(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년05월15일
(11) 등록번호 10-1735719
(24) 등록일자 2017년05월08일

(51) 국제특허분류(Int. Cl.)
G06F 9/44 (2006.01) *G06F 15/16* (2006.01)
(21) 출원번호 10-2012-7021031
(22) 출원일자(국제) 2011년01월11일
심사청구일자 2014년01월06일
(85) 번역문제출일자 2012년08월10일
(65) 공개번호 10-2012-0125292
(43) 공개일자 2012년11월14일
(86) 국제출원번호 PCT/US2011/020790
(87) 국제공개번호 WO 2011/088022
국제공개일자 2011년07월21일
(30) 우선권주장
61/294,266 2010년01월12일 미국(US)
(56) 선행기술조사문헌
KR1020030071750 A*
JP2002108801 A*
KR1020070100708 A*
US20050022175 A1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
구글 인코포레이티드
미국 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600 (우:94043)
(72) 발명자
데 로스 레예스 앤드류
미국 캘리포니아주 94061 레드우드 시티 팜 힐 블러바드 에이피티. 1 4028
(74) 대리인
박장원

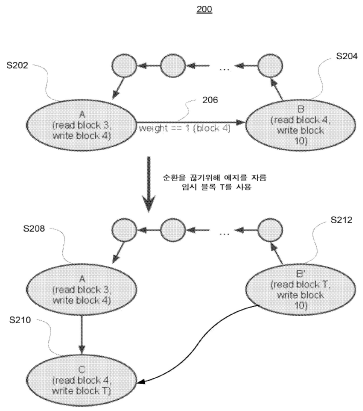
전체 청구항 수 : 총 16 항 심사관 : 최정권

(54) 발명의 명칭 **운영 체제 자동 업데이트 절차**

(57) 요 약

본 발명은 클라이언트 디바이스(302, 306, 308, 310, 312)에 있는 운영 체제의 차등 업데이트에 관한 것이다. 델타 업데이트 파일(100)은 구 인스톨 파티션으로부터 복사(port)할, 메모리(326)에 있는 새 인스톨 파티션상에서 수행될 오퍼레이션들의 순서 리스트를 포함한다. 이진 차등 압축 알고리즘이 업데이트를 위한 차이를 결정하는데 사용될 수 있고, 클라이언트 디바이스로 전송되는 데이터를 압축하는데 사용될 수 있다. 파티션에 있는 블록들은 서로(S202, S204)에게 순환적으로 의존할 수 있다. 에지들이 순환들을 끊기 위해 잘린다. 시스템 오퍼레이션 동안, 디스크 이미지들이 스캔된다(도 3에서 1). 그래프가 생성되고(도 3에서 3), 에지 가중치들이 블록들의 개수와 동일하게 설정된다. 이런 전제하에, 이어 순환들이 끊어지고(도 3에서 4), 토폴로지화 정렬로부터 도출된 마지막 순서가 생산된다(도 3에서 5).

대 표 도



명 세 서

청구범위

청구항 1

컴퓨터의 운영 체제(operating system)에 대한 업데이트를 생성하는 방법으로서,

상기 운영 체제의 현재 버전의 버전 넘버(version number)를 식별하는 단계;

프로세서로, 상기 운영 체제의 현재 버전을 상기 운영 체제의 새 버전으로 업데이트하기 위한 오퍼레이션들의 순서 리스트(ordered list)를 생성하는 단계, 상기 프로세서는 상기 새 버전과 관련된 모든 데이터 블록들에 대한 순서 리스트를 얻기 위해, 상기 운영 체제의 새 버전의 각 레귤러 파일(regular file)에 대해 반복(iteration)들을 수행하며, 상기 순서 리스트에 있는 각각의 오퍼레이션은 클라이언트 디바이스의 파티션(partition) 내의 하나 이상의 특정 데이터 블록(block)에 적용가능하고;

상기 순서 리스트의 오퍼레이션들 중 대응하는 오퍼레이션에 대한 데이터의 묶음(chunk)을 각각 나타내는 데이터 블랍(data blob)들의 순서 세트(ordered set)를 생성하는 단계로서, 상기 데이터 블랍들의 순서 세트의 각각의 데이터 블랍은, 상기 순서 리스트의 오퍼레이션들 중 해당 데이터 블랍에 대응하는 오퍼레이션에 기초하여 압축되거나 압축되지 않는, 데이터 블랍들의 순서 세트 생성 단계; 및

상기 프로세서로,

차등 업데이트 파일(differential update file)이 실제 업데이트 파일임을 나타내는 매직 넘버 표시자(magic number indicator),

상기 매직 넘버 바로 다음의, 운영 체제의 새 버전을 식별하는 새 버전 넘버,

상기 새 버전 넘버 바로 다음의, 상기 오퍼레이션들의 순서 리스트를 포함하는 프로토콜 버퍼(protocol buffer),

상기 프로토콜 버퍼 바로 다음의, 상기 데이터 블랍들의 순서 세트의 블록, 및

상기 순서 리스트 내의 하나 이상의 오퍼레이션들의 각각의 데이터 블랍의 블록 바로 다음의, 파일 끝 표시자(EOF: end of file)

를 포함하는 상기 차등 업데이트 파일을 어셈블링(assembling)하는 단계를 포함하며,

상기 오퍼레이션들은:

상기 파티션 내의 데이터 블록들 중 적어도 하나가 상기 운영 체제의 새 버전을 위한 상기 클라이언트 디바이스의 새 파티션 내의 다른 블록에 복사되는 복사 오퍼레이션(copy operation);

상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록이 메모리에 독출되고, 상기 차등 업데이트 파일의 데이터 블랍을 이용하여 상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록에 차등 루틴(difference routine)을 수행하는 차등 오퍼레이션(difference operation);

상기 차등 업데이트 파일의 선택된 데이터 블랍이 상기 새 파티션 내의 특정된 블록(specified block)들에 기록되도록 구성되는 대체 오퍼레이션(replace operation); 및

압축된 데이터 블랍이 상기 차등 업데이트 파일 내에 포함되고, 그 압축된 데이터 블랍이 상기 운영 체제의 새 버전을 위한 새 파티션 내의 선택된 특정된 블록들에 쓰여지도록 구성되는 압축 해제 오퍼레이션으로의 대체(replace) 중 하나 이상을 포함하는 것을 특징으로 하는 방법.

청구항 2

삭제

청구항 3

삭제

청구항 4

제1항에 있어서,

상기 순서 리스트에 있는 각 오퍼레이션은 파일 오브젝트(file object)와 관련되고, 상기 방법은

각 파일 오브젝트를 위한 그래프에서 정점(vertex)을 생성하는 단계; 및

각 블록을 나타내는 벡터(vector)를 생성하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 5

제4항에 있어서,

각 블록의 벡터에 대한 리더(reader)와 라이터(writer) 파라미터를 설정하는 단계와;

다른 리더와 라이터를 갖는 각 블록에 대하여, 상기 그래프에 상기 라이터로부터 리더까지의 에지(edge)를 생성하는 단계를 더 포함하고,

상기 에지는 자신과 관련된 소스 파일 오퍼레이션(source file operation)이 시작하기 전에 완료되어야 할 파일 오퍼레이션을 가리키는 것을 특징으로 하는 방법.

청구항 6

제5항에 있어서,

상기 각 에지는 가중치(weight)를 갖고,

상기 가중치는 상기 그래프에서 각 에지와 관련된 블록들의 개수를 식별하는 것을 특징으로 하는 방법.

청구항 7

제5항에 있어서,

상기 그래프가 순환(cycle)들을 포함하면, 상기 방법은 각 순환을 끊는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 8

제7항에 있어서,

상기 순환들 중 주어진 하나를 끊는 단계는

상기 주어진 순환과 관련된 최저 가중치 에지(a lowest-weight edge)를 찾는 단계; 및

상기 최저 가중치 에지를 잘라내는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 9

제8항에 있어서,

상기 최저 가중치 에지를 잘라내는 단계는

익스텐트(extent)를 스크래치 공간(scratch space)에 복사하는 오퍼레이션을 나타내는 새 노드를 생성하는 단계; 및

상기 최저 가중치 에지의 소스 노드(source node)로 하여금 상기 새 노드를 가리키도록 하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 10

제9항에 있어서,

상기 최저 가중치 에지의 소스 노드로 하여금 상기 새 노드를 가리키도록 하는 단계는,

새 복사 오퍼레이션이 상기 오퍼레이션들의 순서 리스트에 대해 정확한 순서로 행해지도록, 상기 최저 가중치 에지의 목적 노드(destination node)로부터 새 에지를 만드는 단계를 포함하는, 방법.

청구항 11

컴퓨터의 운영 체제를 위한 업데이트를 생성하는 디바이스로서,

상기 운영 체제와 관련된 차등 업데이트 정보를 저장하기 위한 메모리; 및

상기 메모리에 결합된 프로세서를 포함하고,

상기 프로세서는,

상기 운영 체제의 현재 버전의 버전 넘버를 식별하고;

새 버전과 관련된 모든 데이터 블록들에 대한 순서 리스트를 얻기 위해, 상기 운영 체제의 새 버전의 각 레귤러 파일에 대해 반복들을 수행하는 것을 포함하여, 상기 차등 업데이트 정보를 사용하여 상기 운영 체제의 현재 버전을 상기 운영 체제의 새 버전으로 업데이트하기 위한 오퍼레이션들의 순서 리스트를 생성하고 - 상기 순서 리스트 내에 있는 각각의 오퍼레이션은 클라이언트 디바이스의 파티션의 하나 이상의 특정 데이터 블록에 적용가능함 -;

상기 순서 리스트의 오퍼레이션들 중 대응하는 오퍼레이션에 대한 데이터의 묶음을 각각 나타내는 데이터 블랍들의 순서 세트를 생성하고 - 상기 데이터 블랍들의 순서 세트의 각각의 데이터 블랍은, 상기 순서 리스트의 오퍼레이션들 중 해당 데이터 블랍에 대응하는 오퍼레이션에 기초하여 압축되거나 압축되지 않음 -;

차등 업데이트 파일이 실제 업데이트 파일임을 나타내는 매직 넘버 표시자, 상기 운영 체제의 새 버전을 식별하는 새 버전 넘버, 및 상기 오퍼레이션들의 순서 리스트를 포함하는 프로토콜 버퍼를 포함하는 상기 차등 업데이트 파일을 어셈블링하고;

상기 차등 업데이트 파일을 상기 메모리에 저장하도록 구성되며,

상기 오퍼레이션들은:

상기 파티션 내의 데이터 블록들 중 적어도 하나가 상기 운영 체제의 새 버전을 위한 상기 클라이언트 디바이스의 새 파티션 내의 다른 블록에 복사되는 복사 오퍼레이션(copy operation);

상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록이 메모리에 독출되고, 상기 차등 업데이트 파일의 데이터 블랍을 이용하여 상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록에 차등 루틴(difference routine)을 수행하는 차등 오퍼레이션(difference operation);

상기 차등 업데이트 파일의 선택된 데이터 블랍이 상기 새 파티션 내의 특정된 블록(specified block)들에 기록되도록 구성되는 대체 오퍼레이션(replace operation); 및

압축된 데이터 블랍이 상기 차등 업데이트 파일 내에 포함되고, 그 압축된 데이터 블랍이 상기 운영 체제의 새 버전을 위한 새 파티션 내의 선택된 특정된 블록들에 쓰여지도록 구성되는 압축 해제 오퍼레이션으로의 대체(replace) 중 하나 이상을 포함하며;

상기 순서 리스트 내에 있는 각각의 오퍼레이션은 파일 오브젝트와 관련되고,

상기 프로세서는,

각 파일 오브젝트를 위한 그래프에서 정점을 생성하고,

각 블록을 나타내는 벡터를 생성하도록 더 구성된 것을 특징으로 하는 디바이스.

청구항 12

삭제

청구항 13

제11항에 있어서,

상기 프로세서는

각 블록의 상기 벡터에 대한 리더 및 라이터 파라미터를 설정하는 수단; 및

다른 리더 및 라이터를 갖는 각 블록에 대하여, 상기 그래프에서 상기 라이터로부터 상기 리더까지의 에지를 생성하도록 더 구성되고,

상기 에지는 자신과 관련된 소스 파일 오퍼레이션(source file operation)이 시작하기 전에 완료되어야 할 파일 오퍼레이션을 가리키는 것을 특징으로 하는 디바이스.

청구항 14

제13항에 있어서,

상기 그래프가 순환들을 포함하면, 상기 프로세서는 각 순환을 끊도록 더 동작가능한 것을 특징으로 하는 디바이스.

청구항 15

제14항에 있어서,

상기 순환들 중 주어진 하나를 끊는 것은

상기 주어진 순환과 관련된 최저 가중치 에지를 찾는 단계; 및

상기 최저 가중치 에지를 잘라내는 단계를 포함하는 것을 특징으로 하는 디바이스.

청구항 16

제15항에 있어서,

상기 프로세서가

익스텐트를 스크래치 공간에 복사하는 오퍼레이션을 나타내는 새 노드를 생성하고, 상기 최저 가중치 에지의 소스 노드로 하여금 상기 새 노드를 가르키도록 함으로써, 상기 최저 가중치 에지를 잘라내도록 동작가능한 것을 특징으로 하는 디바이스.

청구항 17

제16항에 있어서,

상기 최저 가중치 에지의 소스 노드로 하여금 상기 새 노드를 가르키도록 하는 상기 프로세서는 또한,

새 복사 오퍼레이션이 상기 오퍼레이션들의 순서 리스트에 대해 정확한 순서로 행해지도록, 상기 최저 가중치 에지의 목적 노드로부터 새 에지를 만드는 것을 특징으로 하는 디바이스.

청구항 18

컴퓨터 판독가능 저장 매체로서,

컴퓨터에 의해 실행되었을 때, 상기 컴퓨터로 하여금 컴퓨터의 운영 체제를 위한 업데이트를 생성하는 방법을 수행하게 하는, 컴퓨터 프로그램의 컴퓨터 판독가능 인스트럭션들이 저장되며,

상기 방법은,

상기 운영 체제의 현재 버전의 버전 넘버를 식별하는 단계;

프로세서로, 상기 운영 체제의 현재 버전을 상기 운영 체제의 새 버전으로 업데이트하기 위한 오퍼레이션들의 순서 리스트를 생성하는 단계로서, 상기 프로세서는 상기 새 버전과 관련된 모든 데이터 블록들에 대한 순서 리스트를 얻기 위해, 상기 운영 체제의 새 버전의 각 레귤러 파일에 대해 반복들을 수행하며, 상기 순서 리스트 내에 있는 각각의 오퍼레이션은 클라이언트 디바이스의 파티션의 하나 이상의 특정 데이터 블록에 적용가능한 것인, 오퍼레이션들의 순서 리스트 생성 단계;

상기 순서 리스트의 오퍼레이션들 중 대응하는 오퍼레이션에 대한 데이터의 묶음을 각각 나타내는 데이

터 블랍들의 순서 세트를 생성하는 단계로서, 상기 데이터 블랍들의 순서 세트의 각각의 데이터 블랍은, 상기 순서 리스트의 오퍼레이션들 중 해당 데이터 블랍에 대응하는 오퍼레이션에 기초하여 압축되거나 압축되지 않는, 데이터 블랍들의 순서 세트 생성 단계; 및

상기 프로세서로, 차등 업데이트 파일이 실제 업데이트 파일임을 나타내는 매직 넘버 표시자, 운영 체제의 새 버전을 식별하는 새 버전 넘버, 및 상기 오퍼레이션들의 순서 리스트를 포함하는 프로토콜 버퍼를 포함하는 상기 차등 업데이트 파일을 어셈블링하는 단계를 포함하고,

상기 오퍼레이션들은:

상기 파티션 내의 데이터 블록들 중 적어도 하나가 상기 운영 체제의 새 버전을 위한 상기 클라이언트 디바이스의 새 파티션 내의 다른 블록에 복사되는 복사 오퍼레이션(copy operation);

상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록이 메모리에 독출되고, 상기 차등 업데이트 파일의 데이터 블랍을 이용하여 상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록에 차등 루틴(difference routine)을 수행하는 차등 오퍼레이션(difference operation);

상기 차등 업데이트 파일의 선택된 데이터 블랍이 상기 새 파티션 내의 특정된 블록(specified block)들에 기록되도록 구성되는 대체 오퍼레이션(replace operation); 및

압축된 데이터 블랍이 상기 차등 업데이트 파일 내에 포함되고, 그 압축된 데이터 블랍이 상기 운영 체제의 새 버전을 위한 새 파티션 내의 선택된 특정된 블록들에 쓰여지도록 구성되는 압축 해제 오퍼레이션으로의 대체(replace) 중 하나 이상을 포함하며;

상기 순서 리스트 내에 있는 각각의 오퍼레이션은 파일 오브젝트와 관련되고,

상기 방법은,

각 파일 오브젝트를 위한 그래프에서 정점을 생성하는 단계,

각 블록을 나타내는 벡터를 생성하는 단계를 더 포함하는 것을 특징으로 하는 컴퓨터 판독가능 저장 매체.

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

청구항 24

클라이언트 디바이스로서,

운영 체제의 현재 버전을 저장하기 위한 메모리; 및

상기 메모리에 결합된 프로세서를 포함하고,

상기 프로세서는,

상기 운영 체제의 현재 버전을 위한 업데이트에 관하여 상기 운영 체제의 현재 버전을 식별하는 버전 넘버를 포

함하는 요청을 원격 디바이스에 전송하고;

차등 업데이트 파일이 실제 업데이트 파일임을 나타내는 매직 넘버 표시자, 상기 운영 체제의 새 버전을 식별하는 새 버전 넘버, 오퍼레이션들의 순서 리스트를 포함하는 프로토콜 버퍼, 및 데이터 블랍들의 순서 세트를 포함하는 데이터 블록을 포함하는 상기 차등 업데이트 파일을 상기 원격 디바이스로부터 수신하고 - 상기 데이터 블랍들의 순서 세트의 각각의 데이터 블랍은, 상기 순서 리스트의 오퍼레이션들 중 대응하는 오퍼레이션에 대한 데이터의 묶음을 나타내며, 상기 순서 리스트의 오퍼레이션들 중 해당 데이터 블랍에 대응하는 오퍼레이션에 기초하여 압축되거나 압축되지 않고, 상기 순서 리스트에 있는 각각의 오퍼레이션은 클라이언트 디바이스의 파티션(partition) 내의 하나 이상의 특정 데이터 블록(block)에 적용가능하고, 상기 오퍼레이션들은:

상기 파티션 내의 데이터 블록들 중 적어도 하나가 상기 운영 체제의 새 버전을 위한 상기 클라이언트 디바이스의 새 파티션 내의 다른 블록에 복사되는 복사 오퍼레이션(copy operation),

상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록이 메모리에 독출되고, 상기 차등 업데이트 파일의 데이터 블랍을 이용하여 상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록에 차등 루틴(difference routine)을 수행하는 차등 오퍼레이션(difference operation),

상기 차등 업데이트 파일의 선택된 데이터 블랍이 상기 새 파티션 내의 특정된 블록(specified block)들에 기록되도록 구성되는 대체 오퍼레이션(replace operation), 및

압축된 데이터 블랍이 상기 차등 업데이트 파일 내에 포함되고, 그 압축된 데이터 블랍이 상기 운영 체제의 새 버전을 위한 새 파티션 내의 선택된 특정된 블록들에 쓰여지도록 구성되는 압축 해제 오퍼레이션으로의 대체(replace) 중 하나 이상을 포함함 -;

매직 넘버를 검증하고;

상기 프로토콜 버퍼로부터 상기 오퍼레이션들의 순서 리스트를 추출하고;

상기 운영 체제의 현재 버전을 상기 운영 체제의 새 버전으로 업데이트하기 위해, 상기 오퍼레이션들의 순서 리스트를 실행함으로써 차등 업데이트를 수행하도록 구성되며,

상기 프로세서는 상기 차등 업데이트 파일이 상기 클라이언트 디바이스로 스트리밍되는 동안, 상기 차등 업데이트 파일을 메모리에 저장하지 않고, 상기 운영 체제의 새 버전으로 상기 운영 체제의 상기 차등 업데이트를 수행하도록 동작가능한 것을 특징으로 하는 클라이언트 디바이스.

청구항 25

삭제

발명의 설명

기 술 분 야

[0001] 본 출원은 2010년 1월 12일자로 출원된 미국 가출원 번호 제61/294,266호에 대한 우선권을 주장하며, 해당 출원에 개시된 내용은 참조로서 모두 포함된다.

배 경 기 술

[0002] 운영 체제(operating system)와 같은 소프트웨어 패키지는 새로운 특징(feature)을 소개하거나 에러 및 어드레스 보안 결함(address security flaw)을 수정하기 위해, 때때로 업데이트(update) 될 수 있다. 용량이 큰 애플리케이션을 위한 파일 사이즈로 인하여, 업데이트를 이용하여 새 패키지 전체를 전송하고 인스톨하는 것은 불편하거나 비효율적일 수 있다. 하나의 솔루션은 클라이언트에게 차등 업데이트(differential update)를 전송하는데, 이 차등 업데이트는 이전 소프트웨어에서 특정 변경(specific change)만을 담당한다. 만약 차등 업데이트가 정확하게 수행되지 않으면, 수정된 소프트웨어가 불완전하게 동작하거나 또는 전혀 동작하지 않을 수 있다.

발명의 내용

해결하려는 과제

[0003] 본 발명은 전반적으로 운영 체제에 관한 것이다. 보다 상세하게는, 운영 체제의 버전(version)을 업데이트하는

것에 관한 것이다.

과제의 해결 수단

[0004] 일 실시예에 따라서 컴퓨터 판독가능 운영 체제에 대한 업데이트를 생성하는 방법이 제공된다. 이 방법은 상기 운영 체제의 현재 버전의 버전 넘버(version number)를 식별하는 단계; 프로세서로, 상기 운영 체계의 현재 버전을 상기 운영 체계의 새 버전으로 업데이트하기 위한 오퍼레이션들의 순서 리스트(ordered list)를 생성하는 단계-여기서, 상기 프로세서는 상기 새 버전과 관련된 모든 데이터 블록들에 대한 순서 리스트를 얻기 위해, 상기 운영 체제의 새 버전의 각 레귤러 파일(regular file)에 대해 반복(iteration)들을 수행함-; 및 상기 프로세서로, 차등 업데이트 파일(differential update file)이 실제 업데이트 파일임을 나타내는 매직 넘버 표시자(magic number indicator), 운영 체계의 새 버전을 식별하는 새 버전 넘버, 및 상기 오퍼레이션들의 순서 리스트를 포함하는 프로토콜 버퍼(protocol buffer)를 포함하는 상기 차등 업데이트 파일을 어셈블링(assembling)하는 단계를 포함한다.

[0005] 일 예시에서, 상기 순서 리스트에 있는 하나 이상의 오퍼레이션 각각은 데이터의 묶음(a chunk of data)을 나타내는 각 데이터 블랍(data blob)과 관련된다. 여기서, 상기 차등 업데이트 파일은 상기 각 데이터 블랍을 포함하도록 어셈블링된다. 다른 예시에서, 상기 순서 리스트에 있는 하나 이상의 오퍼레이션 각각은 클라이언트 디바이스의 파티션(partition) 내의 하나 이상의 특정 데이터 블록(block)에 적용가능하다. 상기 오퍼레이션들은, 상기 파티션 내의 데이터 블록들 중 적어도 하나가 상기 운영 체계의 새 버전을 위한 상기 클라이언트 디바이스의 새 파티션 내의 다른 블록에 복사되는 복사 오퍼레이션(copy operation); 상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록이 메모리에 독출되고, 상기 차등 업데이트 파일의 데이터 블랍을 이용하여 상기 데이터 블록들 중 적어도 하나의 주어진 데이터 블록에 차등 루틴(difference routine)을 수행하는 차등 오퍼레이션(difference operation); 상기 차등 업데이트 파일의 선택된 데이터 블랍이 상기 새 파티션 내의 특정된 블록(specified block)들에 기록되도록 구성되는 대체 동작(replace operation); 및 압축된 데이터 블랍이 상기 차등 업데이트 파일 내에 포함되고, 그 압축된 데이터 블랍이 상기 운영 체계의 새 버전을 위한 새 파티션 내의 선택된 특정된 블록들에 쓰여지도록 구성되는 압축 해제 오퍼레이션으로의 대체(replace) 중 하나 이상을 포함한다.

[0006] 일 대안예로서, 상기 순서 리스트에 있는 각 오퍼레이션은 파일 오브젝트(file object)와 관련되고, 상기 방법은 각 파일 오브젝트를 위한 그래프에서 정점(vertex)을 생성하는 단계; 및 각 블록을 나타내는 벡터(vector)를 생성하는 단계를 더 포함한다. 일 변형예에서, 상기 방법은 각 블록의 벡터에 대한 리더(reader)와 라이터(writer) 파라미터를 설정하는 단계; 및 다른 리더와 라이터를 갖는 각 블록에 대하여, 상기 그래프에 상기 라이터로부터 리더까지의 에지(edge)를 생성하는 단계를 더 포함함다. 상기 에지는 자신과 관련된 소스 파일 오퍼레이션(source file operation)이 시작하기 전에 완료되어야 할 파일 오퍼레이션을 가리킨다. 예시에서, 각 에지는 가중치(weight)를 갖고, 상기 가중치는 상기 그래프에서 각 에지와 관련된 블록들의 개수를 식별한다. 다른 예시에서, 상기 그래프가 순환(cycle)들을 포함하면, 상기 방법은 각 순환을 끊는 단계를 더 포함한다. 이 경우, 상기 순환들 중 주어진 하나를 끊는 단계는 상기 주어진 순환과 관련된 최저 가중치 에지(a lowest-weight edge)를 찾는 단계; 및 상기 최저 가중치 에지를 잘라내는 단계를 포함할 수 있다. 여기서, 상기 최저 가중치 에지를 잘라내는 단계는 익스텐트(extent)를 스크래치 공간(scratch space)에 복사하는 오퍼레이션을 나타내는 새 노드를 생성하는 단계; 및 상기 최저 가중치 에지의 소스 노드(source node)로 하여금 상기 새 노드를 가리키도록 하는 단계를 포함할 수 있다. 그리고 새 노드는 새 복사 오퍼레이션이 상기 새 복사 오퍼레이션의 커스터머(customer) 전에 행해진다는 것을 보장하기 위해 상기 최저 가중치 에지의 목적 노드(destination node)로부터 만들어질 수 있다.

[0007] 본 발명의 다른 실시예에 따라서 컴퓨터 판독가능 운영 체제를 위한 업데이트를 생성하는 디바이스가 제공된다. 상기 디바이스는 상기 운영 체제와 관련된 차등 업테이트 정보를 저장하기 위한 메모리; 및 상기 메모리에 결합된 프로세서를 포함한다. 상기 프로세서는 상기 운영 체제의 현재 버전의 버전 넘버를 식별하고; 상기 차등 업데이트 정보를 사용하여, 상기 운영 체계의 현재 버전을 상기 운영 체계의 새 버전으로 업데이트하기 위한 오퍼레이션들의 순서 리스트를 생성하고-여기서, 상기 오퍼레이션들의 순서 리스트는 상기 새 버전과 관련된 모든 데이터 블록들에 대한 순서 리스트를 얻기 위해, 상기 운영 체제의 새 버전의 각 레귤러 파일에 대해 반복들을 수행하는 것을 포함함-; 차등 업데이트 파일이 실제 업데이트 파일임을 나타내는 매직 넘버 표시자, 상기 운영 체계의 새 버전을 식별하는 새 버전 넘버, 및 상기 오퍼레이션들의 순서 리스트를 포함하는 프로토콜 버퍼를 포함하는 상기 차등 업데이트 파일을 어셈블링하고; 및 상기 차등 업데이트 파일을 상기 메모리에 저장하도록 구

성된다.

[0008] 일 예시에서, 상기 순서 리스트에 있는 각 오퍼레이션은 파일 오브젝트와 관련되고, 상기 프로세서는 각 파일 오브젝트를 위한 그래프에서 정점을 생성하고, 각 블록을 나타내는 벡터를 생성하도록 더 구성된다. 대안예에서, 상기 프로세서는 각 블록의 상기 벡터에 대한 리더 및 라이터 파라미터를 설정하는 수단; 및 다른 리더 및 라이터를 갖는 각 블록에 대하여, 상기 그래프에서 상기 라이터로부터 상기 리더까지의 에지를 생성하도록 더 구성된다. 상기 에지는 자신과 관련된 소스 파일 오퍼레이션(source file operation)이 시작하기 전에 완료되어야 할 파일 오퍼레이션을 가리킨다. 이 경우, 상기 그래프가 순환들을 포함하면, 상기 프로세서는 각 순환을 끊도록 선택적으로 동작가능하다. 여기서, 상기 순환들 중 주어진 하나를 끊는 것은 상기 주어진 순환과 관련된 최저 가중치 에지를 찾는 단계; 및 상기 최저 가중치 에지를 잘라내는 단계를 포함할 수 있다. 이 경우, 상기 프로세스가 익스텐트를 스크래치 공간에 복사하는 오퍼레이션을 나타내는 새 노드를 생성하고, 상기 최저 가중치 에지의 소스 노드로 하여금 상기 새 노드를 가르키도록 함으로써, 상기 최저 가중치 에지를 잘라내도록 동작가능할 수 있다. 새 노드는 새 복사 오퍼레이션이 상기 새 복사 오퍼레이션의 커스터머 전에 행해진다는 것을 보장하기 위해 상기 최저 가중치 에지의 목적 노드로부터 만들어질 수 있다.

[0009] 또 다른 실시예에서, 유형의 컴퓨터 판독가능 저장 매체가 컴퓨터 프로그램의 컴퓨터 판독가능 인스트럭션들을 저자안다. 이 인스트럭션들은 컴퓨터에 의해 실행되었을 때, 상기 컴퓨터로 하여금 컴퓨터 판독가능 운영 체제를 위한 업데이트를 생성하는 방법을 수행하게 한다. 상기 방법은 상기 운영 체제의 현재 버전의 버전 넘버를 식별하는 단계; 프로세서로, 상기 운영 체계의 현재 버전을 상기 운영 체계의 새 버전으로 업데이트하기 위한 오퍼레이션들의 순서 리스트를 생성하는 단계－여기서, 상기 프로세서는 상기 새 버전과 관련된 모든 데이터 블록들에 대한 순서 리스트를 얻기 위해, 상기 운영 체제의 새 버전의 각 레귤러 파일에 대해 반복들을 수행함－; 및 상기 프로세서로, 차등 업데이트 파일이 실제 업데이트 파일임을 나타내는 매직 넘버 표시자, 운영 체계의 새 버전을 식별하는 새 버전 넘버, 및 상기 오퍼레이션들의 순서 리스트를 포함하는 프로토콜 버퍼를 포함하는 상기 차등 업데이트 파일을 어셈블링하는 단계를 포함한다.

[0010] 일 예시에서, 상기 순서 리스트에 있는 각 오퍼레이션은 파일 오브젝트와 관련되고, 상기 방법은 각 파일 오브젝트를 위한 그래프에서 정점을 생성하는 단계; 및 각 블록을 나타내는 벡터를 생성하는 단계를 더 포함한다. 이 경우, 상기 방법은 각 블록의 벡터에 대한 리더(reader)와 라이터(writer) 파라미터를 설정하는 단계와; 다른 리더와 라이터를 갖는 각 블록에 대하여, 상기 그래프에 상기 라이터로부터 리더까지의 에지를 생성하는 단계를 더 포함할 수 있다. 상기 에지는 자신과 관련된 소스 파일 오퍼레이션이 시작하기 전에 완료되어야 할 파일 오퍼레이션을 가리킨다. 여기서, 상기 그래프가 순환들을 포함하면, 상기 방법은 상기 주어진 순환과 관련된 최저 , 상기 최저 가중치 에지를 잘라냄으로써, 상기 순환들 각각을 끊는 단계를 더 포함할 수 있다. 상기 최저 가중치 에지를 잘라내는 단계는 익스텐트를 스크래치 공간에 복사하는 오퍼레이션을 나타내는 새 노드를 생성하는 단계; 및 상기 최저 가중치 에지의 소스 노드로 하여금 상기 새 노드를 가리키도록 하는 단계를 구비할 수 있다. 그리고 새 복사 오퍼레이션이 상기 새 복사 오퍼레이션의 커스터머 전에 행해진다는 것을 보장하기 위해 상기 최저 가중치 에지의 목적 노드로부터 만들어질 수 있다.

[0011] 또 다른 실시예에서, 클라이언트 디바이스는 운영 체제의 현재 버전을 저장하기 위한 메모리; 및 상기 메모리에 결합된 프로세서를 포함한다. 상기 프로세서는 상기 운영 체제의 현재 버전을 위한 업데이트에 관한 요청을 원격 디바이스에 전송하고－여기서, 상기 요청은 상기 운영 체제의 현재 버전을 식별하는 버전 넘버를 포함함－; 상기 원격 디바이스로부터 차등 업데이트 파일을 수신하고－여기서, 상기 차등 업데이트 파일은 자신이 실제 업데이트 파일임을 나타내는 매직 넘버 표시자와, 상기 운영 체계의 새 버전을 식별하는 새 버전 넘버, 및 상기 오퍼레이션들의 순서 리스트를 포함하는 프로토콜 버퍼를 포함함－; 상기 매직 넘버를 검증하고; 상기 프로토콜 버퍼로부터 상기 오퍼레이션들의 순서 리스트를 추출하고; 상기 운영 체제의 현재 버전을 상기 운영 체제의 새 버전으로 업데이트하기 위해, 상기 오퍼레이션들의 순서 리스트를 실행함으로써 차등 업데이트를 수행하도록 구성된다.

[0012] 일 예시에서, 상기 프로세서는 상기 차등 업데이트 파일이 상기 클라이언트 디바이스로 스트리밍되는 동안, 상기 차등 업데이트 파일을 메모리에 저장하지 않고, 상기 운영 체제의 새 버전으로 상기 운영 체제의 상기 차등 업데이트를 수행하도록 동작가능하다.

발명의 효과

[0013] 본 발명은 컴퓨터 시스템 오퍼레이션과 이러한 컴퓨터 시스템을 위한 애플리케이션의 업데이트를 포함한, 그러

나 이에 한정되지는 않는 광범위한 산업상의 이용가능성을 가진다.

도면의 간단한 설명

[0014] 도 1은 본 발명의 양태에 따른 파일 포맷을 예시한다.

도 2는 본 발명이 양태에 따라서 순환(cycle)을 끊기 위해 에지들을 잘라내기 위한 프로세스이다.

도 3은 본 발명의 양태에 따라서 차등 업데이트 프로세스를 예시한다.

도 4a와 도 4b는 본 발명에서 사용하기 위한 컴퓨터 시스템들을 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0015] 본 발명의 양태, 특징, 및 효과는 바람직한 실시예들과 첨부 도면들에 대해 후술하는 상세 설명을 참조하여 고려될 때, 인정될 것이다. 다른 도면들에서의 동일한 참조 번호는 동일하거나 또는 유사한 구성요소를 식별할 수 있다.

[0016] 때때로 운영 체제의 다른 특징들이 업데이트된다. 이러한 업데이트는 예정된 업데이트 프로세스를 실행하는 클라이언트 컴퓨터로 전송될 수 있다. 이것은 클라이언트에 대해 파일들을 삭제, 변경, 및/또는 추가하는 것을 포함할 수 있다. 하나의 프로세스에서, 업데이트는 운영 체제 공급자(operating system provider)에 의해 준비된다. 이것은 클라이언트에게 행해져야 할 변경 내용을 지시하는 차등 업데이트 파일을 생성하는 것을 포함할 수 있다. 이어, 차등 업데이트 파일은 각 클라이언트에게 전송되어 업데이트된다. 클라이언트들은 업데이트를 수행하기 위해 그 파일에 있는 오퍼레이션들을 순차적으로 실행한다.

[0017] 운영 체제를 저장하고 있는 파티션은 하나 이상의 블록(block)으로 구성될 수 있으며, 각 블록은 파일 정보 또는 부기 정보(bookkeeping information)를 저장하고 있다. 각 블록은 예컨대 4k 바이트(byte) 일 수 있으나, 본 발명은 이에 한정되거나 다른 특정된 블록 크기(size)에 한정되지 않는다. 운영 체제는 그 자체에 컴퓨터 프로세서에서 실행되는 데이터, 컴퓨터 하드웨어 리소스(resource)들을 관리하는 데이터, 및 다양한 애플리케이션 소프트웨어의 효율적인 실행을 위한 통상 서비스들을 제공하는 데이터 뿐만 아니라, 프로그램들(예컨대, 인스트럭션)을 포함한다. 운영 체제를 업데이트할 때, 새 파티션이 생성될 수 있다. 새 파티션은 운영 체제의 기존 버전으로 사전 형성(pre-populated)된다. 결과로서 생기는 인스톨 파티션 상의 각 디스크 블록(루트 파일 시스템을 포함함)은 운영 체제 밴더에 의해 바람직하게는 정확하게 특정되어(비트 대 비트(bit for bit)), 서버상에 표시(예를 들어, 검증된 부트용)될 수 있게 된다.

[0018] 업데이트는 가능한 작게 되는 것이 바람직하다. 하나의 시나리오에서, 많은 델타(차등) 업데이트들이 리부팅(rebooting)없이 인스톨(install) 될 수 있도록, 업데이트들이 적정 위치에 적용된다. 따라서 만약 사용자가 버전 N으로 부팅되고, N+1이 발행(release)되면, 해당 사용자(클라이언트)는 N -> N+1 업데이트를 다운로드하고, 그것을 인스톨한다. 이어, 사용자는 N+2가 새로 발행될 때까지 계속해서 N으로 부팅된다. 이어, 사용자는 N+1 -> N+2 업데이트를 다운로드하고, 그것을 N+1의 적소에 인스톨한다.

[0019] 일 예시에서, 클라이언트는 업데이트 서버에 접속하여, 현재의 인스톨 파티션에 인스톨된 시스템의 버전 넘버를 제공한다. 서버는 클라이언트에게 증분(incremental)(델타) 업데이트를 제공하며, 그 증분 업데이트는 클라이언트에게 다운로드된다. 델타 업데이트 파일(delta update file)은 구(존재하는) 버전으로부터 새 버전을 취하는 인스톨 파티션에서 수행할 오퍼레이션들의 순서 리스트를 포함한다.

[0020] 바람직하게, 업데이트 파일은 클라이언트에 의해 수행될 오퍼레이션들의 순서 리스트이다. 각 오퍼레이션은 파티션의 특정된 블록들에서 동작한다. 각 오퍼레이션은 델타 파일(delta file) 내부에 선택 데이터(optional data) "블랍(blob)"을 포함할 수 있다. 용어 "블랍"은 바이트(byte)들의 무작위적(arbitrary) 집합을 가질 수 있는 데이터의 대형 묶음(large chunk)을 가리킨다. 여기에는 "복사"(Copy), "구별"(Diff), "대체"(Replace), 및 "압축 해제으로 대체"(Replace with Uncompression)를 포함하는 몇 가지 유형의 인스톨 오퍼레이션들이 있다. "복사"에서는, 현재 인스톨 파티션 내의 일부 블록들이 새 인스톨 파티션 내의 다른 블록들로 복사된다. "구별"의 경우에는, 일부의 블록들이 메모리로 독출되고, 이진 차등 루틴(binary difference routine)이 첨부된 데이터 블랍을 이용하여 수행된다. 그 결과들은 새 인스톨 파티션 내의 특정된 블록들에 기록된다. "대체"의 경우에는, 상기 첨부된 데이터 블랍은 새 인스톨 파티션 내의 특정된 블록들에 기록된다. 압축 프로세스가 실행될 수도 있는데, 압축된 데이터 블랍이 클라이언트에게 전송되는 경우에는 압축 해제(uncompress)이 수행되어, 그 결과가 새 인스톨 파티션 내의 특정된 블록들에 기록된다. 여기서, Gzip, Bzip2, 또는 그 밖의 다른 압축 알고

리즘이 사용될 수 있다.

[0021] 일 시나리오에서, 패치 프로그램(patch program)은 구 파일이 인스톨 파티션이라는 것을 지시받는다. 전체 파티션을 메모리로 독출하는 프로그램을 갖기 보다는, 파일을 가져오기 위해 메모리에서 어떤 블록을 판독할 것인지가 지시된다. 이어, 패치 오퍼레이션이 메모리에서 수행된다. 최종적으로, 그 결과가 디바이스의 시작 시점을 제외하고, 새 인스톨 파티션에 바로 기록된다. 대신에, 이 프로그램은 결과를 어떤 블록에 기록할지를 지시받는다.

[0022] 델타/차등 업데이트 파일이 후술의 논의에 따라서 생성될 수 있다. 업데이트 파일 포맷은 도 1의 파일 포맷(100)에 도시된 바와 같을 수 있다. 특히, 그 포맷은 바람직하게 "매직 넘버" 표시자 또는 해당 파일이 실제적으로 업데이트 파일임을 보여주는 온건성 체크(sanity check)를 포함한다. 일 경우에, 매직 넘버는 "CrAU"와 같은 단구(short phrase)의 2진 버전을 갖는 4바이트를 포함한다. 다음으로 각각이 8 바이트일 수 있는 버전 넘버와, 프로토콜 버퍼 오프셋(protocol buffer offset) 및 길이(length)가 이어진다. 프로토콜 버퍼가 그 뒤에 이어진다. 프로토콜 버퍼는 클라이언트가 순서에 따라 수행해야 할 일련의 인스트럭션들이다. 다음으로, 하나 이상의 데이터 블랍과 파일 끝 표시자(EOF;end of file)가 이어서 제공된다. 순서를 부여하는 것이 이 도면에 도시된 바와 같을 수 있지만, 다른 구성 요소들이 필요에 따라 다르게 배열될 수도 있다.

[0023] 일 시나리오에서, "익스텐트"의 개념이 사용된다. 익스텐트는 디스크 블록들의 인접 범위이다. 예를 들면, 블럭들{10, 11, 12, 13, 14, 15, 17, 18}을 특정하기보다는, 이것들이 {(10, 6), (17, 2)} (익스텐트들의 리스트)를 특정하기 위해 더 간단해질 수 있다. 델타 업데이트들을 생성하기 위한 예시적 프로세스는 다음과 같다.

[0024]
```
message Manifest {
```

[0025]
```
message InstallOperation {
```

[0026]
```
        enum CompressionType {
```

[0027]
```
                NO_CHANGE = 0, // file is unchanged; just move data
```

[0028]
```
                        blocks
```

[0029]
```
                BSDIFF = 1, // Read source data blocks as old file,
```

[0030]
```
                        included binary blob is diff, output to new blocks
```

[0031]
```
                FULL = 2, // Output included binary blob to new
```

[0032]
```
                        blocks
```

[0033]
```
                FULL_BZIP = 3 // Bunzip binary blob into new blocks
```

[0034]
```
}
```

[0035]
```
        uint64 blob_offset; // if present, the offset in the update image
```

[0036]
```
                of the binary blob for this file
```

[0037]
```
        uint64 blob_length; // if present, binary blob length
```

[0038]
```
                message extent {
```

[0039]
```
                uint64 offset; // in blocksize
```

[0040]
```
                uint64 length; // in blocksize
```

[0041]
```
}
```

[0042]
```
        repeated extent input_extents;
```

[0043]
```
        repeated extent output_extents;
```

[0044]
```
}
```

[0045]
```
repeated InstallOperation install_operations;
```

[0046]
```
}
```

[0047] 리스트(manifest)은 인스톨 오퍼레이션 리스트를 나타낸다. 입력 익스텐트들은 독출될 블록을 나타낸다. 출력 익스텐트들은 기록될 블록을 나타낸다.

[0048] 델타 업데이트를 생성하기 위하여, 반복들이 새 파일 시스템상의 각 레귤러 파일에 수행되어, 각 레귤러 파일이 갖는 모든 데이터 블록의 순서 리스트를 얻는다. 얻어진 순서 리스트는 다음의 포맷을 갖는 파일 스트럭처(file structure)에 저장된다.

[0049] struct Extent {

[0050] uint64 start;

[0051] uint64 length;

[0052] }

[0053] struct File {

[0054] string path; // path within the filesystem

[0055] vector<Extent> dst_extents; // ordered list of all

[0056] extents on the new filesystem

[0057] vector<Extent> src_extents; // Applies only for

[0058] NO_CHANGE and BSDIFF

[0059] enum CompressionType; // one of: NO_CHANGE, BSDIFF,

[0060] FULL, FULL_BZIP

[0061] }

[0062] 결국, 각 파일 오브젝트는 프로토콜 버퍼 내의 인스톨오퍼레이션(InstallOperation) 메시지로 변환될 것이다. 각 파일에 대하여, 그것을 압축할 최적의 방식을 결정하는 것이 바람직하다. 여기에는 4가지 경우가 있다. 한 경우에는, 파일이 변경되지 않았다. 다른 세 개의 경우에는, 파일이 변경되었다. 일 시나리오에서, 변경된 파일은 Bzip2 또는 다른 압축 알고리즘을 사용하여 가장 작은 크기로 압축될 수 있다. 다른 시나리오에서는, 파일이 변환되었고, 그 파일이 가장 작게 압축 해제되었다. 세 번째 시나리오에서는, 파일이 변경되었고, 그 파일이 가장 작은 2진-딥스(binary-diffs)이다. 만일 파일이 새 것이라면, 즉 그 파일이 운영 체제의 구 버전에 존재하지 않았음을 의미하면, 상기 경우들 중 두 가지만 적용한다. 데이터는 전부 전송될 수 있고, 또는 압축되어 전부 전송될 수 있다.

[0063] 본 발명의 일 양태에 따르면, 정점이 각 파일 오브젝트를 위해서 그래프에 생성된다. 그래프의 변(side)을 따라서 벡터가 인스톨 파티션 내에는 각 블록(block)을 표현하기 위해 생성된다.

[0064] struct Block {

[0065] InstallOperation* reader;

[0066] InstallOperation* writer;

[0067] }

[0068] vector<Block> blocks; // length is the size of the

[0069] install partition

[0070] 이어, 프로세스는 각 파일 오브젝트 내의 각 블록에 걸쳐 진행된다. 각 블록에 대하여, 리더 및 라이터 파라미터들이 상기 블록의 벡터를 위해 설정된다.

[0071] 다음, 반복들이 블록의 어레이(array)를 통하여 수행된다. 다른 리더와 라이터를 갖는 각 블록에 대하여(둘 다 넌-널(non-null)임), 에지(화살표)가 라이터에서 리더까지 그래프 상에 생성된다. (지시된)그래프에 있는 에지(edge)는 반드시 해당 에지(edge)의 소스(source) 파일 오퍼레이션이 시작하기 전에 완료되어야 하는 파일 오퍼레이션을 가리킨다. 따라서 이 프로세스는 만약 블록에 다른 파일 오퍼레이션들에 의해 판독(read)과 기록(write)이 모두 수행되는 경우, 그 블록은 기록되기 전에 판독된다는 것을 보장하기 위해 행해진다. 에지가 그

래프 상에서 블록을 나타내므로, 에지의 가중치(edge's weight)는 블록들의 개수이다.

[0072] 이 점에서, 결과는 순환들을 갖는 그래프일 가능성이 크다. 순환들은 깨어져야 한다. 상기 순환들은 1975년 3월, 시엠 제이.콤풋.(SIAM J. Comput.)에 의해 "지시된 그래프의 제반 기본 회로 조사"(vol.4 no.1)으로 발표된 도널드 B. 존슨의 회로 조사 알고리즘(circuit finding algorithm)을 이용하여 찾아질 수 있는데, 상기 문헌의 모든 내용은 본 명세서에 참조로서 포함된다. 또한, 순환은 데이비드 앱스테인(David Eppstein)에 의해 "타잔의 강건하게 연결된 구송 요소 알고리즘"(Ed., Wikipedia)으로 발표된 타잔 알고리즘(Tarjan's Algorithm)을 이용하여 찾아질 수 있는데, 본 문헌에 있는 모든 내용은 본 명세서에 참조로서 포함된다. 각 순환에 대하여, 최저 가중치 에지가 발견되어 잘린다. 에지는 다음과 같이 잘릴 수 있다. 먼저, 일부 익스텐트들을 스크래치 공간으로 복사하는 오퍼레이션을 나타내는 새 노드(new node)를 만든다. 이어, 에지의 소스 노드가 새 노드를 가리키게 한다. 여기서, 에지는 잘린 노드의 새 노드로의 목적 노드로부터 만들어져, 새 복사 오퍼레이션이 상기 복사의 커스터머 전에 일어나게 한다. 바람직하게, 잘려지는 에지에 의해 표현되는 블럭들로부터 판독되기 보다는, 스크래치 공간으로부터 판독되도록 잘린 에지의 목적 노드를 변경한다.

[0073] 순환을 끊기 위해 에지를 잘라내는 예시가 도 2의 프로세스(200)에 도시되어 있다. 화살표들은 오퍼레이션이 가리키고 있는 오퍼레이션은 그 오퍼레이션 전에 행해져야 한다는 것을 가리키고, 포인팅 오퍼레이션이 포인팅된 대상이 요구하는 데이터를 덮어쓸 수 있기 때문에 그 순서로 일어나야 한다는 것을 가리키는 오퍼레이션을 나타낸다. 이 도면에 도시된 바와 같이, S202에서 오퍼레이션 A가 블록 4에 기록하기 위해 블록 3을 판독한다. 그리고 S204에서 오퍼레이션 B는 블록 10에 기록하기 위해 블록 4를 판독한다. 상기 순환은 A와 B 사이의 에지(화살표(206))를 자름으로써 끊어지는데; 이것은 임시 블록(temp block) T를 이용함으로써 완료될 수 있다. 이로서, 도시된 바와 같이, S208에서 오퍼레이션 A는 블록 3을 판독하고 블록 4를 기록한다. S210에서, 동작 C는 블록 4를 판독하고, 블록 T를 기록한다. 그리고 S212에서, 오퍼레이션 B'는 블록 T를 판독하고, 블록 10을 기록한다. 상기 순환이 끊어지면, 토폴로지화 정렬(topological sort)이 모든 노드들을 정리하기 위하여 사용될 수 있다. 이것은 모든 파일-데이터 블록들의 인스톨레이션(installation)을 포함한다.

[0074] 일 예시에서, 한 에지를 선택하고 그 선택된 에지를 B'->C->A->Temp 와 같이, 자를 수 있다. 이 경우, 임시 오퍼레이션은 블록 A를 임시 영역으로 복사할 것이다. B는 이후 임시 영역으로부터 판독할 수 있는 B'로 변경될 것이다. 그러나 만약 두 번째 에지(예컨대, C -> A)가 잘리면, 그 결과는 B'->C->Temp2 및 A'->Temp가 될 것이다. 이것은 두 개의 완전히 독립적인 그래프로 도출될 것이며, 오퍼레이션의 최종 순서는 다음과 같이 배열되었을 것이다.

[0075] Temp2 (클라이언트가 이것을 가장 먼저 실행함)

[0076] C

[0077] B'

[0078] Temp

[0079] A' (클라이언트는 이것을 가장 마지막으로 실행함)

[0080] 그러나 이 경우, B'는 임시 오퍼레이션에 의해 기록된 데이터를 판독할 것이지만, 임시는 B' 이후에 일어난다. 이것은 A->B가 잘리는 에지일 때 변형이 완료된 그래프가 임시가 반드시 B' 전에 일어나야 한다고 구체적으로 명시하지 않기 때문에, 발생할 수 있다. 이러한 상황을 회피하고, 이를 명확히 하기 위하여 다른 에지가 추가된다. 그 결과는 아래와 같다.

[0081] A->B->C->A (original)

[0082] cut A->B to obtain:

[0083] B'->C->A->Temp<-B'

[0084] Then, cut C->A to obtain:

[0085] B'->C'->Temp2<-A->Temp<-B'

[0086] 이제, 오퍼레이션의 마지막 순서가 그래프에 기초하여 선택되었다면, 그 오퍼레이션들은 정확한 순서로 행해질 것이다.

[0087] 예시적인 프로세스가 도 3에 도시되어 있다. 단계 1에서, 디스크 이미지들이 스캔된다. 블록 인덱스들 0 ~ 7은

다른 이미지들(예컨대, "sh," "foo," "bar," 및 "dog")과 관련될 수 있다. 새 이미지들은 동일 또는 다른 블록들과 관련될 수 있다. 따라서 "foo"는 이제 블록 4 대신에 블록 2-3과 관련될 수 있다. "Sh"는 블록 2-3 대신에 블록 4-5와 관련될 수 있다. "Bar"는 블록 5 대신에 블록 6과 관련될 수 있다. 그리고 새 이미지 "cat"은 블록 7과 관련될 수 있다.

[0088] 단계 2에서, 파일 오퍼레이션들이 생성된다. 이는 새 파티션상의 파일에 상응한다. 이것 이후에, 블록 벡터가 생성된다. 블록 벡터는 순서 리스트이고, 이 순서 리스트의 각 요소는 주어진 블록에 대한 파일 오퍼레이션을 포함한다. 각 이미지는 하나 이상의 소스 블록(source block)과 하나 이상의 목적지 블록(destination block)을 가질 수 있다. 따라서 "foo"에 대한 소스 블록은 블록 4이고, "foo"의 목적지 블록들은 "dst: (2,2)"와 같이 도시되는데, "dst: (2,2)"는 첫번째 블록은 블록 2이고 2개의 블록이 할당되어 있음을 나타낸다. 이 부분은 또한 각 이미지가 클라이언트에게 어떻게 제공될 수 있는지를 보여준다. 따라서 "foo"와 "bar"는 bsdiff와 같은 이진 차등 알고리즘(binary difference algorithm)을 이용하여 제공될 수 있으며, "sh"는 단지 복사될 수 있고, "cat"은 압축 없이 전부 전송될 수 있다. Bsdiff는 두 파일 간의 차이(different)를 산출하는 알려진 알고리즘이다.

[0089] 단계 3에 도시된 바와 같이, 에지 가중치(예컨대, 1 또는 2)가 블록들의 개수와 같은 그래프가 생성된다. 단계 4에서, 순환들이 끊어진다. 바람직하게, 순환(들)은 각 순환 내에서 최저 가중치 에지를 자름으로써 끊어진다. 따라서 이 예시에서는, "sh"부터 "foo"까지의 화살표를 포함하는(가중치 1을 갖음) 순환이 끊어지고, 변경된 foo(foo')가 얻어진다. 그리고 단계 5에서, 토폴로지화 정렬로부터 도출된 최종 순서가 제시된다. 이것은 클라이언트 측에서 실행되는 순서이다.

[0090] 파일 데이터를 세팅한 후, 클라이언트는 남겨진 데이터를 모든 비-파일-데이터 블록(non-file-data block)들 내에 압축 해제하는 최종 인스톨 오퍼레이션(final InstallOperation)으로 비-파일-데이터 블록들을 덮어쓸 것이다. 전형적예시에서, 이것은 압축된 약 2메가바이트(megabyte) 데이터이다. 이 데이터에 델타 압축 프로세스를 수행할 수도 있다.

[0091] 프로토콜 버퍼(모든 오퍼레이션들을 리스트함)는 파일의 시작 부분에 행해지기 때문에, 업데이트가 디스크에 저장될 필요는 없다. 이는 서버로부터 스트리밍하는 동안 적용될 수 있다. 업데이트가 운영 체제 밴더에 의해 서명되었는지 확인하는 것이 바람직하다. 시스템이 업데이트의 적용하고, 델타 업데이트 서명이 검증된 이후가 될 때까지 그것을 부팅할 수 있다고 나타내기 시작할 수 있다.

[0092] 본 명세서에 기재된 실시예들은 실제로 최고 압축율을 제공하도록 발견되었다. 이것은 또한 운영 체제 밴더가 미래에 대안적인 압축 스킴(scheme)을 사용하는 것을 허용한다. 예를 들면, 구글이 개발한 프로세스로서 코제트(Courgette)가 있는데, 이는 bsdiff를 대체할 수 있다.

[0093] 다른 가능한 솔루션은 파티션 전체를 델타-압축(delta-compress)하는 것이다. 그러나 bsdiff는 그것의 메모리 요구 사양(memory requirements)이 너무 높기 때문에, 이 시나리오에서는 실행 불가할 것이다. 패칭(patching)하는 동안, bsdiff는 원본과 새 파일들을 저장하기 위해 1기가 바이트(gigabyte)를 초과할 수 있는 충분한 메모리를 필요로 한다. 또다른 델타 압축 프로그램인 Xdelta는 슬라이딩 윈도우(sliding window)를 사용한다. 테스트에서, 이것은 좋지 못한 압축 결과(예컨대, 수백 메가바이트)를 나타냈다. 또 다른 대안은 rdiff를 사용하는 것인데, rdiff는 델타 파일 내에 오직 변화된 블록들만을 저장함으로써 동작한다. 이것은 슬라이딩 윈도우를 사용하므로, 블록들이 배열될 필요가 없다. 테스트할 때, 전체 파티션에 대한 rdiff 델타는 104메가바이트였다. 이것은 상술한 절차에 따라서 사용될 수 있는 약 10메가바이트를 상당히 상회한다.

[0094] 향후, 파일 레벨로 rdiff(즉, rsync-) 스타일 델타 압축을 사용하는 것이 가능할 수 있다. 이것은 향후 bsdiff와 함께 사용될 수도 있다.

[0095] 본 발명의 양태에 따른 업데이트 절차는 후술하는 예시적 컴퓨터 시스템을 구현될 수 있다. 도 4a는 본 발명에서 단독으로 또는 네트워크로 연결된 구성으로 사용될 수 있는 다양한 컴퓨팅 디바이스를 묘사하는 컴퓨터 시스템의 개념도를 나타낸다. 예를 들어, 이 도면은 휴대용 전자 디바이스(예컨대, 휴대전화기(310)과 PDA(312))와 같은 다른 유형의 디바이스 뿐만 아니라, 복수의 컴퓨터(302, 304, 306, 및 308)가 있는 컴퓨터 네트워크(300)를 예시한다. 그러나 본 발명은 이에 한정되지 않으며, 넷북(netbooks) 및 패드-타입 휴대용 컴퓨터(pad-type handheld computers, 도시되지 않음)를 포함하는 다른 디바이스들도 사용될 수 있다. 이러한 디바이스들은 로컬 또는 직접 접속(314)을 통해 상호연결되거나, 및/또는 유선 및 무선일 수 있는 LAN, WAN, 인터넷 등과 같은 통신 네트워크(316)를 통해 접속될 수 있다.

[0096] 각 디바이스는 예컨대 하나 이상의 프로세싱 디바이스를 포함할 수 있고, 예를 들어, CRT, LCD, 플라즈마 스크린 모니터(plasma screen monitor), TV, 프로젝터 등을 포함할 수 있는 디스플레이(322) 뿐만 아니라, 사용자 입력기(예컨대, 키보드(318)와 마이스(320)) 및/또는 다양한 다른 유형의 입력 디바이스(예컨대, 펜-입력기(pen-inputs), 조이스틱(joysticks), 버튼(buttons), 터치 스크린(touch screens) 등)을 구비할 수 있다. 컴퓨터(302, 304, 306 및 308) 각각은 개인용 컴퓨터, 서버 등일 수 있다. 단지 예시로서, 컴퓨터(302와 306)는 개인용 컴퓨터일 수 있고, 컴퓨터(304)는 서버일 수 있고, 컴퓨터(308)는 랩탑(laptop)일 수 있다.

[0097] 도 4b에 도시된 바와 같이, 컴퓨터(302,304)와 같은 각 컴퓨터는 프로세서(324), 메모리/저장소(326) 및 컴퓨터에서 일반적으로 보여지는 다른 구성요소들을 포함한다. 예컨대, 메모리/저장소(326)는 프로세서(324)에 의해 액세스가능한 정보를 저장하는데, 그 정보는 프로세서(324)에 의해 실행될 수 있는 인스트럭션들(328)과 그 프로세서에 의해 검색(rertive), 조작(manipulate), 또는 저장될 수 있는 데이터(330)를 포함한다. 서버에서 그 인스트럭션들(328)은 하나 이상의 클라이언트 디바이스에 의해 인스톨될 차등 업데이트(differential update)를 생성하기 위한 오퍼레이션들을 포함할 수 있다. 그리고 클라이언트 디바이스에서 그 인스트럭션들은 차등 업데이트를 수행하기 위한 오퍼레이션들을 포함할 수 있다.

[0098] 상기 데이터는 클라이언트 디바이스로의 서비스를 위해 데이터베이스에서 유지관리되는 하나 이상의 차등 업데이트를 포함할 수 있다. 상기 메모리/저장소는 프로세서가 액세스 가능한 정보를 저장할 수 있는 모든 유형 또는 모든 디바이스(예컨대, 하드 드라이브, ROM, RAM, CD-ROM, 플래시 메모리(flash memory), 기록 가능(write-capable) 또는 판독 전용(read-only) 메모리)일 수 있다. 프로세서(324)는 임의 개수의 잘 알려진 프로세서(예컨대, 인텔 코포레이션(Intel Corporation)에서 출시된 프로세서들과 어드벤스드 마이크로 디바이스(Advanced Micro Device)들)을 포함할 수 있다. 대안적으로, 프로세서는 오퍼레이션을 수행하기 위한 전용 컨트롤러(예컨대, ASIC 또는 다른 프로세싱 디바이스)일 수도 있다.

[0099] 인스트럭션들(328)은 프로세서(들)에 의해 직접적으로 실행(예컨대, 기계 코드)되거나 간접적으로 실행(예컨대, 스크립트)될 임의의 인스트럭션 세트를 포함할 수 있다. 이러한 점에서, 용어들 "인스트럭션들", "단계들", 및 "프로그램들"은 본 명세서에서 서로 혼용하여 사용될 수 있다. 인스트럭션들은 오브젝트 코드(object code) 또는 소스 코드의 모듈들과 같이, 임의의 컴퓨터 언어 또는 포맷으로 저장될 수 있다. 본 발명에 따른 인스트럭션의 루틴들, 기능들, 및 방법들이 이하에서 보다 상세하게 설명된다.

[0100] 인스트럭션들(328)에 따라 프로세서(324)가 데이터(330)를 검색, 저장, 또는 변경할 수 있다. 데이터는 데이터 집합(collection of data)으로서 저장될 수 있다. 예를 들어, 본 발명은 임의의 특정 데이터 구조로 제안되지는 않지만, 상기 데이터는 컴퓨터 레지스터(computer register), 복수의 다른 필드(field)와 레코드(record)를 갖는 테이블(table)과 같은 상관 데이터베이스(relational database), XML 문서와 같은 웹 페이지 캐시(cache) 등에 저장될 수 있다.

[0101] 데이터는 또한 이진수 값(binary values), ASCII, 또는 유니코드(Unicode)와 같은(하지만 이에 제한되지 않음) 임의의 컴퓨터 판독가능 포맷으로 포맷될 수 있다. 또한, 상기 데이터는 관련 정보(예컨대, 설명 텍스트(descriptive text), 전용 코드(proprietary code), 포인터(pointer), 다른 메모리(다른 네트워크 위치들을 포함함)에 저장된 데이터에 대한 참조, 또는 그 관련 데이터를 계산하는 기능에 의해 사용되는 정보)를 식별하기에 충분한 임의 정보를 포함할 수 있다. 이에 더하여, 주어진 항목은 하나 이상의 파일, 데이터베이스에 저장된 데이터 세트(data set), 웹 캐시(web cache) 등을 포함할 수 있다. 상기 데이터의 크기와 내용에 기초하여, 그 데이터의 일부가 저장되거나 분리되어 유지관리될 수 있다.

[0102] 도 4b에서 프로세서(324)와 메모리(326)가 동일한 블록 내에 존재하는 것으로 기능적으로 도시되어 있지만, 이는 프로세서와 메모리가 동일한 물리적 하우징 또는 위치 내에 저장될 수도 있고, 저장되지 않을 수도 있는 다수의 프로세서와 메모리를 실질적으로 포함할 수 있다는 것으로 이해될 것이다. 예를 들면, 인스트럭션과 데이터의 일부 또는 모두는 삭제가능한 CD-ROM, DVD-ROM 또는 플래시 드라이브(flash drive), 및 판독-전용 컴퓨터 칩(read-only computer chip) 내에 있는 다른 어떤 것에 저장될 수 있다. 인스트럭션과 데이터의 일부 또는 전부는 물리적으로는 프로세서로부터 떨어진, 그러나 여전히 프로세서에 의해 액세스 가능한 위치에 저장될 수 있다. 마찬가지로, 프로세서는 동시에 동작할 수 있거나 동작 불가할 수 있는 프로세스들의 집합을 실제로 포함할 수 있다. 데이터는 배포될 수 있고, 다수의 메모리(326, 예컨대, 하드 드라이브들) 전역에 저장될 수 있다.

[0103] 일 양태에서, 서버(304)는 모바일 전화기(310)과 PDA(312)와 같은 디바이스 뿐만 아니라, 하나 이상의 클라이언트 컴퓨터(302, 306, 및/또는 308)와 통신할 수 있다. 각각의 클라이언트 컴퓨터 또는 다른 클라이언트 디바이스는 서버(304)와 유사하게, 하나 이상의 사용자 입력 디바이스(318, 320) 및 디스플레이(322)와 같은 사용자

출력 디바이스 뿐만 아니라, 프로세서, 메모리, 및 인스트럭션들로 구성될 수 있다. 각 클라이언트 컴퓨터는 개인에 의한 사용이 의도된 범용 컴퓨터일 수 있으며, 중앙 처리 장치("CPU"), 디스플레이, CD-ROM 또는 DVD 드라이브, 하드 드라이브, 마우스, 키보드, 터치 감지 스크린(touch-sensitive screen), 스피커, 마이크로전화기, 모뎀 및/또는 라우터(전화, 케이블 등)와 같이 개인용 컴퓨터에 일반적으로 있는 모든 구성요소와, 이러한 요소들이 서로 연결하기 위해 사용되는 모든 구성 요소를 포함한다.

[0104] 서버(304), 사용자 컴퓨터들, 및 그 밖의 디바이스들은 다른 컴퓨터들과 직접 통신 및 간접 통신(예컨대, 네트워크(16)를 통해)할 수 있다. 불과 몇 개의 컴퓨팅 디바이스들만이 도 4a와 도 4b에 묘사되어 있지만, 전형적인 시스템은 네크워크의 다른 노드에 존재하는 각기 다른 컴퓨터로, 많은 수의 연결된 서버과 클라이언트를 포함할 수 있다는 것이 인식되어야만 한다. 네트워크(316)와 중간의 노드들은 다양한 구성(configuration)과 프로토콜을 가질 수 있는데, 그 프로토콜들로는 인터넷, 인트라넷, 가상 사설망(virtual private networks), 광역 통신망(wide area networks), 로컬 네트워크, 하나 이상의 기업에 소속된 통신 프로토콜들을 사용하는 사설 네트워크들, 이더넷, WiFi, 블루투스, 또는 TCP/IP가 있다.

[0105] 임의의 중간 노드를 포함하는 네트워크상의 통신은 다른 컴퓨터들과 데이터를 송수신할 수 있는 임의의 디바이스에 의해 가능해질 수 있는데, 이 디바이스로는 모뎀(예컨대, 다이얼-업(dial-up) 또는 케이블), 네트워크 인터페이스, 및 무선 인터페이스가 있다. 서버(304)는 웹 서버일 수 있다. 상술된 것처럼 정보가 전송되거나 수신될 때 특정 장점들이 얻어질 수도 있지만, 본 발명의 그 밖의 양태들은 정보 전송에 대해 임의의 특정 방식으로 한정되지 않는다. 예컨대, 일부 양태들에서는, 정보가 매체(예컨대, 디스크, 테이프, CD-ROM)를 통해 전송되거나, 또는 다이얼-업 모뎀을 통해 두 컴퓨터 시스템 사이에 직접적으로 전송될 수 있다. 또 다른 양태에서는, 특정 정보가 비-전자 포맷(non-electronic format)으로 전송되어, 시스템에 수동(manually)으로 입력될 수도 있다.

[0106] 또한, 본 명세서에 기재된 시스템 및 방법에 따른 컴퓨터와 사용자 디바이스는 인스트럭션들을 처리하고 인간들 및 다른 컴퓨터들과 데이터를 송수신할 수 있는 임의 디바이스를 포함할 수 있으며, 그 디바이스들로는 로컬 저장 능력이 없는 네트워크 컴퓨터, 모뎀을 갖는 PDA(예컨대, PDA(312)), 인터넷-이용가능 무선 전화기(예컨대, 모바일 전화기(310)), 넷북, 및 패드-타입 휴대용 컴퓨터가 있다.

[0107] 도 4a에 도시된 바와 같이, 네트워크(300)는 또한 클라이언트 디바이스들에게 차등 업데이트를 제공하는 업데이트 데이터베이스(332)를 포함할 수 있다. 업데이트 데이터베이스는 서버(304)에 직접적으로 또는 간접적으로 연결될 수 있다. 대안예로서, 업데이트 데이터베이스(332)는 서버(304)의 일부분이거나 또는 서버(304)와 논리적으로 관련된 것일 수 있다.

[0108] 본 발명은 특정한 실시예들을 참조하여 설명되었지만, 이는 이러한 실시예들은 본 발명의 원리들과 애플리케이션들의 단지 예시에 불과한 것으로 이해되어야 한다. 따라서 다양한 변형예들이 이 예시적 실시예들에 대해 만들어질 수 있고, 다른 배열예들이 첨부된 청구범위에 의해 정의되는 본 발명의 사상 및 범주에서 벗어나지 않으면서 창안될 수 있다는 것이 이해되어야만 한다. 예를 들어, 특정 실시예들이 운영 체제와 관련되어 설명되고 있지만, 본 발명의 양태에 따른 차등 업데이트는 다른 소프트웨어 패키지, 애플리케이션, 및 서비스로도 사용될 수 있다. 이에 더하여, 특정 프로세스가 첨부된 도면에서 특정 순서로 제시되었지만, 이러한 프로세서들은 본 명세서에서 리어한 순서가 명시적으로 개시되어 있지 않은 한, 어떤 특정 순서로도 제안되지 않으며, 다른 순서 또는 동시에 수행될 수 있다. 그리고 달리 특별하게 언급된 바가 없는 한, 추가 프로세스들이 추가되거나 다른 프로세스들이 생략될 수 있다.

[0109] [산업상의 응용가능성]

[0110] 본 발명은 컴퓨터 시스템 오퍼레이션과 이러한 컴퓨터 시스템을 위한 애플리케이션의 업데이트를 포함한, 그러나 이에 한정되지는 않는 광범위한 산업상의 이용가능성을 가진다.

부호의 설명

[0111] 302: 사용자 시스템

304: 업데이트 서버

316: 네트워크

322: 디스플레이

324: 프로세서

326: 메모리

328: 인스트럭션들

330: 데이터

도면

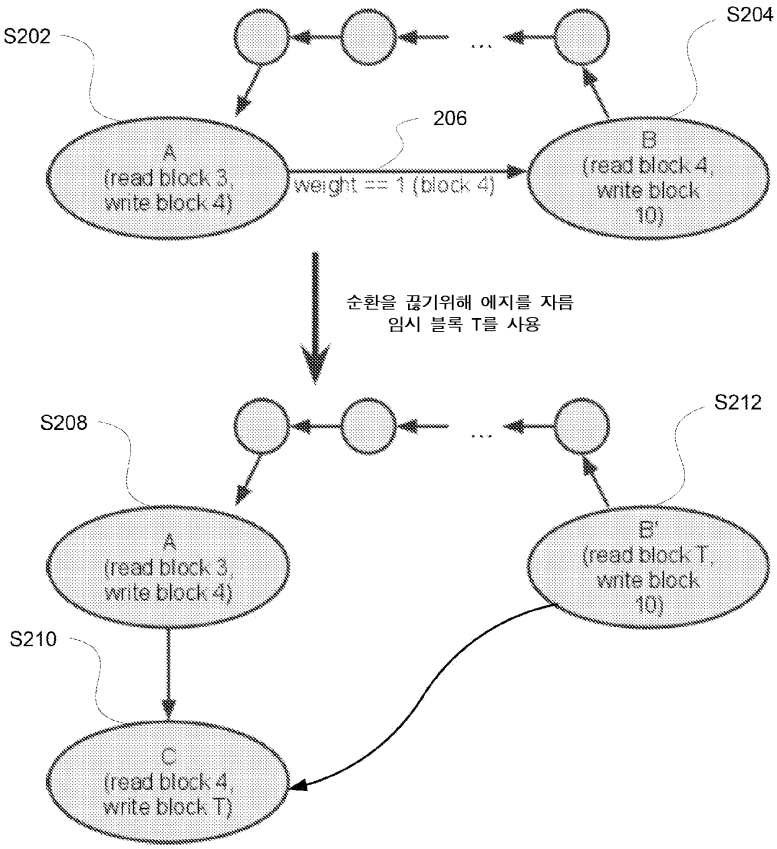
도면1

100

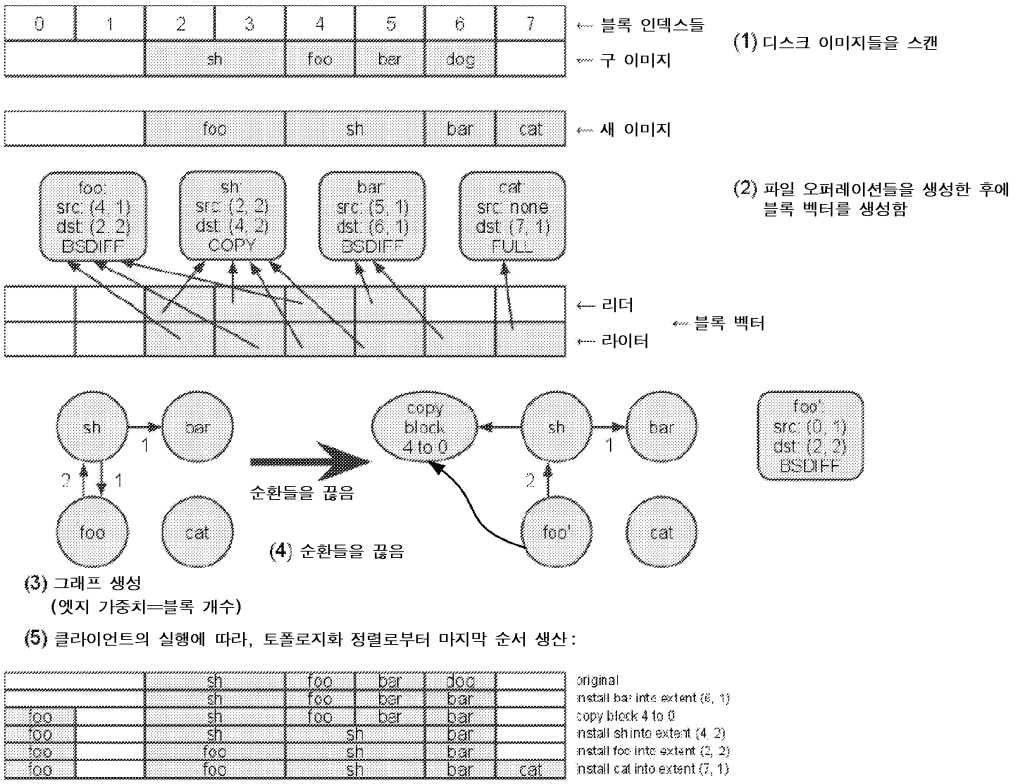
매직 넘버	버전 넘버	프로토콜 버퍼 오프셋	프로토콜 버퍼 길이	프로토콜 버퍼	데이터 블랍들	EOF

도면2

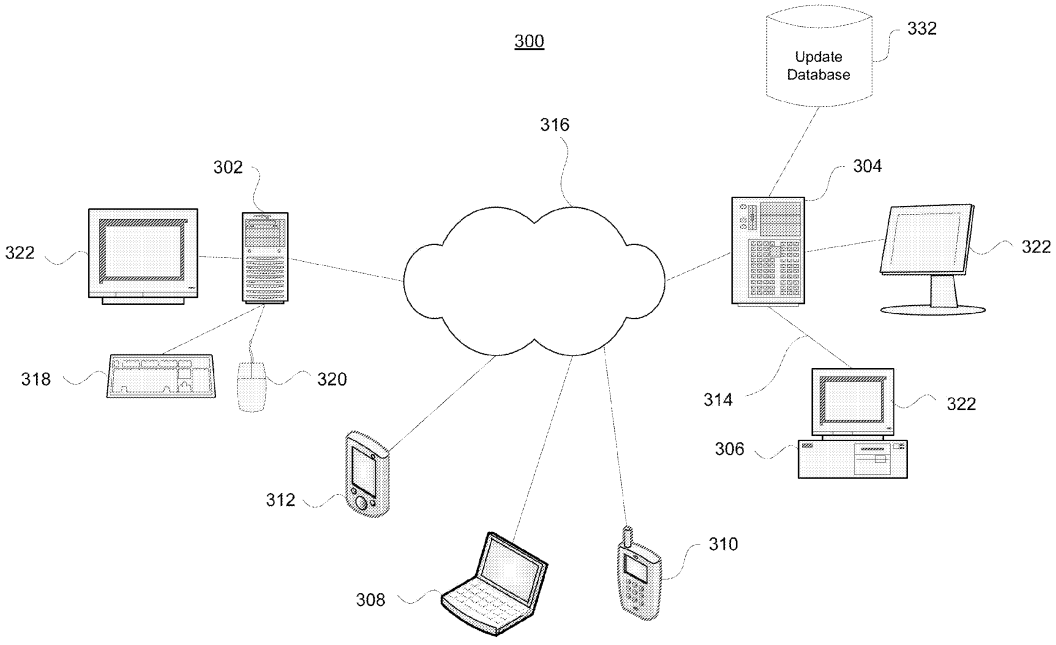
200



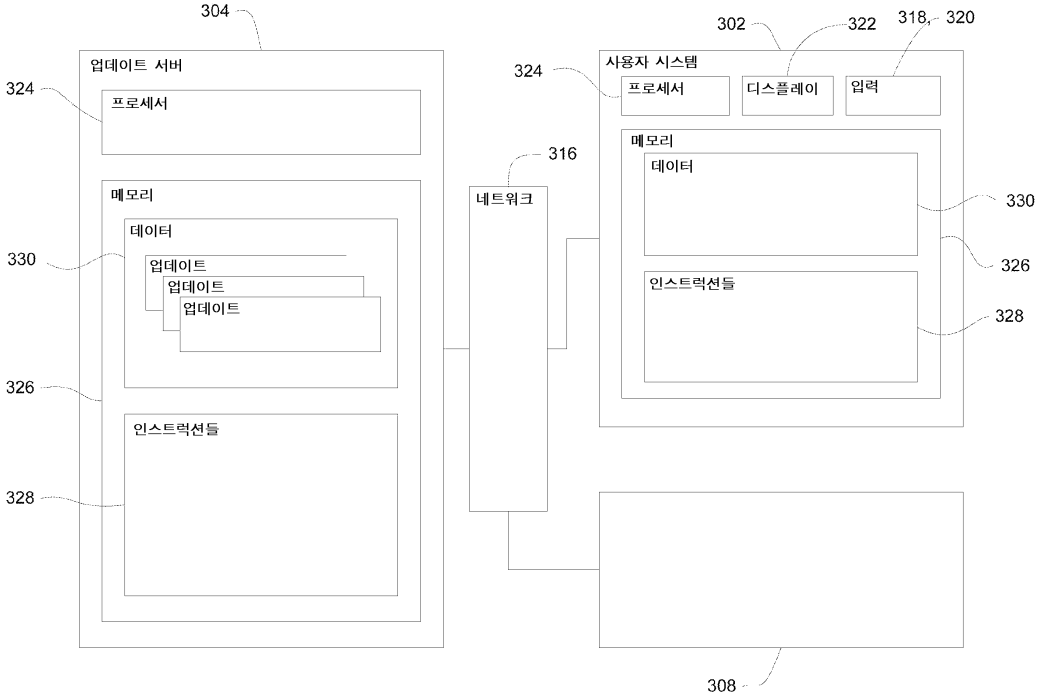
도면3


0 1 2 3 4 5 6 7
sh foo bar dog
← 블록 인덱스들
← 구 이미지
(1) 디스크 이미지들을 스캔
foo sh bar cat
← 새 이미지
foo: src: (4, 1) dst: (2, 2) BSDIFF
sh: src: (2, 2) dst: (4, 2) COPY
bar: src: (5, 1) dst: (6, 1) BSDIFF
cat: src: none dst: (7, 1) FULL
(2) 파일 오퍼레이션들을 생성한 후에 블록 벡터를 생성함
← 리더
← 라이터
← 블록 벡터
sh bar foo cat
1 2 1
순환들을 끊음
copy block 4 to 0
(4) 순환들을 끊음
foo': src: (0, 1) dst: (2, 2) BSDIFF
(3) 그래프 생성
(엣지 가중치=블록 개수)
(5) 클라이언트의 실행에 따라, 토폴로지화 정렬로부터 마지막 순서 생산:
original
install bar into extent (6, 1)
copy block 4 to 0
install sh into extent (4, 2)
install foo into extent (2, 2)
install cat into extent (7, 1)

도면4a


300
Update Database
332
316
302
304
322
322
318
320
314
322
306
312
310
308

도면4b


304
업데이트 서버
324
프로세서
메모리
데이터
330
업데이트
업데이트
업데이트
326
인스트럭션들
328
316
네트워크
302
322
318
320
324
사용자 시스템
프로세서
디스플레이
입력
메모리
데이터
330
326
인스트럭션들
328
308