(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2008-0033264
(43) 공개일자 2008년04월16일

(51) Int. Cl.
G06F 17/30 (2006.01) *G06F 12/06* (2006.01)
G06F 17/40 (2006.01)
(21) 출원번호 10-2008-7001016
(22) 출원일자 2008년01월14일
심사청구일자 없음
번역문제출일자 2008년01월14일
(86) 국제출원번호 PCT/US2006/026853
국제출원일자 2006년07월10일
(87) 국제공개번호 WO 2007/011576
국제공개일자 2007년01월25일
(30) 우선권주장
11/181,440 2005년07월14일 미국(US)

(71) 출원인
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
바라크리쉬난, 쇼바나
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
하베와라, 사로쉬, 사이러스
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
(뒷면에 계속)
(74) 대리인
양영준, 백만기

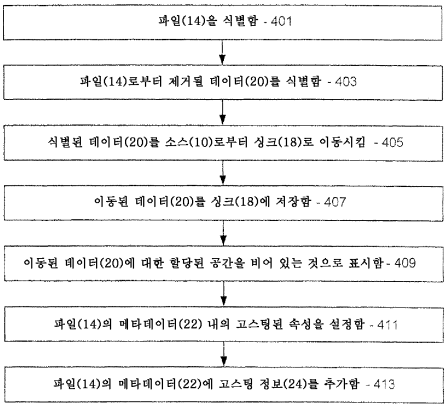
전체 청구항 수 : 총 19 항

(54) 공간을 비워두기 위해 저장 볼륨 상의 파일로부터 대안의장소로의 데이터 이동

(57) 요 약

컴퓨팅 장치는 파일 시스템에 의해 저장 볼륨 상에 저장되고 이러한 파일 시스템에 통해 액세스되는 파일을 갖는다. 파일은 데이터 및 데이터에 관련된 메타데이터를 포함하는 것으로 정의되며, 데이터가 볼륨 상의 장소를 거의 차지하지 않고 파일이 축소된 고스팅된 형태로 있도록 파일의 데이터의 적어도 일부분이 파일로부터 제거되어 대안의 장소에 저장된다. 고스팅된 파일은, 대안의 장소로부터 데이터를 검색하고 이러한 검색된 데이터를 이러한 고스팅된 파일과 연관시킴으로써, 사용하기 위해 재구성되어 재구성된 파일을 형성한다.

대표도 - 도4



(72) 발명자

수리야나라야난, 구한

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이

로버트, 크리스토프, 프랭크

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이

테오도시우, 단

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이

브조르너, 니콜라이, 에스.

미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이

특허청구의 범위

청구항 1

컴퓨팅 장치로서,

저장 볼륨,

상기 저장 볼륨을 관리하는 파일 시스템, 및

상기 파일 시스템에 의해 상기 저장 볼륨 상에 저장되고 이러한 파일 시스템을 통해 액세스되는 파일을 가지며,

상기 파일은 데이터 및 상기 데이터에 관련된 메타데이터를 포함하는 것으로 정의되고,

상기 데이터가 상기 볼륨 상의 장소를 거의 차지하지 않고 상기 파일이 축소된 고스팅된 형태로 있도록 상기 파일의 상기 데이터의 적어도 일부분이 상기 파일로부터 제거되어 대안의 장소에 저장되며,

상기 고스팅된 파일은, 상기 대안의 장소로부터 상기 데이터를 검색하고 이러한 검색된 데이터를 이러한 고스팅된 파일과 연관시킴으로써, 사용하기 위해 재구성되어 재구성된 파일을 형성하는 것인 컴퓨팅 장치.

청구항 2

제1항에 있어서, 상기 컴퓨팅 장치는 지점 서버이고,

상기 대안의 장소는 상기 지점 서버로부터 원격지에 있는 허브 서버이며,

상기 허브 서버는 복수의 이러한 지점 서버에 서비스를 제공하는 것인 컴퓨팅 장치.

청구항 3

제1항에 있어서, 상기 고스팅된 파일은 오래된 것(stale) 또는 관련없는 것(irrelevant) 중 적어도 하나로 판정된 것인 컴퓨팅 장치.

청구항 4

제1항에 있어서, 상기 볼륨 상의 상기 고스팅된 파일은 상기 파일 시스템에 의해 파일 장소에 논리적으로 저장되어 있고,

상기 고스팅된 파일을 고스팅하는 것은 그의 파일 장소를 변경하지 않으며, 그에 의해 상기 고스팅된 파일 및 그의 장소를 식별하기 위해 사용자가 상기 파일 시스템을 통해 상기 볼륨을 브라우징할 수 있는 것인 컴퓨팅 장치.

청구항 5

제1항에 있어서, 상기 볼륨 상에 존재하는 상기 고스팅된 파일의 상기 메타데이터는 상기 고스팅 이전에 상기 파일로부터의 거의 모든 메타데이터를 포함하고, 또한 상기 대안의 장소로부터 상기 파일에 대한 데이터를 검색하기 위해 이용되는 고스팅 정보도 포함하는 것인 컴퓨팅 장치.

청구항 6

제1항에 있어서, 상기 파일은 주 데이터(primary data) 및 보조 데이터(secondary data)를 포함하는 데이터를 포함하는 것으로 정의되고,

상기 주 데이터는 상기 보조 데이터보다 더 크고,

적어도 상기 파일의 상기 주 데이터가 상기 파일로부터 제거되어 상기 대안의 장소에 저장되는 것인 컴퓨팅 장치.

청구항 7

저장 볼륨, 상기 저장 볼륨을 관리하는 파일 시스템, 및 상기 파일 시스템에 의해 상기 저장 볼륨 상에 저장되고 이러한 파일 시스템을 통해 액세스되는 파일을 갖는 컴퓨팅 장치와 관련한 방법으로서,

상기 파일은 데이터 및 상기 데이터와 관련된 메타데이터를 포함하는 것으로 정의되고,

상기 파일의 적어도 일부분이 차지하고 있는 상기 볼륨 상의 공간을 비워두는 상기 방법은,

상기 파일을 식별하는 단계,

상기 파일로부터 제거될 상기 데이터의 적어도 일부분을 식별하는 단계,

상기 파일이 축소되어 고스팅된 형태로 있도록 상기 식별된 데이터를 상기 파일로부터 저장될 상기 대안의 장소로 이동시키는 단계,

상기 이동된 데이터가 이전에 차지하고 있던 상기 볼륨 상의 공간을 비어 있는 것으로 표시하는 단계, 및

상기 대안의 장소로부터 상기 파일의 상기 이동된 데이터를 검색하는 데 이용될 수 있는 정보를 포함한 고스팅 정보를 포함하도록 현재-고스팅된 파일의 상기 메타데이터를 수정하는 단계를 포함하는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 8

제7항에 있어서, 상기 식별된 데이터를 상기 파일로부터 상기 대안의 장소로 이동시키고 이러한 이동된 데이터를 상기 파일의 ID(identification)와 함께 상기 대안의 장소에 저장하는 단계를 포함하는, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 9

제7항에 있어서, 상기 이동된 데이터가 이전에 차지하고 있던 공간을 비어 있는 것으로 표시하는 결과로서 상기 볼륨 상의 상기 고스팅된 파일의 물리적 크기는 상기 파일의 상기 메타데이터에 나타낸 바와 같이 감소되고 상기 볼륨 상의 상기 고스팅된 파일의 논리적 파일 크기는 상기 파일의 상기 메타데이터에 나타낸 바와 같이 감소되지 않는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 10

제7항에 있어서, 이러한 파일이 현재 고스팅되어 있음을 보여주는 속성을 설정하기 위해 상기 현재-고스팅된 파일의 상기 메타데이터를 수정하는 단계를 더 포함하며, 그에 의해 상기 컴퓨팅 장치 상의 애플리케이션이 이러한 속성으로부터 상기 파일이 현재 고스팅되어 있는 것으로 판정할 수 있고, 따라서 이러한 파일이 그의 상기 데이터에 액세스하기 이전에 먼저 비고스팅된 형태로 재구성되어야만 한다는 것을 이러한 애플리케이션이 알게 되는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 11

제7항에 있어서, 상기 파일이 오래된 것(stale) 또는 관련없는 것(irrelevant) 중 적어도 하나라고 판정한 것에 기초하여 상기 파일을 식별하는 단계를 포함하는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 12

제7항에 있어서, 상기 볼륨 상의 상기 고스팅된 파일이 상기 파일 시스템에 의해 파일 장소에 논리적으로 저장되고,

상기 방법은 상기 고스팅된 파일의 상기 파일 장소를 변경하지 않는 단계를 포함하고, 그에 의해 사용자가 상기 고스팅된 파일 및 그의 장소를 식별하기 위해 상기 파일 시스템을 통해 상기 볼륨을 브라우징할 수 있는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 13

제7항에 있어서, 상기 파일은 주 데이터 및 보조 데이터를 포함하는 데이터를 포함하는 것으로 정의되고, 상기 주 데이터는 상기 보조 데이터보다 더 크며,

상기 방법은,

상기 주 데이터의 적어도 일부분을 상기 파일로부터 제거될 데이터로서 식별하는 단계, 및

상기 파일이 축소되어 고스팅된 형태로 있도록 상기 식별된 주 데이터를 상기 파일로부터 저장될 상기 대안의 장소로 이동시키는 단계를 포함하는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 14

제7항에 있어서, 상기 고스팅된 파일을 재구성하는 단계를 더 포함하며,

이러한 재구성하는 단계는,

상기 고스팅된 파일의 상기 이동된 데이터에 액세스하려는 요청을 수신하는 단계,

상기 볼륨 상에서 상기 고스팅된 파일을 찾아내는 단계,

상기 파일의 상기 메타데이터에서 상기 고스팅 정보를 식별하는 단계,

상기 식별된 고스팅 정보에 기초하여 상기 대안의 장소에서 상기 파일의 상기 이동된 데이터를 찾아내는 단계,

상기 찾아낸 데이터의 적어도 일부분을 상기 대안의 장소로부터 상기 컴퓨팅 장치로 이동시키는 단계, 및

상기 재구성된 파일을 형성하기 위해 상기 이동된 데이터를 이러한 고스팅된 파일과 연관시키는 단계를 포함하는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 15

제14항에 있어서, 이러한 파일이 고스팅되어 있음을 보여주는 속성을 리셋하기 위해 상기 재구성된 파일의 상기 메타데이터를 수정하는 단계를 더 포함하는, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 16

제14항에 있어서, 상기 재구성된 파일을 형성하기 위해 상기 이동된 데이터를 상기 고스팅된 파일과 연관시키는 상기 단계는 상기 파일에 상기 볼륨 상에서 상기 이동된 데이터가 차지하게 될 공간을 할당하고 상기 이동된 데이터를 상기 할당된 공간에 저장하는 단계를 포함하는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 17

제14항에 있어서, 상기 고스팅 정보를 제거하기 위해 상기 재구성된 파일의 상기 메타데이터를 수정하는 단계를 더 포함하는, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 18

제14항에 있어서, 상기 파일 시스템이 상기 고스팅된 파일의 상기 이동된 데이터에 액세스하려는 상기 요청을 수신하고 상기 고스팅된 파일이 이러한 이동된 데이터를 포함하지 않는다는 것을 발견하여 에러를 반환하는 단계를 포함하고,

상기 컴퓨팅 장치는 고스팅 필터를 더 가지며,

상기 고스팅 필터는 상기 에러를 가로채기하고 상기 파일 시스템을 이용하여 상기 가로채기된 에러에 기초하여 상기 고스팅된 파일의 상기 메타데이터로부터 상기 고스팅 정보를 획득하며, 상기 획득된 고스팅 정보에 기초하여 상기 고스팅된 파일의 재구성을 트리거하는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

청구항 19

제18항에 있어서, 상기 고스팅 필터는 상기 대안의 장소에 액세스하지 않는 하위-레벨 구성자(lower-level construct)이고,

상기 컴퓨팅 장치는 고스팅 관리자를 더 가지며,

상기 고스팅 관리자는 상기 대안의 장소에 액세스하고 상기 고스팅 필터와 인터페이스하는 상위-레벨 구성자

(higher-level construct)이고,

상기 고스팅 필터는 상기 고스팅 관리자에 상기 고스팅된 파일의 재구성을 수행하도록 요청함으로써 그 재구성을 트리거하는 것인, 파일의 적어도 일부분이 차지하고 있는 볼륨 상의 공간을 비워두는 방법.

명 세 서

기 술 분 야

<1> 본 발명은 저장 볼륨 상의 공간을 비워두기 위해 데이터가 저장 볼륨(storage volume) 상에 저장된 컴퓨터 파일로부터 대안의 장소로 이동 또는 "고스팅(ghost)"될 수 있게 해주는 아키텍처 및 방법에 관한 것이다. 보다 상세하게는, 본 발명은 고스팅된 파일(ghosted file)의 나머지가 저장 볼륨 상에 있고 대안의 장소에 있는 고스팅된 데이터(ghosted data)가 필요한 경우 검색되고 고스팅된 파일에 다시 배치되어 디고스팅된 파일(de-ghosted file)이 얻어지는 이러한 아키텍처 및 방법에 관한 것이다.

배 경 기 술

<2> 퍼스널 컴퓨터, 컴퓨터 서버, 기타 등등 등의 컴퓨팅 장치에서, 공지된 바와 같이, 데이터는 일반적으로 컴퓨팅 장치의 하나 이상의 로컬 저장 볼륨 상에 존재하는 로컬 컴퓨터 파일의 형태로 컴퓨팅 장치 상에 영속적으로 저장된다. 각각의 이러한 저장 볼륨은 컴퓨팅 장치 등의 하드 드라이브 상에 존재할 수 있고, 이러한 저장 볼륨은 컴퓨팅 장치 상에서 실행되는 파일 시스템에 의해 구성되고, 그를 통해 액세스되며, 다른 방식으로 제어될 수 있으며, 이 역시 공지된 바와 같다.

<3> 때때로, 저장 볼륨 상의 컴퓨터 파일들 중 일부, 다수 및 아마도 심지어 대부분이 관심 없는 것이고 "쓸모없는(cold)" 것으로 간주될 수 있는 경우가 있을 수 있다. 즉, 이러한 쓸모없는 파일들은, 예를 들어, 다소 긴 시간 동안 액세스되지 않았고 및/또는 다소 긴 시간 동안 액세스될 가능성이 없으며, 따라서 볼륨 상에 남아 있을 진정한 가치가 거의 없다.

<4> 물론, 이러한 쓸모없는 파일들은 저장 볼륨으로부터 간단히 삭제될 수 있으며, 이러한 볼륨 상에 공간이 필요한 경우에 특히 그렇다. 그렇지만, 대부분의 사용자가 단지 공간을 생성하기 위해 파일을 삭제하려고 하지 않는다는 것을 잘 알 것이다. 그에 부가하여, 단지 사용하지 않음을 알기 때문에 쓸모없는 파일을 삭제하는 것은 나쁜 습관으로 생각된다. 어쨌든, 쓸모없는 파일이, 비록 필요하다고 생각되지도 않고 필요하다고 예견되지도 않더라도, 그럼에도 불구하고 어떤 장래의 시점에서 필요하게 될지도 모를 경우가 있을 수 있다.

<5> 이러한 상황에서, 이러한 쓸모없는 파일이 볼륨 상에 여전히 존재를 유지할 수 있게 해주면서 이러한 쓸모없는 파일로부터 대안의 장소로 데이터를 이동시킴으로써 볼륨 상에 공간을 생성할 수 있으면 유용할 것이다. 즉, 쓸모없는 파일을 더 작은 "고스팅된" 형태로 볼륨 상에 남겨두면서, 이러한 쓸모없는 파일로부터 대안의 장소로 데이터를 이동시킴으로써 쓸모없는 파일을 "고스팅(ghost)"하거나 전체 파일을 대안의 장소로 복사할 수 있으면 유용할 것이다. 따라서, 실제로 고스팅된 파일이 컴퓨팅 장치 상에서 필요하게 되는 경우, 이러한 고스팅된 파일에 대한 데이터가 대안의 장소로부터 검색될 수 있고, 고스팅된 파일이 그에 기초하여 재구성될 수 있으며, 이어서 이러한 재구성된 파일이 이용될 수 있다.

<6> 다른 시나리오에서, 조직 등의 데이터 파일이 아마도 허브(hub) 등의 중앙 장소로부터 조직의 다수의 지점에 복제되어야 하는 경우가 있을 수 있다. 예를 들어, 건축 설계 회사는 그의 건축 설계 파일 전부가 이러한 회사의 몇개의 지점 중 어느 지점에서도 이용가능하도록 하고자 할 수 있다.

<7> 이러한 상황에서, 각각의 파일의 복사본을 각각의 지점으로 복제하고 모든 지점에서의 모든 파일을 최신의 것으로 유지하기 위해 네트워크화된 시스템이 구축될 수 있다. 예를 들어, 이러한 시스템에서, 중앙 집중식 허브 서버는 각각의 이러한 파일을 저장하고, 허브에 있는 각각의 파일의 복사본을 네트워크를 통해 각각의 지점에 있는 지점 서버 등으로 배포하기 위해 복제 서비스가 이용된다. 그렇지만, 조직의 파일의 수가 증가함에 따라 또 각각의 파일의 크기가 증가함에 따라 또 지점의 수가 증가함에 따라, 네트워크를 통한 트래픽의 양이 아마도 가용 대역폭을 초과하는 정도까지 증가한다는 것을 잘 알 것이다. 게다가, 허브 서버에 있는 파일들 전부의 전체 크기가 증가함에 따라, 실제로 각각의 지점 서버가 허브 서버로부터 복제되는 파일들 전부를 저장하기에 충분한 가용 공간을 갖지 않을 수 있다.

<8> 그럼에도, 이전의 시나리오와 유사하게, 특정의 지점의 지점 서버 상의 컴퓨터 파일의 일부, 다수 및 아마도 심지어 대부분이 관심 없는 것이고 불필요한 것으로 간주될 수 있다. 이러한 쓸모없는 파일들은 예를 들어 특정

의 지점과 관련이 없는 일들에 관한 것일 수 있고, 특정의 지점을 통해 액세스될 가능성이 없으며, 따라서 특정의 지점에 대한 지점 서버 상에 복제될 진정한 가치가 거의 없다. 예를 들어, Pennsylvania주 Wilkes-Barre 소재의 건축 설계 회사의 지점 사무소는 Florida주 Boynton Beach 소재의 회사의 지점 사무소에 의해 처리되는 프로젝트에 관련된 건축 설계 파일을 가질 필요가 거의 없다.

<9> 이러한 상황에서, 이전의 시나리오와 유사하게, 조직에 대한 허브 서버로부터 이용가능한 모든 다른 비관련 파일들을 특정의 지점의 지점 서버 상에 단지 부분적으로만 저장하면서 이러한 특정의 지점에 관련된 그 파일들만을 이러한 특정의 지점의 지점 서버 상에 완전히 저장할 수 있으면 유용할 것이다. 따라서, 이전의 시나리오와 비슷한 방식으로, 특정의 지점의 지점 서버에 있는 비관련 파일들을 '고스팅'함으로써 각각의 비관련 파일이 지점 서버에 더 작은 '고스팅된' 형태로 남아 있도록 할 수 있으면 유용할 것이다. 따라서, 다시 말하면, 실제로 고스팅된 파일이 지점 서버에서 필요하면, 이러한 고스팅된 파일에 대한 데이터가 허브 서버로부터 검색될 수 있고, 고스팅된 파일이 그에 기초하여 재구성될 수 있으며, 이어서 이러한 재구성된 파일이 이용될 수 있다.

<10> 그에 따라, 로컬 볼륨 또는 지점 서버 등의 소스(source)에 있는 파일이 복제 또는 고스팅됨으로써 그의 데이터가 대안의 장소 또는 허브 서버 등의 싱크(sink)에 저장되게 하고 또 이에 따라 소스에 있는 파일이, 필요한 경우 재구성될 수 있는, 축소된 또는 고스팅된 형태로 있게 할 수 있는 방법 및 메카니즘이 필요하다. 특히, 이러한 고스팅된 파일이, 필요에 따라 작성되고 재구성될 수 있게 해주는 이러한 방법 및 메카니즘이 필요하다.

발명의 상세한 설명

<11> 상기한 필요성이 컴퓨팅 장치가 저장 볼륨, 이 저장 볼륨을 관리하는 파일 시스템, 및 파일 시스템에 의해 저장 볼륨 상에 저장되고 이러한 파일 시스템을 통해 액세스되는 파일을 갖는 본 발명에 의해 적어도 부분적으로 만족된다. 이 파일은 데이터 및 상기 데이터에 관련된 메타데이터를 포함하는 것으로 정의되고, 상기 데이터가 상기 볼륨 상의 장소를 거의 차지하지 않고 상기 파일이 축소되어 고스팅된 형태로 있도록 상기 파일의 상기 데이터의 적어도 일부분이 상기 파일로부터 제거되어 대안의 장소에 저장된다. 상기 고스팅된 파일은, 상기 대안의 장소로부터 상기 데이터를 검색하고 이러한 검색된 데이터를 이러한 고스팅된 파일과 연관시킴으로써, 사용하기 위해 재구성되어 재구성된 파일을 형성한다.

<12> 상기 파일을 고스팅하기 위해, 상기 파일이 식별되고, 상기 파일로부터 제거될 상기 데이터의 적어도 일부분이 식별되며, 상기 파일이 축소되어 고스팅된 형태로 있도록 상기 식별된 데이터가 상기 파일로부터 저장될 상기 대안의 장소로 이동된다. 상기 이동된 데이터가 이전에 차지하고 있던 상기 볼륨 상의 공간이 이에 따라 비어 있는 것으로 표시된다. 그에 부가하여, 상기 대안의 장소로부터 상기 파일의 상기 이동된 데이터를 검색하는 데 이용될 수 있는 정보를 포함한 고스팅 정보를 포함하도록 현재-고스팅된 파일의 상기 메타데이터가 수정된다.

<13> 상기 고스팅된 파일은 상기 고스팅된 파일의 상기 이동된 데이터에 액세스하려는 요청을 수신할 시에 재구성된다. 이에 응답하여, 상기 고스팅된 파일을 상기 볼륨 상에서 찾아내고, 상기 파일의 상기 메타데이터에서 상기 고스팅 정보가 식별되며, 상기 식별된 고스팅 정보에 기초하여 상기 대안의 장소에서 상기 파일의 상기 이동된 데이터를 찾아낸다. 그 후에, 상기 찾아낸 데이터가 상기 대안의 장소로부터 상기 컴퓨팅 장치로 이동되고, 상기 재구성된 파일을 형성하기 위해 상기 이동된 데이터가 이러한 고스팅된 파일과 연관된다.

실 시 예

<26> 이상의 요약은 물론 본 발명의 실시예들에 대한 이하의 상세한 설명은 첨부 도면과 관련하여 읽어보면 잘 이해될 것이다. 본 발명의 설명을 위해, 현재 양호한 실시예들이 도면에 도시되어 있다. 그렇지만, 잘 알 것인 바와 같이, 본 발명이 도시된 정확한 구성 및 수단으로 한정되지 않는다.

<27> 컴퓨터 환경

<28> 도 1 및 이하의 논의는 본 발명 및/또는 그의 일부분이 구현될 수 있는 적당한 컴퓨팅 환경의 간략한 전체적인 설명을 제공하기 위한 것이다. 꼭 그럴 필요는 없지만, 본 발명은 일반적으로 클라이언트 워크스테이션 또는 서버 등의 컴퓨터에 의해 실행되는 프로그램 모듈 등의 컴퓨터 실행가능 명령어와 관련하여 기술된다. 일반적으로, 프로그램 모듈은 특정의 태스크를 수행하거나 특정의 추상 데이터 유형을 구현하는 루틴, 프로그램, 객체, 컴포넌트, 데이터 구조, 기타 등등을 포함한다. 게다가, 본 발명 및/또는 그의 일부분이 핸드-헬드 장치, 멀티-프로세서 시스템, 마이크로프로세서-기반 또는 프로그램가능 가전 제품, 네트워크 PC, 미니컴퓨터, 메인프레임 컴퓨터, 기타 등등을 비롯한 다른 컴퓨터 시스템 구성에서 실시될 수 있다는 것을 잘 알 것이다.

본 발명은 또한 통신 네트워크를 통해 연결되어 있는 원격 처리 장치에 의해 태스크들이 수행되는 분산 컴퓨팅 환경에서 실시될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 로컬 및 원격 메모리 저장 장치 둘다에 위치할 수 있다.

<29> 도 1에 도시된 바와 같이, 예시적인 범용 컴퓨팅 시스템은, 처리 장치(121), 시스템 메모리(122), 및 시스템 메모리를 비롯한 여러가지 시스템 컴포넌트를 처리 장치(121)에 연결시키는 시스템 버스(123)를 포함하는, 종래의 퍼스널 컴퓨터(120), 기타 등등을 포함한다. 시스템 버스(123)는 메모리 버스나 메모리 컨트롤러, 주변 장치 버스, 및 각종의 버스 아키텍처 중 임의의 것을 사용하는 로컬 버스를 비롯한 몇가지 유형의 버스 구조 중 임의의 것일 수 있다. 시스템 메모리는 판독 전용 메모리(ROM)(124) 및 랜덤 액세스 메모리(RAM)(125)를 포함한다. 시동 중과 같은 때에, 퍼스널 컴퓨터(120) 내의 구성요소들 간에 정보를 전달하는 데 도움을 주는 기본적인 루틴을 포함하는 기본 입/출력 시스템(BIOS)(126)이 ROM(124)에 저장되어 있다.

<30> 퍼스널 컴퓨터(120)는 또한 하드 디스크(도시 생략)로부터 판독을 하고 그에 기록을 하는 하드 디스크 드라이브(127), 이동식 자기 디스크(129)로부터 판독을 하거나 그에 기록을 하는 자기 디스크 드라이브(128), 및 CD-ROM 또는 기타 광 매체 등의 이동식 광 디스크(131)로부터 판독을 하거나 그에 기록을 하는 광 디스크 드라이브(130)를 포함할 수 있다. 하드 디스크 드라이브(127), 자기 디스크 드라이브(128), 및 광 디스크 드라이브(130)는 각각 하드 디스크 드라이브 인터페이스(132), 자기 디스크 드라이브 인터페이스(133), 및 광 드라이브 인터페이스(134)에 의해 시스템 버스(123)에 연결된다. 이들 드라이브 및 그와 연관된 컴퓨터 판독가능 매체는 퍼스널 컴퓨터(120)에 대한 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 및 기타 데이터의 비휘발성 저장을 제공한다.

<31> 본 명세서에 기술된 예시적인 환경이 하드 디스크, 이동식 자기 디스크(129) 및 이동식 광 디스크(131)를 이용하고 있지만, 컴퓨터에 의해 액세스가능한 데이터를 저장할 수 있는 기타 유형의 컴퓨터 판독가능 매체도 역시 예시적인 운영 환경에서 사용될 수 있다는 것을 잘 알 것이다. 이러한 기타 유형의 매체는 자기 카세트, 플래쉬 메모리 카드, 디지털 비디오 디스크, 베르누이 카트리지, 랜덤 액세스 메모리(RAM), 판독 전용 메모리(ROM), 기타 등등을 포함한다.

<32> 운영 체제(135), 하나 이상의 애플리케이션 프로그램(136), 기타 프로그램 모듈(137) 및 프로그램 데이터(138)를 비롯한 다수의 프로그램 모듈이 하드 디스크, 자기 디스크(129), 광 디스크(131), ROM(124) 또는 RAM(125)에 저장될 수 있다. 사용자는 키보드(140) 및 포인팅 장치(142) 등의 입력 장치를 통해 퍼스널 컴퓨터(120)에 명령 및 정보를 입력할 수 있다. 다른 입력 장치(도시 생략)는 마이크, 조이스틱, 게임 패드, 위성 안테나, 스캐너, 기타 등등을 포함할 수 있다. 이들 및 다른 입력 장치는 종종 시스템 버스에 연결된 직렬 포트 인터페이스(146)를 통해 처리 장치(121)에 연결되어 있지만, 병렬 포트, 게임 포트, 또는 USB(universal serial bus) 등의 기타 인터페이스에 의해 연결될 수 있다. 모니터(147) 또는 기타 유형의 디스플레이 장치도 비디오 어댑터(148) 등의 인터페이스를 통해 시스템 버스(123)에 연결되어 있다. 모니터(147)에 부가하여, 퍼스널 컴퓨터는 일반적으로 스피커 및 프린터 등의 기타 주변 출력 장치(도시 생략)를 포함한다. 도 1의 예시적인 시스템은 또한 호스트 어댑터(155), SCSI(Small Computer System Interface) 버스(156), 및 SCSI 버스(156)에 연결된 외장형 저장 장치(162)를 포함한다.

<33> 퍼스널 컴퓨터(120)는 원격 컴퓨터(149) 등의 하나 이상의 원격 컴퓨터에의 논리적 접속을 사용하여 네트워크화된 환경에서 동작할 수 있다. 원격 컴퓨터(149)는 또하나의 퍼스널 컴퓨터, 서버, 라우터, 네트워크 PC, 피어 장치 또는 기타 통상의 네트워크 노드일 수 있으며, 일반적으로 퍼스널 컴퓨터(120)와 관련하여 상기한 요소들 중 다수 또는 그 전부를 포함하지만, 메모리 저장 장치(150)만이 도 1에 도시되어 있다. 도 1에 도시된 논리적 접속은 근거리 통신망(LAN)(151) 및 원거리 통신망(WAN)(152)을 포함한다. 이러한 네트워킹 환경은 사무실, 전사적 컴퓨터 네트워크, 인트라넷 및 인터넷에서 통상적인 것이다.

<34> LAN 네트워킹 환경에서 사용될 때, 퍼스널 컴퓨터(120)는 네트워크 인터페이스 또는 어댑터(153)를 통해 LAN(151)에 연결되어 있다. WAN 네트워킹 환경에서 사용될 때, 퍼스널 컴퓨터(120)는 일반적으로 인터넷 등의 WAN(152)을 통해 통신을 설정하는 모뎀(154) 또는 기타 수단을 포함한다. 내장형 또는 외장형일 수 있는 모뎀(154)은 직렬 포트 인터페이스(146)를 통해 시스템 버스(123)에 연결된다. 네트워크화된 환경에서, 퍼스널 컴퓨터(120) 또는 그의 일부분과 관련하여 도시된 프로그램 모듈들은 원격 메모리 저장 장치에 저장될 수 있다. 도시된 네트워크 접속이 예시적인 것이며 컴퓨터들 간에 통신 링크를 설정하는 기타 수단이 사용될 수 있다는 것을 잘 알 것이다.

<35> 파일의 고스팅 및 재구성

<36> 본 발명에서, 이제 도 2를 참조하면, 퍼스널 컴퓨터 또는 컴퓨터 서버 기타 등등의 컴퓨팅 장치(10)는 하드 드라이브 또는 영속적 RAM 드라이브, 기타 등등의 저장 볼륨(12)을 가지며, 이 볼륨(12)은 다수의 컴퓨터 파일(14)를 저장하고 있고, 볼륨(12) 상의 파일들(14)은 컴퓨팅 장치(10) 상에서 실행 중인 파일 시스템(16)에 의해 구성되고, 그를 통해 액세스되며, 그에 의해 다른 방식으로 제어된다. 잘 알 수 있는 바와 같이, 컴퓨팅 장치(10), 볼륨(12), 파일(14) 및 파일 시스템(16)은 본 발명의 정신 및 범위를 벗어나지 않고 임의의 유형의 컴퓨팅 장치, 볼륨, 파일 및 파일 시스템일 수 있다.

<37> 본 발명의 일 실시예에서, 컴퓨팅 장치(10)는 퍼스널 컴퓨터, 기타 등등이고, 이 컴퓨팅 장치의 볼륨(12) 상의 파일들 중 적어도 일부는 이러한 파일(14)이 예를 들어 다소 긴 시간 동안 액세스되지 않았고 및/또는 다소 긴 시간 동안 액세스되지 않을 가능성이 있다는 점에서 쓸모없는 것으로 판정되었으며, 따라서 볼륨(12) 상에 남아 있을 진정한 가치가 거의 없다. 그렇지만, 유의할 점은 파일(14)의 쓸모없음(coldness)은 본 발명의 정신 및 범위를 벗어나지 않고 임의의 적절한 방식으로 정의될 수 있다는 것이다.

<38> 어쨌든, 쓸모없다고 판정할 시에, 파일(14)이 볼륨(12)으로부터 삭제되지 않고 그 대신에 그 파일의 적어도 일부 데이터(20)를 대안의 장소(18)로 이동시킴으로써 볼륨(12) 상에서의 크기가 감소된다. 이러한 대안의 장소(18)는 컴퓨팅 장치에 로컬일 수 있거나 그로부터 원격지에 있을 수 있다. 일반적으로, 이러한 대안의 장소(18)는 컴퓨팅 장치(10)에 적절히 연결되어 있으며, 본 발명의 정신 및 범위를 벗어나지 않고 임의의 적절한 저장 장소일 수 있다. 예를 들어, 대안의 장소(18)는 저장 장치(10) 상의 다른 볼륨(12), 다른 컴퓨팅 장치(10) 상의 다른 볼륨(12), 서버에 있는 파일 웨어하우스(file warehouse), 원격 서버에 있는 장기 저장 장치(long-term storage device), 기타 등등일 수 있다.

<39> 대안의 장소에서, 그 다음에 상기 데이터(20)를 이러한 쓸모없는 파일(14)로부터 이러한 대안의 장소(18)로 이동시킴으로써 볼륨(12) 상의 공간이 비워질 수 있다. 중요한 것은, 이러한 쓸모없는 파일(14)로부터 데이터(20)가 이동되었지만, 이러한 쓸모없는 파일(14)이, 비록 축소된 또는 '고스팅된' 형태이지만, 볼륨(12) 상에 존재 또는 '고스트'로서 남아 있다는 것이다. 따라서, 실제로 고스팅된 파일(14)이 컴퓨팅 장치(10)에서 필요하게 되면, 그 파일의 데이터(20)를 대안의 장소(18)로부터 검색하여 이러한 검색된 데이터(20)를 이러한 고스팅된 파일(14)과 재연관시켜 재구성된 파일(14)을 형성함으로써 이러한 고스팅된 파일(14)이 재구성된다. 잘 알 수 있는 바와 같이, 그 다음에 일단 재구성되었으면, 파일(14)은 이어서 실제로 이용될 수 있다.

<40> 본 발명의 다른 실시예에서, 컴퓨팅 장치(10)는 지점 서버, 기타 등등이고, 그의 볼륨(12) 상의 파일들(14) 중 적어도 일부는, 이러한 파일이 예를 들어 이러한 지점 서버와 연관된 지점 사무소와 관련이 없다는 점에서, 무관한 것으로 판정된다. 물론, 파일(14)에 대한 무관성(irrelevance)이 본 발명의 정신 및 범위를 벗어나지 않고 임의의 적절한 방식으로 정의될 수 있다.

<41> 어쨌든, 이전과 같이, 무관한 파일(14)이 그의 적어도 일부 데이터(20)가 없는 축소된 형식으로 지점 서버(10)의 볼륨(12) 상에 유지된다. 여기에서, 이러한 데이터(20)는 조직(이 조직의 지점 사무소가 지점임)에 의해 유지되는 중앙 집중식 허브 서버 등의 대안의 장소(18)에 저장된다. 다시 말하면, 이러한 허브 서버(18)는 지점 서버(10)와 적절히 연결된다.

<42> 허브 서버(18)에서, 그 다음에, 지점 서버(10)의 볼륨(12) 상에서 사용되는 공간이 최소화될 수 있고, 이러한 무관한 파일(14)로부터의 이러한 상기 데이터(20)를 이러한 허브 서버(18)에 저장함으로써 이러한 공간을 채우는데 필요한 대역폭도 최소화될 수 있다. 중요한 것은, 이전과 유사하게, 이러한 무관한 파일(14)로부터의 데이터(20)가 지점 서버(10)에 존재하지 않지만, 이러한 무관한 파일(14)이, 비록 축소된 또는 '고스팅된' 형태이지만, 지점 서버의 볼륨(12) 상에 존재 또는 '고스트'로서 남아 있다는 것이다. 따라서, 실제로 고스팅된 파일(14)이 지점 서버(10)에서 필요하게 되는 경우, 이러한 고스팅된 파일(14)은, 그의 데이터(20)를 허브 서버(18)로부터 검색하고 이러한 검색된 데이터(20)를 이러한 고스팅된 파일(14)과 재연관시켜 재구성된 파일(14)을 형성함으로써, 재구성된다. 다시 말하면, 일단 재구성되었으면, 파일(14)이 실제로 이용될 수 있다.

<43> 일반화하기 위해, 퍼스널 컴퓨터 등의 컴퓨팅 장치(10) 및 대안의 장소(18), 즉 지점 서버(10) 및 허브 서버(19)를 포함하는 구성이든지 또는 다른 구성이든지 간에, '고스팅'은 도 2에 나타낸 바와 같이 소스(10)에 있는 볼륨(12) 상의 파일(14)을 필요로 하며, 여기서 파일(14)이 단지 이러한 파일(14)과 연관된 데이터(20)의 어떤 일부분을 포함하지 않는 축소된 형식으로 볼륨(12) 상에 존재하는지가 판정된다. 따라서, 도 2에서 알 수 있는 바와 같이, 이러한 데이터(20)는 소스(10)에 적절히 연결된 싱크(18)에 저장된다.

<44> 이러한 싱크(18)에서, 볼륨(12) 상의 고스팅된 파일(14)이 축소된 또는 "스텁(stub)" 형태로 존재한다는 점에서

소스(10)의 볼륨(12) 상의 사용된 공간이 최소화될 수 있다. 따라서, 실제로 고스팅된 파일(14)이 소스(10)에서 필요하게 되면, 싱크(18)로부터 그 파일의 데이터(20)를 검색하고 이러한 검색된 데이터(20)를 이러한 고스팅된 파일(14)과 재연관시켜 재구성된 파일(14)을 형성함으로써 이러한 고스팅된 파일(14)이 재구성된다. 다시 말하면, 재구성되면, 파일(14)이 필요에 따라 소스(10)에서 이용될 수 있다.

<45> 본 발명의 일 실시예에서, 소스(10)의 볼륨 상에 존재하는 고스팅된 파일(14)의 스텁이 원래의 파일(14)과 동일한 볼륨(12) 상의 장소에 저장된다. 이에 따라, 파일 시스템(16)이 디렉토리 형식에 따라 또는 다른 방식으로 볼륨(12)을 구성하는지에 상관없이, 특정의 장소에서 고스팅된 파일(14)을 찾는 파일 시스템(16)은 이러한 장소에서 파일(14)에 대한 스텁, 즉 고스팅된 파일(14)을 찾아야만 한다. 중요한 것은, 소스(10)의 볼륨(12) 상에 존재하는 고스팅된 파일(14)이 원래의 비고스팅된 파일(14)로부터의 모든 메타데이터(22)를 포함하고, 그 중에서도 특히, 싱크(18)로부터 파일(14)에 대한 데이터(20)를 검색하는 데 이용될 수 있는 고스팅 정보(24)도 포함한다. 잘 알 수 있는 바와 같이, 이러한 고스팅 정보(24)는 파일(14)에 대한 메타데이터(22)에 추가될 수 있거나 파일(14)의 다른 장소에 저장될 수 있다.

<46> 그에 따라, 볼륨(12) 상의 파일들(14) 중 어떤 것은 고스팅되고 볼륨(12) 상의 파일들(14) 중 어떤 것은 고스팅되어 있지 않더라도, 사용자 등은 파일 시스템(16)에 의해 소스(10)의 전체 볼륨(12)을 브라우징할 수 있다. 상세하게는, 그 중에서도 특히, 고스팅된 파일(14)을 식별하고 크기 정보, 날짜 정보, 기타 등등을 제공하기 위해 이러한 브라우징 동안에 고스팅된 파일(14)과 연관된 메타데이터(22)가 이용될 수 있다. 사용자가 고스팅된 파일(14)에 액세스하려고 시도할 때, 그 파일의 데이터(20)는 고스팅된 파일(14)에 존재하는 고스팅 정보(24)에 기초하여 검색되고, 고스팅된 파일(14)이 비고스팅된 파일(14)로 재구성되며, 비고스팅된 파일(14)이 실제로 사용자에 의해 액세스될 수 있다.

<47> 이제 도 3a를 참조하면, 소스(10)에 있는 볼륨(12) 상에 존재할 수 있는 비고스팅된 파일(14)이 헤더, 기타 등등으로 구성될 수 있는 메타데이터(22)를 포함한다는 것을 알 수 있으며, 여기서 이러한 메타데이터(22)는 논리적 파일 크기, 볼륨(12) 상에서의 크기, 생성 시간, 수정 시간, 액세스 시간, 액세스 퍼미션(access permission), 및 각종의 파일 속성 등의 비고스팅된 파일(14)의 데이터(20)에 관한 정보를 포함한다. 그에 부가하여, 명백한 바와 같이, 소스(10)에 있는 볼륨(12) 상에 존재하는 이러한 비고스팅된 파일(14)은 파일(14)의 실제 데이터(20)를 포함한다. 여기서 유의할 점은 이러한 실제 데이터(20)가 주 데이터(primary data) 및 대체 데이터(alternate data)로서 구성될 수 있으며, 여기서 주 데이터는 애플리케이션(30) 등(도 2)에 의해 이용될 수 있는 파일(14)로부터의 데이터(20)인 반면, 대체 데이터는 다른 용도를 위해 생성된 파일(14)로부터의 데이터이다. 단지 한 예로서, 이러한 대체 데이터는 아마도 소스(10)에 있는 애플리케이션(30) 또는 파일 시스템(16)에서 파일(14)의 표현을 디스플레이할 때 이용될 수 있는 "썸네일" 등의 그래픽 표현을 포함할 수 있다.

<48> 어쨌든, 크기로 볼 때 주 데이터가 데이터(20)의 대부분일 수 있다. 그에 따라, 비고스팅된 파일(14)이 실제로 고스팅되어 있을 때, 실제로는 데이터(20)의 주 데이터만이 비고스팅된 파일(14)의 나머지로부터 이동되어 고스팅된 파일(14)을 형성하고, 도 3a에 나타낸 바와 같이, 이러한 주 데이터만이, 아마도 고스팅된 파일(14)의 ID와 함께, 싱크(18)에 저장될 수도 있다. 물론, 본 발명의 정신 및 범위를 벗어나지 않고 파일(14)의 다른 부분도 파일(14)의 고스팅 동안에 이동될 수 있다.

<49> 그렇지만, 잘 알 수 있는 바와 같이, 비고스팅된 파일(14)이 실제로는 지점 서버(10) 및 허브 서버(18)와 관련하여 고스팅되어 있을 때(여기서, 허브 서버(18)는 파일(14)을 지점 서버(10)로 복제함), 실제로는 데이터(20)의 주 데이터가 소스(10)에 있는 복제된 파일(14)의 나머지로부터 이동되어 고스팅된 파일(14)을 형성하지만, 도 3b에 나타낸 바와 같이, 이러한 파일(14)의 전체가, 아마도 고스팅된 파일(14)의 ID와 함께, 싱크(18)에 저장된다.

<50> 비고스팅된 파일(14)을 고스팅된 파일(14)로 변환하기 위해(즉, 파일(14)을 고스팅하기 위해), 이제부터 도 3 및 도 4를 참조하면, 파일(14)이 먼저 식별되고(단계 401), 그 후에 파일(14)로부터 이동될 데이터(20)가 식별된다(단계 403). 다시 말하면, 이러한 데이터(20)는 일반적으로 데이터(20)의 주 데이터이지만, 이러한 데이터(20)로부터의 다른 데이터일 수 있다. 어쨌든, 식별된 데이터(20)는 이어서 적절한 전송 메카니즘 및 통로를 통해 소스(10)로부터 싱크(18)로 이동되고(단계 405), 이러한 이동된 데이터(20)는 이어서 적절한 형식으로 싱크(18)에 저장된다(단계 407). 잘 알 수 있는 바와 같이, 데이터(20)를 싱크(18)에 저장하는 것은 본 발명의 정신 및 범위를 벗어나지 않고 임의의 적절한 형식으로 행해질 수 있다. 예를 들어, 데이터(20)는 원하는 경우 압축 및/또는 암호화될 수 있고, 이러한 데이터(20)는 파일(14)을 재구성하라는 요청 시에 비교적 쉽게 검색될 수 있는 방식으로 저장되어야만 한다. 예를 들어, 이러한 데이터(20)는 검색에 이용되는 ID에 따라 저장될 수

있으며, 이에 대해서는 이하에서 보다 상세히 기술할 것이다.

<51> 단계(405)에서와 같이 데이터(20)가 일단 이동되었으면, 잘 알 수 있는 바와 같이, 소스(10)의 볼륨(12) 상에서 이러한 데이터(20)가 차지하던 공간이 파일(14)에 할당된 채로 있을 필요가 없다. 그에 따라, 이러한 할당된 공간은 비워져 있는 것으로 표시될 수 있고, 그 결과 현재-고스팅된 파일(14)이 산재(sparse)되어 있는 것으로 표시된다. 중요한 것은, 이러한 할당된 공간이 고스팅된 파일(14)에 의해 사용되지 않더라도, 현재-고스팅된 파일(14)의 논리적 파일 크기가 이러한 파일(14)에 대한 메타데이터(22) 또는 다른 곳에서 변경되어서는 안된다는 것이며, 그럼에도 파일(14)의 볼륨(12) 상에서의 크기가 실제로는 이러한 비워진 공간을 반영하기 위해 변경될 수 있다.

<52> 그에 부가하여, 본 발명의 일 실시예에서, 현재-고스팅된 파일(14)에 대한 메타데이터(22)는 이러한 파일(14)이 현재 고스팅되어 있음을 보여주기 위해 변경된다(단계 411). 그렇게 하기 위해, 예를 들어, '고스팅됨(ghosted)' 속성이 메타데이터(22)에 설정될 수 있다. 이러한 설정된 고스팅됨 속성은 주로 파일(14)이 실제로는 고스팅되어 있다는 문의측 애플리케이션(30) 등에의 신호로서 이용될 수 있다. 이러한 신호로, 문의측 애플리케이션(30)은 파일(14)이 실제로는 고스팅되어 있다는 것 뿐만 아니라 이러한 파일(14)이 먼저 비고스팅된 형태로 재구성되어야만 한다는 점에서 이러한 파일(14)에 액세스하는 데는 비용이 수반된다는 것도 알 수 있다. 잘 아는 바와 같이, 이러한 비용은, 그 중에서도 특히, 싱크(18)로부터 파일(14)에 대한 데이터(20)에 액세스하는 데 필요한 대역폭, 데이터(20)에 필요한 볼륨(12) 상의 공간, 및/또는 싱크(18)에 있는 데이터(20)에 액세스하고 그에 기초하여 파일(14)을 재구성하기 위한 지연시간의 관점에서의 비용일 수 있다.

<53> 또한, 본 발명의 일 실시예에서, 현재-고스팅된 파일(14)에 대한 메타데이터(22)는 상기한 고스팅 정보(24)를 포함하도록 변경된다(단계 413). 다시 말하면, 이러한 고스팅 정보(24)는, 그 중에서도 특히, 싱크(18)로부터 파일(14)에 대한 데이터(20)를 검색하는 데 이용될 수 있는 정보를 포함한다. 예를 들어, 이러한 고스팅 정보(24)는 싱크(18)에 데이터(20)와 함께 저장되어 그 데이터(20)를 식별해주는 ID는 물론, 싱크(18)를 어떻게 찾아내는지, 파일(14)을 어떻게 재구성하는지, 기타 등등에 관한 정보를 비롯한, 고스팅된 파일(14)에 관련된 기타 정보를 저장하는 데 이용될 수 있는 저장소의 양을 포함할 수 있다. 고스팅 정보(24)에서의 이러한 저장소는 가변적인 또는 고정된 양일 수 있으며, 후자의 경우에 16 킬로바이트 정도로 제한될 수도 있다. 잘 알 수 있는 바와 같이, 이러한 고스팅 정보(24)는 주로 파일(14)을 재구성하는 데 이용되는 구성자(construct)가 어떤 것이든 그 구성자에 의해 이용되고, 일반적으로 상기한 문의측 애플리케이션(30) 등에 의해 사용되지 않으며, 그렇지만 이러한 사용은 그럼에도 불구하고 본 발명의 정신 및 범위를 벗어나지 않고 있을 수 있다.

<54> 이제 잘 알 것인 바와 같이, 고스팅되어 있는 고스팅된 파일(14)은 짧은 시간 동안, 긴 시간 동안, 아주 긴 시간 동안, 또는 영원히, 적어도 볼륨(12)이 더 이상 동작하지 않을 때까지 소스(10)의 볼륨(12) 상에 그대로 존재할 수 있다. 그렇지만, 임의의 시점에서, 이제 도 5를 참조하면, 고스팅된 파일(14)의 저장된 데이터(20)에 액세스하려는 요청이 애플리케이션(30), 이러한 애플리케이션(30)에 있는 사용자, 기타 등등으로부터 수신(단계 501)될 수 있다는 것을 잘 알 것이다. 물론, 이러한 요청 이전에, 이러한 고스팅된 파일(14)의 메타데이터(22)에 액세스하려는 하나 이상의 요청도 수신될 수 있는 일이 역시 가능하다. 예를 들어, 상기한 애플리케이션(30)은 파일(14)이 실제로 고스팅되어 있는지를 판정하기 위해 이러한 고스팅된 파일(14)의 메타데이터(22) 내의 설정된 고스팅됨 속성에 액세스할 수 있거나, 볼륨(12)을 제어하는 파일 시스템(16)이 디렉토리 리스트 등을 컴파일하는 도중에 메타데이터(22)에 액세스할 수 있다. 후자의 경우에 주목할 만한 것은, 파일(14)의 표현을 디스플레이할 때 이용될 수 있는 상기하나 그래픽을 획득하기 위해, 파일 시스템(16), 애플리케이션(30) 또는 다른 개체가 데이터(20)의 대체 데이터에도 액세스할 수 있다는 것이다.

<55> 어쨌든, 단계(501)에서와 같이 고스팅된 파일(14)의 저장된 데이터(20)에 액세스하려는 요청에 응답하여, 고스팅된 파일(14)은 실제로 다음과 같은 방식으로 재구성된다. 사전 준비로서, 파일(14)을 볼륨(12) 상에서 찾아내고(단계 503), 그 후에 파일(14)의 메타데이터(22) 내의 고스팅 정보(24)가 식별된다(단계 505). 잘 알 수 있는 바와 같이, 싱크(18)에 저장되어 있는 파일(14)의 데이터(20)는 이러한 식별된 고스팅 정보(24)에 기초하여 찾아내고(단계 507), 이러한 찾아낸 데이터(20)는 이어서 적절한 전송 메카니즘 및 통로에 의해 싱크(18)로부터 소스(10)로 이동될 수 있다(단계 511).

<56> 물론, 이러한 데이터(20)를 소스(10)에 있는 파일(14)에 저장하기 위해, 이러한 데이터(20)가 소스(10)의 볼륨(12) 상에서 차지하는 공간이 할당되어야만 하고(단계 509), 그 결과 더 이상 고스팅되어 있지 않은 파일(14)이 더 이상 산재되어 있는 것으로 표시되지 않는다. 이제 알 수 있는 바와 같이, 데이터(20)가 실제로 파일(14)을 재구성하기 위해 파일(14)로 이동된 후에, 더 이상 고스팅되어 있지 않은 파일(14)에 대한 메타데이터(22)는,

예를 들어, 이러한 메타데이터(22) 내의 고스팅됨 속성을 리셋하는 등에 의해, 이러한 파일(14)이 고스팅되어 있지 않음을 보여주기 위해 변경된다(단계 513). 또한, 더 이상 고스팅되어 있지 않은 파일(14)에 대한 메타데이터(22)는 고스팅 정보(24)를 제거하기 위해 변경된다(단계 515). 그에 따라, 파일(14)은 이제 고스팅되지 않은 형태로 되어 있다.

<57> 이제 잘 알 것인 바와 같이, 고스팅된 파일(14)의 생성을 가져오는 2가지 주요 시나리오가 있다. 첫번째 시나리오에서, 파일(14)은 볼륨(12) 상에 고스팅된 파일(14)로서 직접 생성된다. 이러한 고스트 생성은 지점 서버(10) 및 허브 서버(18)에서 상기한 것 등의 복제 아키텍처에서 일반적일 수 있다. 이러한 복제 아키텍처에서, 몇개의 지점 서버(10) 각각을 중앙의 허브 서버(18)로부터의 모든 파일의 비고스팅된 복사본으로 채우는 일은 엄청난 양의 대역폭을 필요로 한다. 그에 따라, 그 대신에, 복제 엔진이 지점 서버(10) 상에 단지 파일(14)의 고스팅된 복사본만을 생성하기로 결정할 수 있다. 임의의 특정의 지점 서버(10)에 있는 임의의 특정의 고스팅된 파일(14)과 연관된 데이터(20)는 이어서 이러한 특정의 지점 서버(10)에서 이러한 특정의 고스팅된 파일(14)을 비고스팅된 파일(14)로 재구성하기 위해 요구 시에 또 필요할 때에 허브 서버(18)로부터 검색될 수 있다.

<58> 두번째 시나리오에서, 파일(14)은 볼륨(12) 상에 비고스팅된 파일(14)로서 생성되며, 얼마간 나중에 이러한 볼륨(12) 상에서 고스팅된 파일(14)로 변환된다. 이러한 비고스트 생성은 퍼스널 컴퓨터, 기타 등등 및 대안의 장소(18)에서 상기한 것 등의 공간-절감 아키텍처에서 일반적일 수 있다. 이러한 공간-절감 아키텍처에서, 퍼스널 컴퓨터, 기타 등등의 볼륨(12) 상의 공간은, 예를 들어, 거의 사용되지 않는 파일(14)과 연관된 데이터(20)를 다른 매체일 수 있는 대안의 장소(18)로 또는 다른 시스템으로 이동시킴으로써 회수(reclaim)된다. 첫번째 시나리오에서와 같이, 퍼스널 컴퓨터, 기타 등등에 있는 임의의 특정의 고스팅된 파일(14)과 연관된 데이터(20)는 이러한 특정의 고스팅된 파일(14)을 비고스팅된 파일(14)로 재구성하기 위해 요구 시에 또 필요할 때에 대안의 장소(18)로부터 검색될 수 있다.

<59> 본 발명의 일 실시예에서, 사용자 또는 애플리케이션(30)을 대신하여 고스팅된 파일(14)과 관련하여 소스(10)에서 수행되는 도 5의 동작들은 실제로는, 도 2에서 알 수 있는 바와 같이, 고스팅 필터(26)의 도움을 받아 수행된다. 이러한 동작들은 사용자 또는 애플리케이션(30)이 모르게(transparently) 또는 알게(with notice) 수행될 수 있지만, 모르게 하는 것(transparency)이 이러한 사용자 또는 애플리케이션(30)에 더 나을 수 있다. 상세하게는, 사용자 또는 애플리케이션(30)이 고스팅된 파일(14)의 이동된 데이터(20)에 액세스하려고 시도할 때, 파일 시스템(16)은 이러한 액세스 요청을 수신할 시에 고스팅된 파일(14)이 이동된 데이터(20)를 포함하지 않음을 발견하고 따라서 고스팅 필터(26)가 가로채기할 에러를 반환한다. 이러한 가로채기(interception) 시에, 고스팅 필터(26)는 파일 시스템(16)을 이용하여 고스팅된 파일(14)에 대한 메타데이터(22)로부터 고스팅 정보(24)를 획득하고, 획득된 고스팅 정보(24)에 기초하여, 이러한 고스팅 필터(26)는, 이하에서 더 상세히 기술하는 바와 같이, 액세스 요청이 궁극적으로 존중될 수 있도록 이러한 고스팅된 파일(14)의 재구성을 트리거한다.

<60> 잘 알 수 있는 바와 같이, 고스팅 필터(26)는 기능이 그다지 없고 싱크(18) 등의 네트워크 자원에 액세스하지 않는 하위-레벨 구성자(lower-level construct)일 수 있다. 이러한 상황에서, 도 2에 나타낸 바와 같이, 고스팅 필터(26)는 고스팅 관리자(28) 등의 상위-레벨 구성자와 인터페이스할 수 있으며, 여기서 이러한 고스팅 관리자(28)는 부가적인 고스팅 기능 및 싱크(18) 등의 네트워크 자원에의 직접적인 액세스를 포함한다. 게다가, 이러한 상황에서, 고스팅 필터(26)가 이러한 고스팅된 파일(14)의 이러한 재구성을 트리거할 때, 도 5와 관련하여 도시된 바와 같이, 고스팅 관리자(28)가 소스(10)에 대한 이러한 재구성 기능의 대부분을 수행한다는 것을 잘 알 것이며, 이에 대해서는 이하에서 더 상세히 기술한다. 이러한 고스팅 관리자(28)는 또한, 도 4와 관련하여 나타낸 바와 같이, 소스(10)에 대한 고스팅 기능의 대부분을 수행할 수 있다.

<61> 고스팅된 파일(14)에 데이터(20)를 요청하는 애플리케이션(30)

<62> 고스팅된 파일(14)을 재구성하는 것에 대해서는 이미 도 4와 관련하여 이상에서 상세히 기술하였지만, 이러한 고스팅된 파일(14)의 데이터(20)의 어떤 부분을 읽고자 하는 애플리케이션(30) 등의 관점에서 이러한 프로세스를 재탐방(revisit)하는 것이 도움이 된다. 잘 알 수 있는 바와 같이, 이러한 파일(14)로부터 이러한 데이터(20)를 읽고자 하는 이러한 애플리케이션(30)은 일반적으로 파일(14)과 관련하여 파일 시스템(16)에 열기 명령을 발행하고 이어서 이러한 열린 파일(14)과 관련하여 파일 시스템(16)에 읽기 명령을 발행함으로써 이러한 기능을 달성한다. 본 발명의 일 실시예에서, 이러한 애플리케이션(30)은 이러한 열기 및 읽기 명령을 계속하여 발행하지만, 이러한 명령은 고스팅되고 있는 문제의 파일(14)에 기초하여 얼마간 다르게 해석된다. 이러한 차이점은 애플리케이션(30)에 투명하고 거의 문제가 되지 않지만, 애플리케이션(30)은 고스팅된 파일(14)이 재구성되는 과정에서 얼마간의 지연시간을 겪을 수 있으며, 여기서 이러한 지연시간은 대부분 데이터(20)를 싱크

(18)로부터 소스(10)로 이동시키는 것으로 인할 것일 수 있다.

<63> 어쨌든, 이제 도 6을 참조하면, 애플리케이션(30)은 볼륨(12) 상의 특정의 고스팅된 파일(14)에 대해 열기 명령을 발행하는 것으로 프로세스를 시작하고 이러한 열기 명령은 궁극적으로 파일 시스템(16)에 의해 수신된다(단계 601). 주목할 만한 것은, 애플리케이션(30)이, 이러한 열기 명령을 발행할 시에, 특정의 파일(14)이 실제로 고스팅되어 있는지를 알 것으로 예상되지 않지만, 실제로는 애플리케이션(30)이 고스팅됨 속성이 파일(14)에 대해 설정되어 있는지를 보고하도록 파일 시스템(16)에 명령함으로서 이러한 판정을 할 수 있다는 것이다.

<64> 상기한 바와 같이, 파일 시스템(16)은, 열기 명령을 수신할 시에, 고스팅된 파일(14)이 이동된 데이터(20)를 포함하지 않으며 따라서 에러를 반환(단계 603)한다는 것에 유의하고, 고스팅 필터(26)는 이러한 반환된 에러를 가로채기하여 그에 기초하여 문제의 파일(14)이 실제로는 고스팅된 형식으로 되어 있다는 것을 인식한다(단계 605). 그에 따라, 고스팅 필터(26) 자체가 파일 시스템(16)에 이러한 고스팅된 파일(14)로부터 고스팅 정보(24)를 검색하도록 명령하고 실제로 이러한 고스팅 정보(24)를 수신한다(단계 607). 이러한 고스팅 정보(24)에 기초하여, 고스팅 필터(26)는 버퍼 및 고스팅된 파일(14)에 대응하는 핸들을 생성하고 애플리케이션(30)으로부터의 열기 명령에 대한 (정상적인) 응답으로서 그 핸들을 애플리케이션(30)에 전달한다(단계 609).

<65> 유의할 점은 애플리케이션(30)에 의해 수신되는 고스팅된 파일(14)에 대한 핸들이 열린 고스팅된 파일(14)의 인스턴스를 나타내고, 예를 들어, 읽기 명령 또는 닫기 명령 등의 이러한 파일(14)과 관련하여 추가적인 명령을 발행할 때 이러한 파일(14)에 대한 참조로서 애플리케이션(30)에 의해 이용된다는 것이다. 그렇지만, 중요한 것은 고스팅 필터(26)에 의해 생성되고 애플리케이션(30)으로부터의 명령의 핸들 파라미터로서 이용되는 고스팅된 파일(14)에 대한 이러한 핸들은 이러한 명령이 파일 시스템(16)이 아니라 고스팅 필터(26)로 직접 전달되게 한다.

<66> 역시 유의할 점은 고스팅된 파일(14)에 대한 버퍼가, 예를 들어, 고스팅된 파일(14)의 고스팅 정보(24) 등의 고스팅된 파일(14)에 관련된 정보를 저장하기 위해 고스팅 필터(26)에 의해 이용된다는 것이다. 이에 따라, 고스팅된 파일(14)이 고스팅 필터(26)에 의해 조작되는 과정에서, 이러한 고스팅 정보(24)에 대한 변경이 버퍼에 기록될 수 있고 파일(14)에 다시 기입될 필요가 없다. 물론, 고스팅 필터(26)가 고스팅된 파일(14)을 조작하는 일을 끝마친 후에, 버퍼에 있는 이러한 기록된 변경은 필요한 경우 파일(14)에 대한 고스팅 정보(24)에 다시 기입될 수 있다.

<67> 어쨌든, 고스팅 필터(26)로부터의 고스팅된 파일(14)에 대한 핸들로, 애플리케이션(30)은 볼륨(12) 상의 특정의 고스팅된 파일(14)과 관련하여 읽기 명령을 계속하여 발행할 수 있으며, 여기서 이러한 읽기 명령은 핸들, 파일에 대한 데이터(20)에의 오프셋, 이러한 오프셋부터 시작하여 이러한 데이터(20)에 대한 읽기 길이, 기타 등등을 비롯한 파라미터를 갖는다. 그에 따라, 이러한 핸들을 파라미터로서 갖는 것에 기초한 이러한 읽기 명령은 궁극적으로 파일 시스템(16)이 아니라 고스팅 필터(26)에 의해 수신된다(단계 611). 그렇지만, 명백한 바와 같이, 데이터(20)는 고스팅된 파일(14)에 실제로 존재하지 않으며, 그에 따라 고스팅 필터는 추가적인 처리를 기다리기 위해 읽기 명령을 큐, 기타 등등에 저장한다(단계 613).

<68> 잘 알 것인 바와 같이, 이러한 추가적인 처리는 주로 싱크(18)로부터 고스팅된 파일(14)에 대한 데이터(20)를 획득하는 것을 포함한다. 상세하게는, 고스팅 필터(26)는 싱크(18)에 있는 고스팅된 파일(14)의 데이터(20)에 기초하여 고스팅된 파일(14)의 재구성을 개시하고(단계 615), 여기서 이러한 재구성은 다음과 같은 방식으로 행해질 수 있다.

<69> 고스팅 필터(26)가 고스팅 관리자(28)와 인터페이스하여 상위-레벨 고스팅 기능 및 싱크(18) 등의 네트워크 자원에의 직접적인 액세스를 수행할 수 있다는 것을 상기하면서, 고스팅 필터(26)는 싱크(18)로부터 고스팅된 파일(14)에 대한 데이터(20)의 어떤 부분을 획득하라는 요청을 고스팅 관리자(28)에 발행하고(단계 615a), 여기서 이러한 요청은 고스팅된 파일(14)의 고스팅 정보(24)로부터의 ID, 데이터(20)에의 오프셋, 이러한 오프셋부터 시작하여 획득될 데이터(20)의 길이를 포함한다. 그 후에, 고스팅 관리자(28)는 실제로 싱크(18)로부터 이러한 데이터(20)의 요청된 부분을 획득한다(단계 615b). 아마도, 고스팅 관리자(28)는 그렇게 획득하기 위한 모든 필요한 기능을 포함하고, 당업자라면 알고 있고 그에게 명백하며 여기에 상세히 기술할 필요가 없는 방식으로 그렇게 한다. 그에 따라, 본 발명의 정신 및 범위를 벗어나지 않고 그렇게 획득하는 임의의 특정의 방법이 이용될 수 있다.

<70> 유의할 점은 단계(615b)에서와 같이 고스팅 관리자(28)가 싱크(18)로부터 데이터(20)의 요청된 부분을 획득하는 과정에서, 얼마간의 지연이 있을 수 있다는 것이다. 예를 들어, 이러한 데이터(20)가 네트워크를 통해 획득될

수 있으며, 이 경우에, 아마도 몇 밀리초 정도, 몇 초 정도, 또는 그 이상의 네트워크 지연이 있을 수 있다. 이에 따라, 고스팅 관리자(28), 고스팅 필터(26), 및/또는 다른 개체는 싱크(18)에 요청되었지만 아직 그로부터 획득하지 못한 데이터(20)의 각각의 부분을 나타내는 대기중 데이터 리스트(pending data list)를, 아마도 대응하는 요청 시간과 함께, 유지하고자 할 수 있다. 잘 알 수 있는 바와 같이, 데이터의 각각의 요청된 부분의 ID는 그와 같이 요청될 시에 대기중 데이터 리스트에 추가되고, 수신되어 고스팅된 파일(14)에 저장될 시에 이러한 대기중 데이터 리스트로부터 제거된다. 이러한 대기중 데이터 리스트에서, 데이터(20)에 대한 모든 대기중 요청이 오랫동안 모니터링되고 어떤 기간 내에 만족되지 않은 요청들이 타임아웃될 수 있다.

<71> 고스팅 관리자(28)가 실제로 싱크(18)로부터 이러한 데이터(20)의 이러한 요청된 부분을 획득하면, 이러한 고스팅 관리자(28)는 이러한 요청된 부분을 고스팅 필터(26)에 전달하고(단계 615c), 이러한 고스팅 필터(26)는 이어서 파일 시스템(16)에 대한 적절한 명령에 의해 이러한 요청된 부분을 문제의 파일(14)의 적절한 장소에 기입한다(단계 615d). 이러한 요청된 부분을 문제의 파일(14)의 적절한 장소에 기입하는 것은 당업자에게 공지되어 있거나 명백하며, 따라서 본 명세서에 상세히 기재할 필요가 없다. 그에 따라, 본 발명의 정신 및 범위를 벗어나지 않고 그와 같이 기입하는 임의의 특정의 방법이 이용될 수 있다.

<72> 물론, 파일(14)에 대한 데이터(20) 전부가 그에 기입될 때까지 단계(615a-615d)를 여러번 반복할 필요가 있을 수 있으며, 따라서 이러한 단계들은 필요에 따라 반복된다. 파일(14)에 대한 데이터(20) 전부가 실제로 그에 기입되면, 고스팅 필터(26)는, 고스팅 정보(24)를 제거하는 것 및 고스팅됨 속성을 리셋하는 것을 비롯하여, 이러한 현재-재구성된 파일(14)의 메타데이터(22)를 교정하기 위해 필요에 따라 파일 시스템(16)에 명령을 발행한다(단계 617). 그에 부가하여, 고스팅 필터(26)는 큐로부터 파일(14)에 대한 모든 읽기 명령을 검색하고 추가적인 처리를 위해 이러한 읽기 명령을 파일 시스템(16)으로 전달하며(단계 619), 그에 의해 애플리케이션(30)으로부터의 읽기 명령은 실제로 문제의 데이터(20)로 응답된다.

<73> 유의할 점은 열린 파일(14)에 대한 핸들이 고스팅 필터(26)와 연관된 채로 있으며 이러한 현재-재구성된 파일(14)과 관련하여 애플리케이션(30)으로부터의 명령을 이러한 고스팅 필터(26)로 계속하여 보낸다는 것이다. 그에 따라, 고스팅 필터(26)가 이러한 명령을 파일 시스템(16)으로 전달할 수 있거나, 고스팅 필터(26)가 이 핸들을 파일 시스템(16)과 재연관시킬 수 있거나, 또는 고스팅 필터가 이 핸들을 파일 시스템(16)과 재연관시키도록 다른 구성자에 명령할 수 있거나, 기타 등등이 있을 수 있다.

<74> 고스팅된 파일(14)의 부분적 재구성

<75> 잘 알 수 있는 바와 같이, 파일(14)로부터의 고스팅된 데이터(20) 전부보다 적은 것이 요청측 애플리케이션(30)에서 요구되는 상황에서 고스팅된 파일(14)을 완전히 재구성할 필요가 없다. 이에 따라, 애플리케이션(30)이 2 기가바이트의 데이터(20)의 특정의 오프셋에 있는 단지 1, 2, 12 또는 100 킬로바이트만이 필요한 것으로 판정할 수 있는 경우, 싱크(18)로부터 2 기가바이트의 데이터(20)를 획득할 필요가 없으며, 그 대신에 특정의 오프셋에 있는 그 킬로바이트만큼의 필요한 데이터(20)만을 획득하면 된다. 게다가, 이러한 상황에서, 고스팅된 파일(14)을 단지 부분적으로만 재구성함으로써, 상당량의 불필요한 데이터(20)를 싱크(18)로부터 소스(10)로 전송할 필요가 없게 되고, 그렇게 하는 데 필요한 대역폭이 그에 대응하여 감소된다.

<76> 고스팅된 파일(14)을 단지 부분적으로만 재구성함으로써, 애플리케이션(30)은 파일(14)의 완전한 재구성을 트리거하지 않고 필요한 경우 단지 몇 바이트의 파일만을 읽을 수 있다. 이에 따라, 예를 들어, 비디오 파일(14)의 첫번째 프레임만이 필요한 경우, 10 또는 심지어 100 기가바이트 정도일 수 있는 고스팅된 파일(14)에 대한 전체 데이터(20)량이 아니라 이러한 첫번째 프레임이 싱크(18)에 있는 파일(14)에 대한 데이터(20)로부터 획득된다. 부분적으로 재구성하는 것은 애플리케이션(30)으로부터의 특정의 읽기 요청을 만족시키기 위해 고스팅된 파일(14)에 대한 싱크(18)에 있는 데이터(20)를 필요한 만큼만 페치하며, 더 이상 페치하지 않는다.

<77> 유의할 점은, 고스팅된 파일(14)을 부분적으로 재구성할 때, 싱크(18)로부터 획득된 데이터(20)의 일부가 실제로 이러한 싱크(18)로부터 제거될 수도, 제거되지 않을 수도 있다는 것이다. 한편으로, 획득된 이러한 부분은 소스(10)에 저장되고, 따라서 더 이상 싱크(18)에 유지될 필요가 없다. 그렇지만, 반면에, 실제로 싱크(18)로부터 획득된 부분을 삭제하는 것이 실제로는 더 많은 노력을 필요로 할 수 있다. 게다가, 적어도 어떤 환경에서, 예를 들어, 싱크가 허브 서버(18)이고 소스가 지점 서버(10)인 경우 등에, 이러한 획득된 부분은 다른 소스(10)에 의한 액세스를 위해 싱크(18)에 유지되어야만 한다.

<78> 잘 알 수 있는 바와 같이, 고스팅된 파일(14)이 실제로 부분적으로 구성되는 경우, 고스팅 필터(26)가 이러한 부분이 고스팅된 파일(14)에 존재하는지를 판정할 수 있도록 고스팅된 파일(14)의 어느 부분이 실제로 재구성되

는지에 유의하기 위해 기록이 유지될 수 있다. 그에 따라, 본 발명의 일 실시예에서, 이러한 기록이 이러한 고스팅된 파일(14)의 메타데이터(22) 내의 고스팅 정보(24)에 유지된다. 상세하게는, 재구성되고 그에 따라 고스팅된 파일(14)에 존재하는 고스팅된 파일(14)의 데이터(20)의 각각의 인접한 섹션에 대해, 고스팅 정보(24)는 그에 대해 섹션의 시작 및 인접한 이러한 섹션의 양을 기술하는 길이를 기술하는 오프셋을 포함하는 섹션 참조를 포함한다.

<79> 일반적으로, 부분적으로 재구성된 데이터(20)의 상기한 기록을 비롯한 고스팅된 파일(14)에 대한 고스팅 정보(24)는 고스팅 필터(26)에 의해 유지된다. 상기한 바와 같이, 이러한 고스팅 필터(26)가 파일 시스템(16)을 통해 이러한 고스팅 정보(24)를 고스팅된 파일(14)의 메타데이터(22)에 직접 유지하고 갱신할 수 있지만, 이러한 유지 및 갱신은 파일 시스템(16)에 의해 수행되는 다른 동작들과 인터페이스할 수 있다. 이에 따라, 고스팅 필터(26)는 고스팅된 파일(14)을 조작하는 도중에 그 대신에 처음에 이러한 메타데이터(22)로부터 이러한 고스팅 정보(24)를 획득하고 이러한 고스팅 정보(24)를 도 6의 단계(609)에서와 같이 고스팅된 파일(14)과 관련하여 생성된 버퍼에 저장하고, 이어서 고스팅된 파일(14)을 조작하는 과정 동안에 이러한 고스팅 정보(24)를 유지 및 갱신하고, 종료할 시에 버퍼로부터의 이러한 고스팅 정보(24)를 고스팅된 파일(14)에 대한 메타데이터(22)에 기입한다.

<80> 유의할 점은 때때로 고스팅된 파일(14)의 부분적 재구성이, 예를 들어, 소스(10)에서의 전원 또는 네트워크 연결의 상실에 의해 중단될 수 있다는 것이다. 이와 마찬가지로, 고스팅된 파일(14)에 대한 버퍼가 상실될 수 있으며, 전원 단절의 경우에 또 버퍼가 휘발성 RAM, 기타 등등에 유지되는 경우에 특히 그렇다. 이러한 상황에서, 유지 및 갱신되는 고스팅 정보(24)가 상실되고 버퍼로부터 파일(14)에 대한 메타데이터(22)에 기입되지 않으며, 사실상 이러한 고스팅 정보(24)가 상실될 뿐만 아니라, 비록 그와 연관된 모든 부분적으로 재구성된 데이터(20)가 소스(10)에 부분적으로 존재할지라도 특히 이러한 고스팅 정보(24) 없이 이러한 데이터(20)를 찾아낼 수 없는 한, 그 데이터(20)도 상실된다. 그에 따라, 본 발명의 일 실시예에서, 유지되고 갱신되는 이러한 고스팅 정보(24)는 버퍼로부터 파일(14)에 대한 메타데이터(22)에 주기적으로, 예를 들어, 1분 정도마다 한번씩 기입된다. 이에 따라, 기껏해야 겨우 1분 정도의 이러한 고스팅 정보(24) 및 그와 연관된 부분적으로 재구성된 데이터(20)가 버퍼의 상실로 인해 상실될 수 있다.

<81> 본 발명의 일 실시예에서, 고스팅된 파일(14)이 이러한 고스팅된 파일(14)에 데이터(20)를 요청하는 애플리케이션(30)으로부터의 적절한 명령에 기초하여 단지 부분적으로 재구성된다. 그에 따라, 이러한 애플리케이션(30)은 특정의 파일(14)에 대해 고스팅됨 속성이 설정되어 있는지를 판정하여 이러한 파일(14)이 실제로 고스팅되어 있는지를 판정하기 위해 먼저 파일 시스템(16)으로 검사를 해야만 하며, 그러한 경우, 애플리케이션(30)은 적절한 명령으로 이러한 고스팅된 파일(14)에 대한 데이터(20)의 일부분 또는 일부분들의 부분적인 재구성을 요청할 수 있다.

<82> 부분적인 재구성을 고려하여, 본 발명의 일 실시예에서, 고스팅 필터(26)는 도 6의 단계(611)에서 주어진 것 등의 고스팅된 파일(14)과 관련한 읽기 명령에 응답하여 먼저 대응하는 버퍼에 저장된 이러한 고스팅된 파일(14)에 대한 고스팅 정보(24)에서의 임의의 섹션 참조를 검토하는 것으로 그에 응답하고, 이어서 섹션 참조로부터 요청된 데이터(20) 또는 그의 일부분이 소스(10)에 있는 고스팅된 파일(14)에 이미 존재하는지를 판정한다. 요청된 데이터(20) 전부가 실제로 존재하는 경우, 이러한 데이터(20)는 싱크(18)로부터 이러한 데이터(20)를 획득할 필요없이 소스(10)에 있는 고스팅된 파일(14)로부터 읽어온다. 요청된 데이터(20)의 일부분만이 존재하는 경우, 데이터(20)의 이러한 존재하는 부분은 싱크(18)로부터 이러한 데이터(20)를 획득할 필요없이 소스(10)에 있는 고스팅된 파일(14)로부터 읽어오고, 데이터(20)의 나머지는 싱크(18)로부터 획득되고 이어서 상기한 바와 같이 읽어온다. 요청된 데이터(20)의 어느 것도 존재하지 않는 경우, 이러한 데이터(20)의 전부는 싱크(18)로부터 획득되고 이어서 상기한 바와 같이 읽어온다.

<83> 요약하기 위해, 이제 도 7을 참조하면, 애플리케이션(30)으로부터의 명령에 응답하여 고스팅된 파일(14)(이로부터의 부분적 재구성이 필요할 수 있음)을 열기 위해(단계 701), 고스팅 필터(26)는 다시 버퍼 및 고스팅된 파일(14)에 대응하는 핸들을 생성하고, 단계(609)에서와 같이 애플리케이션(30)으로부터의 열기 명령에 대한 (정상적인) 응답으로서 이 핸들을 애플리케이션(30)으로 전달한다(단계 703). 그에 부가하여, 고스팅 필터(26)는 처음에 고스팅된 파일(14)에 대한 메타데이터(22)로부터 고스팅 정보(24)를 획득하고 이러한 고스팅 정보(24)를 생성된 버퍼에 저장한다(단계 705).

<84> 이전과 같이, 고스팅 필터(26)로부터의 고스팅된 파일(14)에 대한 핸들로, 애플리케이션(30)은 볼륨(12) 상의 특정의 고스팅된 파일(14)의 데이터(20)의 일부분에 대한 읽기 명령을 발행하며, 여기서 이러한 읽기 명령은 핸

들을 포함하고, 단계(611)에서와 같이 일부분의 오프셋 및 길이를 정의한다(단계 707). 여기에서, 읽기 명령에 응답하여, 고스팅 필터(26)는 고스팅된 파일(14)에 대한 버퍼 내의 고스팅 정보(24)에 기초하여 정의된 이러한 일부분이 소스(10)에 존재하는 고스팅된 파일(14)에 적어도 부분적으로 이미 존재하는지를 판정한다(단계 709). 이러한 판정을 하는 것은 당업자에게 공지되어 있고 명백하며, 따라서 본 명세서에 상세히 기술할 필요가 없다. 그에 따라, 이러한 판정을 하는 방법이 본 발명의 정신 및 범위를 벗어나지 않고 이용될 수 있다.

<85> 다시 말하면, 데이터(20)의 일부분 전부가 실제로 존재하는 경우, 이러한 데이터의 이러한 일부분은 싱크(18)로부터 획득될 필요가 없다(단계 711a). 데이터(20)의 일부분의 단지 일부만이 존재하는 경우, 데이터(20)의 일부분의 나머지가 싱크(18)로부터 획득된다(단계 711b). 데이터(20)의 일부분의 어느 것도 존재하지 않는 경우, 데이터(20)의 이러한 일부분 전부가 싱크(18)로부터 획득된다(단계 711c). 중요한 것은, 단계(711b, 711c)에서와 같이 싱크(18)로부터 데이터(20)의 임의의 일부를 획득할 시에, 고스팅 필터(26)는 데이터(20)의 일부가 이제 고스팅된 파일(14)과 함께 존재하여 고스팅된 파일(14)로 재구성되어 있다는 것을 적절히 반영하기 위해 버퍼에 저장되어 있는 고스팅된 파일(14)에 대한 고스팅 정보(24)를 갱신한다(단계 713). 어쨌든, 데이터(20)의 요청된 일부분이 이제 소스(10)에 존재함으로써, 이러한 요청된 일부분은 이제 실제로 단계(619)에서와 같이 애플리케이션(30)에 의해 읽혀질 수 있다(단계 715).

<86> 상기한 바와 같이, 고스팅 필터(26)는 고스팅 정보(24)를 그의 가장 최근의 형태로 버퍼로부터 파일(14)에 대한 메타데이터(22)에 주기적으로 기입하며, 그에 따라 버퍼가 상실되는 경우에도 이러한 고스팅 정보(24) 및 그와 연관되어 있는 부분적으로 재구성된 데이터(20)가 완전히 상실되지 않는다. 그에 부가하여, 예를 들어, 애플리케이션(30)의 명령 등으로 고스팅된 파일(14)이 닫히면(단계 719), 고스팅 필터(26)는 고스팅 정보(24)를 그의 가장 최근의 형태로 이러한 버퍼로부터 파일(14)에 대한 메타데이터(22)에 기입함으로써 버퍼를 폐쇄하며(close out)(단계 721), 물론 고스팅된 파일(14)이 완전히 재구성되지 않은 것으로 가정한다. 이에 따라, 가장 최근의 형태로 있는 이러한 고스팅 정보(24)가 단계(705)에서와 같이 얼마간 나중에 다시 검색될 수 있다.

<87> 부분적으로 재구성된 고스팅된 파일(14)의 고속 읽기

<88> 애플리케이션(30)은 고스팅된 파일(14)에 대한 읽기 명령을 발행할 시에 고스팅된 파일(14)의 상태를 고려하지 않으며, 상세하게는 고스팅된 파일(14)이 이미 부분적으로 재구성되었는지 및/또는 이미 부분적으로 재구성되고 있는 중인지를 고려하지 않는다. 즉, 이제 도 8을 참조하면, 고스팅된 파일(14)에 대한 특정의 읽기 명령이 데이터(20)의 특정의 일부분을 지정하고 또 이러한 특정의 일부분이 고스팅된 파일(14)에 이미 존재하는 데이터(20)를 포함하는 제1 세그먼트, 싱크(18)로부터 고스팅된 파일(14)로 복사되기 위해 대기 중인 데이터(20)를 포함하는 제2 세그먼트, 및 고스팅된 파일(14)에 존재하지 않지만 그 대신에 싱크(18)에만 저장되어 있는 데이터(20)를 포함하는 제3 세그먼트에 대응하는 경우가 있을 수도 있다.

<89> 잘 알 수 있는 바와 같이, 데이터(20)의 특정의 일부분 전부가 싱크(18)로부터 고스팅된 파일(14)로 복사되도록 이러한 읽기 명령이 처리되어야 하는 경우, 이러한 처리는 이러한 특정의 일부분에 대응하는 적어도 제1 및 제2 세그먼트에 대해 중복되고 낭비적이다. 상세하게는, 잘 알 수 있는 바와 같이, 이러한 데이터(20)가 고스팅된 파일(14)에 이미 존재하는 한, 제1 세그먼트에 대응하는 데이터(20)를 복사하는 것은 불필요하고, 이러한 데이터(20)가 싱크(18)로부터 고스팅된 파일(14)로 복사되기 위해 이미 대기중에 있는 한, 제2 세그먼트에 대응하는 데이터(20)를 복사하는 것은 불필요하다. 사실상, 이러한 데이터(20)가 고스팅된 파일(14)에 존재하지 않고 이러한 고스팅된 파일(14)로 복사되도록 아직 요청되지도 않은 한, 제3 세그먼트에 대응하는 데이터(20)만이 싱크(18)로부터 고스팅된 파일(14)로 복사되면 된다.

<90> 본 발명의 일 실시예에서, 고스팅 필터(26)는 먼저 데이터(20)의 이러한 특정의 일부분과 관련하여 이미 존재하는 고스팅된 파일(14) 내의 대응하는 세그먼트들(즉, 제1 세그먼트들), 이미 대기중인 고스팅된 파일(14) 내의 대응하는 세그먼트들(즉, 제2 세그먼트들) 및 존재하거나 대기중이지 않은 고스팅된 파일(14) 내의 대응하는 세그먼트들(즉, 제3 세그먼트들)을 식별하고, 이어서 실제로 이러한 제3 세그먼트만을 획득함으로써 데이터(20)의 특정의 일부분에 대한 이러한 읽기 명령을 처리한다. 사실상, 고스팅 필터(26)는 읽기 명령으로부터 싱크(18)로부터 실제로 읽혀질 필요가 없는 세그먼트들 전부를 제거한다(strip out). 이러한 제거 동작을 수행함으로써, 이러한 '고속 읽기'가 실제로 필요한 그 데이터(20)만을 싱크로부터 획득하고 고스팅된 파일(14)에 이미 존재하거나 대기중인 데이터(20)를 획득하지 않는다는 점에서 읽기 명령이 더 빠르게 처리된다. 이에 따라, 고스팅 필터(26)에 의해 수행되는 고속 읽기의 결과, 애플리케이션(30)으로부터의 읽기 명령에 대한 더 빠른 응답이 얻어지고, 그에 수반하여 이러한 읽기 명령에 필요한 대역폭의 양을 감소시킨다.

<91> 본 발명의 일 실시예에서, 고스팅 필터(26)는 이러한 고스팅된 파일(14)에 대해 유지되는 고스팅 정보(24)를 참

조하여 이미 존재하는 고스팅된 파일(14) 내의 각각의 제1 세그먼트를 식별한다. 상기한 바와 같이, 이러한 고스팅 정보(24)는 이러한 고스팅된 파일(14)에 대한 메타데이터(22)에 위치하는 것으로 말해질 수 있거나 이러한 고스팅된 파일(14)에 대응하는 버퍼에 위치하는 것으로 말해질 수 있다. 이와 마찬가지로, 본 발명의 일 실시예에서, 고스팅 필터(26)는 도 6의 단계(615b)와 관련하여 상기한 대기중 데이터 리스트 내의 정보를 참조하여 이미 대기중인 고스팅된 파일(14) 내의 각각의 제2 세그먼트를 식별한다. 이러한 고스팅 정보(24) 및 대기중 데이터 리스트를 참조하여 이러한 제1 및 제2 세그먼트를 식별하는 것은 당업자에게는 공지되어 있거나 명백하며, 따라서 본 명세서에 상세히 기술될 필요가 없다. 그에 따라, 이러한 식별은 본 발명의 정신 및 범위를 벗어나지 않고 임의의 적절한 방식으로 수행될 수 있다.

<92> 이제 잘 알 수 있는 바와 같이, 제1 및 제2 세그먼트가 식별되었으면, 그 나머지가 제3 세그먼트(들)이고, 소거(elimination) 프로세스에 의해, 존재하거나 대기중이 아닌 고스팅된 파일(14)의 이러한 제3 세그먼트가 식별된다. 물론, 식별되면, 이러한 제3 세그먼트는 실제로 싱크(18)에 요청될 수 있고 수신 시에 고스팅된 파일(14)에 복사될 수 있다. 유의할 점은, 요청되면, 각각의 이러한 제3 세그먼트가 사실상 대기중인 제2 세그먼트가 된다는 것이다. 역시 유의할 점은, 수신되어 고스팅된 파일(14)에 복사되면, 각각의 제2 세그먼트는 사실상 존재하는 제1 세그먼트가 된다. 마지막으로 유의할 점은, 세그먼트가 제1 세그먼트로 되면, 이러한 제1 세그먼트는 애플리케이션(30)으로부터의 읽기 명령에 응답하여 실제로 애플리케이션(30)에 의해 읽혀질 수 있다는 것이다.

<93> 적어도 어떤 상황에서, 비록 애플리케이션(30)이 고스팅된 파일(14) 내의 특정의 데이터(20)에 대해 읽기 명령을 발행하지 않았더라도, 고스팅 필터(26)가 그럼에도 불구하고, 특히 고스팅 필터(26)가 다른 방식으로 사용되지 않는 경우에, 이러한 읽기 명령을 예견하여 이러한 데이터(20)를 싱크(18)로부터 고스팅된 파일(14)로 이동시킨다는 것을 잘 알 수 있다. 단지 한 예로서, 고스팅된 오디오 파일(14), 고스팅된 비디오 파일(14), 또는 고스팅된 멀티미디어 파일(14) 등의 컨텐츠를 스트리밍할 때, 특정의 기간 T0에서의 데이터(20)에 대한 읽기 명령에 뒤이어서 그 다음 기간 T1에서의 데이터(20)에 대한 읽기 명령이 있을 것으로 예견하는 것은 당연하다. 이러한 상황에서, 고스팅 필터(26)는, 다른 방식으로 사용되고 있지 않은 경우, 애플리케이션(30)으로부터의 특정의 읽기 명령 없이도 기간 T1 동안에 싱크(18)로부터 이러한 데이터(20)를 획득할 기회를 이용할 수 있다. 물론, 그렇게 함에 있어서, 고스팅 필터(26)는 상기한 방식으로 고속 읽기로서 이러한 동작을 수행할 수 있다.

<94> 요약하기 위해, 이제 도 9를 참조하면, 고스팅 필터(26)는 다음과 같은 방식으로 애플리케이션(30)으로부터의 읽기 명령에 응답하여 고속 읽기를 수행한다. 사전 준비로서, 읽기 명령이 실제로 애플리케이션(30)으로부터 수신되고, 이 때 이러한 읽기 명령은 고스팅된 파일(14)로부터 읽혀질 데이터(20)의 일부분 또는 범위를 지정한다(단계 901). 일반적으로, 다시 말하면, 이러한 범위는 데이터(20)와 관련한 오프셋 및 길이로서 표현된다.

<95> 그 후에, 고스팅 필터(26)는 먼저 이러한 범위의 데이터(20) 내에서 이미 존재하는 고스팅된 파일(14) 내의 데이터(20)의 대응하는 제1 세그먼트를 식별한다(단계 903). 다시 말하면, 이러한 식별은 이러한 고스팅된 파일(14)에 대해 유지되는, 즉 이러한 고스팅된 파일(14)에 대한 메타데이터(22)에 또는 이러한 고스팅된 파일(14)에 대응하는 버퍼에 유지되는 고스팅 정보(24)를 참조하여 수행된다. 식별된 제1 세그먼트가 요청된 범위의 데이터(20) 전부를 포함하는 경우, 고스팅된 파일(14)은 읽기 명령을 만족시키는 데 필요한 정도까지 이미 재구성되어 있으며, 이러한 읽기 명령은 따라서 싱크(18)로부터의 임의의 추가적인 데이터(20) 복사 없이 또 임의의 대기중 데이터(20)를 기다리지 않고 완료될 수 있다(단계 905).

<96> 그렇지만, 식별된 제1 세그먼트가 요청된 범위의 데이터(20) 전부를 포함하지는 않는 경우, 고스팅된 파일(14)은, 적어도 대기중 데이터(20)에 기초하여, 실제로 읽기 명령을 만족시키기에 충분한 정도까지 재구성되어야만 하며, 그에 따라, 고스팅 필터(26)는 제1 세그먼트가 아닌 요청된 범위의 데이터(20)의 각각의 세그먼트의 범위를 포함하는 제1 세트를 계산한다(단계 907). 그 후에, 이전과 유사하게, 고스팅 필터(26)는 제1 세트 내에서 대기중에 있는 고스팅된 파일(14) 내의 데이터(20)의 대응하는 제2 세그먼트를 식별한다(단계 909). 다시 말하면, 이러한 식별은 대기중 데이터 리스트를 참조하여 수행된다. 식별된 제2 세그먼트가 제1 세트 전부를 포함하는 경우, 고스팅된 파일(14)은 읽기 명령을 만족시키는 데 필요한 정도까지 재구성되기 위해 대기중이며, 이러한 읽기 명령은 모든 대기중 데이터(20)가 실제로 고스팅된 파일(14)로 복사될 때 완료될 수 있다(단계 911).

<97> 이제 잘 알 수 있는 바와 같이, 식별된 제2 세그먼트가 제1 세트 전부를 포함하지 않는 경우, 고스팅된 파일(14)은 싱크(18)로부터 복사될 데이터(20)에 기초하여 읽기 명령을 만족시키는 데 필요한 정도까지 실제로 재구성되어야만 하며, 그에 따라 고스팅 필터(26)는 제1 세그먼트도 아니고 제2 세그먼트도 아닌 요청된 범위의 데이터(20)의 각각의 세그먼트의 범위를 포함하는 제2 세트, 즉 제3 세그먼트를 계산한다(단계 913). 그 후에,

고스팅 필터(26)는 제2 세트/제3 세그먼트가 싱크(18)로부터 고스팅된 파일(14)로 복사되도록 요청한다(단계 915).

<98> 이윽고, 다시 말하면, 요청된 각각의 제3 세그먼트는 대기중인 제2 세그먼트가 되고, 모든 복사가 완료(단계 917)될 때까지 실제로 복사된 각각의 제2 세그먼트는 존재하는 제1 세그먼트가 된다. 그 후에, 읽기 명령의 요청된 범위가 이러한 읽기 명령에 응답하여 애플리케이션(30)에 의해 실제로 읽혀질 수 있다(단계 919).

<99> 파일(14)의 재고스팅/고스팅

<100> 잘 알 수 있는 바와 같이, 재구성된 또는 부분적으로 재구성된 고스팅된 파일(14)이 어떤 시점에서, 예를 들어, 소스(10)에 부가적인 공간이 필요한 경우, 재고스팅될 수 있다. 그에 부가하여, 역시 잘 알 수 있는 바와 같이, 고스팅된 적이 없는 파일(14)이 이와 마찬가지로 어떤 시점에서 유사한 이유로 고스팅될 수 있다.

<101> 상세하게는, 고스팅된 파일(14)이 부분적으로 또는 완전히 재구성되면 또는 파일(14)이 소스(10)에 설치되면, 이러한 파일(14)이 재고스팅(re-ghost) 또는 고스팅(ghost)(이후부터 재고스팅이라고 함)되지 않는 한, 이러한 파일(14)은 소스(10)에 이러한 형태로 계속 존재한다. 이러한 재고스팅은, 예를 들어, 소스(10)에 공간이 필요한 것으로 판정하는 것, 파일(14)이 어떤 기간 동안 액세스되지 않은 것으로 판정하는 것, 기타 등등일 수 있는 어떤 이벤트에 의해 트리거된다. 게다가, 특히 지점 서버(10) 및 허브 서버(18)와 관련하여 파일(14)을 재고스팅함으로써, 소스(10) 또는 싱크(18)에 있는 파일(14)에 대한 어떤 변경도, 파일(14)을 최신의 것으로 유지하기 위해, 싱크(18) 또는 소스(10)로 각각 복제될 수 있다.

<102> 본 발명의 일 실시예에서, 소스(10)에서의 재고스팅은 마지막 액세스 시간, 볼륨 상에 남아있는 빈 공간, 데이터에의 액세스 빈도수, 다른 곳에서 일어나는 파일(14)에 대한 수정, 기타 등등의 인자들을 고려하는 재고스팅 알고리즘에 따라 수행된다. 재고스팅 알고리즘은, 이러한 인자들을 고려할 시에, 고스팅된 파일(14)이 재고스팅된 후 얼마지 않아 다시 재구성될 확률을 감소시키는 것을 목표로 갖는다. 따라서, 이러한 고스팅 및 재고스팅을 위해 필요한 대역폭이, 비록 감소되지는 않더라도, 최소화될 수 있다. 일반적으로, 반드시 그럴 필요는 없지만, 재고스팅 알고리즘은 소스(10)에 있는 애플리케이션(30)에 의해 또는 소스(10)에 있는 고스팅 관리자(28)에 의해 수행되지만, 본 발명의 정신 및 범위를 벗어나지 않고 다른 개체가 이러한 재고스팅 알고리즘을 수행할 수 있다.

<103> 소스(10)에 있는 파일(14)을 재고스팅하는 데 이용되는 재고스팅 알고리즘은 적어도 부분적으로 구성가능 정책에 기초하여 정의될 수 있다. 상세하게는, 이러한 재고스팅 알고리즘은 적어도 부분적으로 재구성가능 정책에 기초하여 트리거될 수 있으며, 이러한 재고스팅 알고리즘은 적어도 부분적으로 재구성가능 정책에 기초하여 특정의 파일(14)을 재고스팅하기로 결정할 수 있다. 각각의 경우에, 이러한 정책은 소스(10)의 사용자에 의해, 소스(10)의 관리자에 의해, 기타 등등에 의해 구성가능할 수 있다.

<104> 재고스팅 알고리즘과 관련하여 이용되는 가능한 트리거링 파라미터는 이하의 것들을 포함하지만 이에 한정되는 것은 아니다.

<105> - 미리 정해진 기간이 만료될 때 활성화되는 주기적 트리거(periodic trigger),

<106> - 소스(10)에서의 볼륨(12)에 있는 빈 공간이 어떤 양보다 작게 될 때 또는 소스(10)에서의 볼륨(12)에 있는 사용된 공간이 어떤 양을 초과할 때 활성화되는 공간 트리거(space trigger),

<107> - 미리 정해진 기간이 만료될 때, 그렇지만 소스(10)에서의 빈 공간이 어떤 양보다 작게 되거나 소스(10)에서의 사용된 공간이 어떤 양을 초과하는 경우에만 활성화되는 주기적-공간 트리거(periodic-space trigger),

<108> - 고스팅 필터(26)가 볼륨(12)이 남아 있는 빈 공간을 갖지 않음을 신호하기 위해 파일 시스템(16)에 의해 반환되는 볼륨-충만 에러(full-volume error)을 알 때 활성화되는 볼륨-충만 트리거(full-volume trigger),

<109> - 미리 정해진 수의 바이트가 소스(10)로 다운로드될 때 활성화되는 바이트-다운로드 완료 트리거(bytes-downloaded trigger), 및

<110> - 사용자, 관리자, 기타 등등에 의해 활성화될 수 있는 수동 트리거(manual trigger).

<111> 물론, 재고스팅 알고리즘을 트리거하는 것은 이러한 트리거들 중 하나 또는 이러한 트리거들의 조합에 기초하여 행해질 수 있다.

<112> 특정의 파일(14)을 재고스팅할지를 판정하기 위한 가능한 선택 인자들은 이하의 것들을 포함하지만, 이에 한정

되는 것은 아니다.

<113> - 파일(14)이 액세스된 마지막 시간, 이에 의해 더 오래된 액세스 시간을 갖는 파일(14)이 우선적으로 재고스팅될 수 있음,

<114> - 파일(14)과 연관된 다운로드 시간, 이에 의해 더 오래된 다운로드 시간을 갖는 파일(14)이 우선적으로 재고스팅될 수 있음,

<115> - 파일(14)에 대한 파일 크기, 이에 의해 더 큰 파일(14)이 우선적으로 재고스팅될 수 있음,

<116> - 파일(14)에 대한 파일 유형, 이에 의해 특정의 확장자를 갖는 파일(14)이 우선적으로 재고스팅될 수 있음,

<117> - 파일(14)에 대한 파일 속성, 이에 의해, 예를 들어, 시스템 파일인 파일(14)은 재고스팅되지 않고 숨겨진 파일(14)이 재고스팅됨,

<118> - 다수의 유사한 파일(14)이 존재하는지 여부, 이에 의해 유사한 것으로 생각되는 파일(14)이 우선적으로 재고스팅될 수 있음,

<119> - 파일(14)이 싱크(18)에서 수정되었는지 여부, 이에 의해 이러한 파일(14)이 오래된 것(out-of-date)으로 생각될 수 있는 그 안에 있는 데이터(20)를 제거하기 위해 재고스팅됨,

<120> - 파일(14)이 소스(10)에서 수정되었는지 생성되었는지, 이에 의해 이러한 파일(14)이 이러한 수정/생성을 보존하기 위해 재고스팅되도록 선택되지 않거나 이러한 수정/생성을 싱크(18)로 복사하기 위해 재고스팅되도록 선택될 수 있음,

<121> - 특정의 기간에 걸쳐 파일(14)에의 액세스 빈도수,

<122> - 특정의 기간에 걸쳐 파일(14)에의 액세스 횟수,

<123> - 예를 들어, 동일한 폴더에 있는, 동일한 컨텐츠 세트에 있는, 기타 등등에 있는 기타 관련된 파일(14)의 액세스 빈도수/액세스 횟수/마지막 액세스 시간, 및

<124> - 예를 들어, 리스트, 메서드, XML 파일, 기타 등등에 의한 우선적으로 재고스팅될 특정의 파일 또는 특정 유형의 파일의 외부 입력.

<125> 물론, 재고스팅 알고리즘은 단일의 선택 인자 또는 이러한 선택 인자들의 조합을 이용할 수 있다.

<126> 유의할 점은 재고스팅을 위해 선택된 파일(14)이 실제로 재고스팅될 수 있거나 그 대신에 우선적 재고스팅을 위한 후보로서 명될 수 있을 뿐이라는 것이다. 후자의 경우에, 상세하게는, 특정의 중단 트리거(stop trigger)에 도달될 때까지만 재고스팅이 수행될 수 있다. 잘 알 수 있는 바와 같이, 이러한 중단 트리거는 재고스팅 세션을 개시하는 트리거링 파라미터에 대응할 수 있거나, 재고스팅을 위한 후보 파일(14)을 선택하는 기준에 대응할 수 있거나, 본 발명의 정신 및 범위를 벗어나지 않는 다른 트리거일 수 있다.

<127> 역시 유의할 점은, 어떤 중단 트리거에 도달될 때까지만 재고스팅이 수행되는 경우에, 이러한 파일들(14)에 걸친 재고스팅을 평활화(smooth out)하기 위해 재고스팅될 후보인 파일(14)에 대한 선택 기준에 의존하는 것이 바람직할 수 있다는 것이다. 예를 들어, 후보 파일(14)이 소스(10)에서 적어도 2주의 마지막 액세스 날짜를 갖는 것에 기초하여 선택된 경우, 중단 트리거가 활성화될 때까지, 먼저 2개월보다 더 큰 마지막 액세스 날짜를 갖는 이러한 후보들 전부를 재고스팅하고, 이어서 필요한 경우 1개월보다 큰 마지막 액세스 날짜를 갖는 이러한 후보들 전부를 재고스팅하며, 이어서 필요한 경우 3주, 기타 등등보다 큰 마지막 액세스 날짜를 갖는 이러한 후보들 전부를 재고스팅하는 것이 바람직할 수 있다. 이와 마찬가지로, 후보 파일(14)이 10 메가바이트의 최소 파일 크기를 갖는 것에 기초하여 선택되는 경우, 중단 트리거가 활성화될 때까지, 먼저 1 기가바이트보다 큰 파일 크기를 갖는 이러한 후보들 전부를 재고스팅하고, 이어서 필요한 경우 100 메가바이트보다 큰 파일 크기를 갖는 이러한 후보들 전부를 재고스팅하며, 이어서 필요한 경우 50 메가바이트, 기타 등등보다 큰 파일 크기를 갖는 이러한 후보들 전부를 재고스팅하는 것이 바람직할 수 있다. 잘 알 수 있는 바와 같이, 어느 시나리오든지, 후보 파일(14)의 리스트를 생성하고, 리스트를 정렬하며, 하나 이상의 재고스팅 라운드(rounds of re-ghosting)를 발생하고, 기타 등등을 하기 위해 얼마간의 처리가 필요하다.

<128> 또한 유의할 점은 소스(10) 및 싱크(18)에서의 파일(14)의 일관성(consistency)을 유지하기 위해 재고스팅이 이용될 수 있으며, 특히 서로 다른 버전의 파일(14)이 양 장소에 위치할 수 있는 경우에 특히 그렇다는 것이다. 이에 따라, 파일(14)이 소스(10)에서 재구성되고 그의 데이터(20)가 수정되지만 싱크(18)에 있는 대응하는 데이

터(20)가 수정되지 않는 경우, 싱크(18)에 있는 데이터(20)를 소스(10)에 있는 데이터(20)로 대체하기 위해 재고스팅이 수행될 수 있다. 이와 유사하게, 소스(10)에 있는 파일(14)의 데이터(20)가 수정되지 않고 싱크(18)에 있는 대응하는 데이터(20)가 수정되는 경우, 이러한 파일의 나중의 재구성이 싱크(18)에 있는 데이터(20)를 소스(10)로 복사할 것으로 예상하여, 소스(10)에 있는 이러한 데이터(20)를 삭제하기 위해서만 재고스팅이 수행될 수 있다. 물론, 소스(10)에 있는 파일(14)의 데이터(20)가 수정되고 싱크(18)에 있는 대응하는 데이터(20)도 수정되는 경우, 충돌이 존재하며, 그에 따라 파일(14)을 재고스팅해야 하는지 또 그러한 경우 어떻게 재고스팅해야 하는지를 판정하기 위해 적절한 충돌 규칙이 조회되어야만 한다.

<129> 요약하기 위해, 이제 도 10을 참조하면, 볼륨(12) 상의 파일(14)의 재고스팅은 재고스팅 알고리즘에 따라 수행될 수 있으며, 여기서 이러한 재고스팅 알고리즘은 어떤 개체에 의해 다음과 같은 방식으로 수행된다. 사전 준비로서, 어떤 이벤트에 의해 재고스팅 알고리즘이 트리거되고(단계 1001), 여기서 이러한 트리거는 개체에 의해 내부적으로 발생되거나 이러한 개체에 의해 외부로부터 수신될 수 있다. 이러한 트리거 시에, 재고스팅 알고리즘은 어떤 선택 기준에 기초하여 완전히 및 부분적으로 재구성된 파일(14) 및 전혀-고스팅되지 않은 파일(14) 중에서 파일(14)을 선택한다(단계 1003).

<130> 이 시점에서, 재고스팅 알고리즘은 단지 선택된 파일(14)을 재고스팅함으로써 시작될 수 있거나(단계 1005), 그 대신에 선택된 파일(14)을 가능한 재고스팅에 대한 후보 파일(14)로 간주할 수 있다(단계 1007). 후자의 경우에, 후보 파일(14)은 이어서 중단 트리거가 활성화될 때까지 하나 이상의 라운드에서 재고스팅을 위해 선택된다. 상세하게는, 각각의 라운드에 대해, 한 그룹의 후보 파일(14)이 선택되고(단계 1009), 이러한 선택된 그룹이 재고스팅되며(단계 1011), 중단 트리거가 활성화되었는지의 판정이 행해진다. 그러한 경우, 프로세스가 종료된다(단계 1013). 그렇지 않은 경우, 단계(1009)에서와 같이 다른 그룹을 선택하기 위해 되돌아감으로써 프로세스가 계속된다(단계 1015).

<131> 일반 고스팅

<132> 본 명세서에서 지금까지 기술한 바와 같이, 특정의 소스(10)에서 고스팅되는 모든 파일(14)은 상세하게는 그의 데이터(20)가 단일의 싱크(18)에 존재하도록 고스팅된다. 그렇지만, 잘 알 수 있는 바와 같이, 특정의 소스(10)에 있는 각각의 고스팅된 파일(14)의 데이터(20)가, 도 11에 나타낸 바와 같이, 복수의 이러한 싱크(18) 중 임의의 것에 존재할 수 있는 경우가 있을 수 있다.

<133> 상세하게는, 잘 알 수 있는 바와 같이, 본 발명의 고스팅 필터(26)가 반드시 파일(14)로부터의 데이터(20)를 단일의 싱크(18)로 고스팅하기 위해 단일의 고스팅 관리자(28)와 함께 동작하는 것으로 한정될 필요는 없다. 그 대신에, 고스팅 필터(26)는 실제로 복수의 고스팅 관리자(28)와 함께 동작할 수 있으며, 이 경우 각각의 이러한 고스팅 관리자(28)는 복수의 싱크(18) 중 특정의 것과 관련하여 고스팅 기능을 수행한다.

<134> 예를 들어, 지점 서버(10) 및 허브 서버(18)와 관련하여, 특정의 지점 서버(10)는 복수의 허브 서버(18)와 인터페이스할 수 있다. 이에 따라, 특정의 지점 서버(10)와 인터페이스하는 하나의 허브 서버(18)가 제1 소스로부터의 데이터(20)를 가지는 반면 이러한 특정의 지점 서버(10)와 인터페이스하는 다른 허브 서버(18)가 제2 소스로부터의 데이터(20)를 갖는 경우가 있을 수 있다. 이와 마찬가지로, 예를 들어, 컴퓨팅 장치(10) 및 대안의 장소(18)와 관련하여, 특정의 컴퓨팅 장치(10)는 다수의 대안의 장소에 데이터(20)를 저장할 수 있다. 이에 따라, 특정의 컴퓨팅 장치(10)에 대한 한 대안의 장소(18)가 제1 특정의 유형의 파일(14)로부터의 데이터(20)를 저장하는 것으로 지명되는 반면, 특정의 컴퓨팅 장치(10)에 대한 다른 대안의 장소(18)가 제2 특정의 유형의 파일(14)로부터의 데이터(20)를 저장하는 것으로 지명되는 경우가 있을 수 있다. 물론, 이용되는 싱크(18)의 수 및 이러한 싱크들(18) 간에 데이터(20)를 분할하는 기준이 본 발명의 정신 및 범위를 벗어나지 않고 임의의 적절한 수 및 기준일 수 있다.

<135> 어쨌든, 특정의 소스(10)와 함께 다수의 싱크(18)가 이용되는 경우, 어느 싱크(18)가 이러한 소스(10)의 특정의 고스팅된 파일(14)로부터의 데이터(20)를 갖는지를 식별하고 또 어느 대응하는 고스팅 관리자(28)가 이러한 싱크(18)에 액세스하기 위해 이용되어야만 하는지를 식별하기 위한 메카니즘이 필요하다. 그에 따라, 본 발명의 일 실시예에서, 이러한 식별은, 도 11에 나타낸 바와 같이, 이러한 특정의 파일(14)과 연관된 고스팅 정보(24)에 유지된다.

<136> 그 결과, 고스팅 필터(26)는, 임의의 특정의 고스팅된 파일(14)을 만나 그로부터 고스팅 정보(24)를 읽을 시에, 이러한 특정의 고스팅된 파일(14)과 관련하여 이용될 고스팅 관리자(28)의 ID를 이러한 고스팅 정보(24)로부터 획득할 수 있고, 그에 기초하여, 적절한 경우 이러한 싱크(18)에 액세스하기 위해 이러한 식별된 고스팅 관리자

(28)와 통신을 할 수 있다. 사실상, 고스팅 필터(26)는 고스팅 관리자(28) 전부에 대해 범용적인 것이며, 고스팅 필터(26)가 이러한 특정의 고스팅된 파일(14)과 관련하여 임의의 다른 고스팅 관리자(28)와 통신을 해서는 안되는 한, 식별된 고스팅 관리자(28)는 이러한 특정의 고스팅된 파일(14)을 제어 또는 "소유"한다.

<137> 아마도, 각각의 고스팅 관리자(28)는, 고스팅 필터(26)가 이러한 문제들로 걱정할 필요가 없도록, 그에 대응하는 싱크(18)와 통신하는 데 필요한 모든 기능 및 정보를 포함한다. 고스팅 필터(26)에게는, 특정의 고스팅된 파일(14)과 관련하여 식별된 고스팅 관리자(28)와 통신을 하는 일이 대응하는 싱크(18)에 액세스하기 위해 필요한 전부이며, 이러한 고스팅 필터(26)는 실제로 이러한 식별된 고스팅 관리자(28)가 이러한 대응하는 싱크(18)와 어떻게 통신을 하는지, 이러한 식별된 고스팅 관리자(28)가 이러한 대응하는 싱크(18)를 어떻게 찾아내는지, 기타 등등으로 걱정할 필요가 없다.

<138> 도 4 및 도 5에 나타낸 프로세스와 유사하게, 이제 도 12를 참조하면, 파일(14)의 일반적 고스팅 및 재구성이 다음과 같은 방식으로 수행된다. 아마도, 이러한 고스팅은 특정의 ID(identification)를 갖는 특정의 고스팅 관리자(28)에 의해 개시되어 실제로 고스팅 필터(26)에 의해 수행되며, 이러한 고스팅 관리자(28)는 이에 따라 상기 ID 및 고스팅될 파일(14)의 ID와 함께 적절한 고스팅 요청을 고스팅 필터(26)로 전송하고(단계 1201), 그 후에 고스팅 필터(26)는 필요에 따라 데이터(20)를 파일(14)로부터 제거한다(단계 1203). 고스팅 필터(26)는 이어서 이러한 제거된 데이터(20)를 필요에 따라 그의 ID를 통해 요청측 고스팅 관리자(28)로 전달하고(단계 1205), 이러한 고스팅 관리자(28)는 이어서 이러한 고스팅 관리자(28)에 포함되어 있는 프로토콜이 무엇이든지 간에 그 프로토콜에 기초하여 이러한 제거된 데이터를 대응하는 싱크(18)로 전달한다.

<139> 이 점에서 유의할 점은, 싱크(18) 및 파일(14)의 유형에 따라, 예를 들어, 싱크(18)가 문제의 파일(14)을 소스(10)로 판독 전용 방식으로 복제한 경우 등에, 고스팅 관리자(28)가 실제로 이러한 제거된 데이터를 싱크(18)로 전송하지 않기로 선택할 수 있다는 것이다. 물론, 이러한 경우에, 고스팅 관리자(28)로부터의 고스팅 요청이 불필요한 것인 단계(1205)를 수행하지 않도록 고스팅 필터에 알려줄 수 있다.

<140> 어쨌든, 이전과 같이, 고스팅 필터(26)는 '고스팅됨' 속성을 설정하고 고스팅 정보(24)를 추가(단계 1207)함으로써 현재-고스팅된 파일의 메타데이터(22)를 수정한다. 중요한 것은, 이러한 고스팅 정보가 고스팅된 파일(14)을 재구성할 때 나중에 사용하기 위해 고스팅 관리자(28)의 ID를 포함해야 한다는 것이다. 이에 따라, 얼마간 나중에 고스팅 관리자(28) 또는 애플리케이션(30)이 이러한 고스팅된 파일(14)의 데이터(20)에의 액세스를 요청할 때, 이 요청은 궁극적으로 상기한 바와 같이 고스팅 필터(26)에 도달하고(단계 1209), 이러한 고스팅 필터(26)는 파일(14)의 메타데이터(22)에서 고스팅 정보(24)를 찾아낸다(단계 1211).

<141> 다시 말하면, 싱크(18)에 저장되어 있는 파일(14)의 데이터(20)는 이러한 식별된 고스팅 정보(24)에 기초하여 찾아내지만, 이 경우에 고스팅 필터(26)는 먼저 이러한 고스팅 정보(24) 내에서 이러한 고스팅된 파일(14)을 책임지고 있는 고스팅 관리자(28)의 ID를 찾아내고(단계 1213), 이러한 ID를 사용하여 대응하는 싱크(18)로부터 이러한 데이터(20)를 실제로 획득하려는 요청을 대응하는 고스팅 관리자(28)로 전달한다(단계 1215). 아마도, 이러한 고스팅 관리자(28)는 이러한 싱크(18)로부터 이러한 데이터(20)를 실제로 획득하고, 그 데이터(20)를 고스팅 필터(26)에 제공하며(단계 1217), 이러한 고스팅 필터(26)는 이어서 이러한 데이터(20)를 문제의 파일(14)로 재구성한다(단계 1219).

<142> 결론

<143> 본 발명과 관련하여 수행되는 프로세스를 실시하는 데 필요한 프로그래밍은 비교적 간단하며 프로그래밍 당업자에게는 명백할 것이다. 그에 따라, 이러한 프로그래밍이 본 명세서에 첨부되어 있지 않다. 본 발명의 정신 및 범위를 벗어나지 않고 본 발명을 실시하는 데 임의의 특정의 프로그래밍이 이용될 수 있다.

<144> 이상의 설명에서, 본 발명이 로컬 볼륨(12), 컴퓨팅 장치(10) 또는 지점 서버(10) 등의 소스(10)에 있는 파일(14)이 복제 또는 고스팅되어 소스에 있는 파일(14)이 이에 따라 필요한 경우에 재구성될 수 있는 축소된 또는 고스팅된 형태로 있도록 그의 데이터(20)가 대안의 장소(18) 또는 허브 서버(18) 등의 싱크(18)에 저장되게 할 수 있는 새롭고 유용한 방법 및 메카니즘을 포함한다는 것을 알 수 있다. 고스팅된 파일(14)은 필요에 따라 형성 및 재구성될 수 있다.

산업상 이용 가능성

<145> 본 발명의 개념을 벗어나지 않고 상기한 실시예들에 여러 변경이 행해질 수 있다는 것을 잘 알 것이다. 일반적으로, 따라서 본 발명이 개시된 특정의 실시예들로 한정되지 않고 첨부된 청구항들에 의해 정의되는 본 발명의

정신 및 범위 내의 여러 수정들을 포괄하는 것으로 보아야 한다는 것을 잘 알 것이다.

도면의 간단한 설명

<14> 도 1은 본 발명의 측면들 및/또는 그의 일부분이 포함될 수 있는 범용 컴퓨터 시스템을 나타낸 블록도.

<15> 도 2는 본 발명의 실시예들에 따른, 파일의 데이터가 싱크에 저장되도록 소스에서 고스팅된 파일을 나타낸 블록도.

<16> 도 3a 및 도 3b는 본 발명의 실시예들에 따른, 데이터만이 싱크에 저장되어 있고(도 3a) 전체 파일이 싱크에 저장되어 있는(도 3b) 도 2의 파일 및 데이터를 나타낸 블록도.

<17> 도 4는 본 발명의 실시예들에 따른, 파일을 도 2의 싱크로 고스팅하는 데 수행되는 주요 단계들을 나타낸 흐름도.

<18> 도 5는 본 발명의 일 실시예에 따른, 도 2의 싱크로부터 고스팅된 파일을 재구성하는 데 수행되는 주요 단계들을 나타낸 흐름도.

<19> 도 6은 본 발명의 일 실시예에 따른, 도 2의 싱크로부터 고스팅된 파일을 재구성하는 데 수행되는 주요 단계들을 보다 상세히 나타낸 흐름도.

<20> 도 7은 본 발명의 일 실시예에 따른, 도 2의 싱크로부터 고스팅된 파일을 부분적으로 재구성하는 데 수행되는 주요 단계들을 나타낸 흐름도.

<21> 도 8은 본 발명의 일 실시예에 따른, 도 2의 파일의 데이터에 대한 여러가지 상태를 나타낸 블록도.

<22> 도 9는 본 발명의 일 실시예에 따른, 도 8에 나타낸 일부분의 데이터의 세그먼트들의 상태에 기초하여 도 2의 싱크로부터 고스팅된 파일의 적어도 일부분을 효율적으로 재구성하는 데 수행되는 주요 단계들을 나타낸 흐름도.

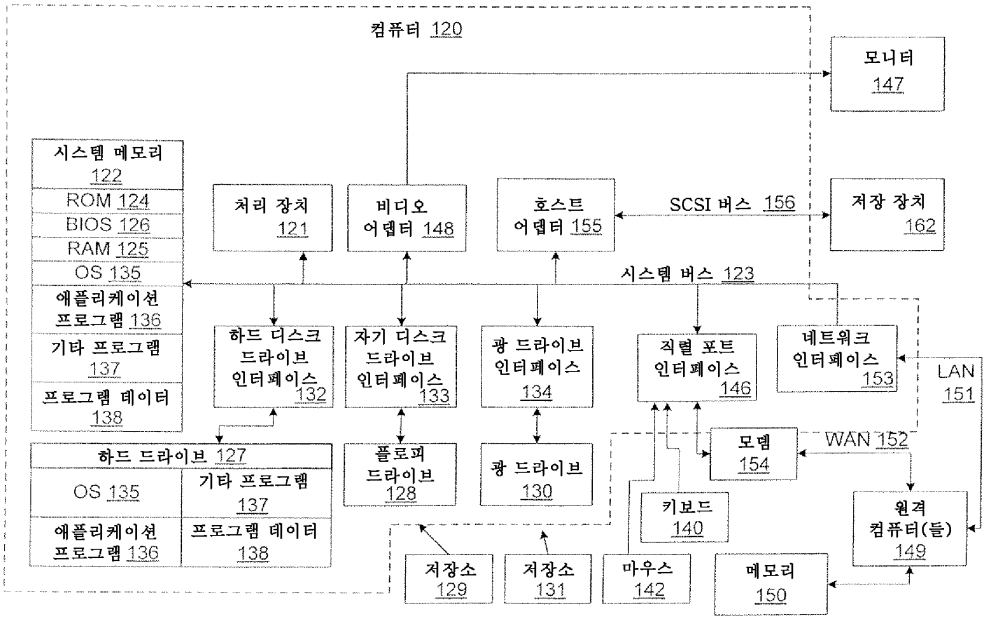
<23> 도 10은 본 발명의 일 실시예에 따른 도 2의 싱크로 파일을 재고스팅(re-ghost)할 때 수행되는 주요 단계들을 나타낸 흐름도.

<24> 도 11은 본 발명의 일 실시예에 따른, 소스가 단일의 일반 고스팅 필터(generic ghosting filter) 및 각각의 싱크에 대응하는 고스팅 관리자(ghosting manager)를 갖는, 복수의 싱크와 연관된 도 2의 소스를 나타낸 블록도.

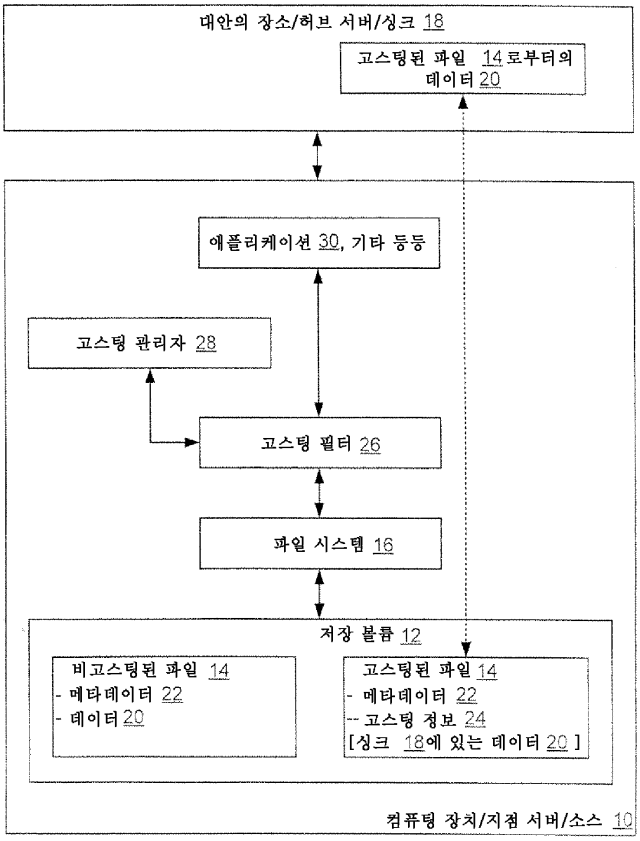
<25> 도 12는 본 발명의 일 실시예에 따른, 파일을 고스팅 및 재구성하는 데 도 11의 고스팅 필터에 의해 수행되는 주요 단계들을 나타낸 흐름도.

도면

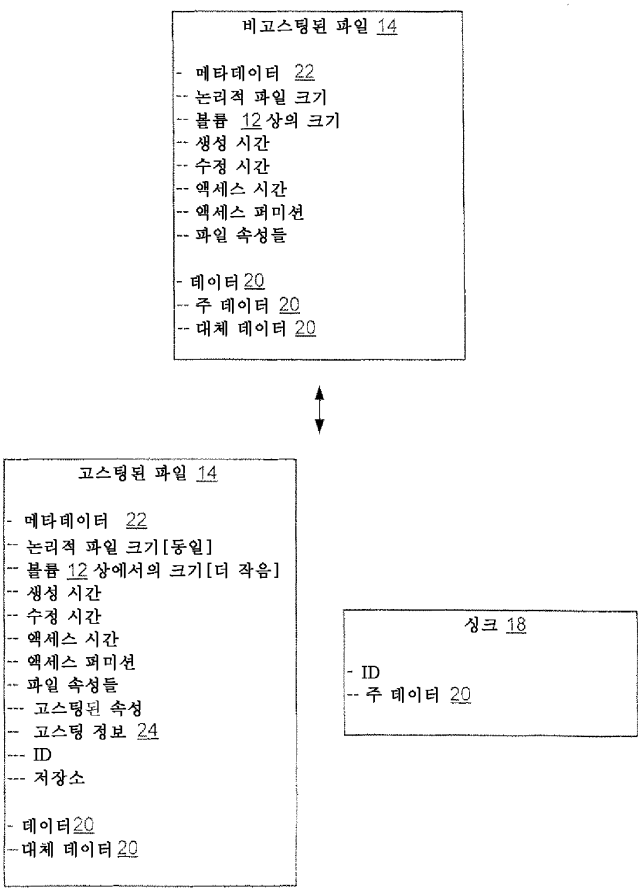
도면1


컴퓨터 120
모니터 147
시스템 메모리 122
ROM 124
BIOS 126
RAM 125
OS 135
애플리케이션 프로그램 136
기타 프로그램 137
프로그램 데이터 138
처리 장치 121
비디오 어댑터 148
호스트 어댑터 155
SCSI 버스 156
저장 장치 162
시스템 버스 123
하드 디스크 드라이브 인터페이스 132
자기 디스크 드라이브 인터페이스 133
광 드라이브 인터페이스 134
직렬 포트 인터페이스 146
네트워크 인터페이스 153
LAN 151
하드 드라이브 127
OS 135
기타 프로그램 137
애플리케이션 프로그램 136
프로그램 데이터 138
플로피 드라이브 128
광 드라이브 130
모뎀 154
WAN 152
키보드 140
원격 컴퓨터(들) 149
저장소 129
저장소 131
마우스 142
메모리 150

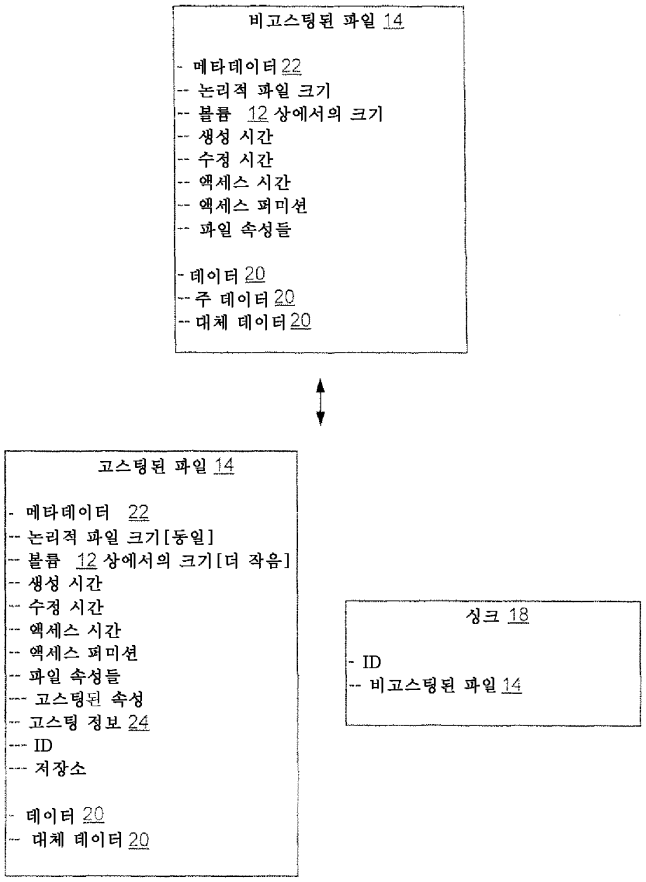
도면2


대안의 장소/허브 서버/싱크 18
고스팅된 파일 14로부터의 데이터 20
애플리케이션 30, 기타 등등
고스팅 관리자 28
고스팅 필터 26
파일 시스템 16
저장 볼륨 12
비고스팅된 파일 14
- 메타데이터 22
- 데이터 20
고스팅된 파일 14
- 메타데이터 22
-- 고스팅 정보 24
[싱크 18에 있는 데이터 20]
컴퓨팅 장치/지점 서버/소스 10

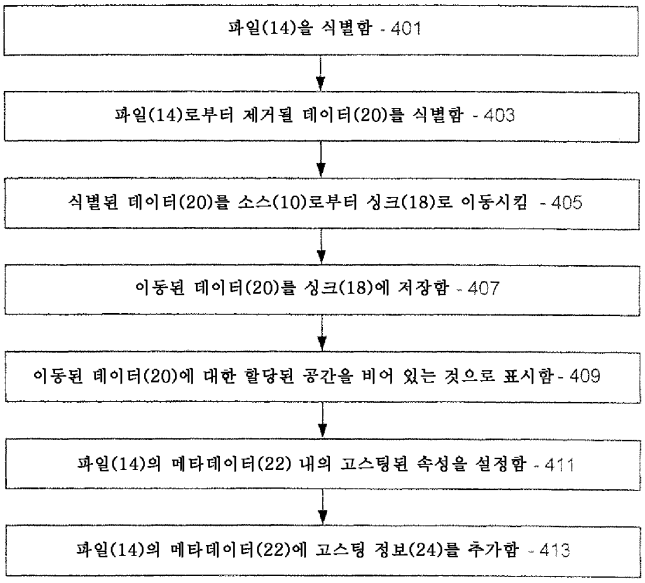
도면3a


비고스팅된 파일 14
- 메타데이터 22
-- 논리적 파일 크기
-- 블륨 12 상의 크기
-- 생성 시간
-- 수정 시간
-- 액세스 시간
-- 액세스 퍼미션
-- 파일 속성들
- 데이터 20
-- 주 데이터 20
-- 대체 데이터 20
고스팅된 파일 14
- 메타데이터 22
-- 논리적 파일 크기[동일]
-- 블륨 12 상에서의 크기[더 작음]
-- 생성 시간
-- 수정 시간
-- 액세스 시간
-- 액세스 퍼미션
-- 파일 속성들
--- 고스팅된 속성
-- 고스팅 정보 24
--- ID
--- 저장소
- 데이터 20
-- 대체 데이터 20
싱크 18
- ID
-- 주 데이터 20

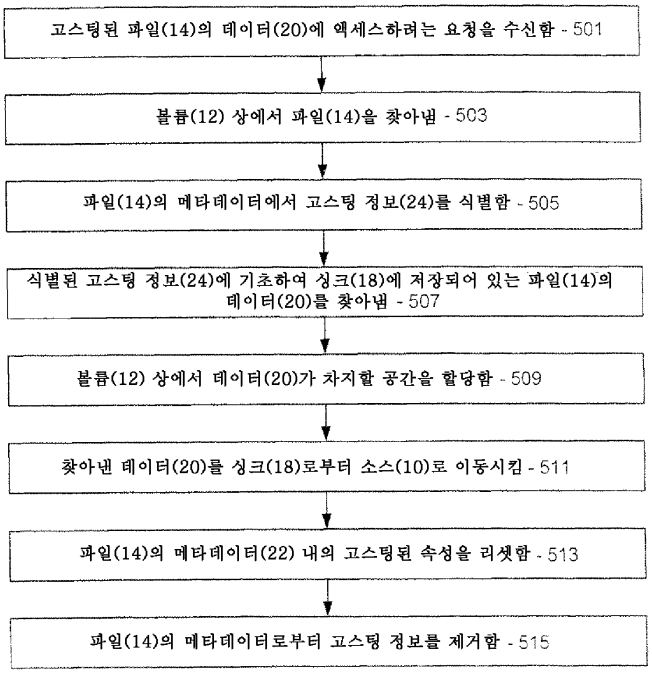
도면3b


비고스팅된 파일 14
- 메타데이터 22
-- 논리적 파일 크기
-- 볼륨 12 상에서의 크기
-- 생성 시간
-- 수정 시간
-- 액세스 시간
-- 액세스 퍼미션
-- 파일 속성들
- 데이터 20
-- 주 데이터 20
-- 대체 데이터 20
고스팅된 파일 14
- 메타데이터 22
-- 논리적 파일 크기[동일]
-- 볼륨 12 상에서의 크기[더 작음]
-- 생성 시간
-- 수정 시간
-- 액세스 시간
-- 액세스 퍼미션
-- 파일 속성들
--- 고스팅된 속성
-- 고스팅 정보 24
--- ID
--- 저장소
- 데이터 20
-- 대체 데이터 20
싱크 18
- ID
-- 비고스팅된 파일 14

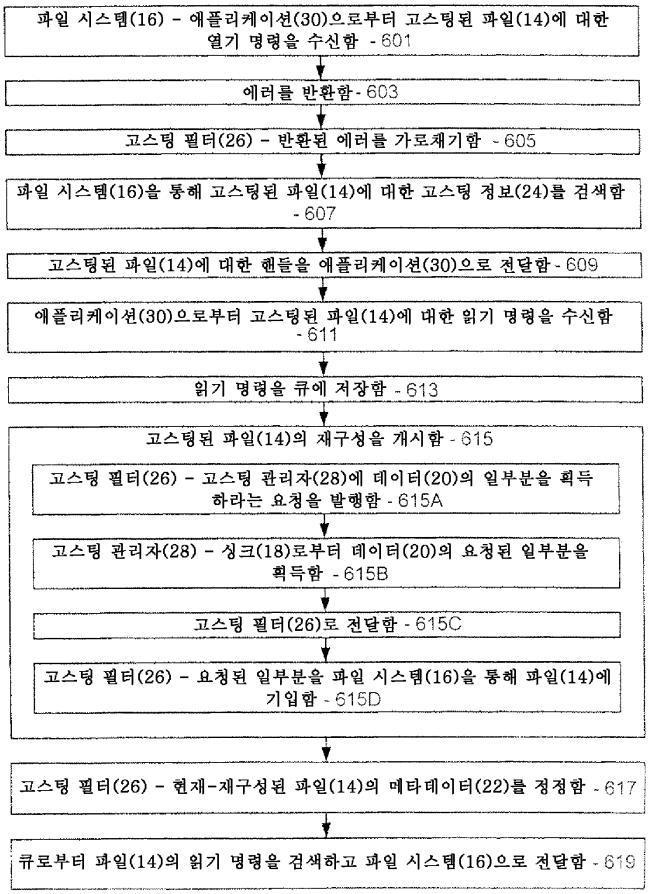
도면4


파일(14)을 식별함 - 401
파일(14)로부터 제거될 데이터(20)를 식별함 - 403
식별된 데이터(20)를 소스(10)로부터 싱크(18)로 이동시킴 - 405
이동된 데이터(20)를 싱크(18)에 저장함 - 407
이동된 데이터(20)에 대한 할당된 공간을 비어 있는 것으로 표시함 - 409
파일(14)의 메타데이터(22) 내의 고스팅된 속성을 설정함 - 411
파일(14)의 메타데이터(22)에 고스팅 정보(24)를 추가함 - 413

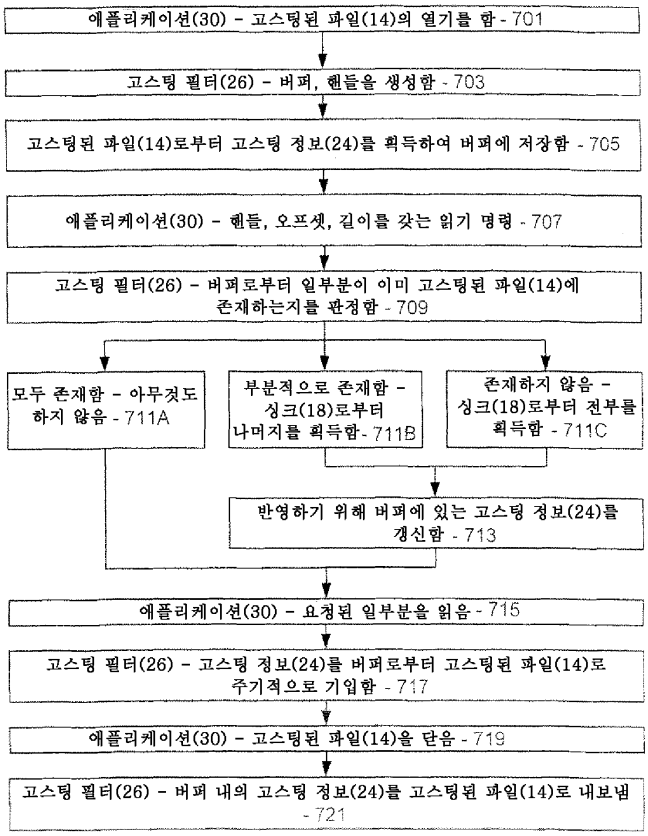
도면5


고스팅된 파일(14)의 데이터(20)에 액세스하려는 요청을 수신함 - 501
볼륨(12) 상에서 파일(14)을 찾아냄 - 503
파일(14)의 메타데이터에서 고스팅 정보(24)를 식별함 - 505
식별된 고스팅 정보(24)에 기초하여 싱크(18)에 저장되어 있는 파일(14)의 데이터(20)를 찾아냄 - 507
볼륨(12) 상에서 데이터(20)가 차지할 공간을 할당함 - 509
찾아낸 데이터(20)를 싱크(18)로부터 소스(10)로 이동시킴 - 511
파일(14)의 메타데이터(22) 내의 고스팅된 속성을 리셋함 - 513
파일(14)의 메타데이터로부터 고스팅 정보를 제거함 - 515

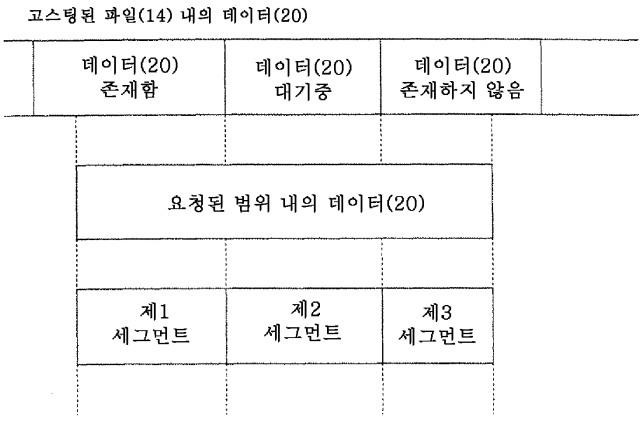
도면6


파일 시스템(16) - 애플리케이션(30)으로부터 고스팅된 파일(14)에 대한 열기 명령을 수신함 - 601
에러를 반환함 - 603
고스팅 필터(26) - 반환된 에러를 가로채기함 - 605
파일 시스템(16)을 통해 고스팅된 파일(14)에 대한 고스팅 정보(24)를 검색함 - 607
고스팅된 파일(14)에 대한 핸들을 애플리케이션(30)으로 전달함 - 609
애플리케이션(30)으로부터 고스팅된 파일(14)에 대한 읽기 명령을 수신함 - 611
읽기 명령을 큐에 저장함 - 613
고스팅된 파일(14)의 재구성을 개시함 - 615
고스팅 필터(26) - 고스팅 관리자(28)에 데이터(20)의 일부분을 획득하라는 요청을 발행함 - 615A
고스팅 관리자(28) - 싱크(18)로부터 데이터(20)의 요청된 일부분을 획득함 - 615B
고스팅 필터(26)로 전달함 - 615C
고스팅 필터(26) - 요청된 일부분을 파일 시스템(16)을 통해 파일(14)에 기입함 - 615D
고스팅 필터(26) - 현재-재구성된 파일(14)의 메타데이터(22)를 정정함 - 617
큐로부터 파일(14)의 읽기 명령을 검색하고 파일 시스템(16)으로 전달함 - 619

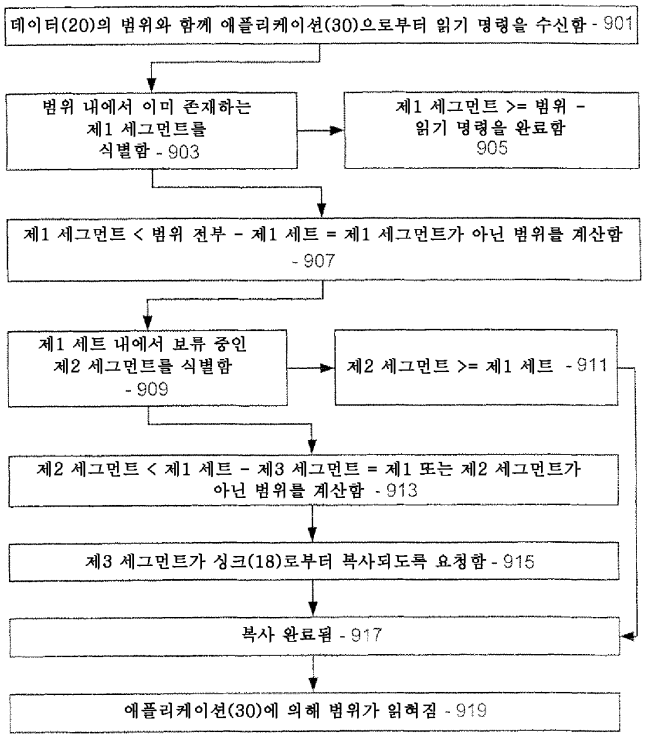
도면7


애플리케이션(30) - 고스팅된 파일(14)의 열기를 함 - 701
고스팅 필터(26) - 버퍼, 핸들을 생성함 - 703
고스팅된 파일(14)로부터 고스팅 정보(24)를 획득하여 버퍼에 저장함 - 705
애플리케이션(30) - 핸들, 오프셋, 길이를 갖는 읽기 명령 - 707
고스팅 필터(26) - 버퍼로부터 일부분이 이미 고스팅된 파일(14)에 존재하는지를 판정함 - 709
모두 존재함 - 아무것도 하지 않음 - 711A
부분적으로 존재함 - 싱크(18)로부터 나머지를 획득함 - 711B
존재하지 않음 - 싱크(18)로부터 전부를 획득함 - 711C
반영하기 위해 버퍼에 있는 고스팅 정보(24)를 갱신함 - 713
애플리케이션(30) - 요청된 일부분을 읽음 - 715
고스팅 필터(26) - 고스팅 정보(24)를 버퍼로부터 고스팅된 파일(14)로 주기적으로 기입함 - 717
애플리케이션(30) - 고스팅된 파일(14)을 닫음 - 719
고스팅 필터(26) - 버퍼 내의 고스팅 정보(24)를 고스팅된 파일(14)로 내보냄 - 721

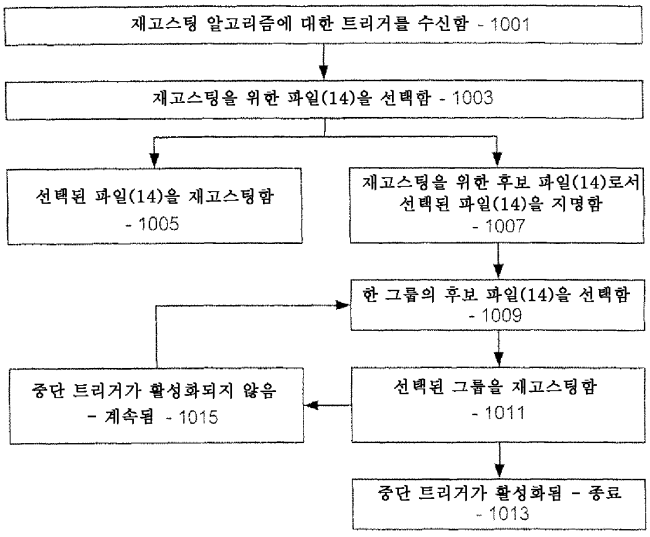
도면8


고스팅된 파일(14) 내의 데이터(20)
데이터(20) 존재함
데이터(20) 대기중
데이터(20) 존재하지 않음
요청된 범위 내의 데이터(20)
제1 세그먼트
제2 세그먼트
제3 세그먼트

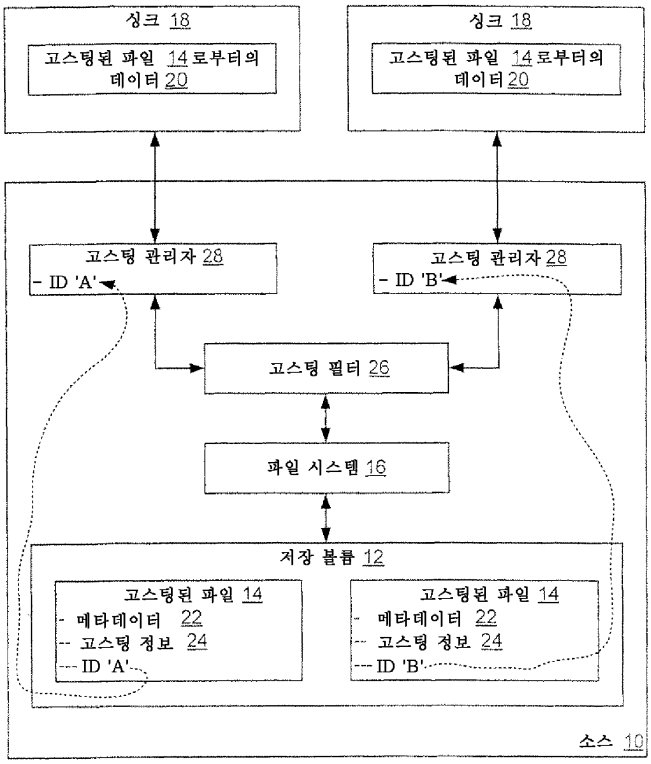
도면9


데이터(20)의 범위와 함께 애플리케이션(30)으로부터 읽기 명령을 수신함 - 901
범위 내에서 이미 존재하는 제1 세그먼트를 식별함 - 903
제1 세그먼트 >= 범위 - 읽기 명령을 완료함 905
제1 세그먼트 < 범위 전부 - 제1 세트 = 제1 세그먼트가 아닌 범위를 계산함 - 907
제1 세트 내에서 보류 중인 제2 세그먼트를 식별함 - 909
제2 세그먼트 >= 제1 세트 - 911
제2 세그먼트 < 제1 세트 - 제3 세그먼트 = 제1 또는 제2 세그먼트가 아닌 범위를 계산함 - 913
제3 세그먼트가 싱크(18)로부터 복사되도록 요청함 - 915
복사 완료됨 - 917
애플리케이션(30)에 의해 범위가 읽혀짐 - 919

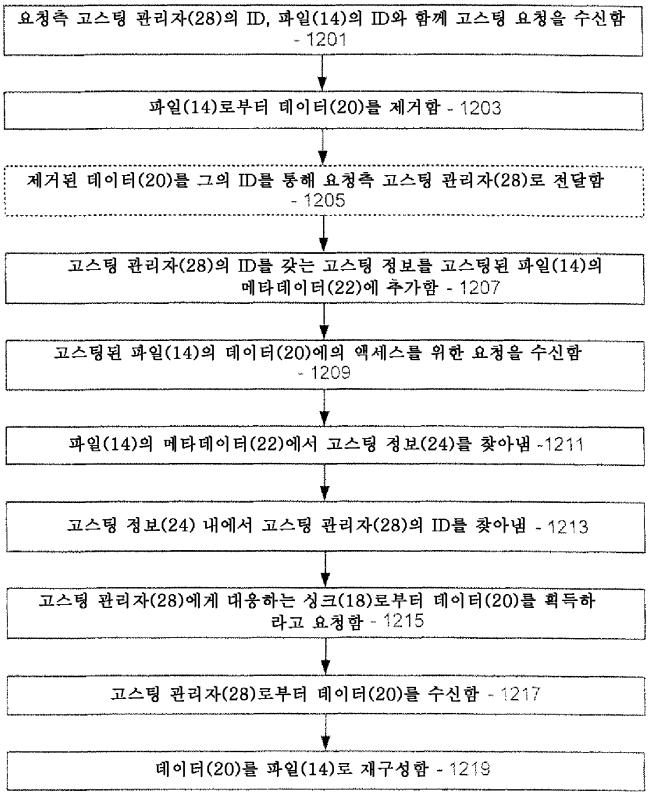
도면10


재고스팅 알고리즘에 대한 트리거를 수신함 - 1001
재고스팅을 위한 파일(14)을 선택함 - 1003
선택된 파일(14)을 재고스팅함 - 1005
재고스팅을 위한 후보 파일(14)로서 선택된 파일(14)을 지명함 - 1007
한 그룹의 후보 파일(14)을 선택함 - 1009
중단 트리거가 활성화되지 않음 - 계속됨 - 1015
선택된 그룹을 재고스팅함 - 1011
중단 트리거가 활성화됨 - 종료 - 1013

도면11


싱크 18
고스팅된 파일 14로부터의 데이터 20
싱크 18
고스팅된 파일 14로부터의 데이터 20
고스팅 관리자 28
- ID 'A'
고스팅 관리자 28
- ID 'B'
고스팅 필터 26
파일 시스템 16
저장 볼륨 12
고스팅된 파일 14
- 메타데이터 22
-- 고스팅 정보 24
--- ID 'A'
고스팅된 파일 14
- 메타데이터 22
-- 고스팅 정보 24
--- ID 'B'
소스 10

도면12


요청측 고스팅 관리자(28)의 ID, 파일(14)의 ID와 함께 고스팅 요청을 수신함 - 1201
파일(14)로부터 데이터(20)를 제거함 - 1203
제거된 데이터(20)를 그의 ID를 통해 요청측 고스팅 관리자(28)로 전달함 - 1205
고스팅 관리자(28)의 ID를 갖는 고스팅 정보를 고스팅된 파일(14)의 메타데이터(22)에 추가함 - 1207
고스팅된 파일(14)의 데이터(20)에의 액세스를 위한 요청을 수신함 - 1209
파일(14)의 메타데이터(22)에서 고스팅 정보(24)를 찾아냄 -1211
고스팅 정보(24) 내에서 고스팅 관리자(28)의 ID를 찾아냄 - 1213
고스팅 관리자(28)에게 대응하는 싱크(18)로부터 데이터(20)를 획득하라고 요청함 - 1215
고스팅 관리자(28)로부터 데이터(20)를 수신함 - 1217
데이터(20)를 파일(14)로 재구성함 - 1219