



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0005769
(43) 공개일자 2025년01월10일

(51) 국제특허분류(Int. Cl.)
H04L 9/00 (2022.01) G06F 3/06 (2006.01)
H04L 67/289 (2022.01) H04L 9/06 (2006.01)
H04L 9/32 (2006.01)
(52) CPC특허분류
H04L 9/50 (2022.05)
G06F 3/0628 (2013.01)
(21) 출원번호 10-2023-0085951
(22) 출원일자 2023년07월03일
심사청구일자 2023년07월03일

(71) 출원인
인하대학교 산학협력단
인천광역시 미추홀구 인하로 100(용현동, 인하대학교)
(72) 발명자
김덕환
서울특별시 양천구 목동로 230, 103동 1004호(목동, 아이파크아파트)
키크마틀로
인천광역시 미추홀구 인하로164번길 9, 301호(용현동)
(74) 대리인
양성보

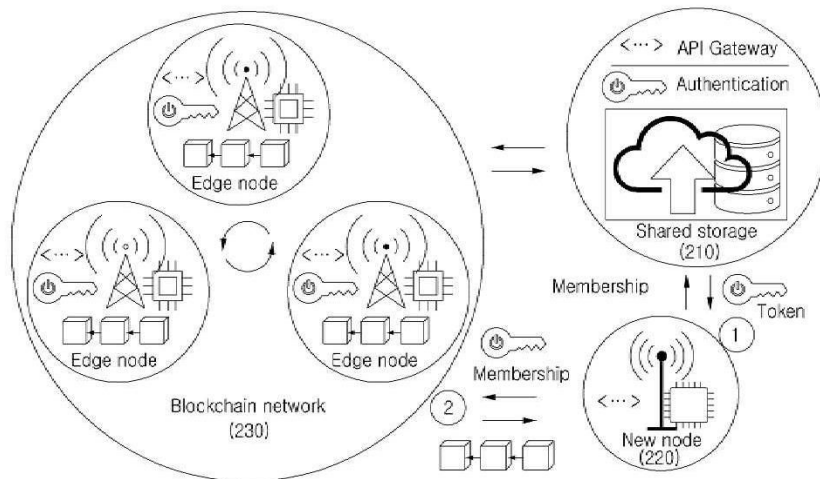
전체 청구항 수 : 총 8 항

(54) 발명의 명칭 엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 방법 및 시스템

(57) 요약

엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 방법 및 시스템이 제시된다. 본 발명에서 제안하는 엣지 컴퓨팅에서의 대용량 데이터 처리 시스템은 새 노드로부터 멤버십 요청을 수신 받아 상기 새 노드의 ID를 인코딩 알고리즘의 키로 사용하여 새 토큰을 생성하는 공유 스토리지 및 상기 새 노드로부터 상기 생성된 토큰을 사용하여 블록체인 네트워크 내의 각 요청에 서명을 받고, 상기 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용하는 블록체인 네트워크를 포함한다.

대표도



(52) CPC특허분류

H04L 67/289 (2022.05)

H04L 9/0643 (2013.01)

H04L 9/3213 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711184751
과제번호	2021R1F1A1050750
부처명	과학기술정보통신부
과제관리(전문)기관명	한국연구재단
연구사업명	기본연구
연구과제명	[Ezbaro] IoT 데이터를 위한 연합 심층학습을 통한블록체인지향 분산 엣지 컴퓨팅
솔루션에 대한 연구	
기 여 율	6/10
과제수행기관명	인하대학교
연구기간	2023.03.01 ~ 2024.02.29

이 발명을 지원한 국가연구개발사업

과제고유번호	1711126440
과제번호	2019-0-00064-003
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	방송통신산업기술개발사업
연구과제명	[Ezbaro] 커넥티드 카를 위한 지능형 모바일 엣지 클라우드솔루션 개발(3차년도)
기 여 율	4/10
과제수행기관명	인하대학교
연구기간	2021.01.01 ~ 2021.12.31

명세서

청구범위

청구항 1

새 노드로부터 멤버십 요청을 수신 받아 상기 새 노드의 ID를 인코딩 알고리즘의 키로 사용하여 새 토큰을 생성하는 공유 스토리지; 및

상기 새 노드로부터 상기 생성된 토큰을 사용하여 블록체인 네트워크 내의 각 요청에 서명을 받고, 상기 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용하는 블록체인 네트워크

를 포함하는 엣지 컴퓨팅에서의 대용량 데이터 처리 시스템.

청구항 2

제1항에 있어서,

상기 블록체인 네트워크는,

API(Application Programming Interface)와 토큰 API를 모두 갖춘 하이브리드 네트워크를 사용하는 레코드체인(Recordchain)을 통한 데이터 관리를 수행하고,

공유 스토리지에 데이터를 업데이트하고 해당 레코드를 블록체인 네트워크 내에서 공유하고,

공유 스토리지에서 데이터를 삭제하고 해당 레코드를 블록체인 네트워크 내에서 공유하며,

상기 새 노드의 각 상태에 대한 세부 정보가 포함된 데이터 수정 내역 목록을 반환하는

엣지 컴퓨팅에서의 대용량 데이터 처리 시스템.

청구항 3

제2항에 있어서,

상기 블록체인 네트워크는,

상기 레코드체인을 통한 데이터의 업데이트 및 삭제에서 레코드 인덱싱을 위해 해당 데이터에 관한 수정 사항을 상태 변수에 저장하고, 상기 상태 변수에 따라 수정된 데이터에 대한 해시가 복제되어 저장되고,

상태 필드의 값이 초기에 해당 데이터가 수정되지 않음을 의미하는 1로 설정되며, 공유 스토리지에 데이터가 업데이트될 때마다 원래 데이터의 동일한 해시로 새 레코드가 생성되고 상태 필드 값이 업데이트된 데이터의 해시로 전송되어 상태 필드에 해시 값이 있는 경우 데이터가 이전 상태에서 업데이트되었음을 의미하고, 삭제 시 상태 필드 값이 0으로 설정되어 원본 데이터의 상태를 확인하는

엣지 컴퓨팅에서의 대용량 데이터 처리 시스템.

청구항 4

제2항에 있어서,

상기 블록체인 네트워크는,

상기 새 노드의 각 상태에 대하여 해시에 할당된 수정 내역을 정의하고, 상기 해시를 사용하여 데이터 기록을 추적하고, 데이터 기록 추적 시 추가 지연을 방지하기 위해 상기 레코드체인을 통해 공유 스토리지의 OBT(optimal browse table)를 사용하여 수정된 데이터에 대해 각 수정 내역을 추적하기 위한 검색 경로를 나타내는

엣지 컴퓨팅에서의 대용량 데이터 처리 시스템.

청구항 5

새 노드가 공유 스토리지로 멤버십 요청을 전송하는 단계;

상기 공유 스토리지가 상기 새 노드의 ID를 인코딩 알고리즘의 키로 사용하여 새 토큰을 생성하는 단계;

상기 새 노드는 상기 생성된 토큰을 사용하여 블록체인 네트워크 내의 각 요청에 서명하는 단계; 및

상기 블록체인 네트워크가 데이터가 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용하는 단계

를 포함하는 엣지 컴퓨팅에서의 대용량 데이터 처리 방법.

청구항 6

제1항에 있어서,

상기 상기 블록체인 네트워크가 데이터가 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용하는 단계는,

API(Application Programming Interface)와 토큰 API를 모두 갖춘 하이브리드 네트워크를 사용하는 레코드체인(Recordchain)을 통한 데이터 관리를 수행하고,

공유 스토리지에 데이터를 업데이트하고 해당 레코드를 블록체인 네트워크 내에서 공유하고,

공유 스토리지에서 데이터를 삭제하고 해당 레코드를 블록체인 네트워크 내에서 공유하며,

상기 새 노드의 각 상태에 대한 세부 정보가 포함된 데이터 수정 내역 목록을 반환하는

엣지 컴퓨팅에서의 대용량 데이터 처리 방법.

청구항 7

제6항에 있어서,

상기 상기 블록체인 네트워크가 데이터가 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용하는 단계는,

상기 레코드체인을 통한 데이터의 업데이트 및 삭제에서 레코드 인덱싱을 위해 해당 데이터에 관한 수정 사항을 상태 변수에 저장하고, 상기 상태 변수에 따라 수정된 데이터에 대한 해시가 복제되어 저장되고,

상태 필드의 값이 초기에 해당 데이터가 수정되지 않음을 의미하는 1로 설정되며, 공유 스토리지에 데이터가 업데이트될 때마다 원래 데이터의 동일한 해시로 새 레코드가 생성되고 상태 필드 값이 업데이트된 데이터의 해시로 전송되어 상태 필드에 해시 값이 있는 경우 데이터가 이전 상태에서 업데이트되었음을 의미하고, 삭제 시 상태 필드 값이 0으로 설정되어 원본 데이터의 상태를 확인하는

엣지 컴퓨팅에서의 대용량 데이터 처리 방법.

청구항 8

제2항에 있어서,

상기 상기 블록체인 네트워크가 데이터가 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용하는 단계는,

상기 새 노드의 각 상태에 대하여 해시에 할당된 수정 내역을 정의하고, 상기 해시를 사용하여 데이터 기록을 추적하고, 데이터 기록 추적 시 추가 지연을 방지하기 위해 상기 레코드체인을 통해 공유 스토리지의 OBT(optimal browse table)를 사용하여 수정된 데이터에 대해 각 수정 내역을 추적하기 위한 검색 경로를 나타내는

엣지 컴퓨팅에서의 대용량 데이터 처리 방법.

발명의 설명

기술 분야

[0001] 본 발명은 엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 방법 및 시스템에 관한 것이다.

배경 기술

[0002] 최근 몇 년 동안, 블록체인은 많은 분야에서 널리 이용되는 기술이 되었다. 금융계에 처음 도입된 분산되고 안전한 P2P(Peer-to-Peer) 네트워크는 새로운 암호화폐 및 NFT(non-fungible Token) 거래소의 출현으로 이어졌다. 블록체인이 이점을 가져올 것으로 예상되는 분야 중 하나는 엣지 컴퓨팅이다. 클라우드 기반 데이터 센터의 중앙 집중화 한계를 해결하기 위해 엣지 컴퓨팅은 IoT 데이터 처리에 대한 분산형 접근 방식을 제공한다. 엣지 노드는 지리적으로 사용자 평면과 가까운 곳에 위치하므로 데이터를 로컬로 처리할 수 있다. 엣지 컴퓨팅은 이미 분산 엣지 노드를 사용하므로 블록체인 프로토콜을 사용할 가능성이 높다. 레코드체인(Recordchain)이라는 이름의 최근 아키텍처는 블록체인을 엣지 컴퓨팅 환경과 통합하는 이점을 보여준다. 그러나 이 두 시스템 간의 데이터 처리 요구 사항은 기술 불일치 문제가 있다. 엣지 컴퓨팅 노드는 일반적으로 데이터 업데이트/삭제 작업이 일반적인 빅데이터를 생성하는 IoT 장치를 처리한다. 그러나 기존 블록체인은 안정성과 불변성을 위해 모든 노드가 동일한 데이터베이스를 공유해야 하므로 데이터가 블록체인에 삽입되면 변경되지 않는다. 모든 노드가 동일한 블록체인 데이터 복사본을 가지고 있기 때문에 네트워크에서 악의적인 수정을 쉽게 거부할 수 있다. 이 간단한 규칙은 암호화폐를 활성화하여 2009년부터 정부의 간섭 없이 안전한 화폐 교환 방법을 가능하게 했다. 이후 2014년에 이더리움이라는 이름의 네트워크는 스마트 계약의 도입으로 블록체인 기능을 확장했다. 스마트 계약은 블록체인에 배포되고 EVM(Ethereum Virtual Machine)에서 실행되는 실행 가능한 소스 코드이다. 스마트 계약에는 로직이 포함될 수 있기 때문에 스마트 계약과 함께 저장된 데이터도 올바른 로직을 실행하여 업데이트하거나 삭제할 수 있다. 스마트 계약의 개념이 데이터 수정을 가능하게 했지만 빅데이터 처리에 적용할 수 없는 크기와 비용 제한이 있다. 최근 IoT 환경의 블록체인에서 데이터 수정을 가능하게 하는 작업을 수행해 왔다. 많은 최신 작업이 도입되었지만 빅데이터와 즉각적인 데이터 업데이트/삭제 작업을 모두 처리할 수 있는 완벽한 접근 방식은 없다. 게다가, 대부분의 접근 방식은 여전히 기존 블록을 삭제함으로써 가장 긴 체인 규칙이 깨지기 때문에 블록체인의 보안 이점을 유지하지 못한다. 앞서 언급했듯이, Recordchain은 블록체인 프로토콜을 사용하는 빅데이터 처리 시스템에서 솔루션에 근접한다. Recordchain은 빅 데이터용 공유 스토리지라는 새로운 데이터 구조를 사용하고 네트워크에서 관련 해시를 레코드로 공유한다.

[0003] 종래기술에서는 클라우드 스토리지를 위한 블록체인 기반 삭제 기술을 사용하여 또 다른 삭제 지향 방법을 도입했다. 여기서 데이터 수정에 블록체인 네트워크 검증을 사용하여 클라우드 서버의 정직성을 높였다. 그러나 이 기법은 블록체인 데이터 삭제에 완전히 초점을 맞추는 것이 아니라 블록체인을 이용하여 데이터 삭제를 확보하는 것에 초점을 맞추고 있다.

[0004] 또 다른 종래기술에서는 중복 데이터 삭제가 일반적인 공급망 시스템에서 추적 기능을 위해 블록체인을 사용했다. 블록체인은 원치 않는 제품 삭제를 방지하고 추적 기능을 사용하여 삭제된 데이터를 복구하는 데 사용된다.

[0005] 또 다른 종래기술에서는 블록체인 프로토콜을 사용하여 중복된 데이터의 스마트 삭제를 가능하게 했다. 이 방법은 블록체인 특징을 사용하여 중복 제거를 쉽게 확인하고, 중복된 데이터에 대해 자동 삭제를 활성화한다.

[0006] 또 다른 종래기술에서는 데이터 관리가 자동화되는 블록체인 프로토콜을 사용하여 클라우드 컴퓨팅에서 데이터 감사에 대해 기존 방법과 비교하여 또 다른 최첨단 접근 방식을 도입했다. 그러나 이러한 모든 방법은 클라우드 기반 시스템에서 데이터 수정을 활성화하거나 모니터링하기 위한 추가 기능으로 블록체인을 사용하는 데 중점을 둔다.

[0007] IoT 기술에서는 마이크로 트랜잭션에서 합의를 유지하기 위해 탱글(tangle) 및 조정자 노드를 사용한다. 또 다른 종래기술에서는 IoT 데이터 시스템에서 더 높은 처리량을 처리하기 위해 블록체인 프로토콜에 대한 그룹 기반 합의를 도입했다. 이러한 기술은 데이터 삭제에만 초점을 맞추고 있지만, 적용된 환경은 이론적으로 데이터 수정에 대한 새로운 기법을 필요로 한다.

선행기술문헌

특허문헌

[0008] (특허문헌 0001) [1] K. Tulkinbekov and D. -H. Kim, "Blockchain-enabled Approach for Big Data Processing in Edge Computing," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2022.3160838.

발명의 내용

해결하려는 과제

- [0009] 본 발명이 이루고자 하는 기술적 과제는 기존의 빅데이터 지향 아키텍처를 계승하면서도 추가적이고 즉각적인 데이터 수정 기능을 갖춘 새로운 하이브리드 블록체인 아키텍처(Recordchain 2.0)인 엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 방법 및 시스템을 제공하는데 있다. 본 발명의 실시예에 따른 Recordchain 2.0은 데이터 레코드 인덱싱 및 검색, 즉각적인 업데이트/삭제 API 및 불변 데이터 변경 브라우징 기능에 대한 새로운 접근 방식을 사용하고, 시스템 안정성을 향상시키기 위해 Recordchain 2.0은 토큰 기반 API(Application Programming Interface)로 인증된 노드에 대해서만 업데이트/삭제 작업을 허용하고자 한다.

과제의 해결 수단

- [0010] 일 측면에 있어서, 본 발명에서 제안하는 엣지 컴퓨팅에서의 대용량 데이터 처리 시스템은 새 노드로부터 멤버십 요청을 수신 받아 상기 새 노드의 ID를 인코딩 알고리즘의 키로 사용하여 새 토큰을 생성하는 공유 스토리지 및 상기 새 노드로부터 상기 생성된 토큰을 사용하여 블록체인 네트워크 내의 각 요청에 서명을 받고, 상기 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용하는 블록체인 네트워크를 포함한다.
- [0011] 상기 블록체인 네트워크는 API(Application Programming Interface)와 토큰 API를 모두 갖춘 하이브리드 네트워크를 사용하는 레코드체인(Recordchain)을 통한 데이터 관리를 수행하고, 공유 스토리지에 데이터를 업데이트하고 해당 레코드를 블록체인 네트워크 내에서 공유하고, 공유 스토리지에서 데이터를 삭제하고 해당 레코드를 블록체인 네트워크 내에서 공유하며, 상기 새 노드의 각 상태에 대한 세부 정보가 포함된 데이터 수정 내역 목록을 반환한다.
- [0012] 상기 블록체인 네트워크는 상기 레코드체인을 통한 데이터의 업데이트 및 삭제에서 레코드 인덱싱을 위해 해당 데이터에 관한 수정 사항을 상태 변수에 저장하고, 상기 상태 변수에 따라 수정된 데이터에 대한 해시가 복제되어 저장되고, 상태 필드의 값이 초기에 해당 데이터가 수정되지 않음을 의미하는 1로 설정되며, 공유 스토리지에 데이터가 업데이트될 때마다 원래 데이터의 동일한 해시로 새 레코드가 생성되고 상태 필드 값이 업데이트된 데이터의 해시로 전송되어 상태 필드에 해시 값이 있는 경우 데이터가 이전 상태에서 업데이트되었음을 의미하고, 삭제 시 상태 필드 값이 0으로 설정되어 원본 데이터의 상태를 확인한다.
- [0013] 상기 블록체인 네트워크는 상기 새 노드의 각 상태에 대하여 해시에 할당된 수정 내역을 정의하고, 상기 해시를 사용하여 데이터 기록을 추적하고, 데이터 기록 추적 시 추가 지연을 방지하기 위해 상기 레코드체인을 통해 공유 스토리지의 OBT(optimal browse table)를 사용하여 수정된 데이터에 대해 각 수정 내역을 추적하기 위한 검색 경로를 나타낸다.
- [0014] 또 다른 일 측면에 있어서, 본 발명에서 제안하는 엣지 컴퓨팅에서의 대용량 데이터 처리 방법은 새 노드가 공유 스토리지로 멤버십 요청을 전송하는 단계, 상기 공유 스토리지가 상기 새 노드의 ID를 인코딩 알고리즘의 키로 사용하여 새 토큰을 생성하는 단계, 상기 새 노드는 상기 생성된 토큰을 사용하여 블록체인 네트워크 내의 각 요청에 서명하는 단계 및 상기 블록체인 네트워크가 데이터가 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용하는 단계를 포함한다.

발명의 효과

- [0015] 본 발명의 실시예들에 따르면 새로운 하이브리드 블록체인 아키텍처(Recordchain 2.0)인 엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 방법 및 시스템을 통해 기존의 빅데이터 지향 아키텍처를 계승하면서도 추가적이고 즉각적인 데이터 수정 기능을 수행할 수 있다. 본 발명의 실시예에 따른 Recordchain 2.0은 데이터 레코드 인덱싱 및 검색, 즉각적인 업데이트/삭제 API 및 불변 데이터 변경 브라우징 기능에 대한 새로운 접근 방식을 사용하고, 시스템 안정성을 향상시키기 위해 Recordchain 2.0은 토큰 기반 API(Application Programming Interface)로 인증된 노드에 대해서만 업데이트/삭제 작업을 허용할 수 있다. 본 발명의 실시예에 따른 네트워크는 여전히 읽기/쓰기 작업을 위해 공개적으로 액세스할 수 있고, Recordchain 2.0은 모든 블록체인 특징을 유지하면서 엣지 컴퓨팅 환경의 모든 요구를 완벽하게 충족할 수 있다.

도면의 간단한 설명

[0016] 도 1은 본 발명의 일 실시예에 따른 블록체인 개념도이다.

도 2는 본 발명의 일 실시예에 따른 엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 시스템의 구성을 설명하기 위한 도면이다.

도 3은 본 발명의 일 실시예에 따른 엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 방법을 설명하기 위한 흐름도이다.

도 4는 본 발명의 일 실시예에 따른 레코드체인 2.0 블록 구조를 나타내는 도면이다.

도 5는 본 발명의 일 실시예에 따른 평가 워크로드 설정을 나타내는 도면이다.

도 6은 본 발명의 일 실시예에 따른 Matlab 시뮬레이션 결과를 나타내는 도면이다.

도 7은 본 발명의 일 실시예에 따른 평가 결과를 나타내는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0017] 블록체인 프로토콜은 데이터 처리를 위한 안전하고 확장 가능한 분산 솔루션을 제공하기 때문에 최근 엣지 컴퓨팅 시스템에서 등장했다. 레코드체인(Recordchain)은 엣지 노드를 위한 최초의 빅데이터 처리 솔루션으로 도입되었다. 그러나 엣지 노드 요구 사항은 블록체인 프로토콜이 허용하지 않는 지속적인 데이터 업데이트 및 삭제로 확장된다. 본 발명에서는 기존의 빅데이터 지향 아키텍처를 계승하면서도 추가적인 즉각적인 데이터 수정 기능을 갖춘 새로운 하이브리드 블록체인 아키텍처(Recordchain 2.0)를 제안한다. 본 발명의 실시예에 따른 Recordchain 2.0은 데이터 레코드 인덱싱 및 검색, 즉각적인 업데이트/삭제 API 및 불변 데이터 변경 브라우징 기능에 대한 새로운 접근 방식을 사용한다. 시스템 안정성을 향상시키기 위해 Recordchain 2.0은 토큰 기반 API(Application Programming Interface)로 인증된 노드에 대해서만 업데이트/삭제 작업을 허용한다. 그러나 네트워크는 여전히 읽기/쓰기 작업을 위해 공개적으로 액세스할 수 있다. Recordchain 2.0은 모든 블록체인 특징을 유지하면서 엣지 컴퓨팅 환경의 모든 요구를 완벽하게 충족한다.

[0018] 본 발명에서는 모든 레코드체인(Recordchain) 특징을 계승하면서도 즉각적인 데이터 업데이트/삭제 작업을 가능하게 하는 새로운 아키텍처(Recordchain 2.0)를 제안한다. Recordchain 2.0은 즉각적인 데이터 수정을 가능하게 하는 새로운 레코드 인덱싱 방법과 데이터 기록을 쉽게 탐색할 수 있는 새로운 OBT(Optimal Browse Table)을 공유 스토리지에 사용한다. 또한 Recordchain 2.0은 업데이트/삭제 API에 액세스하기 위해서만 승인이 필요한 공용 노드와 권한 있는 노드 모두에 대해 API를 활성화하는 하이브리드 네트워크를 사용한다. Recordchain 2.0은 무결성에 대한 가장 긴 체인 규칙을 유지하면서 스토리지 활용률의 이점을 제공한다.

[0019] 본 발명의 실시예에 따르면, 토큰 기반 멤버십 자격을 통해 노드는 요청 시 데이터 수정 방법에 액세스할 수 있다. 또한, 상태 기반 레코드 인덱싱을 통해 네트워크는 공유 스토리지에서 해당 데이터를 수정할 수 있다. 검색(browse) API를 사용하여 OBT(Optimal browse table)는 데이터 수정 내역을 쉽게 추적할 수 있으며, 공유 스토리지에서 즉각적인 삭제를 통해 빅 데이터 시스템에서 스토리지 활용률을 높일 수 있다. 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0020] 블록체인의 목적은 데이터 보안이 강조되는 분산형 불변 데이터베이스를 만드는 것이기 때문에 데이터 삭제는 불필요한 것으로 여겨졌다. 응용 분야가 증가함에 따라 빅데이터 지향 시스템에 적합한 블록체인의 가능성을 더 많이 고려하기 시작했다. 대부분의 관련 연구에서 데이터 삭제 기술이 명시적으로 논의되지는 않았지만, 적용된 접근 방식은 새로운 연구 주제를 가능하게 하는 데 중점을 두었다.

[0021] 본 발명에서는 실행 가능한 소스 코드를 저장하는 기능을 갖춘 이더리움에서 시작된다. 이를 통해 개발자는 실생활 및 가상 자산을 디지털화하고 소유권을 주장하기 위해 안전한 블록체인 네트워크에 저장할 수 있었다. 로직 저장 및 실행 기능과 함께 스마트 계약의 기능도 데이터 수정으로 확장되었다. 올바른 로직을 호출함으로써 개발자는 이제 스마트 계약 내에서 기존 데이터를 업데이트하고 삭제할 수 있다. 스마트 계약에 자극을 받아 Binance Smart Chains, Polygon, Solana 등과 같은 더 많은 프로젝트가 도입되었다. 그러나 이러한 모든 블록체인은 여전히 모든 데이터베이스가 네트워크에서 브로드캐스트되어야 하는 비트코인의 합의의 대부분 측면을 확장한다. 이로 인해 대기 시간 및 스토리지 비용에 제한이 생긴다. 이러한 문제를 방지하기 위해 이더리움은 규정에 따라 스마트 계약의 크기를 24KB로 제한한다. 또한, 스마트 계약을 게시하면 네트워크 보안에 대한 보상으로 노드에 제공되는 수수료가 발생한다. 다시 말해, 이 두 가지 요구 사항은 개발자들이 비용이 증가하기 때문에 더 큰 스마트 계약을 피하도록 동기를 부여했다. 이는 가치 있는 자산을 디지털화하고 새로운 유형의 암호

화폐로 비즈니스를 가능하게 하는 데 충분할 수 있지만, 엡지 컴퓨팅 환경에서는 확실히 감당할 수 없다.

[0022] 엡지 컴퓨팅은 IoT 환경에 많은 이점을 가져오는 널리 논의되는 주제 중 하나이다. 빅 데이터 처리 시스템에 적용됨에 따라 데이터 보안 및 동적 노드 관리는 안정적인 환경을 구축하는 데 있어 핵심적인 문제이다. 블록체인은 분산 네트워크를 제공함으로써 이러한 문제에 대한 가장 쉬운 솔루션을 제공한다. 본 발명은 Recordchain의 확장 버전으로서, 엡지 컴퓨팅에서 블록체인 통합에 대한 또 다른 중요한 주제인 데이터 수정에 초점을 맞추고 있다. IoT 환경에 적용되는 엡지 컴퓨팅은 대부분이 짧은 시간 동안만 의미가 있는 예측할 수 없는 사용자 데이터를 저장한다. 예를 들어, 커넥티드(connected) 자동차 환경에서 교통 데이터는 인근 자동차에 대한 최신 정보로 정기적으로 업데이트될 수 있는 경우에만 가치가 있다. 또한 트래픽 문제가 해결된 후에는 데이터의 가치가 손실되며 스토리지를 효율적으로 사용하기 위해 삭제할 수 있다. IoT 데이터의 이러한 시간 민감성은 데이터 관리를 필요로 한다. 따라서, 적용된 블록체인 네트워크는 장기적인 지원에 대한 이러한 요구사항을 충족할 수 있어야 하며, 데이터 중복 및 과도한 스토리지 비용을 방지해야 한다.

[0024] 도 1은 본 발명의 일 실시예에 따른 블록체인 개념도이다.

[0025] 블록체인은 데이터가 체인을 형성하는 연결된 블록에 저장되는 데이터베이스 유형이라고 할 수 있다. 이 데이터베이스는 동일한 역할을 가진 여러 노드(110, 120, 130, 140)로 구성된 P2P 네트워크에서 공유된다. 이는 중앙 집중화된 권한이 없는 분산 네트워크이다. 따라서 데이터 관리 및 처리는 노드 간의 일반적인 계약(agreement)에 의해 제어되며, 이를 합의(consensus)라고 한다. 합의의 주요 목적은 데이터 보안과 배포의 일관성을 가능하게 하고 중요한 데이터를 손상시키는 비잔틴(byzantine) 노드를 방지하는 것이다. 이 일반 계약에 따라 노드는 새 블록을 수락할지 거부할지 결정한다. 합의는 모든 노드에서 공유되므로 일반 규칙이나 악의적인 활동을 변경하는 것은 네트워크에서 쉽게 무시될 수 있다. 전통적인 블록체인(예를 들어, 비트코인)은 PoW(Proof-of-Work) 합의를 채택한다. PoW에 따르면, 모든 노드는 블록체인의 복사본을 유지해야 한다. 새로운 블록이 생성되면, 메인 콘텐츠(트랜잭션)가 데이터 세그먼트에 저장되고, 그에 따라 헤더(Header)가 생성된다. 모든 노드는 새로운 블록을 생성할 수 있는 권한을 놓고 서로 경쟁한다. 이 프로세스는 풀에서 보류 중인 모든 트랜잭션을 선택하는 것으로 시작된다. 이러한 트랜잭션을 기반으로 하나의 고유한 Merkle 루트가 생성되고 헤더의 카운트로 설정된다. Merkle 루트를 계산한 후 마지막 블록의 해시(Hash)가 체인을 생성하도록 설정된다. 이것은 블록의 내용을 가정하며, 다음은 블록에 대한 고유 해시를 생성하는 것이다. 이 프로세스는 노드가 $H(\text{nonce} + \text{데이터} + \text{메타데이터})$ 가 지정된 조건을 충족하는 새 nonce 값을 생성하도록 한다. 조건을 충족하는 작업을 난이도라고 하며 네트워크에 의해 동적으로 설정된다. 더 많은 노드가 네트워크에 가입하면 그에 따라 난이도가 증가한다. 비트코인에서는 블록 마이닝 난이도를 새로운 블록 생성 지연(약 10분)으로 설정한다. 이 작업(work)을 완료한 첫 번째 노드는 확인을 위해 결과를 네트워크에 전파한다. 다른 노드는 계산을 확인하고 검증(proof)으로 응답한다. 충분한 검증을 받은 노드는 새로운 블록을 블록체인에 추가하고 기여에 대한 보상(다시 말해, 코인)을 받을 수 있다. PoW는 네트워크의 많은 노드에서 많은 에너지와 노력을 필요로 하는 긴 프로세스이다. 블록이 네트워크에 삽입되어 데이터가 수정되면 다른 Merkle 루트와 다른 블록 해시가 생성된다. 따라서 네트워크 노드는 합의에 의존하여 변경사항을 쉽게 감지하고 무효화할 수 있다.

[0026] 멤버십 규칙 및 권한 요구사항에 따라 블록체인은 다음과 같이 4가지로 분류할 수 있다:

[0027] 공용: 오픈 소스 코드가 있는 순수 오픈 네트워크이다. 누구나 허가나 허가 없이 네트워크에 참여할 수 있다.

[0028] 비공용: 합의는 하나의 노드에서만 제어된다. 노드는 컨트롤러 노드로부터 권한을 얻은 후에 작업에 참여하거나 수행할 수 있다.

[0029] 컨소시엄: 전용 네트워크와 비슷하지만 노드 그룹에 의해 제어된다. 컨트롤러 그룹의 모든 구성원이 구성원 자격을 관리할 수 있다.

[0030] 하이브리드: 공용 네트워크와 개인 네트워크의 결합으로 두 네트워크 모두에서 선택적 특징을 구현한다.

[0031] 이름에서 알 수 있듯이, 공용 블록체인은 가장 많이 분산되어 있으며 중앙 집중식 제어에서 자유롭다. 하지만, 이는 가장 비용이 많이 든다. 누구나 네트워크에 가입할 수 있기 때문에 비잔틴 노드를 제어하려면 PoW와 같은 강력한 합의 알고리즘이 필요하다. 반면, 사실 및 컨소시엄 네트워크는 신뢰할 수 있는 노드로만 작업하는 것이 가장 우선순위가 높은 폐쇄형 네트워크에만 적합하다. 이는 자연스럽게 하이브리드 네트워크를 신뢰할 수 있는 노드를 통해 블록체인 합의를 더 쉽게 달성할 수 있는 연구에 도입했으며 여전히 공용 네트워크를 통해 데이터를 브로드캐스트한다. 특히 블록체인 아키텍처에 새로운 기능(예를 들어, 데이터 수정 등)을 도입할 때 하이브

리드 네트워크는 무결성을 달성하기 위한 다양한 솔루션을 제공한다.

[0032] 앞서 블록체인에 관하여 설명한 바와 같이, 기존의 블록체인 합의는 블록을 검증한 후 데이터 수정을 엄격하게 방지한다. 이는 블록체인 보안을 유지하면서도 빅데이터 시스템과의 통합에 가장 큰 문제가 되는 주요 요인이다. 데이터는 블록 내부에 저장되고 해싱 기법을 사용하여 서로 연결되므로 블록 내부의 특정 데이터를 삭제하거나 업데이트할 수 있는 실질적인 가능성은 없다. 따라서 블록체인에서 데이터를 삭제하는 가장 쉬운 방법은 체인 프로토콜을 위반하지 않고 블록 자체를 삭제하는 것이다. 이론적으로 블록체인 합의를 따르면서 블록 삭제를 활성화하는 방법은 다음 블록 확인 전 마지막 블록을 삭제하는 방법과 제네시스 블록 위치를 업데이트하여 첫 번째 블록을 삭제하는 방법 두 가지가 있다.

[0033] 유한 수명 블록에 대해 리티체인(Littichain)을 구현하여 LBD(Last Block Deletion) 방법을 적용할 수 있다. 두 개의 그래프를 만들고 수명에 따라 블록을 정렬한다. 수명이 가장 긴 것이 첫 번째 순서이다. 실제 위치와 정렬 순서를 유지하기 위해 각 블록은 이전 블록과 상위 블록을 모두 참조한다. 이 시나리오에서는 먼저 삭제할 블록이 마지막에 위치하므로 다음 블록이 도착하기 전에 삭제할 수 있다. 마지막 블록을 삭제하기 전에 다음 블록이 오면 하위 블록이 삭제될 때까지 대기하므로 삭제 지연이 발생한다. 긴 지연을 방지하기 위해 리티체인은 블록이 도착 시간 순서를 따르지 않아도 되는 k-높이 삽입 방법을 사용한다. 상위 블록 순서 체인이 유지되므로 만료 시간 순서를 따름으로써 합의를 달성할 수 있다. 삭제 시간에 따라 블록을 정렬하는 데 집중함으로써 LBD 방법을 구현하기 위한 독특한 솔루션을 제공했다. 그러나 리티체인은 만료 시간에 따라 데이터를 정렬하려면 사전 처리가 필요한 구현 측면에서 한계가 있다. 블록을 삭제하면 전체 데이터 저장소도 손실되므로 블록은 무결성을 위해 유사한 만료 시간 데이터만 저장할 수 있다. 또한 아직 만료되지 않은 하위 블록이 있으면 블록 삭제가 지연되어 원하지 않는 추가 저장소가 생성될 수 있다. 따라서 리티체인이 블록 삭제를 효과적으로 활성화하더라도 옛지 컴퓨팅에서는 중단 간 지연이 허용되지 않는다.

[0034] FBD(First Block Deletion) 방법은 블록 체인 구조에서 해시 참조를 변경해야 하는 경우에는 그다지 효과적이지 않은 것으로 보인다. 그러나 선택적 삭제를 가능하게 함으로써 구현할 수 있다. 특정 임계값 뒤에 새 요약 블록이 생성되는데, 이 블록은 새 데이터를 저장하지 않고 이전 블록을 요약한다. 이 시나리오에서는 삭제된 데이터를 건너뛰고 새 요약 블록이 생성 블록을 직접 참조한다. 이 시나리오에서는 이전 블록에 대한 참조가 요약 블록으로 확장된 새 체인으로 대체된다. 이 간단한 접근 방식은 논리적으로 경로를 잃어 첫 번째 블록을 삭제하는 데 중점을 두므로 새 체인은 최신 데이터만 저장한다. 이 기법은 실용적이고 쉽고 효과적인 것처럼 보일 수 있지만 요약 블록이 생성될 때까지 지연이 발생하므로 옛지 컴퓨팅 요구 사항 측면에서 리티체인과 동일한 단점이 있다. 또한 두 방법 모두 실제 블록을 삭제하는 데 중점을 두며, 이는 블록체인에서 가장 긴 체인 규칙을 위반할 수 있다.

[0035] 앞서 설명된 바와 같이, 기존 블록체인은 무결성과 보안이 최우선인 공용 네트워크를 위해 설계되었다. 보편적인 신뢰성을 달성하기 위해, 가장 긴 체인 규칙은 블록체인 프로토콜에서 가장 효과적인 접근 방식을 입증했다. 이 규칙에 따르면, 가장 긴 체인을 보유한 노드가 합의를 얻고 해당 블록체인이 네트워크의 모든 노드에서 수락된다. 이 시나리오에서 모든 노드는 데이터베이스를 최신 상태로 유지하기 위해 블록체인의 복사본을 더 긴 체인으로 지속적으로 업데이트한다. 이 간단한 규칙은 블록체인 보안에서 중요한 역할을 한다. 공격자가 기존 블록체인 데이터를 변경하려면 업데이트된 데이터로 더 긴 체인을 만들고 네트워크에서 이를 수락해야 한다. 반면에, 새로운 블록을 생성하려면 대량의 계산이 필요하며, 최소 51%의 노드에서 네트워크를 확인해야 한다. 따라서 공격자는 대다수 노드의 지원 없이는 더 긴 체인을 만들 수 없다. 다른 시도는 체인이 가장 긴 네트워크 노드에서 쉽게 거부된다.

[0036] 데이터 수정 시나리오를 고려할 때 LBD와 FBD 방법은 모두 블록체인에서 기존 블록을 삭제하는 데 초점을 맞추고 있으며, 이는 블록체인을 공격자에게 취약하게 만든다. 따라서 데이터 수정 특징을 갖춘 새로운 아키텍처를 설계하려면 안정성 및 보안을 위한 가장 긴 체인 규칙을 고려해야 한다.

[0037] 레코드체인(Recordchain)은 분산 노드에서 빅 데이터를 처리하기 위한 옛지 컴퓨팅 환경을 위해 제안되었다. 이 목표를 달성하기 위해 레코드체인은 실제 데이터를 공유 스토리지라는 독립적인 노드로 분리한다. 내결함성을 제공하기 위해 레코드체인은 공유 스토리지 데이터를 복제하기 위해 데이터 노드도 사용한다. 공유 스토리지를 복제하면 노드가 로컬에서 데이터에 액세스하고 비상 시 새로운 공유 스토리지 위치의 잠재적 후보로 남을 수 있다. 공유 스토리지는 IoT 데이터를 키 값 스토리지에 보관하는 동시에 액세스를 위한 읽기 및 쓰기 API만 제공한다. 따라서 레코드 체인의 기본 구현은 네트워크에서 데이터 수정을 허용하지 않는다. 공유 스토리지에 삽입된 모든 데이터에 대해 레코드체인은 고유한 레코드를 메타데이터로 생성하고 이 레코드는 블록에 삽입되어

전과된다. 새로운 블록은 PoW에 이어 생성되며, 비트코인의 블록체인 규칙 대부분은 여전히 레코드체인에서 상속된다. 또한 옛지 컴퓨팅 환경에서 즉각적인 검증과 낮은 중단 간 대기 시간을 달성하기 위해 고유한 RAC(Record Approval Consensus)가 PoW와 함께 사용된다. 이 프로세스는 각 레코드에 고유한 집합 서명을 할당하고 해당 블록이 생성되기 전에도 보안을 보장한다. RAC는 다음과 같은 세 가지 단계만을 포함하는 경량 합의이다:

- [0038] 브로드캐스트: 처음에는 보류 중인 레코드 목록과 함께 새 레코드가 네트워크에 브로드캐스트 된다.
- [0039] 서명 확인 및 수집: 노드가 브로드캐스트 메시지를 수신하고 확인한다. 노드는 가장 긴 목록만 허용하므로 다른 버전은 제거된다. 그러면 노드는 고유 ID가 포함된 확인 메시지로 응답한다.
- [0040] 커밋: 브로드캐스트 노드는 충분한 확인을 기다린 후 응답하는 ID를 기반으로 고유한 집합 서명을 생성한다. 그런 다음 보류 중인 새 레코드 목록이 다시 브로드캐스트되어 네트워크를 업데이트한다. 이번에는 레코드가 이미 검증되었음을 증명하는 일괄 서명이 할당된다.
- [0041] RAC는 검증을 위해 블록 생성 지연을 방지하므로 옛지 컴퓨팅 환경에 또 다른 중요한 이점을 제공한다. 또한 공유 스토리지는 블록체인 네트워크에서 빅 데이터 처리 기능을 가능하게 한다.
- [0043] 도 2는 본 발명의 일 실시예에 따른 옛지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 시스템의 구성을 설명하기 위한 도면이다.
- [0044] 본 발명의 실시예에 따른 옛지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 시스템을 통해 Recordchain 2.0의 구현 및 새로운 기능을 제안한다. 공유 스토리지 고용 및 레코드 승인 합의를 포함하여 앞서 설명한바와 같이 전체 아키텍처가 레코드체인에서 상속된다. 본 발명의 실시예에 따른 Recordchain 2.0은 기록을 삭제, 업데이트 및 탐색하는 새로운 방법을 사용한다. 보안을 위해 토큰 기반 구성된 프로토콜이 공유 스토리지의 도움을 받아 사용된다. Recordchain 2.0에서는 저장된 데이터에 대한 수정 권한을 제어하기 위해서만 인증이 사용된다. 이 기술은 소유자 노드만 Recordchain 2.0의 기존 데이터를 수정할 수 있도록 한다. 이 새로운 멤버십 프로토콜은 Recordchain 2.0을 공용 및 인증된 방법을 모두 갖춘 하이브리드 네트워크로 만든다.
- [0045] 본 발명의 실시예에 따른 옛지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 시스템은 공유 스토리지 (210) 및 블록체인 네트워크(230)를 포함한다.
- [0046] 공유 스토리지(210)는 새 노드(220)로부터 멤버십 요청을 수신 받아 상기 새 노드(220)의 ID를 인코딩 알고리즘의 키로 사용하여 새 토큰을 생성한다.
- [0047] 블록체인 네트워크(230)는 상기 새 노드(220)로부터 상기 생성된 토큰을 사용하여 블록체인 네트워크 내의 각 요청에 서명을 받고, 상기 토큰으로 서명된 데이터의 경우에만 공유 스토리지(210)에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용한다.
- [0048] 본 발명의 실시예에 따른 Recordchain 2.0은 공개 API와 토큰 API를 모두 갖춘 하이브리드 네트워크를 사용한다. 노드는 공용 네트워크에 쉽게 가입할 수 있다. 그러나 데이터 업데이트 및 삭제 기능의 경우 도 2와 같이 공유 스토리지(Shared storage)(210)에서 새 노드(New node)(220)를 인증해야 한다. 도 2와 같이 새 노드(220)가 먼저 공유 스토리지(210)로 멤버십 요청을 보낸다. 공유 스토리지(210)는 새 노드(220)의 ID를 인코딩 알고리즘의 키로 사용하여 새 토큰을 생성한다. 새 노드(220)는 토큰을 사용하여 블록체인 네트워크(Blockchain network)(230) 내의 각 요청에 서명한다. 블록체인 네트워크(230)는 토큰이 없는 요청도 지원한다. 그러나 토큰은 데이터에 대한 수정 권한을 얻는 데 중요한 역할을 한다. Recordchain 2.0은 데이터가 토큰으로 서명된 경우에만 업데이트 및 삭제 작업을 허용하므로 소유자만 이러한 권한을 가진다. 토큰에는 만료 날짜가 없으므로 노드 실패 후 동일한 노드가 인증을 요청하면 토큰을 다시 생성할 수 있다. 본 발명의 실시예에 따른 Recordchain 2.0은 하이브리드 네트워크를 사용하여 데이터 관리를 위한 세 가지 새로운 방법을 제안한다.
- [0049] 업데이트: 공유 스토리지에 데이터를 업데이트하고 해당 레코드를 블록체인 네트워크 내에서 공유한다.
- [0050] 삭제: 공유 스토리지에서 데이터를 삭제하고 해당 레코드를 블록체인 네트워크 내에서 공유한다.
- [0051] 검색: 상기 새 노드의 각 상태에 대한 세부 정보가 포함된 데이터 수정 내역 목록을 반환한다.

- [0053] 도 3은 본 발명의 일 실시예에 따른 엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 방법을 설명하기 위한 흐름도이다.
- [0054] 본 발명의 실시예에 따른 엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 방법은 새 노드가 공유 스토리지로 멤버십 요청을 전송하는 단계(310), 상기 공유 스토리지가 상기 새 노드의 ID를 인코딩 알고리즘의 키로 사용하여 새 토큰을 생성하는 단계(320), 상기 새 노드는 상기 생성된 토큰을 사용하여 블록체인 네트워크 내의 각 요청에 서명하는 단계(330) 및 상기 블록체인 네트워크가 데이터가 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용하는 단계(340)를 포함한다.
- [0055] 단계(310)에서, 새 노드가 공유 스토리지로 멤버십 요청을 전송하면, 단계(320)에서, 상기 공유 스토리지가 상기 새 노드의 ID를 인코딩 알고리즘의 키로 사용하여 새 토큰을 생성한다.
- [0056] 이후 단계(330)에서, 상기 새 노드는 상기 생성된 토큰을 사용하여 블록체인 네트워크 내의 각 요청에 서명하고, 단계(340)에서 상기 블록체인 네트워크가 데이터가 토큰으로 서명된 데이터의 경우에만 공유 스토리지에 업데이트, 삭제 및 검색을 포함하는 작업을 수행하도록 허용한다.
- [0057] 단계(340)에서, 블록체인 네트워크는 API(Application Programming Interface)와 토큰 API를 모두 갖춘 하이브리드 네트워크를 사용하는 레코드체인(Recordchain)을 통한 데이터 관리를 수행한다.
- [0058] 본 발명의 실시예에 따른 Recordchain 2.0은 공유 스토리지에 데이터를 업데이트하고 해당 레코드를 블록체인 네트워크 내에서 공유하고, 공유 스토리지에서 데이터를 삭제하고 해당 레코드를 블록체인 네트워크 내에서 공유하며, 상기 새 노드의 각 상태에 대한 세부 정보가 포함된 데이터 수정 내역 목록을 반환한다.
- [0059] 본 발명의 실시예에 따른 Recordchain 2.0은 데이터의 업데이트 및 삭제에서 레코드 인덱싱을 위해 해당 데이터에 관한 수정 사항을 상태 변수에 저장하고, 상기 상태 변수에 따라 수정된 데이터에 대한 해시가 복제되어 저장된다.
- [0060] 본 발명의 실시예에 따른 Recordchain 2.0을 통해 상태 필드의 값이 초기에 해당 데이터가 수정되지 않음을 의미하는 1로 설정되며, 공유 스토리지에 데이터가 업데이트될 때마다 원래 데이터의 동일한 해시로 새 레코드가 생성되고 상태 필드 값이 업데이트된 데이터의 해시로 전송되어 상태 필드에 해시 값이 있는 경우 데이터가 이전 상태에서 업데이트되었음을 의미하고, 삭제 시 상태 필드 값이 0으로 설정되어 원본 데이터의 상태를 확인한다. 하는
- [0061] 본 발명의 실시예에 따른 Recordchain 2.0을 통해 상기 새 노드의 각 상태에 대하여 해시에 할당된 수정 내역을 정의하고, 상기 해시를 사용하여 데이터 기록을 추적한다. 데이터 기록 추적 시 추가 지연을 방지하기 위해 상기 레코드체인을 통해 공유 스토리지의 OBT(optimal browse table)를 사용하여 수정된 데이터에 대해 각 수정 내역을 추적하기 위한 검색 경로를 나타낼 수 있다. 도 4를 참조하여 본 발명의 실시예에 따른 Recordchain 2.0 통한 엣지 컴퓨팅에서의 대용량 데이터 처리를 위한 블록체인 활용 방법에 관하여 더욱 상세히 설명한다.
- [0063] 도 4는 본 발명의 일 실시예에 따른 레코드체인 2.0 블록 구조를 나타내는 도면이다.
- [0064] 본 발명의 실시예에 따른 Recordchain 2.0은 데이터 업데이트 및 삭제를 가능하게 하는 새로운 레코드 인덱싱 기술을 사용한다.
- [0065] 블록 구조 및 데이터 수정의 예시를 도 4에 도시하였다. 블록 해시를 연결하고 블록 데이터를 유지 관리하여 체인을 생성하는 방법은 공통 블록 체인 구조에서 상속된다. 도 4는 새로운 방법을 구현하는 데 사용되는 블록 구조의 가장 중요한 구성 요소만 도시한다. 일반적으로 레코드 해시와 인덱스는 기본 인덱스 변수로 사용된다. 본 발명의 실시예에 따른 Recordchain 2.0은 하나의 추가 변수로 약간 수정된 레코드 인덱싱 방식을 사용한다. 이 시나리오에서 해시는 원본 데이터에 대한 고유 해시를 저장하지만 변경할 수 없다. 해당 데이터의 수정 사항은 상태 변수에 저장된다. 이 변수는 수정 유형도 정의한다. 수정된 데이터에 대해 해시가 복제되고 최신 복사본이 최신 버전을 저장한다. 따라서 읽기 방법은 마지막 블록에서 첫 번째 블록까지의 해시를 사용하여 검색하고 최신 버전을 찾는다.
- [0066] 또한, 상태 필드의 값은 처음에 1로 설정된다. 이는 해당 데이터가 아직 수정되지 않았음을 의미한다. 공유 스토리지에서 데이터가 업데이트될 때마다 원래 데이터의 동일한 해시로 새 레코드가 생성되고 상태 필드 값이 업데이트된 데이터의 해시로 전송된다. 따라서 상태 필드에 해시 값이 있으면 데이터가 이전 상태에서 업데이트되

있음을 의미한다. 삭제 시 상태 필드의 값이 0으로 설정되어 원본 데이터의 상태를 확인할 수 있다.

[0067] 도 4는 블록 생성 타임라인에 따른 해당 레코드와의 데이터 수정 상호작용을 보여준다. 첫 번째 블록(블록 A)은 초기 지점(t0)에 생성된다. 블록 A의 모든 레코드는 상태 필드에서 동일한 값을 가진다. 이는 해당 데이터가 수정되지 않았음을 나타낸다. 블록 B를 생성하기 전에 t1에 두 개의 레코드가 수정 표시된다(R2가 업데이트되고 R10이 삭제됨). Recordchain 2.0 프로토콜을 사용하면 데이터의 각 변경 사항에 대한 새 레코드를 만들 수 있다.

[0068] 블록 B에서 볼 수 있듯이, 레코드 R11과 R20은 각각 해시 필드를 기준으로 블록 A의 레코드 R2와 R10에 해당한다. 데이터가 업데이트 되면(R2->R11), 업데이트된 데이터를 기반으로 새 해시가 생성되고 새 레코드의 상태 필드에 저장되지만 원래 데이터 해시는 변경되지 않고 해시 필드에 저장된다. 이렇게 하면 해시 필드와 상태 필드의 값이 달라져 데이터가 업데이트되었음을 증명한다. 또한 블록 B에는 동일한 레코드에 대한 다른 업데이트(R2->R11->R12)가 포함되므로 상태 필드에는 다른 값이 저장된다. 삭제 작업(R10 -> R20)에서는 데이터가 공유 스토리지에서 제거되므로 상태 필드는 그에 따라 0으로 채워진다. 이 규칙에 따라 블록 B의 R12는 t3에서 삭제 표시되며, 이는 블록 C가 새 레코드로 생성될 때 작업 결과를 볼 수 있음을 의미한다. 즉, 데이터 수정 방법은 체인에서 기존 데이터 또는 블록을 업데이트하거나 삭제하지 않고 공유 스토리지에서 수정된 데이터에 대한 새 레코드를 생성한다. Recordchain 2.0은 빅데이터를 저장하도록 설계되었기 때문에 공유 네트워크에서 해당 레코드의 크기는 공유 스토리지 데이터의 크기 다음으로 무시할 수 있다. 따라서 Recordchain 2.0은 블록체인 아키텍처에서 데이터 수정을 달성하는 효율적이고 실용적인 방법을 달성한다. 또한 수정 내역의 기록을 보관하면 데이터 내역을 검색할 수 있으므로 시스템에 무결성이 확보된다.

[0069] 본 발명의 실시예에 따른 Recordchain 2.0은 상태 목록을 반환하는 추가 방법을 제공한다. 여기서 각 상태는 해시에 할당된 수정 내역을 정의한다. 그러나 해시를 사용하여 데이터 기록을 추적하는 것은 비용이 많이 드는 작업이므로 전체 블록 체인을 철저히 검색해야 한다. Recordchain 2.0은 추가 지연을 방지하기 위해 공유 스토리지에 OBT(optimal browse table)를 사용한다. 테이블은 수정된 데이터에 대해서만 초기화되며 각 수정 내역을 추적하기 위한 검색 경로를 보여준다.

[0070] OBT는 임의의 기록 상태를 나타내기 위해 $x(k)$ 를 고려하면, $u_b(k)$ 와 $u_r(k)$ 은 각각 블록별 및 기록별 최적 제어 변수를 나타내며, 다음 상태는 다음과 같이 정의되고,

$$x(k-1) = x(k) \blacktriangle u_B(k) \nabla u_R(k) \quad (1)$$

[0072] 다음의 경계를 사용한다:

$$0 \leq x(k), \quad k = 0, 1, 2, \dots, n$$

$$0 \leq u(k), \quad k = 1, 2, \dots, n$$

[0075] 여기서 식은 블록의 경우 $x(k)$ 에서 $u_b(k)$ 인덱스 UP($\blacktriangle$), 레코드의 경우 $u_r(k)$ 인덱스 IN(∇)으로 읽힌다. 식에서 알 수 있듯이 Recordchain 2.0은 최적의 검색 테이블을 정의하기 위해 계산 방식보다는 레코드 위치에 대한 사실에 주로 의존한다. 또한 정의된 식을 사용하여 전체 이력을 추적하기 위한 최적의 비용은 다음과 같이 정의될 수 있다:

$$J^* = C_0 + \sum_{k=1}^n \{C_k + C_B u_B(k) + C_R u_R(k)\} \quad (2)$$

[0077] 여기서 C_0 , C_k , C_B , C_R 은 시간 상수이다. C_0 과 C_k 는 임의의 레코드를 읽는 데 소요되는 시간이고, C_B 와 C_R 은 각각 특정 블록과 레코드에 액세스하는 데 소요되는 지연 시간이다.

[0078] OBT 먼저 소비자 노드는 새 레코드를 승인하여 데이터 수정 확인을 받은 다음 공유 스토리지에 요청을 보낸다. 앞서 설명한 것처럼 공유 스토리지에서 토큰 기반 업데이트 및 삭제 방법을 사용하여 데이터 수정을 수행한다. 이러한 방법이 실행될 때마다 공유 스토리지는 데이터에 대해 OBT를 초기화하고 레코드의 초기 상태로 이어지는 원래 데이터 해시를 할당한다. 새 레코드는 블록에 대한 메타데이터와 함께 제공되므로 블록 내부에 저장되고 인덱싱되며 이 메타데이터에서 제어 변수를 추출할 수 있다. 그런 다음 새 레코드의 블록 인덱스에서 최신 레코

드의 블록 인덱스를 빼서 $u_B(k)$ 를 계산할 수 있으며, $u_B(k)$ 는 새 레코드 인덱스에 할당된다.

식(1) 및 식(2)에 기초하여, 표 1은 도 4의 예시에 대한 샘플 OBT를 보여준다.

<표 1>

k	x(k)	$u_B(k)$	$u_R(k)$	J^*
0	1	∞	∞	C_0
1	2	1	2	$C_0+C_k+C_B+2C_R$
2	2	0	1	$C_0+2C_k+C_B+3C_R$
3	25	23	9	$C_0+3C_k+24C_B+12C_R$

표 1과 같이, 행은 수정된 데이터를 기반으로 새 레코드가 생성될 때 초기화된다. 원본 데이터는 무한 제어 변수와 함께 인덱스 0에 기록되고 해당하는 제어 변수는 다음 행에 기록된다. 이력 추적 API는 Recordchain 2.0 프로토콜에 따라 마지막 행부터 첫 번째 행까지 테이블을 읽는다. 도 4에 도시된 블록체인의 예를 들어, 마지막 해시 레코드(0d1b754f2)는 블록 C(인덱스 25)에 저장된다. 이 레코드의 경우 블록 제어는 23이고 레코드 제어는 9이다. 이렇게 하면 포인터가 해당 레코드가 저장되는 블록 인덱스 2의 레코드 인덱스 9로 이동한다. 동일한 절차를 수행하면 지정된 해시에 대한 과도한 검색 없이 전체 내역을 추적할 수 있다. 표에는 각 행을 추적하는 비용도 나와 있다. 상수 값은 시스템과 구현 방식에 따라 다르다. 그러나 공유 스토리지는 상수 값을 정의하고 작업에 대한 예상 비용을 업데이트한다. 브라우징 방법의 구현과 OBT의 채용은 정기적인 데이터 업데이트가 필요한 빅 데이터 시스템에서 Recordchain 2.0의 신뢰성을 향상시킨다.

본 발명의 실시예에 따른 Recordchain 2.0은 공유 스토리지의 효율성에 있어서 Recordchain 2.0의 소유자 노드에 대해 직접 데이터 수정이 가능하다. 즉, 소유자 노드는 더 이상의 지연 없이 공유 스토리지에서 데이터를 즉시 쉽게 삭제하거나 업데이트할 수 있다. 이는 공유 스토리지의 스토리지 효율성을 입증한다. 중복 데이터가 삭제되면 해당 크기가 즉시 해제된다.

본 발명의 실시예에 따른 Recordchain 2.0은 공유 해시의 효율성에 있어서 데이터에 해당하는 레코드는 분산 Recordchain 2.0 네트워크에서 삭제되거나 업데이트되지 않는다. 또한 공유 스토리지의 각 변경 사항에 대해 공유 네트워크에 새 레코드가 생성된다. 이는 Recordchain 2.0의 스토리지 비효율성처럼 보일 수 있다. 그러나 레코드 크기는 원래 데이터 크기에 비해 무시할 수 있는 크기(예를 들어, 수바이트)이다. 즉, Recordchain 2.0은 해당 레코드를 저장하여 큰 스토리지를 쉽게 활용할 수 있다.

도 5는 본 발명의 일 실시예에 따른 평가 워크로드 설정을 나타내는 도면이다.

Matlab 시뮬레이션을 수행하기 위해 도 5와 같이 3개의 평가 워크로드를 준비했다. 지정된 수명이 없는 데이터를 별도의 체인에 삽입한다고 가정하고 미리 정의된 수명 워크로드만 시뮬레이션에 사용했다. 총 250만 개의 항목이 100초에서 5000초 사이의 4900개 이상의 서로 다른 수명 동안 균일하게 배포되었다. 이 시나리오에서 각 범위에는 약 500개의 항목이 포함되었다. 모든 데이터는 도 5와 같이 세 가지 유형의 워크로드로 재배열되었다.

워크로드는 도 5(a)에서 최장 우선(longest-first), 도 5(b)에서 최단 우선(shortest-first), 도 5(c)에서 바이모달 분포(bimodal distributions)로 설계되었다.

도 6은 본 발명의 일 실시예에 따른 Matlab 시뮬레이션 결과를 나타내는 도면이다.

도 6(a)는 시나리오의 전체 결과, 도 6(b) 및 도 6(c)는 시뮬레이션 결과에 대한 자세한 정보를 보여준다. 도 6(b)는 모든 워크로드에 대한 블록 종속성의 수에 대한 최소, 최대 및 평균을 보여준다. 도 6에서 보듯이 LBD 방식은 모든 워크로드에서 최대 블록 종속성 수는 동일(~34개)하지만, 워크로드에 따라 최소(2~3), 평균(6~7)이 달랐다. FBD 결과(2와 5 사이)에서 유사한 결과를 보였다. 이러한 결과는 워크로드에 따라 그래프의 높이가 변경되지만 전반적인 추세는 동일하게 유지됨을 증명한다. Recordchain 2.0의 효율성은 도 6(c)에서 보다 명확하게 확인할 수 있으며, 다양한 작업 부하에 따른 삭제 지연이 있다. 도 6(c)는 블록 종속성의 수가 비슷하더라도 워크로드에 따라 지연이 다를 수 있음을 보여준다. LBD 방법은 워크로드 A에 대해 최악의 성능을 보였지만 워크로드 B에 대해서는 좋은 성능을 보였다. 주된 이유는 워크로드 A가 가장 긴 수명 데이터를 먼저 작성하여 다음 블록이 삽입되기 전에 초기 블록이 만료되지 않도록 보장하기 때문이다. 그러나 워크로드 B에서는 더 빠른 만료

시간으로 인해 다음 블록이 삽입되기 전에 첫 번째 블록이 삭제될 수 있다. FBD 방법은 첫 번째 블록을 삭제하는 데 중점을 두므로 Workload B의 경우 가장 긴 지연이, Workload A의 경우 가장 짧은 지연이 나타났다. 이러한 결과에 대한 추론에서 LBD 방법과 유사한 관찰을 할 수 있다. 또한 이들 방법은 Workload C와 유사한 지연을 보인다. 이 수치는 LBD 및 FBD 방법의 삭제 지연이 안정적이지 않지만 워크로드 및 평가 환경에 따라 크게 변한다는 것을 증명한다. 그러나 Recordchain 2.0은 항상 일정한 성능을 가지고 있어 동적 엡지 컴퓨팅 환경에서 완전히 신뢰할 수 있습니다.

[0093] 도 7은 본 발명의 일 실시예에 따른 평가 결과를 나타내는 도면이다.

[0094] 실제 평가를 위해 LDB 및 FBD 방법의 예로 리티체인(Littichain) 및 SDM 블록체인의 프로토타입을 선택했다. 초기 평가를 위해 Matlab 시뮬레이션에서 사용된 것과 유사한 워크로드를 사용했다. 보다 현실적이고 공정한 평가를 위해 쓰기 순서는 특정 순서 없이 무작위로 분배했다. 3개의 블록체인이 있는 랜덤 워크로드에 대한 블록 중속성 수는 도 7(a)에 도시하였다.

[0095] 리티체인은 구성 가능한 변수 k 를 사용하여 k -높이 삽입 아키텍처를 구현한다. k 에 대해 더 작은 값을 설정하면 블록의 선행 순서가 손실되는 만료 시간 순서만 따르게 되며, 이는 차례로 최신 블록을 찾을 수 있는 더 많은 공간을 제공하여 삭제 지연을 줄인다. 평가는 k 에 대해 세 가지 다른 값(∞ , 10 및 0)을 사용했다. 도 7(a)에서 보듯이 k 의 값을 줄임으로써 리티체인은 기하급수적인 속도에서 거의 일정한 성능으로 변화하면서 보다 안정적인 성능을 달성했다.

[0096] 이 3개의 블록체인에 대한 다음 평가는 삭제 기법을 사용하면서 스토리지 효율성을 위해 수행되었다. 각 워크로드를 테스트한 후 각 블록체인에 대한 저장 비용을 기록했으며 그 결과는 도 7(b)에 도시되었다. 이러한 평가에서 스토리지 비용은 데이터 수명 만료부터 다음 값이 기록될 때까지의 스토리지 사용량을 정의한다. 이것은 데이터가 기록되는 다음 데이터를 위해 비워져야 하는 경우에도 스토리지를 사용함을 의미한다. 구성 가능한 변수 k 에 대한 더 높은 값의 도움으로 리티체인은 워크로드 A 및 C에서 SDM보다 더 나은 성능을 달성했다. 그러나 여전히 임의의 워크로드를 처리할 수 없다. 일반적으로 리티체인과 SDM은 스토리지의 70% 이상을 사용했으며 효율성 측면에서 허용할 수 없는 비용을 발생시켰다. 반면 Recordchain 2.0은 공유 스토리지에서 데이터를 즉시 삭제하고 공유 블록체인에 기록을 저장했다. 모든 업데이트에 대한 총 비용은 1% 미만으로 유지되었으며, 이는 전체 데이터 크기에 비해 거의 무시할 수 있는 수준이다.

[0097] 데이터 수정 이력 브라우징을 위해 새로 채용된 OBT의 효율성을 평가하기 위해 보다 광범위한 실험이 수행되었다. 이를 위해 수정 통계의 이력 브라우징을 기반으로 세 가지 방법의 효율성을 비교했다. 알고리즘의 일반적인 관점을 얻기 위해 수정 횟수는 선형적으로 증가했으며 그 결과는 도 7(c)에 도시하였다. 첫 번째 방법(OBT)은 수정의 전체 상태를 검색하기 위해 최적의 찾아보기 테이블을 사용했다. 다음 블록은 일치하는 해시를 찾기 위해 마지막 블록에서 첫 번째 블록까지 선형 검색을 사용했다. 마지막으로 전통적인 접근 방식은 각 상태에 대한 기본 스캔 기능을 사용했다.

[0098] 공정한 평가를 위해 스캔 방법의 구현은 기존 접근 방식을 평가할 때 시작 지연을 피하기 위해 루프에서 호출되었다. 도 7(c)는 전통적인 접근 방식이 빠르게 성장하는 선형 속도로 여전히 최악의 성능을 보였다는 것을 보여준다. 선형 검색은 하나의 루프에서 모든 상태를 검색하여 하나씩 스캔하는 것보다 더 효율적인 접근 방식을 가지고 있었고 더 안정적인 성능을 달성했다. 그러나 루프를 실행하면 평균 지연이 약 2초부터 선형적으로 증가하는 네트워크 지연이 여전히 발생했다. 반면에 OBT는 실행 루프에 의존하지 않고 테이블에서 정확한 인덱스 위치를 가져온다. 지연은 특정 인덱스에서 레코드에 대한 액세스 시간에 의해서만 발생한다. OBT 구현으로 Recordchain 2.0은 데이터 수정 내역을 검색할 때 1초 미만의 지연을 달성했다. 커넥티드 차가 있는 엡지 컴퓨팅 환경을 고려하면 모든 데이터가 지속적으로 업데이트되며, 이력을 빠르게 탐색하면 전체 엔드 투 엔드 대기 시간을 줄이는 데 도움이 된다. 따라서 OBT 방식은 엡지 컴퓨팅 요구 사항을 완전히 충족할 수 있다.

[0100] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적

컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0101] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 구체화(embodiment)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

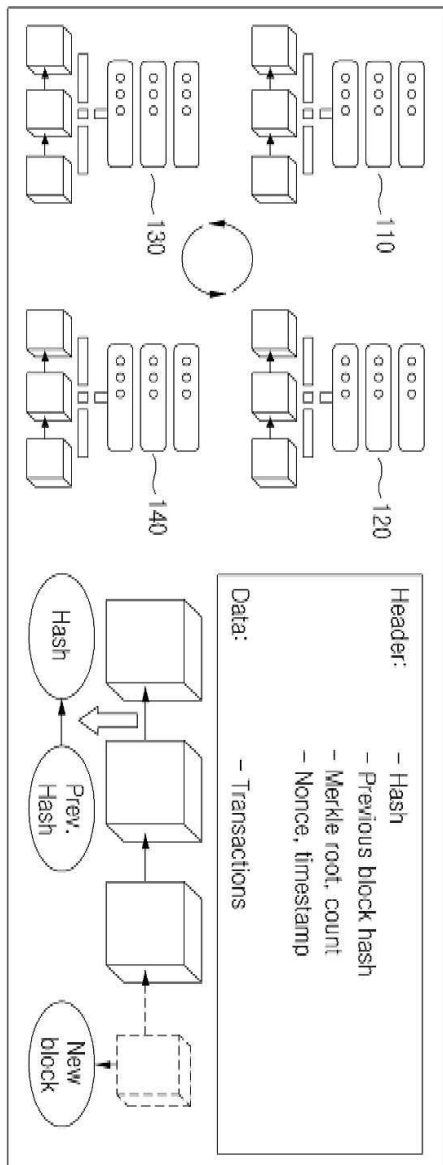
[0102] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다.

[0103] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

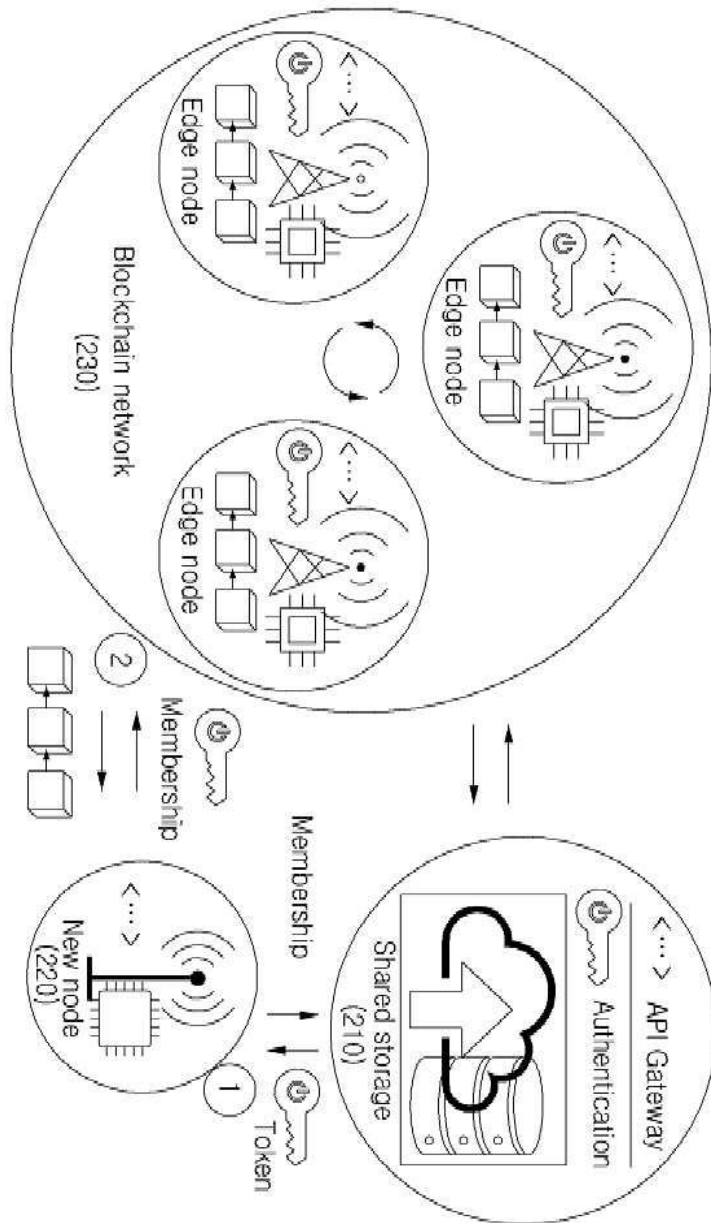
[0104] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

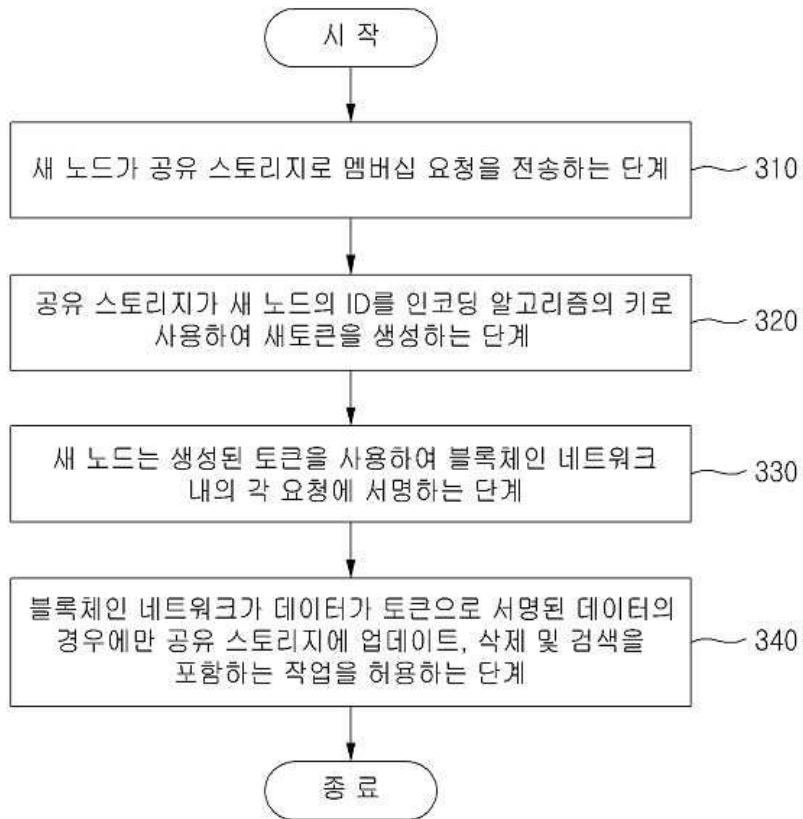
도면1



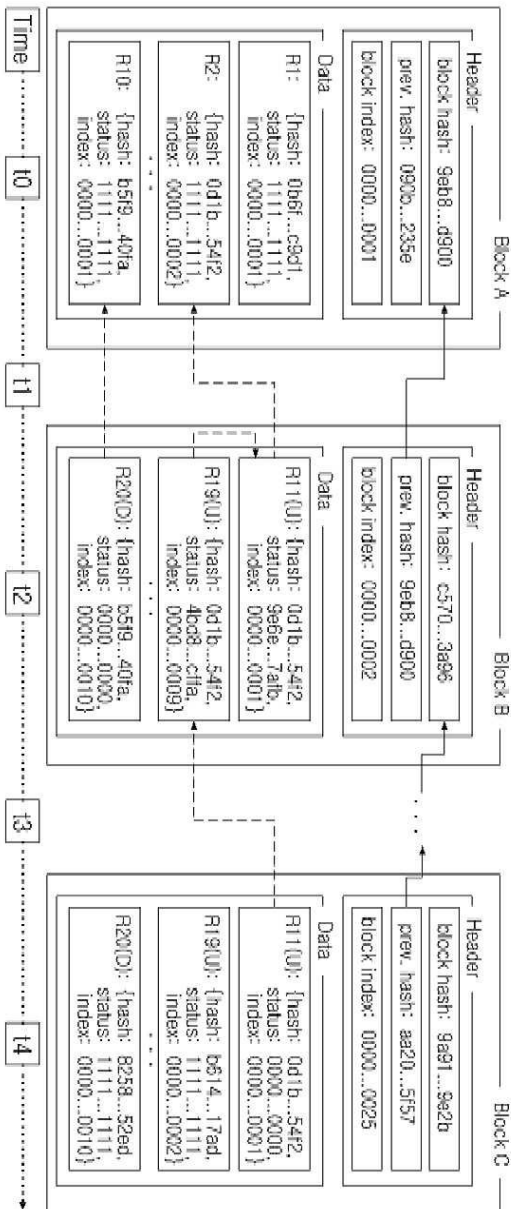
도면2



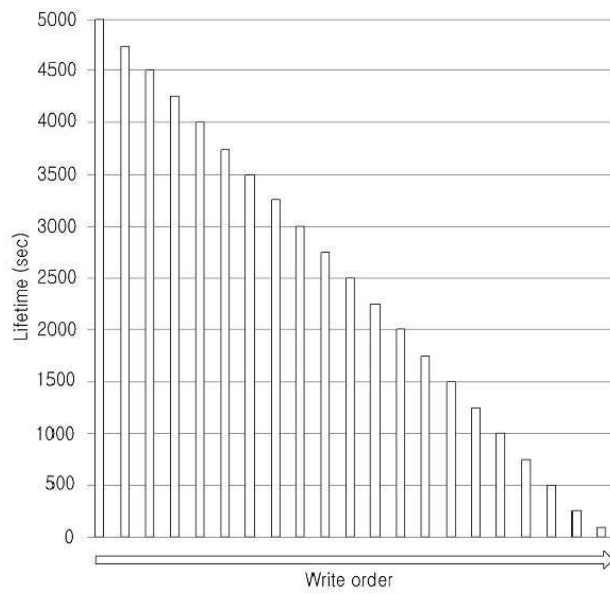
도면3



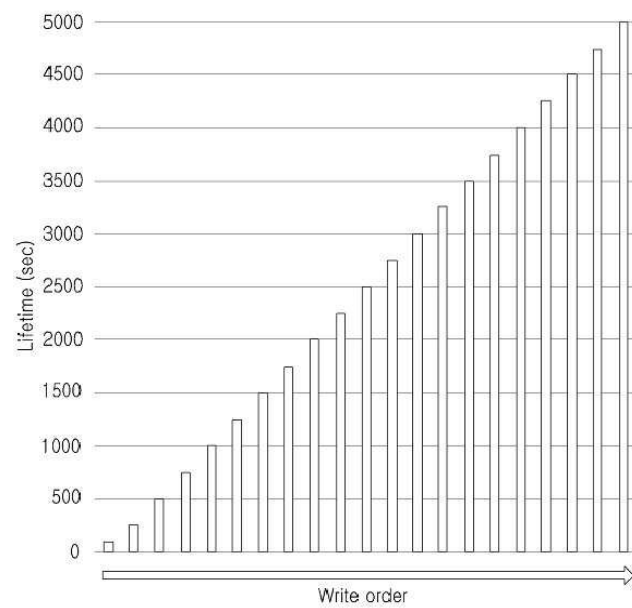
도면4



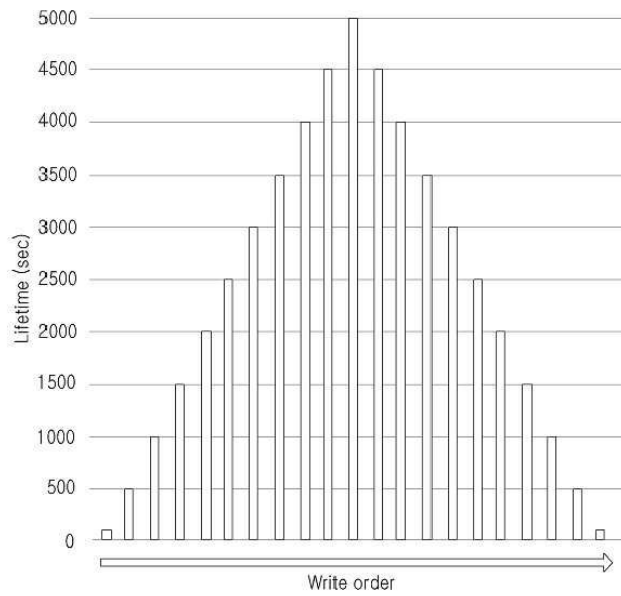
도면5a



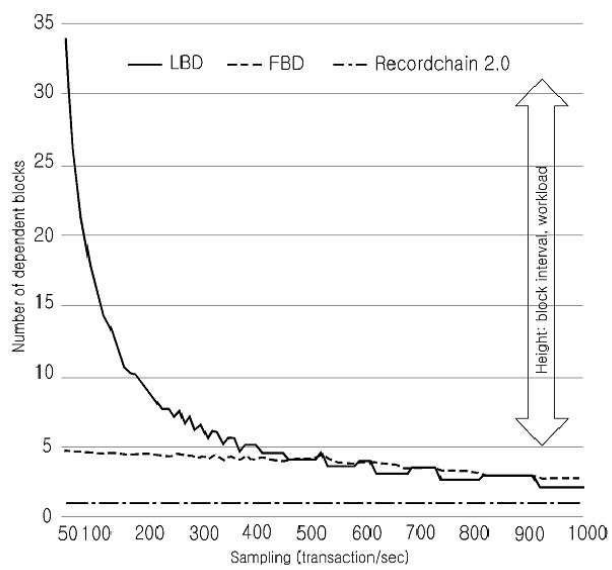
도면5b



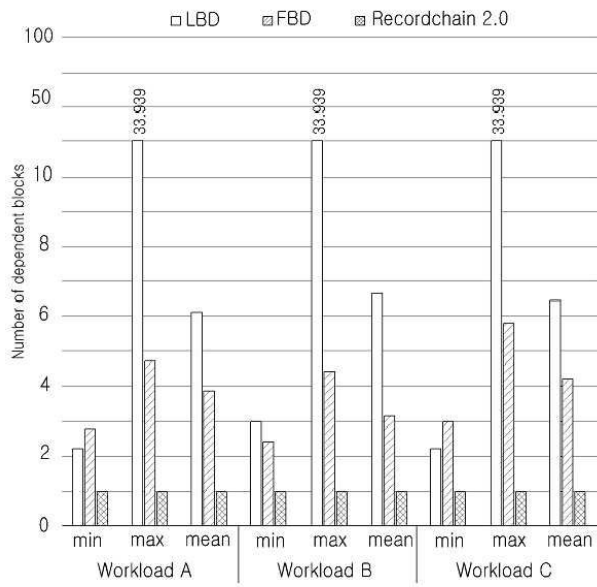
도면5c



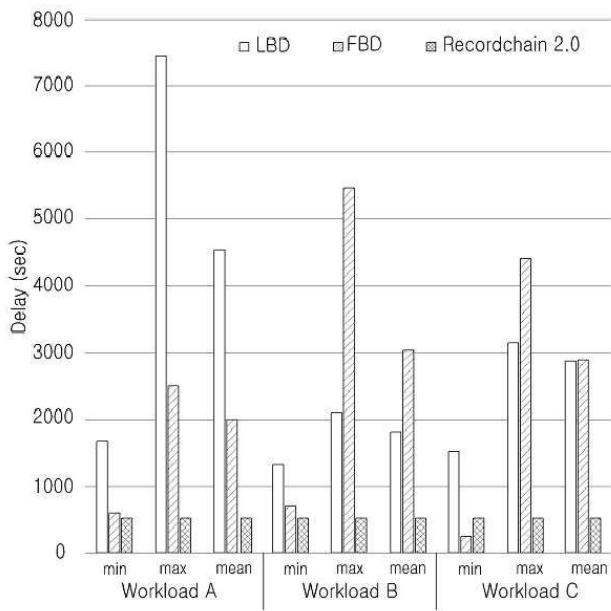
도면6a



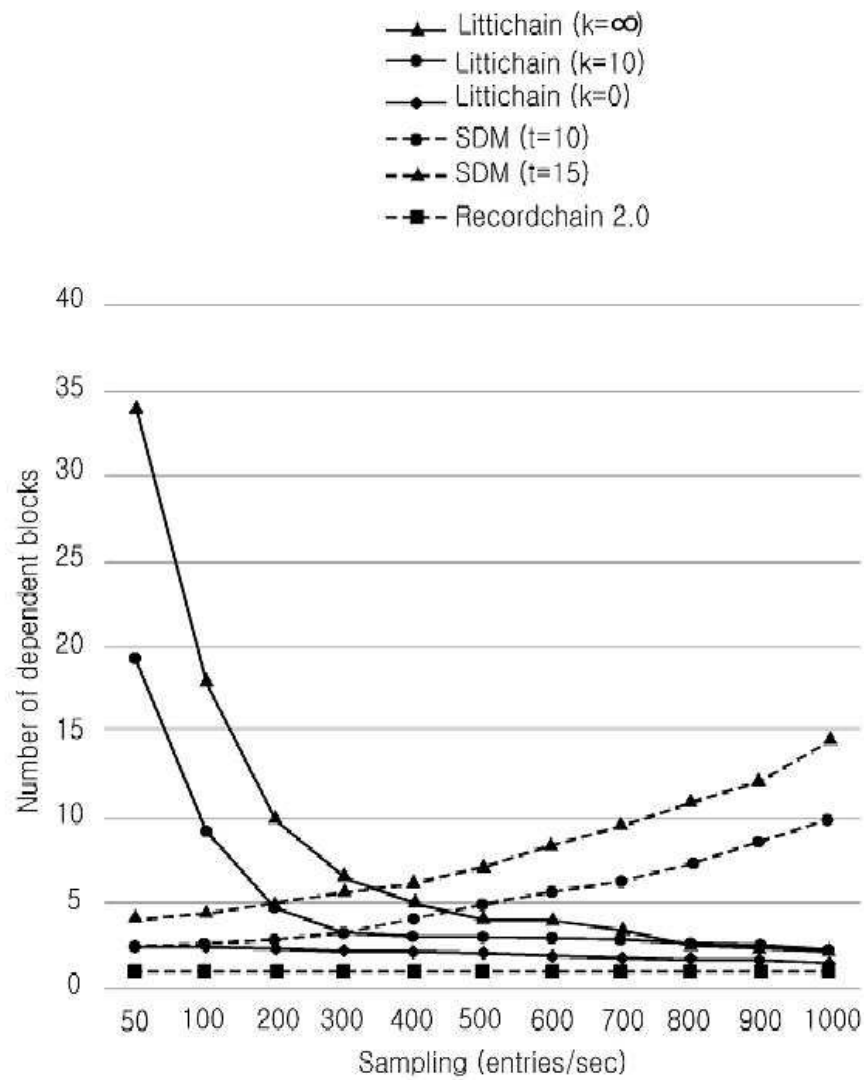
도면6b



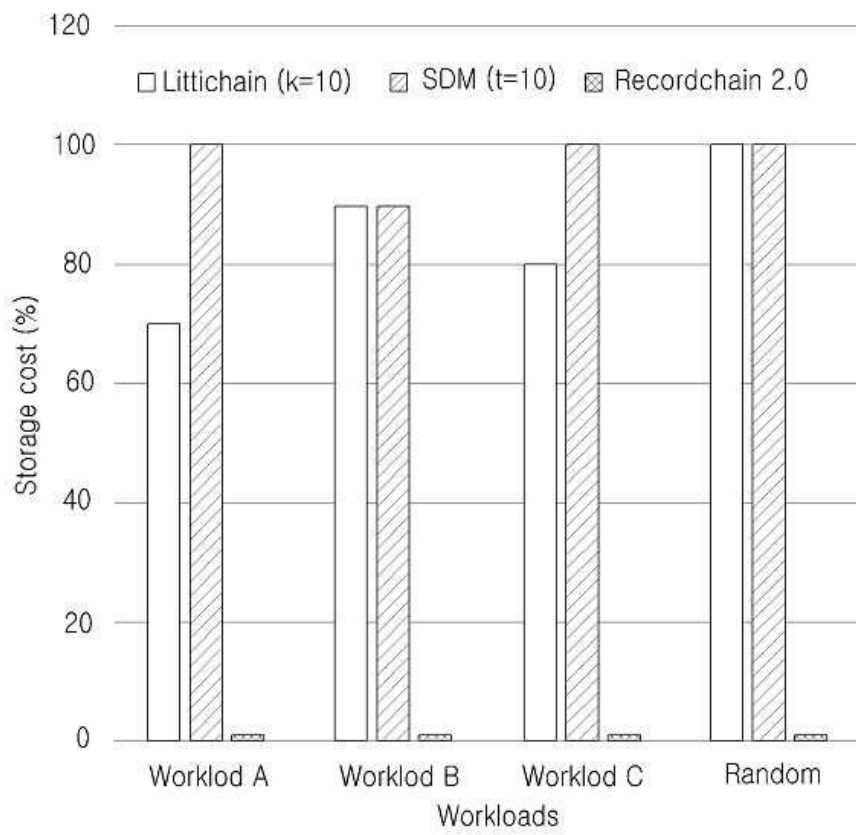
도면6c



도면7a



도면7b



도면7c

