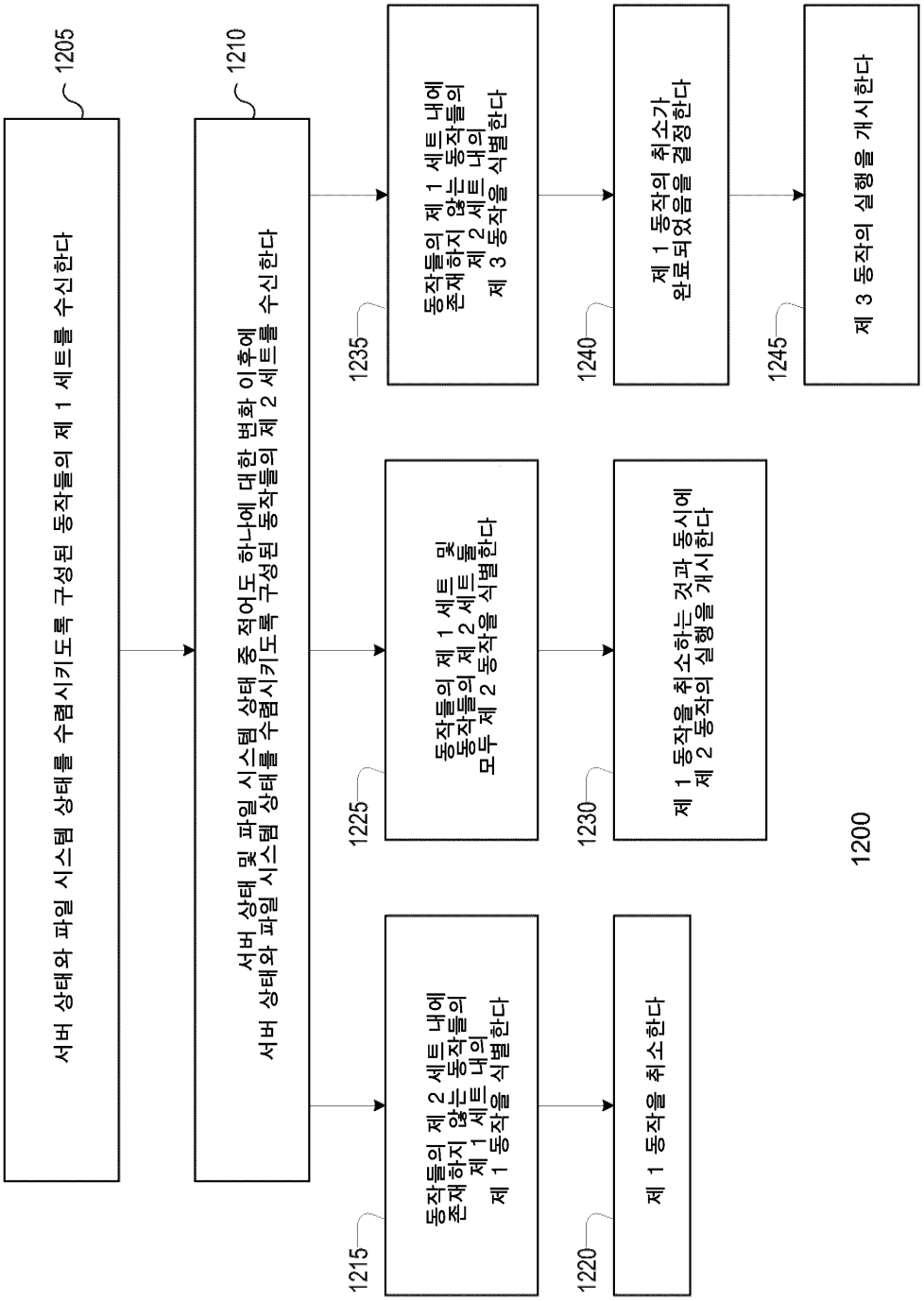
도면10


원격 트리
싱크 트리
로컬 트리
시간 t1
플랜 = {}
/Root
A
사용자가 로컬삭제(A)를 초래한다
시간 t2
플랜 = {로컬삭제(A)}
사용자가 로컬삭제(A)를
취소/실행취소한다
시간 t3
플랜 = {}
로컬삭제(A)가 서버 상태에서
A를 삭제하는데 성공한다
시간 t4
*플랜 = {원격삭제(A)}
시간

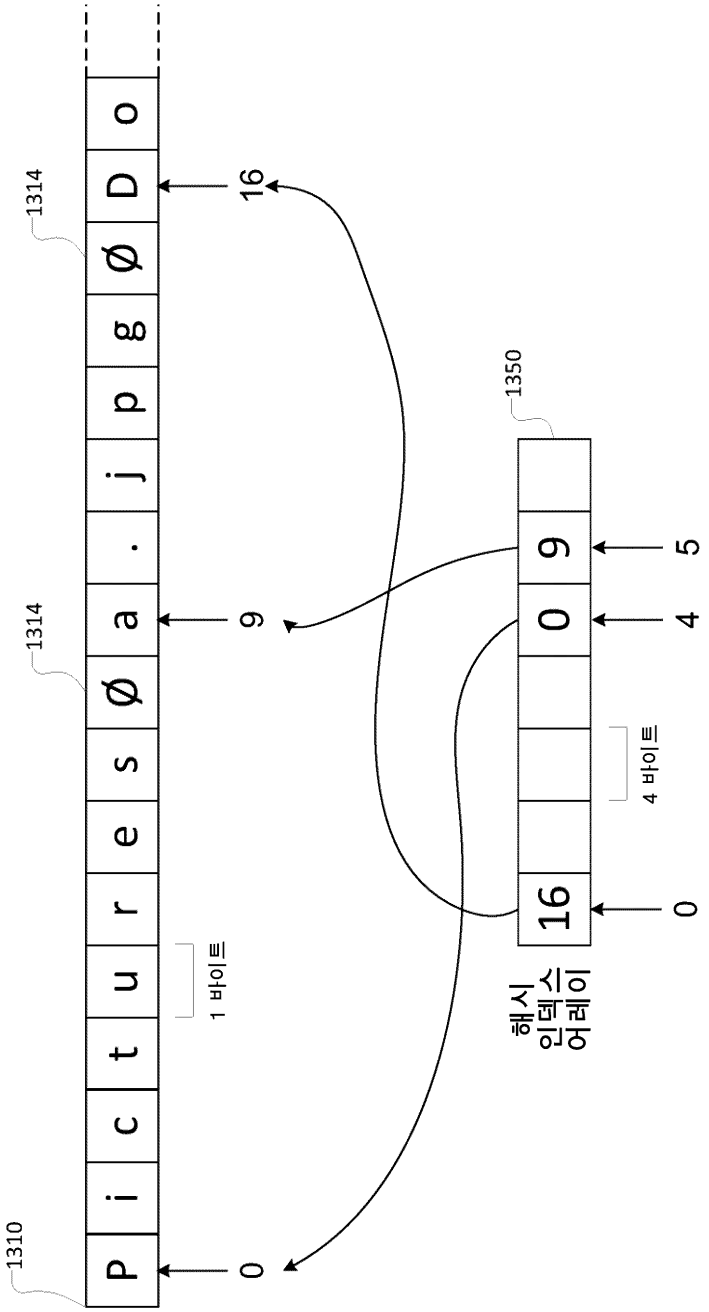
도면11



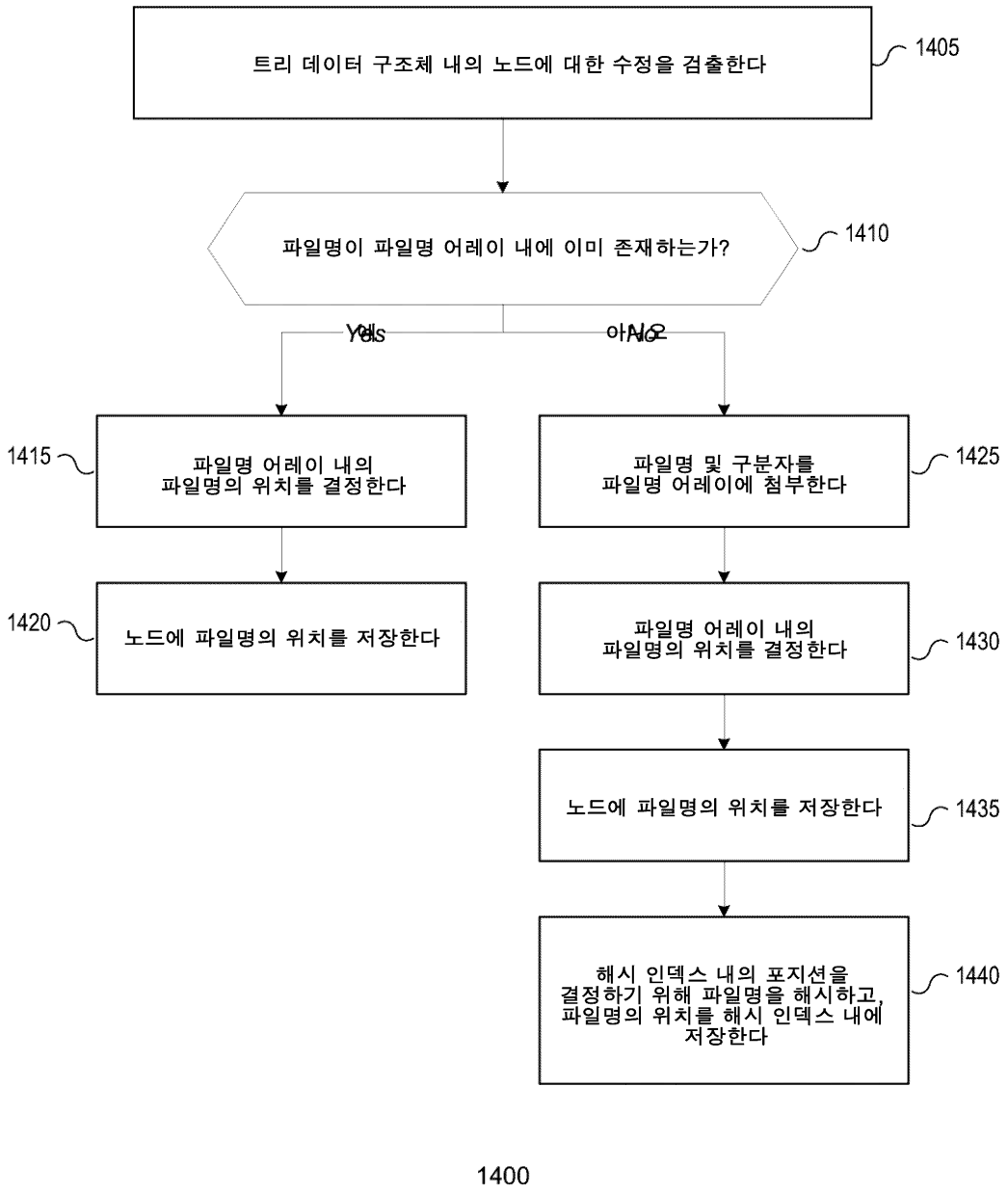
도면12


1205
서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 제 1 세트를 수신한다
1210
서버 상태 및 파일 시스템 상태 중 적어도 하나에 대한 변화 이후에
서버 상태와 파일 시스템 상태를 수렴시키도록 구성된 동작들의 제 2 세트를 수신한다
1215
동작들의 제 2 세트 내에
존재하지 않는 동작들의
제 1 세트 내의
제 1 동작을 식별한다
1220
제 1 동작을 취소한다
1225
동작들의 제 1 세트 및
동작들의 제 2 세트를
모두 제 2 동작을 식별한다
1230
제 1 동작을 취소하는 것과 동시에
제 2 동작의 실행을 개시한다
1235
동작들의 제 1 세트 내에
존재하지 않는 동작들의
제 2 세트 내의
제 3 동작을 식별한다
1240
제 1 동작의 취소가
완료되었음을 결정한다
1245
제 3 동작의 실행을 개시한다
1200

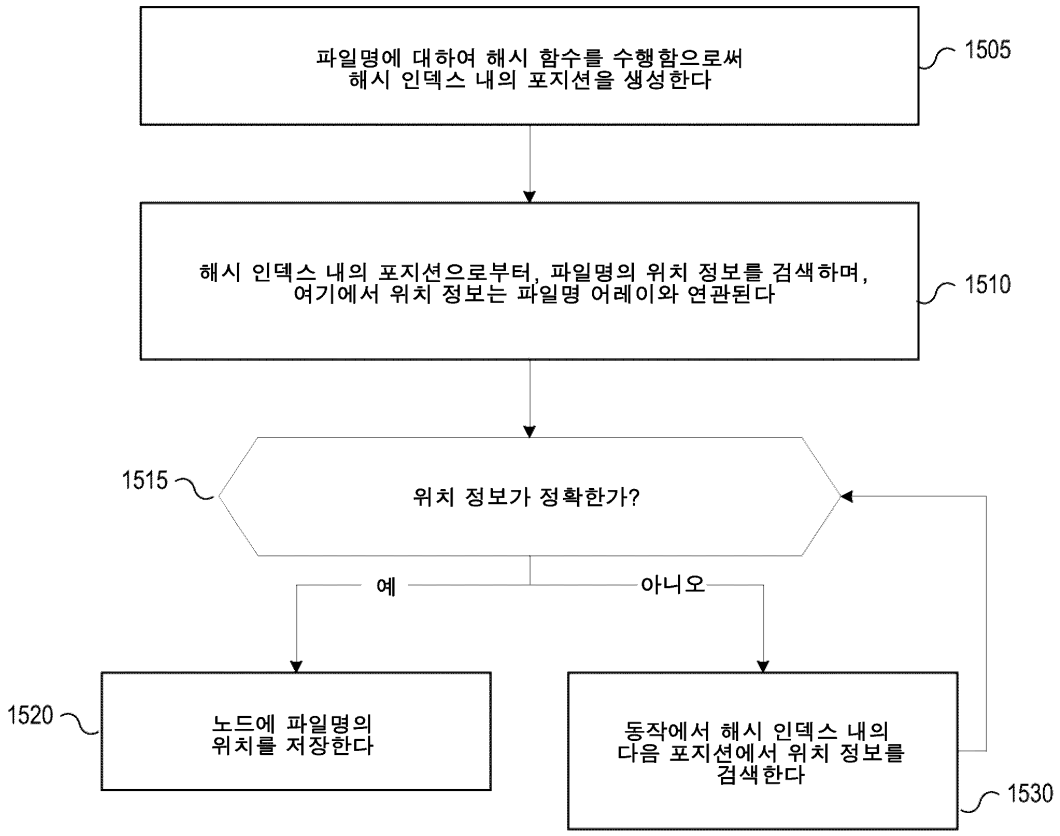
도면13


1310
P
i
c
t
u
r
e
s
Ø
a
.
j
p
g
Ø
D
o
1 바이트
1314
1314
0
9
16
1350
16
0
9
해시 인덱스 어레이
0
4
5
4 바이트

도면14

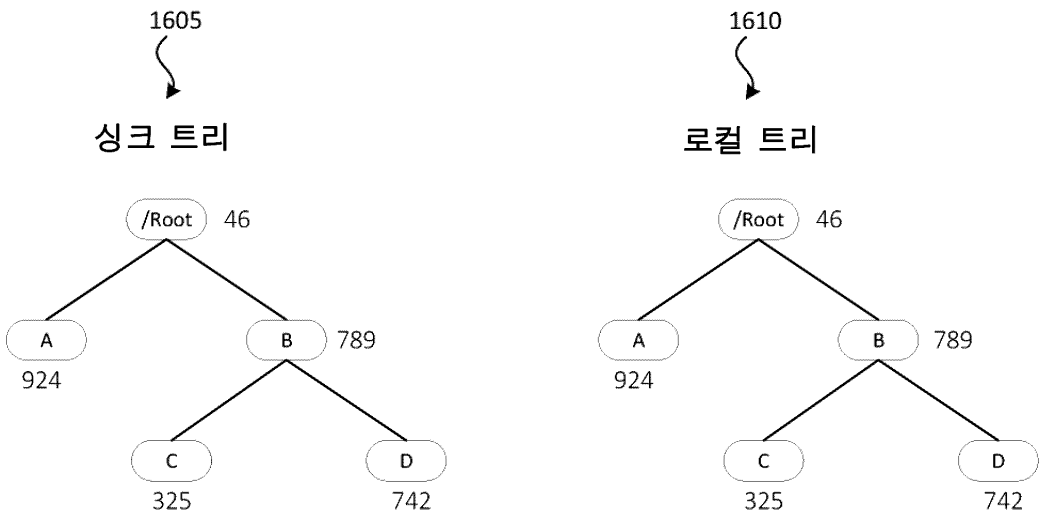

트리 데이터 구조체 내의 노드에 대한 수정을 검출한다
1405
파일명이 파일명 어레이 내에 이미 존재하는가?
1410
Yes
아니오
1415
파일명 어레이 내의
파일명의 위치를 결정한다
1420
노드에 파일명의 위치를 저장한다
파일명 및 구분자를
파일명 어레이에 첨부한다
1425
파일명 어레이 내의
파일명의 위치를 결정한다
1430
노드에 파일명의 위치를 저장한다
1435
해시 인덱스 내의 포지션을
결정하기 위해 파일명을 해시하고,
파일명의 위치를 해시 인덱스 내에
저장한다
1440
1400

도면15

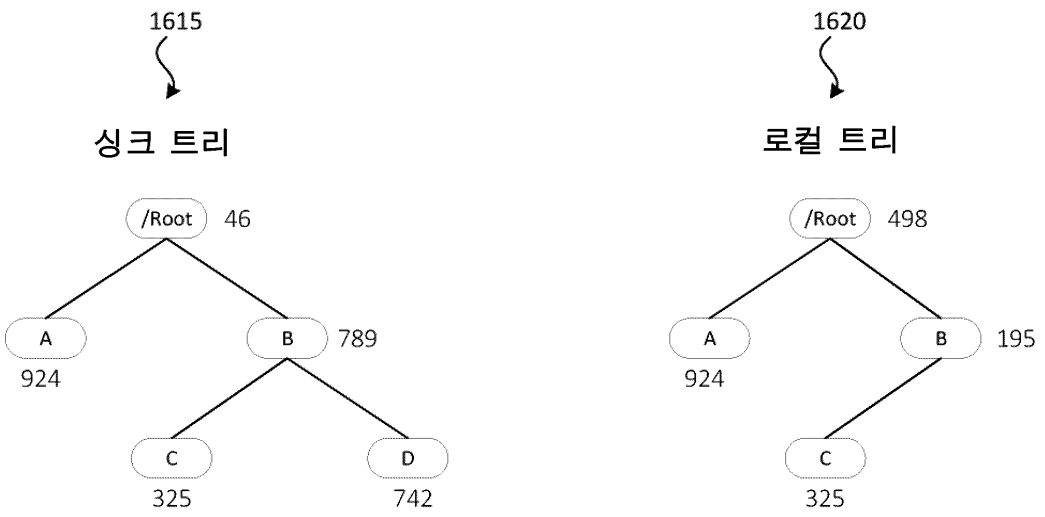

파일명에 대하여 해시 함수를 수행함으로써
해시 인덱스 내의 포지션을 생성한다
1505
해시 인덱스 내의 포지션으로부터, 파일명의 위치 정보를 검색하며,
여기에서 위치 정보는 파일명 어레이와 연관된다
1510
1515
위치 정보가 정확한가?
예
아니오
1520
노드에 파일명의
위치를 저장한다
동작에서 해시 인덱스 내의
다음 포지션에서 위치 정보를
검색한다
1530

1500

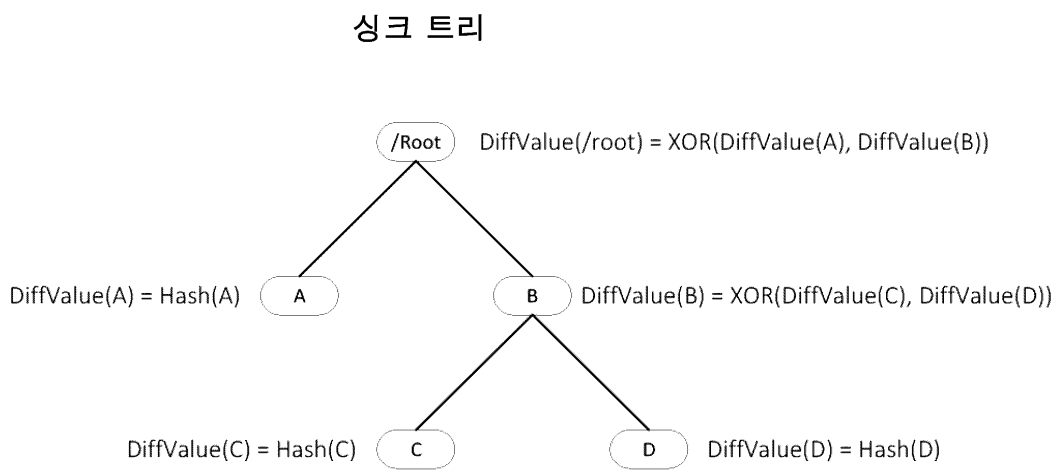
도면16a



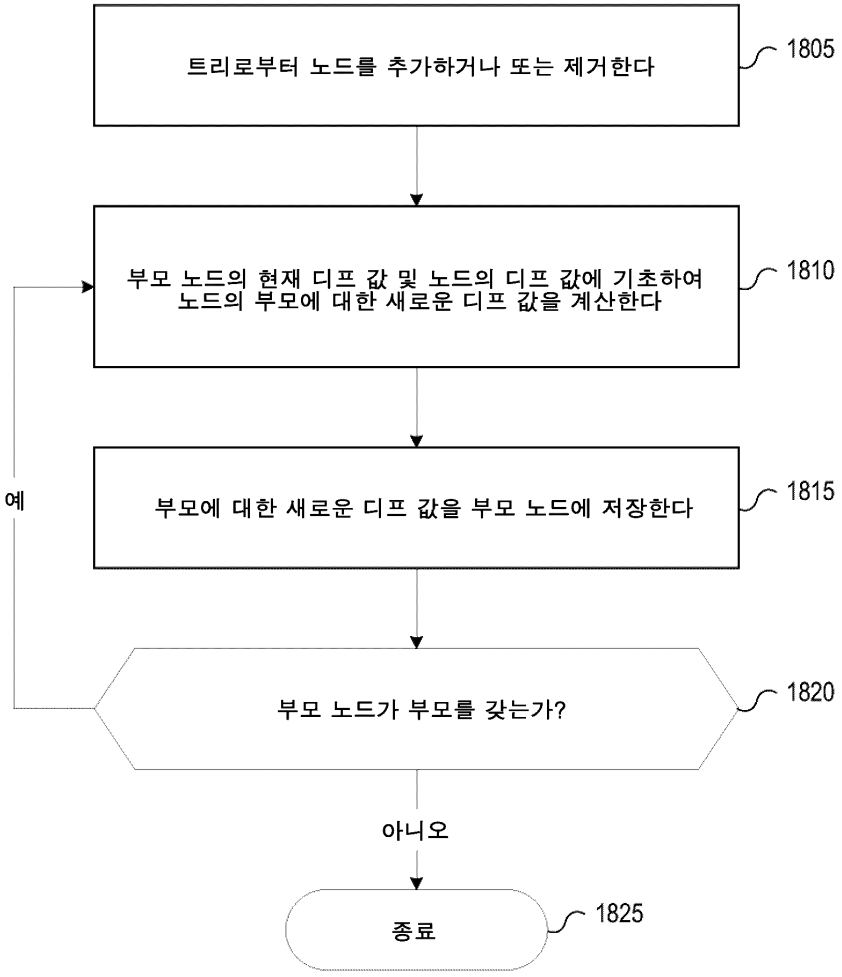
도면16b



도면17

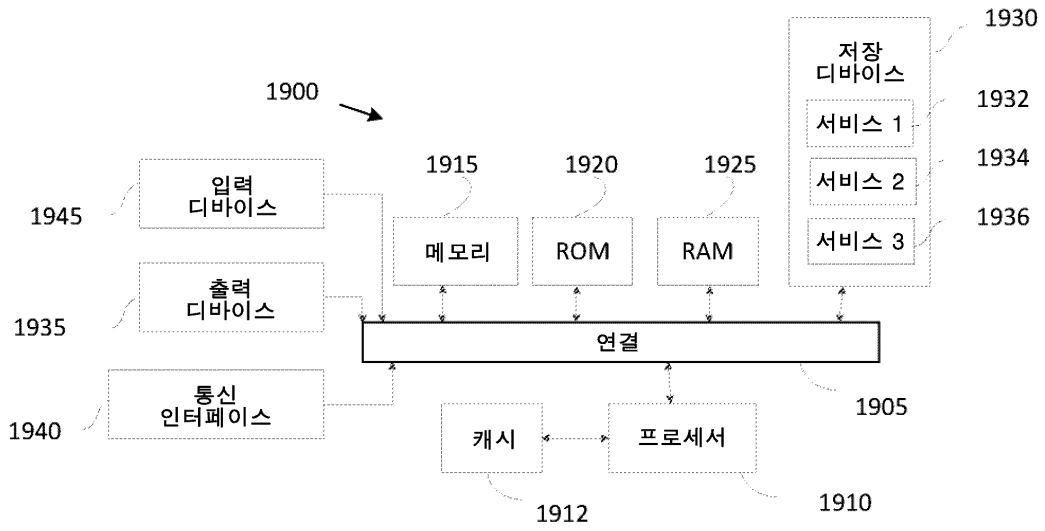


도면18


트리로부터 노드를 추가하거나 또는 제거한다
1805
부모 노드의 현재 디프 값 및 노드의 디프 값에 기초하여 노드의 부모에 대한 새로운 디프 값을 계산한다
1810
부모에 대한 새로운 디프 값을 부모 노드에 저장한다
1815
예
부모 노드가 부모를 갖는가?
1820
아니오
종료
1825

1800

도면19


1900
1930
저장
디바이스
1932
서비스 1
1934
서비스 2
1936
서비스 3
1915
1920
1925
메모리
ROM
RAM
1945
입력
디바이스
1935
출력
디바이스
연결
1905
1940
통신
인터페이스
캐시
프로세서
1912
1910