



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2024년08월16일
(11) 등록번호 10-2695624
(24) 등록일자 2024년08월12일

(51) 국제특허분류(Int. Cl.)
H04L 67/10 (2022.01) G06F 16/27 (2019.01)
G06F 9/50 (2018.01) H04L 67/289 (2022.01)
(52) CPC특허분류
H04L 67/10 (2022.05)
G06F 16/27 (2019.01)
(21) 출원번호 10-2023-0085950
(22) 출원일자 2023년07월03일
심사청구일자 2023년07월03일
(56) 선행기술조사문헌
KR1020200076714 A
(뒷면에 계속)

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

전체 청구항 수 : 총 8 항

심사관 : 이종익

(54) 발명의 명칭 엣지 컴퓨팅을 위한 삭제기반 탱글 시스템 및 그 동작 방법

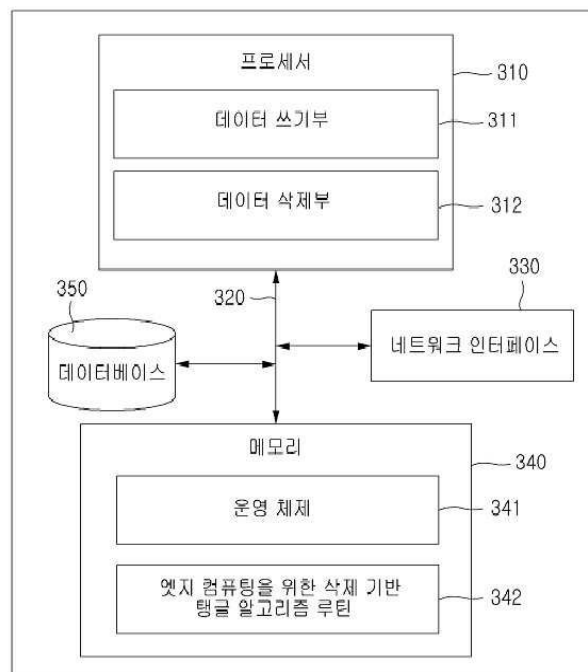
(57) 요약

엣지 컴퓨팅을 위한 삭제기반 탱글 시스템 및 그 동작 방법이 제시된다. 본 발명에서 제안하는 엣지 컴퓨팅을 위한 삭제기반 탱글 시스템의 동작 방법은 엣지 컴퓨팅을 위한 삭제기반 탱글 알고리즘의 현재 상태와 새 노드를 입력 파라미터로 수신하는 단계, 상기 현재 상태에서 검증되지 않은 노드를 획득하여 상기 노드의 생성 시간에

(뒷면에 계속)

대표도 - 도3

300



따라 순서대로 정렬한 후 상기 정렬된 검증되지 않은 노드의 모든 파라미터에 대하여 만료 시간을 포함하는 미리 정해진 검증 제약 조건을 확인하는 단계, 상승(clime-up) 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 만료 시간이 이미 만료된 노드의 경우 제거되어야 하므로, 상기 만료된 노드의 만료 시간보다 만료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 단계 및 노드 생성 시 만료 시간이 제공되는 노드, 만료 시간이 없는 노드 및 만료 시간이 지정되지 않은 노드에 따라 저장 또는 노드 삭제를 수행하도록 하는 단계를 포함한다.

(52) CPC특허분류

G06F 9/5061 (2013.01)

H04L 67/289 (2022.05)

(56) 선행기술조사문헌

US20210091957 A1

KR102586477 B1

US20210091957 A1

KR1020220044306 A

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 7/10

과제수행기관명 인하대학교

연구기간 2023.03.01 ~ 2024.02.29

이 발명을 지원한 국가연구개발사업

과제고유번호 1415183132

과제번호 P0017124

부처명 산업통상자원부

과제관리(전문)기관명 한국산업기술진흥원

연구사업명 신산업융합형임베디드시스템전문인력양성사업

연구과제명 신산업융합 생태계 구축을 위한 인공지능 기반 임베디드시스템고급인력양성(3

차년도)

기 여 율 3/10

과제수행기관명 (사단)한국정보산업연합회

연구기간 2023.03.01 ~ 2024.02.29

공지예외적용 : 있음

명세서

청구범위

청구항 1

엣지 컴퓨팅을 위한 삭제기반 탱글 알고리즘의 현재 상태와 새 노드를 입력 파라미터로 수신하는 단계;

상기 현재 상태에서 검증되지 않은 노드를 획득하여 상기 노드의 생성 시간에 따라 순서대로 정렬한 후 상기 정렬된 검증되지 않은 노드의 모든 파라미터에 대하여 만료 시간을 포함하는 미리 정해진 검증 제약 조건을 확인하는 단계;

상승(clime-up) 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 만료 시간이 이미 만료된 노드의 경우 제거되어야 하므로, 상기 만료된 노드의 만료 시간보다 만료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 단계; 및

노드 생성 시 만료 시간이 제공되는 노드, 만료 시간이 없는 노드 및 만료 시간이 지정되지 않은 노드에 따라 저장 또는 노드 삭제를 수행하도록 하는 단계

를 포함하는 분산 컴퓨팅 방법.

청구항 2

제1항에 있어서,

상승 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 만료 시간이 이미 만료된 노드의 경우 제거되어야 하므로, 상기 만료된 노드의 만료 시간보다 만료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 단계는,

상기 새 노드가 만료 시간이 제공되는 유한 유형인 경우, 상승 쓰기 기법을 이용한 유한 유형 노드 삽입을 위해 새 노드를 검증되지 않은 노드에 대하여 확인하고, 새 노드의 만료 시간이 검증되지 않은 모든 노드의 만료 시간보다 긴 경우 해당 노드의 부모 노드로 올라가 부모 노드의 만료 시간과 비교하고, 상기 부모 노드의 만료 시간과 비교하는 과정을 반복 수행하는

분산 컴퓨팅 방법.

청구항 3

제1항에 있어서,

상승 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 만료 시간이 이미 만료된 노드의 경우 제거되어야 하므로, 상기 만료된 노드의 만료 시간보다 만료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 단계는,

상기 새 노드가 만료 시간이 없는 노드는 만료 시간 불변 유형인 경우, 상승 쓰기 기법을 이용한 불변 유형 노드 삽입을 위해 모든 리프 노드의 만료 시간이 제한되므로 상기 새 노드는 원하는 위치의 부모 노드까지 올라가고,

상기 새 노드가 만료 시간이 지정되지 않은 알 수 없음(Unknown) 유형인 경우, 상승 쓰기 기법을 이용한 알 수 없음 유형 노드 삽입을 위해, 마지막으로 삽입된 노드는 불변 유형이므로 해당 노드의 부모 노드 중 하나를 상기 검증되지 않은 노드에서 찾고, 또 다른 부모 노드는 더 높은 위치로 올라가 다른 불변 유형 노드를 찾는

분산 컴퓨팅 방법.

청구항 4

제1항에 있어서,

상기 노드 생성 시 만료 시간이 제공되는 노드, 만료 시간이 없는 노드 및 만료 시간이 지정되지 않은 노드에

따라 해당 노드의 저장 또는 노드 삭제를 수행하도록 하는 단계는,

만료 시간이 없는 노드는 만료 시간 불변 유형으로서 영구히 저장해야 하고, 만료 시간이 제공되는 노드는 만료 시간 유한 유형이며, 상기 유한 유형 노드 및 불변 유형 노드는 산술을 통해 계산하고, 계산 결과에 따라 해당 노드의 저장 또는 노드 삭제를 수행하고,

유한 유형 노드를 저장하는 경우 유한 유형 노드의 만료 시간에 따라 순서대로 저장되며, 만료 시간이 지정되지 않은 노드는 알 수 없음(Unknown) 유형으로서 만료되거나 또는 만료되지 않고,

알 수 없음 유형 노드를 삭제하는 경우, 알 수 없음 유형 노드를 삭제하기 위해 해당 노드의 부모 노드를 삭제하고, 자식 노드가 만료될 때까지 기다리는

분산 컴퓨팅 방법.

청구항 5

엣지 컴퓨팅을 위한 삭제기반 탱글 알고리즘의 현재 상태와 새 노드를 입력 파라미터로 수신하고, 상기 현재 상태에서 검증되지 않은 노드를 획득하여 상기 노드의 생성 시간에 따라 순서대로 정렬한 후 상기 정렬된 검증되지 않은 노드의 모든 파라미터에 대하여 만료 시간을 포함하는 미리 정해진 검증 제약 조건을 확인하며, 상승(clime-up) 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 만료 시간이 이미 만료된 노드의 경우 제거되어야 하므로, 상기 만료된 노드의 만료 시간보다 만료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 데이터 쓰기부; 및

노드 생성 시 만료 시간이 제공되는 노드, 만료 시간이 없는 노드 및 만료 시간이 지정되지 않은 노드에 따라 저장 또는 노드 삭제를 수행하도록 하는 데이터 삭제부

를 포함하는 분산 컴퓨팅 시스템.

청구항 6

제5항에 있어서,

상기 데이터 쓰기부는,

상기 새 노드가 만료 시간이 제공되는 유한 유형인 경우, 상승 쓰기 기법을 이용한 유한 유형 노드 삽입을 위해 새 노드를 검증되지 않은 노드에 대하여 확인하고, 새 노드의 만료 시간이 검증되지 않은 모든 노드의 만료 시간보다 긴 경우 해당 노드의 부모 노드로 올라가 부모 노드의 만료 시간과 비교하고, 상기 부모 노드의 만료 시간과 비교하는 과정을 반복 수행하는

분산 컴퓨팅 시스템.

청구항 7

제5항에 있어서,

상기 데이터 쓰기부는,

상기 새 노드가 만료 시간이 없는 노드는 만료 시간 불변 유형인 경우, 상승 쓰기 기법을 이용한 불변 유형 노드 삽입을 위해 모든 리프 노드의 만료 시간이 제한되므로 상기 새 노드는 원하는 위치의 부모 노드까지 올라가고,

상기 새 노드가 만료 시간이 지정되지 않은 알 수 없음(Unknown) 유형인 경우, 상승 쓰기 기법을 이용한 알 수 없음 유형 노드 삽입을 위해, 마지막으로 삽입된 노드는 불변 유형이므로 해당 노드의 부모 노드 중 하나를 상기 검증되지 않은 노드에서 찾고, 또 다른 부모 노드는 더 높은 위치로 올라가 다른 불변 유형 노드를 찾는

분산 컴퓨팅 시스템.

청구항 8

제5항에 있어서,

상기 데이터 삭제부는,

만료 시간이 없는 노드는 만료 시간 불변 유형으로서 영구히 저장해야 하고, 만료 시간이 제공되는 노드는 만료 시간 유한 유형이며, 상기 유한 유형 노드 및 불변 유형 노드는 산술을 통해 계산하고, 계산 결과에 따라 해당 노드의 저장 또는 노드 삭제를 수행하고,

유한 유형 노드를 저장하는 경우 유한 유형 노드의 만료 시간에 따라 순서대로 저장되며, 만료 시간이 지정되지 않은 노드는 알 수 없음(Unknown) 유형으로서 만료되거나 또는 만료되지 않고,

알 수 없음 유형 노드를 삭제하는 경우, 알 수 없음 유형 노드를 삭제하기 위해 해당 노드의 부모 노드를 삭제하고, 자식 노드가 만료될 때까지 기다리는

분산 컴퓨팅 시스템.

발명의 설명

기술 분야

[0001] 본 발명은 엣지 컴퓨팅을 위한 삭제기반 탱글 시스템 및 그 동작 방법에 관한 것이다.

배경 기술

[0002] 최근 블록체인 기술은 데이터 보안 및 무결성 측면에서 그 효과가 입증되었다. 암호화폐의 성공은 실생활 응용 프로그램으로서 블록 체인의 최적의 예시이다. 블록체인은 네트워크의 모든 노드에 동일한 데이터를 공유할 수 있는 간단한 프로토콜을 제공한다. 새 데이터가 기록될 때마다 새 블록을 사용하여 이전 상태가 확장된다. 따라서 가장 긴 블록체인을 소유한 사람은 최신 버전의 데이터베이스를 공유한다. 이 간단한 규칙을 따르면 모든 노드는 전체 블록체인의 전체 복사본을 획득한다. 공격자가 블록체인에서 일부 데이터를 변경하려고 시도하는 경우 변경된 버전을 다른 노드와 공유해야 한다. 네트워크에는 이미 원본 버전이 있으므로 변경된 복사본은 쉽게 거부될 수 있다. 또한 대부분의 블록체인은 데이터베이스 공격을 피하기 위해 키-값 저장소를 기본 저장소로 사용한다. 집중적인 연구에 따라 블록체인 기술은 공급망 관리, 클라우드 컴퓨팅에 대한 데이터 프라이버시 제어, 의료 시스템 및 엣지 컴퓨팅과 같은 수많은 분야에 적용되었다.

[0003] 블록체인의 통합이 여러 분야에 도움이 되었지만, 완전한 이점을 얻지 못하는 상당한 기술 불일치가 존재한다. 엣지 컴퓨팅, 클라우드 컴퓨팅 및 데이터 수집 프레임워크와 같은 대규모 데이터 처리 시스템은 일반적으로 예측할 수 없는 데이터를 관리하며, 여기서 대부분은 수명이 짧은 중복 데이터를 이용한다. 데이터를 변경할 수 없는 블록체인에 이를 저장하면 효율성 문제가 발생할 수 있다. 따라서 블록체인 프로토콜은 데이터 삭제를 가능하게 하고 빅 데이터 요구 사항을 충족하도록 변경되어야 한다. 그러나 블록체인에서 데이터 삭제를 허용하는 것은 보안이 핵심인 블록체인의 가장 기본적인 규칙을 심각하게 위반한다. 따라서 빅 데이터 처리를 위해 블록체인을 활성화하는 데 큰 연구 격차가 존재한다. 이러한 문제를 해결하기 위해 이미 블록체인에서 데이터 삭제를 가능하게 하는 여러 최첨단 프로젝트를 도입했다. 제안된 프로젝트의 관찰을 기반으로 블록체인 데이터 삭제 기술을 블록과 선택적 삭제로 분류할 수 있다.

[0004] 분산 컴퓨팅은 투명성 때문에 항상 관심의 중심에 있었다. 공용 네트워크가 데이터 검증에 대한 합의를 달성할 수 있도록 허용하면 타사가 제품 관리, 금전 거래 또는 인증과 같은 다양한 실제 애플리케이션에 참여하는 것을 방지할 수 있다. 그러나 분산 노드를 관리하는 것은 계산을 활성화하는 데 있어 중요한 과제이다. 블록체인과 직접 비순환 그래프의 도입에 따라 최근 이 분야에서 상당한 돌파구가 마련되었다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) [1] IOTA Research Papers. Online available: <https://www.iota.org/foundation/research-papers>. Last accesses: Sep 19, 2022.

(특허문헌 0002) [2] 26. Pyoung, C. K.; Baek, S. J. Blockchain of Finite-Lifetime Blocks With Applications to Edge-Based IoT. IEEE Internet of Things Journal 2020, vol. 7, no. 3, pp. 2102-2116, doi: 10.1109/JIOT.2019.2959599.

발명의 내용

해결하려는 과제

- [0006] 본 발명이 이루고자 하는 기술적 과제는 탱글 아키텍처에서 데이터 삭제를 가능하게 하고, 트랜잭션을 완료 기능에 따라 세 가지 범주로 나누고 상승 쓰기 기법을 사용하는 엡지 컴퓨팅을 위한 삭제기반 탱글 시스템 및 그 동작 방법을 제공하는데 있다. 본 발명의 실시예에 따른 D-탱글이라고 하는 새로운 접근 방식을 통해 유한한 수명 데이터에서 즉각적인 삭제를 가능하게 하고, 알 수 없는 수명 데이터에 대해 요청 시 불변성 및 삭제가 보장 되도록 하고자 한다.

과제의 해결 수단

- [0007] 일 측면에 있어서, 본 발명에서 제안하는 엡지 컴퓨팅을 위한 삭제기반 탱글 시스템의 동작 방법은 엡지 컴퓨팅을 위한 삭제기반 탱글 알고리즘의 현재 상태와 새 노드를 입력 파라미터로 수신하는 단계, 상기 현재 상태에서 검증되지 않은 노드를 획득하여 상기 노드의 생성 시간에 따라 순서대로 정렬한 후 상기 정렬된 검증되지 않은 노드의 모든 파라미터에 대하여 완료 시간을 포함하는 미리 정해진 검증 제약 조건을 확인하는 단계, 상승(clime-up) 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 완료 시간이 이미 완료된 노드의 경우 제거되어야 하므로, 상기 완료된 노드의 완료 시간보다 완료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 단계 및 노드 생성 시 완료 시간이 제공되는 노드, 완료 시간이 없는 노드 및 완료 시간이 지정되지 않은 노드에 따라 저장 또는 노드 삭제를 수행하도록 하는 단계를 포함한다.
- [0008] 상승 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 완료 시간이 이미 완료된 노드의 경우 제거되어야 하므로, 상기 완료된 노드의 완료 시간보다 완료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 단계는 상기 새 노드가 완료 시간이 제공되는 유한 유형인 경우, 상승 쓰기 기법을 이용한 유한 유형 노드 삽입을 위해 새 노드를 검증되지 않은 노드에 대하여 확인하고, 새 노드의 완료 시간이 검증되지 않은 모든 노드의 완료 시간보다 긴 경우 해당 노드의 부모 노드로 올라가 부모 노드의 완료 시간과 비교하고, 상기 부모 노드의 완료 시간과 비교하는 과정을 반복 수행한다.
- [0009] 상승 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 완료 시간이 이미 완료된 노드의 경우 제거되어야 하므로, 상기 완료된 노드의 완료 시간보다 완료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 단계는 상기 새 노드가 완료 시간이 없는 노드는 완료 시간 불변 유형인 경우, 상승 쓰기 기법을 이용한 불변 유형 노드 삽입을 위해 모든 리프 노드의 완료 시간이 제한되므로 상기 새 노드는 원하는 위치의 부모 노드까지 올라가고, 상기 새 노드가 완료 시간이 지정되지 않은 알 수 없음(Unknown) 유형인 경우, 상승 쓰기 기법을 이용한 알 수 없음 유형 노드 삽입을 위해, 마지막으로 삽입된 노드는 불변 유형이므로 해당 노드의 부모 노드 중 하나를 상기 검증되지 않은 노드에서 찾고, 또 다른 부모 노드는 더 높은 위치로 올라가 다른 불변 유형 노드를 찾는다.
- [0010] 상기 노드 생성 시 완료 시간이 제공되는 노드, 완료 시간이 없는 노드 및 완료 시간이 지정되지 않은 노드에 따라 해당 노드의 저장 또는 노드 삭제를 수행하도록 하는 단계는 완료 시간이 없는 노드는 완료 시간 불변 유형으로서 영구히 저장해야 하고, 완료 시간이 제공되는 노드는 완료 시간 유한 유형이며, 상기 유한 유형 노드 및 불변 유형 노드는 산술을 통해 계산하고, 계산 결과에 따라 해당 노드의 저장 또는 노드 삭제를 수행하고, 유한 유형 노드를 저장하는 경우 유한 유형 노드의 완료 시간에 따라 순서대로 저장되며, 완료 시간이 지정되지 않은 노드는 알 수 없음(Unknown) 유형으로서 완료되거나 또는 완료되지 않고, 알 수 없음 유형 노드를 삭제하는 경우, 알 수 없음 유형 노드를 삭제하기 위해 해당 노드의 부모 노드를 삭제하고, 자식 노드가 완료될 때까지 기다린다.
- [0011] 또 다른 일 측면에 있어서, 본 발명에서 제안하는 엡지 컴퓨팅을 위한 삭제기반 탱글 시스템은 엡지 컴퓨팅을 위한 삭제기반 탱글 알고리즘의 현재 상태와 새 노드를 입력 파라미터로 수신하고, 상기 현재 상태에서 검증되지 않은 노드를 획득하여 상기 노드의 생성 시간에 따라 순서대로 정렬한 후 상기 정렬된 검증되지 않은 노드의 모든 파라미터에 대하여 완료 시간을 포함하는 미리 정해진 검증 제약 조건을 확인하며, 상승(clime-up) 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 완료 시간이 이미 완료된 노드의 경우 제거되어야 하므로, 상기 완료된 노드의 완료 시간보다 완료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 데이터 쓰기부 및 노드 생성 시 완료 시간이 제공되는 노드, 완료 시간이 없는 노드 및 완료 시간이 지정되지 않은 노드에 따라 저장 또는 노드 삭제를 수행하도록 하는 데이터 삭제부를 포함한다.

발명의 효과

[0012] 본 발명의 실시예들에 따르면 탱글 아키텍처에서 데이터 삭제를 가능하게 하는 D-탱글이라고 하는 새로운 접근 방식을 통해 트랜잭션을 만료 기능에 따라 세 가지 범주로 나누고 상승 쓰기 기법을 사용한다. 본 발명의 실시예에 따르면, D-탱글이 유한한 수명 데이터에서 즉각적인 삭제를 가능하게 하고, 알 수 없는 수명 데이터에 대해 요청 시 불변성 및 삭제가 보장될 수 있다.

도면의 간단한 설명

[0013] 도 1은 종래기술에 따른 블록체인 아키텍처를 나타내는 도면이다.
 도 2는 본 발명의 일 실시예에 따른 탱글 구조 및 IOTA 네트워크를 설명하기 위한 도면이다.
 도 3은 본 발명의 일 실시예에 따른 엡지 컴퓨팅을 위한 삭제기반 탱글 시스템의 구성을 나타내는 도면이다.
 도 4는 본 발명의 일 실시예에 따른 엡지 컴퓨팅을 위한 삭제기반 탱글 시스템의 동작 방법을 설명하기 위한 흐름도이다.
 도 5는 본 발명의 일 실시예에 따른 D-탱글의 상승 쓰기 기법을 설명하기 위한 도면이다.
 도 6은 본 발명의 일 실시예에 따른 상승 쓰기 기법의 알고리즘을 나타내는 도면이다.
 도 7은 본 발명의 일 실시예에 따른 평가를 위한 워크로드 설정을 나타내는 도면이다.
 도 8은 본 발명의 일 실시예에 따른 평가 결과를 나타내는 도면이다.
 도 9는 본 발명의 일 실시예에 따른 스토리지 사용량을 나타내는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0014] IOTA 탱글(Tangle)은 기존 블록체인과 경쟁할 수 있는 기능을 갖춘 분산 원장 기술에 대한 유망한 접근 방식을 제공한다. 사물인터넷(Internet of Things; IoT) 환경에서 소액결제를 가능하게 하기 위해 IOTA는 거래의 무결성과 불변성을 보장하는 직접 아크릴 그래프를 사용한다. 그러나 IoT 데이터는 시간에 민감하여 일정 기간이 지나면 값이 손실된다. 불변 저장을 위해 이러한 임시 데이터를 저장하는 것은 분산 원장에서 감당할 수 없다. 본 발명에서는 탱글 아키텍처에서 데이터 삭제를 가능하게 하는 D-탱글이라고 하는 새로운 접근 방식을 제안한다. 이 목표를 달성하기 위해 D-탱글은 트랜잭션을 만료 기능에 따라 세 가지 범주로 나누고 상승 쓰기 기법을 사용한다. 본 발명의 실시예에 따른 광범위한 평가는 D-탱글이 유한한 수명 데이터에서 즉각적인 삭제를 가능하게 한다는 것을 증명한다. 또한, 알 수 없는 수명 데이터에 대해 요청 시 불변성 및 삭제가 보장된다. 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0016] 도 1은 종래기술에 따른 블록체인 아키텍처를 나타내는 도면이다.

[0017] 분산 컴퓨팅 시스템 중에서 블록체인은 무한한 가능성과 다양한 애플리케이션을 가진 가장 인기 있는 프로토콜이 되었다. 블록체인은 노드가 일련의 규칙을 따라 합의를 달성하는 공용 네트워크를 가능하게 한다. 도 1은 블록체인 아키텍처의 일반적인 개요를 보여준다. 전통적으로 주요 데이터 단위는 암호화페 거래이다. 트랜잭션이 대량으로 생성되고 블록체인이 네트워크를 통한 대중의 합의에 의존하기 때문에 각 트랜잭션을 개별적으로 처리하는 것은 어렵다. 대신, 블록체인은 블록의 여러 트랜잭션을 그룹화하고 연결된 블록 체인을 만든다(120). 체인의 블록들(121, 122, 123)은 작성 시간에 따라 순서가 매겨지며, 최신 블록(123)은 마지막으로 작성된다. 보안을 위해 블록체인은 도 1에 도시된 바와 같이 P2P 네트워크 노드(111, 112, ..., 116)에 의존한다. 모든 노드가 동일한 블록체인 복사본을 공유한다. 이기종 노드와 동적 노드의 보안 및 안정성을 유지하는 것은 새로운 블록 생성 사이에 지연을 초래하는 복잡한 프로세스이다. 트랜잭션이 예기치 않은 방식으로 생성되면 두 블록 사이에 생성된 트랜잭션이 보류 중인 트랜잭션(130)이라고 하는 데이터 구조에 기록된다. 시스템 요구사항에 따라 블록체인 아키텍처는 다양한 유형의 메커니즘을 사용하여 블록을 생성한다. 비트코인과 이더리움과 같은 가장 유명한 블록체인은 마이닝(Mining)이라는 프로세스에 의존한다. 블록을 만드는 것과 관련하여, 간격이 도착하면 노드는 보류 중인 트랜잭션(130) 목록에서 트랜잭션을 읽고 블록 해시(Hash)를 계산하기 위해 무거운 수학적 퍼즐을 풀어 본다. 퍼즐은 노드가 트랜잭션의 고정 순서와 블록에 대한 고유 해시를 계산하도록 설계되었다. 비트코인에서 노드는 퍼즐을 푸는 데 약 10분이 걸릴 수 있다. 퍼즐을 푼 노드는 결과를 전체 네트워크와 공유하고 확인을 기다린다. 다른 노드가 브로드캐스트 메시지를 수신하면 결과를 확인하고 솔루션이 참인지 확인한다. 다

수의 노드로부터 확인을 받은 후, 소유자 노드는 체인에 새 블록을 추가할 수 있는 권한을 추가로 얻는다. 새 블록이 추가되면 체인의 새 버전이 네트워크에 다시 브로드캐스트된다. 무결성을 위해 노드는 체인이 이전 버전보다 긴 경우에만 허용되는 가장 긴 체인 규칙을 따른다. 두 개 이상의 노드가 동시에 새 블록을 브로드캐스트하면 체인이 긴 노드가 합의를 얻고 네트워크에 의해 확인된다. 블록체인에 대한 노력과 기여와 관련하여 소유자 노드는 암호화폐(다시 말해, 비트코인) 형태의 인센티브를 획득한다. 또한 신규 블록에서 발생하는 모든 거래 수수료는 확인 인센티브로 네트워크 노드에 분배된다. 블록체인 합의는 무거운 계산과 검증을 기반으로 하기 때문에 전체 프로세스를 작업 증명(Proof-of-Work; PoW)이라고도 한다. 임의의 컴퓨터 또는 서버는 PoW 합의를 따르는 한 공용 블록체인 네트워크에 가입할 수 있다. 악의적인 노드가 합의, 규칙 및 트랜잭션 동작을 변경하려고 하면 네트워크에서 업데이트를 쉽게 거부한다.

[0019] 도 2는 본 발명의 일 실시예에 따른 탱글 구조 및 IOTA 네트워크를 설명하기 위한 도면이다.

[0020] 블록체인 합의는 많은 계산이 수반되는 마이닝 프로세스를 기반으로 하기 때문에 채굴자는 보상을 얻기 위해 네트워크에 참여하도록 유인된다. 또한 거래 수수료와 암호화폐에 대한 수요가 동시에 크게 증가한다. 이로 인해 가장 인기 있는 네트워크는 요금이 전송 금액을 초과하는 소액 거래로 인해 감당할 수 없게 된다. 게다가, 블록체인 거래는 블록 생성에 대한 의존도가 높다. 또한, 비트코인은 초당 최대 10개의 거래만 수행할 수 있다. 현재 기술에 대한 수요를 고려할 때, 대부분의 사용자들은 낮은 속도에서 높은 요금을 용인하지 않는다. 이러한 문제를 해결하기 위해 IOTA가 대체 분산 네트워크로 도입되었다. IOTA는 블록체인이 아니지만 직접 비순환 그래프(Direct-Acyclic Graph; DAG)를 기반으로 한다. DAG는 스토리지 시스템으로 사용되는 포레스트(forest) 구조의 한 유형이다. 이름에서 알 수 있듯이 각 예지는 특정 방향(방향성(directed))을 가지며 노드 간에 루프가 허용되지 않는다(비순환(acyclic)). IOTA는 탱글이라는 DAG 기반 구조를 도입한다. IOTA는 블록체인과 비교하여 블록을 채굴하여 거래를 확인하는 것이 아니라 각 거래를 개별적으로 처리한다. 탱글에서 각 트랜잭션은 그래프의 노드로 간주되며 가장자리는 트랜잭션 검증을 지칭한다. 탱글은 합의를 이루기 위해 두 가지 간단한 규칙을 사용한다:

[0021] 1. 각 새 트랜잭션은 두 개의 이전 트랜잭션을 확인해야 한다.

[0022] 2. 확인 후 거래가 확인된다.

[0023] 이러한 규칙에 따라 도 2(a)는 샘플 탱글의 그래프를 보여준다. 우선, 첫 번째 역할을 하는 제네시스 트랜잭션(G)만 존재한다. 규칙 1에 따라 각 트랜잭션은 두 개의 이전 트랜잭션을 확인해야 한다. 그렇지 않으면 네트워크에서 확인할 수 없다. 규칙 2에 따르면, 거래는 보통 두 가지 확인만 한다. 그러나 때때로 더 많은 확인이 발생할 수 있다. 블록체인과 관련하여, 사용자들은 보통 5~6개의 확인 블록을 기다려 거래의 보안을 보장한다. 탱글의 경우 각 트랜잭션에는 가중치(w)라는 특정 숫자가 할당된다. 새 트랜잭션이 저장될 때마다 이전 트랜잭션과 상위 트랜잭션을 확인하는 누적 가중치에 영향을 준다. 도 2(a)에 나타난 바와 같이, 모든 거래는 동일한 가중치, 즉 1을 나타낸다. 새 트랜잭션이 이전 트랜잭션을 확인하면 새 트랜잭션 가중치를 추가하여 가중치가 업데이트된다. 따라서, 트리가 생성될 때 가중치가 증가한다. 누적 가중치가 높을수록 트랜잭션의 보안이 강화된다. 이 간단한 아키텍처 기술은 IOTA 합의를 간단하게 하고 무료로 제공한다. IOTA는 간편함과 거래 수수료의 무료를 위해 소액 거래를 가능하게 한다.

[0024] 블록체인과 관련하여, 사람들은 거래 수수료와 블록 채굴 인센티브로부터 보상을 받기 위해 네트워크에 참여하도록 동기 부여를 받는다. 그러나 IOTA는 무료이며 블록을 채굴하지 않는다. 이는 네트워크 노드에 대한 인센티브 메커니즘에 대한 의문을 제기한다. IOTA 탱글은 두 가지 방식으로 노드에 동기를 부여한다:

[0025] 1. IOTA 네트워크에 가입하면 다른 노드에 의존하지 않고 모든 사용자가 트랜잭션을 확인하고 쓸 수 있다.

[0026] 2. IOTA는 스토리지 활용 멤버십을 제공하므로 노드가 전체 데이터베이스를 복사할 필요가 없다.

[0027] 도 2(b)는 IOTA 네트워크의 예시를 보여준다. 네트워크에는 코디네이터(210)와 노드(221, 222)라는 두 가지 유형의 노드가 있다. 코디네이터(210)는 탱글의 무결성을 제어하는 주 노드 역할을 하는 컴패니(company) 자체를 사용한다. 코디네이터는 새로운 트랜잭션이 도착할 때마다 최적의 알고리즘을 실행하여 검증에 가장 적합한 트랜잭션을 찾는다. 코디네이터 노드(210)를 사용하면 특정 트랜잭션의 무한 검증 및 지연된 구성을 방지할 수 있다. 이렇게 하면 시스템 성능이 안정적이고 안정적으로 유지된다. 또한, 누구나 일반 노드로 네트워크에 가입할 수 있다. 노드가 탱글 데이터베이스를 이어 받을 때 전체 또는 부분 탱글을 복사할 수 있다. 부분 탱글을 복사하면 필요한 부분만 복사되므로 도 2(b)와 같이 새 트랜잭션을 작성할 수 있다.

[0028] 전반적으로 IOTA 탱글은 분산 컴퓨팅에서 또 다른 돌파구를 마련하여 무료 트랜잭션을 가능하게 한다. 더욱이

탱글은 네트워크가 증가하고 트랜잭션 수가 증가함에 따라 성능이 향상되는 확장 가능한 아키텍처를 가지고 있다. 즉, 기록할 트랜잭션이 더 많아지므로 오래된 트랜잭션을 더 빨리 확인할 수 있다. 이 무한한 성능 기능과 스토리지 활용에 대한 공감대는 IOTA를 가장 경쟁력 있는 분산 네트워크로 만든다.

[0029] 분산 컴퓨팅은 에지 컴퓨팅 및 데이터 수집 프레임워크와 같은 IoT 데이터 처리 시스템에서도 등장했다. 그러나 통합은 특정 기술 불일치로 인해 문제에 직면해 있다. IoT 데이터는 예측할 수 없으며 연결된 장치에 의해 대량으로 생성된다. 일반적으로 동일한 데이터에는 업데이트가 필요한 버전이 다르다. 또한 대부분의 IoT 데이터는 쓸모가 없으므로 지속적으로 삭제해야 한다. 그러나 분산 컴퓨팅에서는 동적 노드로 작업할 때 불변 데이터베이스를 유지해야 한다. 블록체인 또는 DAG 프로토콜은 데이터베이스가 공용으로 공유되면 노드가 수정하거나 삭제할 수 없다. 방대한 IoT 데이터를 불변의 분산 데이터베이스에 저장하는 것은 스토리지 효율성 측면에서 많은 비용이 든다. 게다가, 적용 가능성은 제한적이다. 따라서 분산 컴퓨팅에서 데이터 삭제는 IoT 데이터 처리와의 통합을 가능하게 하는 연구의 후속 단계이다.

[0030] 블록체인의 아키텍처를 적용하여 데이터 수정을 추가로 가능하게 하는 데 상당한 발전이 있었다. 이를 달성하기 위해 리티체인(Littichain)[2]은 체인의 끝에서 블록을 제거하여 블록별 삭제 방법을 제안했다. 블록체인은 블록을 연결된 순서로 저장하기 때문에 랜덤 블록을 삭제하는 것은 불가능하며 블록체인 규칙에 강하게 위배된다. 그러나 블록을 참조하는 하위 블록이 없는 경우 블록을 삭제할 수 있다. 즉, 블록을 삭제 순서대로 작성하면 블록체인 프로토콜을 따르는 동안 블록 삭제가 가능하다. 리티체인은 블록 구조에서 다음과 같은 두 가지 유형의 순서를 사용하여 이 목표를 달성한다:

[0031] 1. 도착 시간 순서는 블록의 물리적 순서를 유지하며 블록체인 프로토콜을 따른다.

[0032] 2. 만료 시간 순서 지정은 만료 시간을 기준으로 블록을 찾고 체인의 마지막 블록에서 삭제할 수 있도록 한다.

[0033] 또한 블록체인의 다양한 방법으로 데이터 삭제를 가능하게 하는 몇 가지 다른 기법이 도입되었다. 그러나 대부분의 제안된 시스템은 즉각적인 삭제와 관련된 성능 및 스토리지 활용률 문제로 어려움을 겪고 있다. 블록체인은 블록 생성에 대한 네트워크 확인에 크게 의존하기 때문에 삭제도 동일한 문제에 직면하여 원치 않는 지연을 유발한다.

[0035] 도 3은 본 발명의 일 실시예에 따른 엣지 컴퓨팅을 위한 삭제기반 탱글 시스템의 구성을 나타내는 도면이다.

[0036] 본 발명의 실시예에 따른 IOTA 탱글은 합의 규칙을 유지하기 위해 코디네이터 노드에 의존한다. 코디네이터는 각 새 트랜잭션을 확인하는 데 가장 적합한 트랜잭션을 결정한다. 이렇게 하면 서면 트랜잭션을 지연 없이 가능한 한 빨리 확인할 수 있다. 이를 통해 네트워크는 무결성 규칙을 따르고 트랜잭션 확인에 대한 그룹화를 방지할 수 있다. 따라서 D-탱글 구현의 첫 번째 과제는 데이터 쓰기 프로세스에서 시작된다. 데이터에 만료 시간이 있는 경우 요구 사항은 기존 각도와 비교하여 완전히 달라진다. 처음에는 코디네이터가 트랜잭션의 누적 가중치만 고려한다. 이제, 그것은 또한 수명, 그리고 더 중요한 것은 정의되지 않은 수명을 고려할 필요가 있다. 데이터에 미리 정의된 만료 시간이 있는 경우도 있으며, 요청 시 데이터를 삭제할 위치가 정의되지 않은 경우도 있다. 이러한 데이터를 변경할 수 없는 데이터로 확인하면 삭제될 가능성이 없다. 따라서 일관된 아키텍처를 보장하기 위해 쓰기 기술을 신중하게 업데이트해야 한다. 또한 업데이트된 아키텍처는 만료된 데이터가 예상치 못한 검증 없이 항상 리프 노드로 유지되도록 보장한다. 그렇지 않으면 지연으로 인해 즉시 삭제될 가능성이 줄어들지 않는다. 이러한 모든 과제를 해결하기 위해 본 발명에서는 상응 쓰기 기법과 즉각적인 삭제 기능을 갖춘 D-탱글 아키텍처를 제안한다.

[0037] 본 실시예에 따른 분산 컴퓨팅 시스템(300)은 프로세서(310), 버스(320), 네트워크 인터페이스(330), 메모리(340) 및 데이터베이스(350)를 포함할 수 있다. 메모리(340)는 운영체제(341) 및 엣지 컴퓨팅을 위한 삭제기반 탱글 알고리즘 루틴(342)을 포함할 수 있다. 프로세서(310)는 데이터 쓰기부(311) 및 데이터 삭제부(312)를 포함할 수 있다. 다른 실시예들에서 분산 컴퓨팅 시스템(300)은 도 3의 구성요소들보다 더 많은 구성요소들을 포함할 수도 있다. 그러나, 대부분의 종래기술적 구성요소들을 명확하게 도시할 필요성은 없다. 예를 들어, 분산 컴퓨팅 시스템(300)은 디스플레이나 트랜시버(transceiver)와 같은 다른 구성요소들을 포함할 수도 있다.

[0038] 메모리(340)는 컴퓨터에서 판독 가능한 기록 매체로서, RAM(random access memory), ROM(read only memory) 및 디스크 드라이브와 같은 비소멸성 대용량 기록장치(permanent mass storage device)를 포함할 수 있다. 또한, 메모리(340)에는 운영체제(341)와 엣지 컴퓨팅을 위한 삭제기반 탱글 알고리즘 루틴(342)을 위한 프로그램 코드가 저장될 수 있다. 이러한 소프트웨어 구성요소들은 드라이브 메커니즘(drive mechanism, 미도시)을 이용하여 메모리(340)와는 별도의 컴퓨터에서 판독 가능한 기록 매체로부터 로딩될 수 있다. 이러한 별도의 컴퓨터에서

판독 가능한 기록 매체는 플로피 드라이브, 디스크, 테이프, DVD/CD-ROM 드라이브, 메모리 카드 등의 컴퓨터에서 판독 가능한 기록 매체(미도시)를 포함할 수 있다. 다른 실시예에서 소프트웨어 구성요소들은 컴퓨터에서 판독 가능한 기록 매체가 아닌 네트워크 인터페이스(330)를 통해 메모리(340)에 로딩될 수도 있다.

- [0039] 버스(320)는 분산 컴퓨팅 시스템(300)의 구성요소들간의 통신 및 데이터 전송을 가능하게 할 수 있다. 버스(320)는 고속 시리얼 버스(high-speed serial bus), 병렬 버스(parallel bus), SAN(Storage Area Network) 및/또는 다른 적절한 통신 기술을 이용하여 구성될 수 있다.
- [0040] 네트워크 인터페이스(330)는 분산 컴퓨팅 시스템(300)을 컴퓨터 네트워크에 연결하기 위한 컴퓨터 하드웨어 구성요소일 수 있다. 네트워크 인터페이스(330)는 분산 컴퓨팅 시스템(300)을 무선 또는 유선 커넥션을 통해 컴퓨터 네트워크에 연결시킬 수 있다.
- [0041] 데이터베이스(350)는 엡지 컴퓨팅을 위한 삭제기반 탱글 알고리즘을 위해 필요한 모든 정보를 저장 및 유지하는 역할을 할 수 있다. 도 3에서는 분산 컴퓨팅 시스템(300)의 내부에 데이터베이스(350)를 구축하여 포함하는 것으로 도시하고 있으나, 이에 한정되는 것은 아니며 시스템 구현 방식이나 환경 등에 따라 생략될 수 있고 혹은 전체 또는 일부의 데이터베이스가 별개의 다른 시스템 상에 구축된 외부 데이터베이스로서 존재하는 것 또한 가능하다.
- [0042] 프로세서(310)는 기본적인 산술, 로직 및 분산 컴퓨팅 시스템(300)의 입출력 연산을 수행함으로써, 컴퓨터 프로그램의 명령을 처리하도록 구성될 수 있다. 명령은 메모리(340) 또는 네트워크 인터페이스(330)에 의해, 그리고 버스(320)를 통해 프로세서(310)로 제공될 수 있다. 프로세서(310)는 데이터 쓰기부(311) 및 데이터 삭제부(312)를 위한 프로그램 코드를 실행하도록 구성될 수 있다. 이러한 프로그램 코드는 메모리(340)와 같은 기록 장치에 저장될 수 있다.
- [0043] 데이터 쓰기부(311) 및 데이터 삭제부(312)는 도 4의 단계들(410~440)을 수행하기 위해 구성될 수 있다.
- [0044] 분산 컴퓨팅 시스템(300)은 데이터 쓰기부(311) 및 데이터 삭제부(312)를 포함할 수 있다.
- [0045] 본 발명의 실시예에 따른 데이터 쓰기부(311)는 엡지 컴퓨팅을 위한 삭제기반 탱글 알고리즘의 현재 상태와 새 노드를 입력 파라미터로 수신하고, 상기 현재 상태에서 검증되지 않은 노드를 획득하여 상기 노드의 생성 시간에 따라 순서대로 정렬한 후 상기 정렬된 검증되지 않은 노드의 모든 파라미터에 대하여 만료 시간을 포함하는 미리 정해진 검증 제약 조건을 확인하며, 상승(clime-up) 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 만료 시간이 이미 만료된 노드의 경우 제거되어야 하므로, 상기 만료된 노드의 만료 시간보다 만료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 한다.
- [0046] 본 발명의 실시예에 따른 데이터 쓰기부(311)는 상기 새 노드가 만료 시간이 제공되는 유한 유형인 경우, 상승 쓰기 기법을 이용한 유한 유형 노드 삽입을 위해 새 노드를 검증되지 않은 노드에 대하여 확인하고, 새 노드의 만료 시간이 검증되지 않은 모든 노드의 만료 시간보다 긴 경우 해당 노드의 부모 노드로 올라가 부모 노드의 만료 시간과 비교하고, 상기 부모 노드의 만료 시간과 비교하는 과정을 반복 수행한다.
- [0047] 본 발명의 실시예에 따른 데이터 쓰기부(311)는 상기 새 노드가 만료 시간이 없는 노드는 만료 시간 불변 유형인 경우, 상승 쓰기 기법을 이용한 불변 유형 노드 삽입을 위해 모든 리프 노드의 만료 시간이 제한되므로 상기 새 노드는 원하는 위치의 부모 노드까지 올라간다.
- [0048] 본 발명의 실시예에 따른 데이터 쓰기부(311)는 상기 새 노드가 만료 시간이 지정되지 않은 알 수 없음(Unknown) 유형인 경우, 상승 쓰기 기법을 이용한 알 수 없음 유형 노드 삽입을 위해, 마지막으로 삽입된 노드는 불변 유형이므로 해당 노드의 부모 노드 중 하나를 상기 검증되지 않은 노드에서 찾고, 또 다른 부모 노드는 더 높은 위치로 올라가 다른 불변 유형 노드를 찾는다.
- [0049] 본 발명의 실시예에 따른 데이터 삭제부(312)는 노드 생성 시 만료 시간이 제공되는 노드, 만료 시간이 없는 노드 및 만료 시간이 지정되지 않은 노드에 따라 저장 또는 노드 삭제를 수행하도록 한다.
- [0050] 본 발명의 실시예에 따른 데이터 삭제부(312)는 만료 시간이 없는 노드는 만료 시간 불변 유형으로서 영구히 저장해야 하고, 만료 시간이 제공되는 노드는 만료 시간 유한 유형이며, 상기 유한 유형 노드 및 불변 유형 노드는 산술을 통해 계산하고, 계산 결과에 따라 해당 노드의 저장 또는 노드 삭제를 수행한다.
- [0051] 본 발명의 실시예에 따른 데이터 삭제부(312)는 유한 유형 노드를 저장하는 경우 유한 유형 노드의 만료 시간에 따라 순서대로 저장되며, 만료 시간이 지정되지 않은 노드는 알 수 없음(Unknown) 유형으로서 만료되거나 또는

완료되지 않는다.

- [0052] 본 발명의 실시예에 따른 데이터 삭제부(312)는 알 수 없음 유형 노드를 삭제하는 경우, 알 수 없음 유형 노드를 삭제하기 위해 해당 노드의 부모 노드를 삭제하고, 자식 노드가 완료될 때까지 기다린다.
- [0054] 도 4는 본 발명의 일 실시예에 따른 엣지 컴퓨팅을 위한 삭제기반 탱글 시스템의 동작 방법을 설명하기 위한 흐름도이다.
- [0055] 제안하는 엣지 컴퓨팅을 위한 삭제기반 탱글 시스템의 동작 방법은 엣지 컴퓨팅을 위한 삭제기반 탱글 알고리즘의 현재 상태와 새 노드를 입력 파라미터로 수신하는 단계(410), 상기 현재 상태에서 검증되지 않은 노드를 획득하여 상기 노드의 생성 시간에 따라 순서대로 정렬한 후 상기 정렬된 검증되지 않은 노드의 모든 파라미터에 대하여 완료 시간을 포함하는 미리 정해진 검증 제약 조건을 확인하는 단계(420), 상승(clime-up) 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 완료 시간이 이미 완료된 노드의 경우 제거되어야 하므로, 상기 완료된 노드의 완료 시간보다 완료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 하는 단계(430) 및 노드 생성 시 완료 시간이 제공되는 노드, 완료 시간이 없는 노드 및 완료 시간이 지정되지 않은 노드에 따라 저장 또는 노드 삭제를 수행하도록 하는 단계(440)를 포함한다.
- [0056] 단계(410)에서, 엣지 컴퓨팅을 위한 삭제기반 탱글 알고리즘의 현재 상태와 새 노드를 입력 파라미터로 수신한다.
- [0057] 단계(420)에서, 상기 현재 상태에서 검증되지 않은 노드를 획득하여 상기 노드의 생성 시간에 따라 순서대로 정렬한 후 상기 정렬된 검증되지 않은 노드의 모든 파라미터에 대하여 완료 시간을 포함하는 미리 정해진 검증 제약 조건을 확인한다.
- [0058] 단계(430)에서, 상승 쓰기 기법을 수행하기 위해 상기 정렬된 검증되지 않은 노드 중 상기 완료 시간이 이미 완료된 노드의 경우 제거되어야 하므로, 상기 완료된 노드의 완료 시간보다 완료 시간이 긴 새 노드에 대해서만 검증을 수행하도록 한다.
- [0059] 상기 새 노드가 완료 시간이 제공되는 유한 유형인 경우, 상승 쓰기 기법을 이용한 유한 유형 노드 삽입을 위해 새 노드를 검증되지 않은 노드에 대하여 확인하고, 새 노드의 완료 시간이 검증되지 않은 모든 노드의 완료 시간보다 긴 경우 해당 노드의 부모 노드로 올라가 부모 노드의 완료 시간과 비교하고, 상기 부모 노드의 완료 시간과 비교하는 과정을 반복 수행한다.
- [0060] 상기 새 노드가 완료 시간이 없는 노드는 완료 시간 불변 유형인 경우, 상승 쓰기 기법을 이용한 불변 유형 노드 삽입을 위해 모든 리프 노드의 완료 시간이 제한되므로 상기 새 노드는 원하는 위치의 부모 노드까지 올라간다.
- [0061] 상기 새 노드가 완료 시간이 지정되지 않은 알 수 없음(Unknown) 유형인 경우, 상승 쓰기 기법을 이용한 알 수 없음 유형 노드 삽입을 위해, 마지막으로 삽입된 노드는 불변 유형이므로 해당 노드의 부모 노드 중 하나를 상기 검증되지 않은 노드에서 찾고, 또 다른 부모 노드는 더 높은 위치로 올라가 다른 불변 유형 노드를 찾는다.
- [0062] 단계(440)에서, 노드 생성 시 완료 시간이 제공되는 노드, 완료 시간이 없는 노드 및 완료 시간이 지정되지 않은 노드에 따라 저장 또는 노드 삭제를 수행하도록 한다.
- [0063] 완료 시간이 없는 노드는 완료 시간 불변 유형으로서 영구히 저장해야 하고, 완료 시간이 제공되는 노드는 완료 시간 유한 유형이며, 상기 유한 유형 노드 및 불변 유형 노드는 산술을 통해 계산하고, 계산 결과에 따라 해당 노드의 저장 또는 노드 삭제를 수행하고, 유한 유형 노드를 저장하는 경우 유한 유형 노드의 완료 시간에 따라 순서대로 저장된다.
- [0064] 완료 시간이 지정되지 않은 노드는 알 수 없음(Unknown) 유형으로서 완료되거나 또는 완료되지 않고, 알 수 없음 유형 노드를 삭제하는 경우, 알 수 없음 유형 노드를 삭제하기 위해 해당 노드의 부모 노드를 삭제하고, 자식 노드가 완료될 때까지 기다린다.
- [0065] 본 발명의 일 실시예에 따른 D-탱글 아키텍처의 각 노드는 분산 네트워크에 저장된 독립적인 데이터 또는 트랜잭션을 나타낸다. 따라서 여기서 "노드", "데이터" 및 "트랜잭션"이라는 용어는 서로 바꾸어 사용될 수 있다. 본 발명의 일 실시예에 따른 D-탱글은 DAG에서 데이터 삭제를 활성화하는 것을 목표로 하므로 노드는 완료 시간에 따라 다음 세 가지 유형으로 분류할 수 있다:
- [0066] 유한 - 삽입 시 완료 시간이 제공되는 노드이다.

- [0067] 불변 - 만료 시간이 없는 노드로, 영구히 저장해야 한다.
- [0068] 알 수 없음 - 만료 시간이 지정되지 않은 노드로 만료되거나 만료되지 않을 수 있다.
- [0069] 유한 유형과 불변 유형은 전통적인 산술을 사용하여 계산을 수행할 수 있기 때문에 쉽게 처리할 수 있다. 그러나 이 알 수 없는 유형은 구현하기가 가장 어렵다. IoT 데이터는 예측할 수 없고 동적이기 때문에 만료 시간을 미리 정의하는 것이 항상 가능한 것은 아니다. 데이터가 영구 저장되거나 나중에 삭제되는 알 수 없는 결정을 수반할 가능성이 있다. 만료 시간 및 기타 제약 조건으로 인해 알 수 없는 유형의 데이터를 예측할 수 없으므로 분산 시스템 합의를 유지하면서도 즉각적인 삭제를 보장할 수 없다. 그러나 신뢰할 수 있는 시스템은 두 가지 경우를 모두 고려하고 불변성과 삭제 가능성을 모두 가능하게 해야 한다. 삭제는 노드를 작성하는 동안 노드를 찾는 방법과 상당히 관련이 있기 때문에 최적의 쓰기 기술도 중요하다. 이 목표를 달성하기 위해 D-탱글은 합의 규칙을 유지하면서 삭제 요구 사항을 완전히 충족하는 상승(clime-up) 쓰기 기법을 사용한다. 이러한 모든 기능을 사용하여 도 5는 초기 상태와 상승 쓰기 기법을 사용하는 가능한 세 가지 예시를 도시한다.
- [0071] 도 5는 본 발명의 일 실시예에 따른 D-탱글의 상승 쓰기 기법을 설명하기 위한 도면이다.
- [0072] 도 5(a)는 D-탱글 그래프의 예시와 함께 초기 상태를 도시한다. 그래프는 IOTA 아키텍처에서와 같이 체네시스 노드로 시작한다. 쓰기 기술은 생성 시간 순서대로 불변 유형 노드가 생성 노드 바로 뒤에 자동으로 기록되도록 최적화된다. 불변 유형 노드 후에는 알 수 없는 만료 시간 노드가 저장된다. 이러한 노드를 확인할 때 D-탱글은 알 수 없는 다른 유형의 확인을 엄격하게 방지한다. 알 수 없는 노드를 삭제할 수도 있고 삭제하지 않을 수도 있다. 부모(parent) 노드를 삭제해야 하고 자식(children) 노드가 정지되면 삭제 작업을 수행할 수 없는 영구적인 지연이 발생한다. 알 수 없는 유형에 따라 유한 유형 노드는 만료 시간에 따라 순서대로 저장된다. 이러한 방식으로 D-탱글은 유한 유형 노드에서 즉시 삭제된다. 알 수 없는 노드에서 삭제가 필요한 경우 삭제를 수행하려면 자식 노드가 만료될 때까지 기다려야 한다. 알 수 없는 노드에서 지연이 불가피한 경우에도 D-탱글은 작업이 최종적으로 수행되도록 보장한다. 그래프에서 일부 노드는 자식이 하나만 있을 수 있지만 여전히 확인된 노드로 표시되며, 이는 탱글 규칙과 유사하게 나타날 수 있다. 그러나 노드의 자식 노드에는 다른 노드의 확인이 있으므로 모든 방법이 누적 가중치의 영향을 받으며 노드가 자동으로 확인된다. 완벽한 경우 새 노드는 리프 노드보다 만료 시간이 짧으며 도 5(a)에 표시된 것처럼 이를 확인하기만 한다.
- [0073] 그러나 IoT 데이터 환경은 몇 가지 복잡성을 보인다. 도 5(b)는 상승 쓰기 기법을 이용한 유한 유형 노드 삽입 사례를 보여준다. 전통적인 방법에 따르면, 새로운 노드는 검증되지 않은 노드 중 하나를 확인한다. 그러나 새 노드의 만료 시간은 50으로 확인되지 않은 모든 노드의 만료 시간보다 크다. 따라서 부모 노드로 올라가 유효 기간이 55 및 50인 부모 노드를 확인한다.
- [0074] 도 5(c)는 동일한 기술을 사용한 불변 유형 노드 삽입의 예시를 보여준다. 모든 리프 노드의 만료 시간이 제한되므로 새 노드가 부모 노드로 올라간다. 또한 불변 유형 노드는 불변 유형 노드만 확인할 수 있기 때문에 상승 쓰기 기법은 원하는 위치까지 올라간다.
- [0075] 마지막으로, 도 5(d)는 알 수 없음 유형의 노드의 삽입을 보여준다. 이 경우 마지막으로 삽입한 노드는 불변 유형이므로 부모 노드 중 하나를 검증되지 않은 노드 목록에서 찾는다. 하지만, 또 다른 두 번째 부모 노드는 다른 불변 유형 노드를 찾아 더 높은 위치로 올라간다.
- [0077] 도 6은 본 발명의 일 실시예에 따른 상승 쓰기 기법의 알고리즘을 나타내는 도면이다.
- [0078] 수명 차이 문제를 해결하기 위해 본 발명의 일 실시예에 따른 D-탱글은 코디네이터 노드에 대해 상승 쓰기 기법을 사용한다. 무결성을 달성하고 원래의 탱글 성능을 유지하기 위해 상승 쓰기 기법은 세 가지 규칙을 따른다:
- [0079] 1. 검증되지 않은 노드의 생성 순서에 따라 우선 순위가 가장 높다.
- [0080] 2. 만료된 노드는 탱글에서 제거되어야 한다.
- [0081] 3. 새 노드는 이전 노드의 만료 시간이 긴 경우에만 확인한다.
- [0082] 도 6의 알고리즘은 지정된 규칙에 따라 상승 쓰기 기법에 대한 절차를 제시한다. 이 절차는 탱글의 현재 상태와 새 노드(다시 말해, 데이터)를 입력 매개변수로 가져온다. 알고리즘을 실행한 후 탱글의 새 상태(다시 말해, 새 데이터)가 출력으로 반환된다. 첫 번째 규칙에 따라 7행에 표시된 대로 검증되지 않은 노드를 얻는 것으로 절차가 시작된다. 또한 규칙에 따르면 우선순위는 생성 시간에 따라 설정된다. 즉, 다른 요구 사항이 충족되는 경우 먼저 첫 번째로 생성된 노드를 확인해야 한다. 이를 위해 목록은 8행에 표시된 것처럼 작성 시간(시간 A)별로

정렬된다. 모든 변수를 로드한 후 확인되지 않은 목록은 11~20행에 표시된 것처럼 다른 요구 사항을 확인하기 위해 반복된다. 먼저, 만료 시간을 검사하여 노드 수명이 이미 만료되었는지 여부를 확인한다(13행). 만료된 노드는 두 번째 규칙에 따라 탱글(14행)에서 제거된다. 제거 함수가 호출되면 해당 함수의 부모가 누적 가중치 및 검증 제약 조건을 검사한다. 자식이 없는 노드는 확인되지 않은 목록에 다시 통합된다. 세 번째 규칙의 구현은 15~20행에 나와 있다. 이 조건은 새 데이터 만료 시간이 후보보다 작은지 여부를 확인하기 위해 검사된다. 이 조건이 충족되면 검증이 확인된다. 이 시나리오에서는 확인되지 않은 모든 노드를 반복하여 확인 및 만료 여부를 확인한다. 나머지 절차는 확인되지 않은 노드가 특정 요구 사항을 충족하지 못하는 경우에만 수행된다. 이 문제는 새 노드의 만료 시간이 확인되지 않은 모든 노드(22행)보다 길 때마다 발생할 수 있다. 이러한 방식으로 이 절차는 상위 노드로 올라가 만료 시간에 따라 적합한 후보를 찾는다. 이 단계에서는 삭제가 리프 노드 간에만 발생하므로 노드가 만료 여부를 확인할 필요가 없다. 즉, 상위 노드의 만료 시간이 더 길어지므로 확인할 필요가 없다. 또한 부모 노드는 리프 노드에 의해 이미 검증되었다. 따라서 누적 가중치를 검사할 필요가 없다. 이 절차는 상위 부모를 검색하고 검증에 적합한 노드(24행 및 25행)를 찾는다. 노드가 발견되면 포인터가 후보 노드로 설정되고 계속된다. 결론적으로, 새로운 데이터는 원하는 포인터(31행)를 얻은 후 검증되지 않은 목록에 추가된다.

[0083] 새로 채택된 쓰기 기술은 원하는 부모를 할당할 후 검증되지 않은 데이터 목록을 업데이트하는 것도 고려한다. 또한 각 삽입 후 전체 탱글의 누적 가중치가 다시 계산된다. 전통적인 탱글 규칙에 따르면, 누적 가중치가 높을수록 노드 검증의 신뢰성이 정의된다.

[0084] 본 발명의 실시예에 따른 D-탱글 아키텍처의 이론적 분석의 쓰기 효율성에 있어서, 상승 쓰기 기법의 구현은 삭제 고려 사항의 측면에서 전통적인 쓰기 메커니즘을 지연시키는 것처럼 보일 수 있다. 그러나 이는 성능에 크게 영향을 미치지 않는다. 분산 시스템에서 지연은 주로 네트워크 통신 및 전파 중 응답 대기로 인해 발생한다. D-탱글의 경우 상승 쓰기 기법은 코디네이터 노드에서만 작동하며 기존 로직과 비교하여 몇 가지 추가 로직을 실행한다. 삭제 작업은 새 데이터를 작성하는 것과 유사하게 합의에 의해 확인된다. 따라서 전통적인 검증 성능을 유지하기 위해 쓰기 메커니즘에 상당한 지연이 없다.

[0085] 본 발명의 실시예에 따른 D-탱글 아키텍처의 이론적 분석의 삭제 효율성에 있어서, 삭제는 유한 데이터 삭제와 알 수 없는 데이터 삭제의 두 가지 방법으로 분류할 수 있다. 상승 쓰기 기법을 고려하면 유한 데이터가 자동으로 정렬되어 처음 삭제한 노드는 항상 리프로 저장된다. 따라서 유한 데이터의 삭제는 즉시 수행될 수 있다. 그러나 알 수 없는 데이터는 만료 시간에 관계없이 유한 데이터로 검증할 수 있다. 삭제 요청이 발생하면 자식 노드가 만료될 때까지 알 수 없는 데이터가 대기해야 한다. 알 수 없는 수명 데이터에 대해 가능한 지연을 추정하는 것은 불가능하다. 그러나 이러한 가능성은 워크로드 구성에 따라 추정할 수 있다. 알 수 없는 데이터가 없거나 완전한 알 수 없는 데이터의 워크로드를 고려할 때 알 수 없는 데이터가 서로 유효성을 검사하지 않기 때문에 지연 시간이 감소할 것으로 예상된다. 따라서 알 수 없는 각 데이터에 대해 검증이 수행되는 혼합 워크로드에서 최악의 성능이 발생할 수 있다. 이 경우 지연 시간은 유한 데이터의 만료 시간에 따라 완전히 달라진다.

[0086] 본 발명의 실시예에 따른 D-탱글 아키텍처의 이론적 분석의 스토리지 효율성에 있어서, 전통적인 탱글은 부분 복사를 구현하는 효율적인 아키텍처이다. 동일한 시나리오를 유지하려면 다음 아키텍처도 스토리지 효율적으로 구현해야 한다: 삭제를 사용하도록 설정하면 스토리지 효율성의 의미가 변경된다. 따라서 효율성은 데이터 삭제를 요청한 후 사용 중인 스토리지를 얼마나 빨리 확보할 수 있는지를 나타낸다. 수명 데이터를 알 수 없는 예측할 수 없는 경우를 고려하면 삭제 중에 예상보다 약간 긴 시간 동안 스토리지가 할당될 수 있다. 그러나 D-탱글은 결국 삭제를 보장한다. 따라서 스토리지도 자유롭다. 기존의 접근 방식이 스토리지 할당을 영구적으로 강제한다는 점을 고려할 때, 지연 후 여유 공간이 실제 애플리케이션에서 훨씬 더 효율적인 것으로 보인다.

[0088] 도 7은 본 발명의 일 실시예에 따른 평가를 위한 워크로드 설정을 나타내는 도면이다.

[0089] 본 발명의 일 실시예에 따른 평가를 위해 도 7과 같이 세 가지 워크로드를 설정한다. 워크로드 A(도 7(a)) 및 워크로드 B(도 7(b))는 선형 순서 삭제 시간으로 항목을 작성하는 데 중점을 둔다. 워크로드 C(도 7(c))는 무작위 삭제로 데이터를 사용한다. 워크로드의 기본 구성에는 수명이 미리 정의된 유한 데이터만 포함된다. 결과적으로 알려지지 않은 데이터의 비율은 워크로드와 관련하여 선형적으로 증가했다. 따라서 워크로드의 지정된 비율을 알 수 없는 데이터로 삽입하고 워크로드가 정의한 시간에 삭제 요청을 수행했다. 이러한 유형의 변수 구성은 수명이 다른 다양한 시나리오에서 D-탱글 동작을 관찰하는 데 도움이 된다.

[0091] 도 8은 본 발명의 일 실시예에 따른 평가 결과를 나타내는 도면이다.

- [0092] 도 8과 같이 평균 삭제 대기 시간에 대한 결과를 모든 워크로드에 대해 도시하였다. 알 수 없는 데이터 비율을 0%~95%에서 증가시켜 각 워크로드에 대해 광범위한 평가를 수행한다. 알 수 없는 데이터의 비율은 결과의 공정성을 보장하기 위해 워크로드 전체에 균일하게 분산되었다. 동일한 시나리오를 다른 데이터 크기로 4번 반복하여 동작의 차이점을 추가로 확인했다. 워크로드는 작은 크기의 데이터를 사용하여 준비되었으며 각 샘플링에는 평균 500개의 항목이 있다. 항목 수는 256GB, 512GB, 1TB 및 2TB 데이터의 총 워크로드를 시뮬레이션하도록 적절하게 구성된다. 더 나은 결론을 위해 모든 워크로드와 B + A 통합에 대해 동일한 구성으로 실험을 수행했다.
- [0093] 도 8(a)는 워크로드 A에 대한 결과를 나타낸다. 결과에서 확인할 수 있듯이 D-Tangle은 이 워크로드에서 거의 즉각적인 삭제 성능을 달성한다. 워크로드 A는 데이터를 내림차순으로 삭제하는 데 중점을 둔다. 따라서, 후속 항목은 항상 리프 노드를 확인한다. 이는 알 수 없는 데이터 비율이 증가하더라도 변함없이 마지막에 삽입된 데이터가 먼저 삭제되어 지연을 피할 수 있음을 증명한다. 대기 시간이 선형적으로 증가하는 것처럼 보이지만 모든 경우에 평균 5.2초의 지연을 형성하지만 일반적인 결과는 거의 즉각적인 삭제 성능으로 가정할 수 있다. 따라서 워크로드 A는 모든 조합에 대해 유한 데이터와 알 수 없는 데이터 모두에 대해 D-탱글 아키텍처에 가장 적합한 환경으로 간주될 수 있다.
- [0094] 도 8(b)는 워크로드 B에 대한 평가 결과를 나타낸다. 결과는 알 수 없는 데이터 비율의 40%와 50% 사이에서 최악의 성능을 가진 삼각형 보기를 형성했다. 워크로드 B는 워크로드 A와 반대이므로 분명히 평가에 더 긴 대기 시간이 존재한다. 그러나 알 수 없는 데이터의 비율이 95%까지 증가하면 대기 시간이 크게 줄어든다. 이론적으로 대기 시간은 선형적으로 증가해야 하는 것처럼 보일 수 있다. 그러나 결과가 전체 워크로드의 평균을 기반으로 한다는 점도 고려해야 한다. 데이터의 작은 부분은 대기 시간이 매우 길다. 그러나 전체 평균은 알려지지 않은 비율이 높기 때문에 감소한다. 알 수 없는 데이터가 많아질수록 서로 검증하지 않고 불변 데이터만 검증한다. 또한 비율이 감소하기 때문에 유한한 데이터가 모든 미지 데이터를 검증할 가능성이 적다. 따라서 대부분의 경우 삭제가 더 빠르게 수행될 수 있다.
- [0095] 워크로드 A와 B가 서로 모순되기 때문에 둘의 조합으로 더 많은 평가를 실행하면 더 큰 이미지를 얻는 데 도움이 된다. A+B 조합은 워크로드 B 자체를 실행하는 것과 동일하다. 그러나 B+A의 조합은 D-탱글 성능에 대한 더 큰 개요를 연다. 도 8(c)는 이 평가 결과를 나타낸다. 워크로드 B와 비교할 때, 워크로드 조합은 D-탱글이 알 수 없는 데이터를 확인하도록 강제하여 더 포물선 이미지를 형성한다. 작업 성능이 120초에서 140초로 증가한다. 그러나 일반 이미지는 최소 및 최대 알 수 없는 데이터 비율에서 낮은 대기 시간으로 유지된다. 높이가 더 작은 유사한 이미지가 워크로드 C에 대해 기록된다. 전반적으로 이미지는 다른 결과와 유사하다. 그러나 최악의 지연은 100초로 기록된다. 이러한 결과로부터 B+A의 조합은 특정 구성을 가진 무작위 워크로드의 최악의 경우를 생성한다고 가정한다.
- [0096] 모든 평가에서 모든 워크로드 및 데이터 크기에 대한 초기 대기 시간이 0이고 알 수 없는 데이터 크기가 0%임을 쉽게 관찰할 수 있다. 즉, 불변하고 유한한 영구적인 데이터의 작업 부하를 고려하면 D-탱글은 즉각적인 삭제로 완벽한 성능을 달성한다. 전체 이미지는 최대 120초의 대기 시간을 나타내지만 알 수 없는 데이터의 경우 결국 모든 데이터가 삭제된다.
- [0098] 도 9는 본 발명의 일 실시예에 따른 스토리지 사용량을 나타내는 도면이다.
- [0099] D-탱글 프로토콜의 효율성을 입증하기 위해 스토리지 사용량에 대한 추가 결과가 기록되었다. D-탱글 아키텍처의 스토리지 비용과 효율성을 입증하기 위해 데이터베이스 크기/작업 부하 크기×100% 등식을 사용했다. 이상적으로 데이터베이스 크기는 항상 워크로드 크기와 같아야 한다. 이 경우 스토리지 사용량은 항상 100%이다. 그러나 삭제 지연으로 인해 워크로드 크기에 비해 데이터베이스 크기가 증가할 수 있다. 워크로드에서는 데이터가 삭제되지만 종속성 지연으로 인해 데이터베이스에서는 삭제할 수 없는 경우이다.
- [0100] 도 9은 앞서 언급한 기준에 따른 스토리지 사용의 전체 결과를 나타낸다. 4개의 워크로드는 서로 다른 두 가지 워크로드 구성인 C로 준비되었다. 워크로드 C는 임의의 워크로드이므로 알 수 없는 데이터에 대한 삭제 요청은 간격 사이에 언제든지 호출될 수 있다. 따라서 더 많은 동작을 관찰하기 위해 삭제된 요청의 초기 호출과 지연 호출에 대한 구성이 만들어졌다. 결과는 알 수 없는 데이터 비율이 낮거나 높은 경우 거의 일정한 스토리지 사용량(100%에서 110% 사이)을 나타낸다. 워크로드 B 및 C(초기 삭제 요청 포함)에 대해 50%의 알 수 없는 데이터 비율로 최악의 성능이 관찰되었다. 그러나 유한한 데이터에 비해 삭제 요청이 늦게 수행되도록 워크로드 C를 구성하면 50% 비율에서도 성능이 안정화된다.
- [0102] 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 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0103] 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, 가상장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 구체화(embody)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.

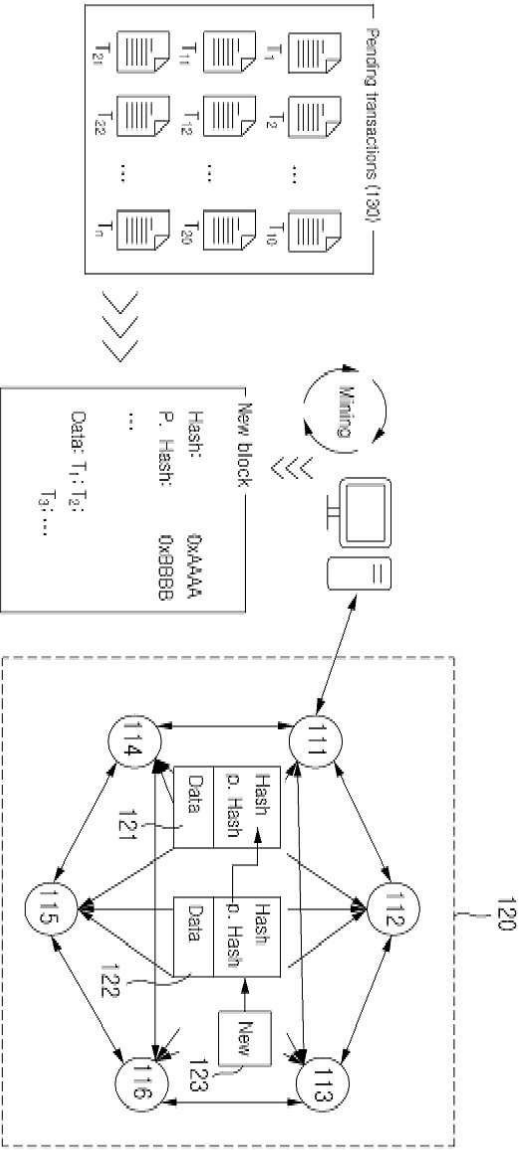
[0104] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic 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.

[0105] 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.

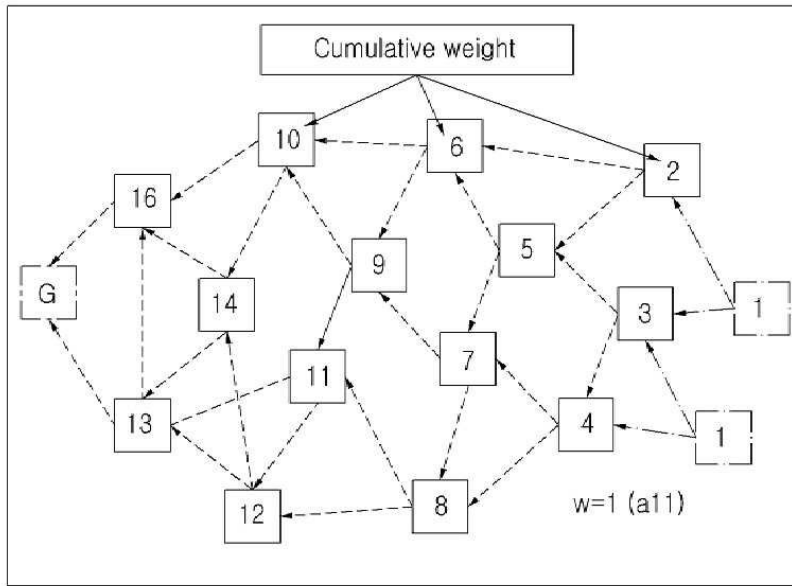
[0106] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

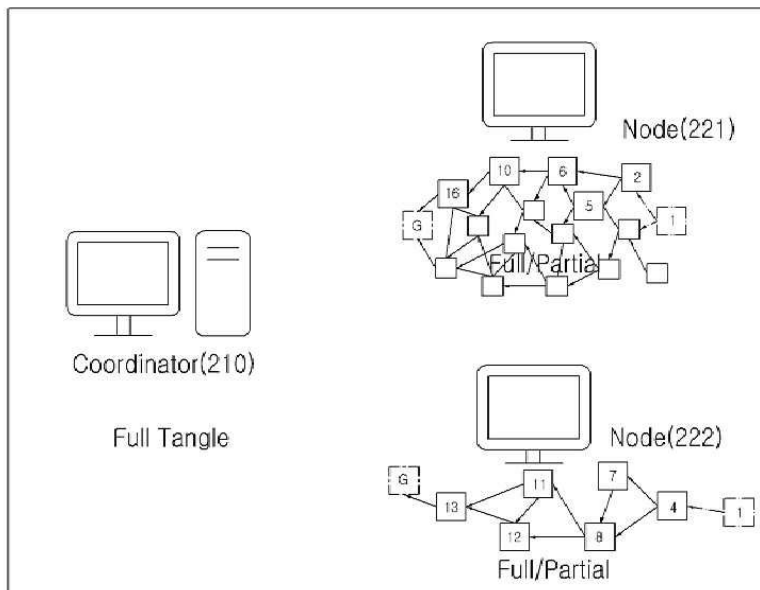
도면1



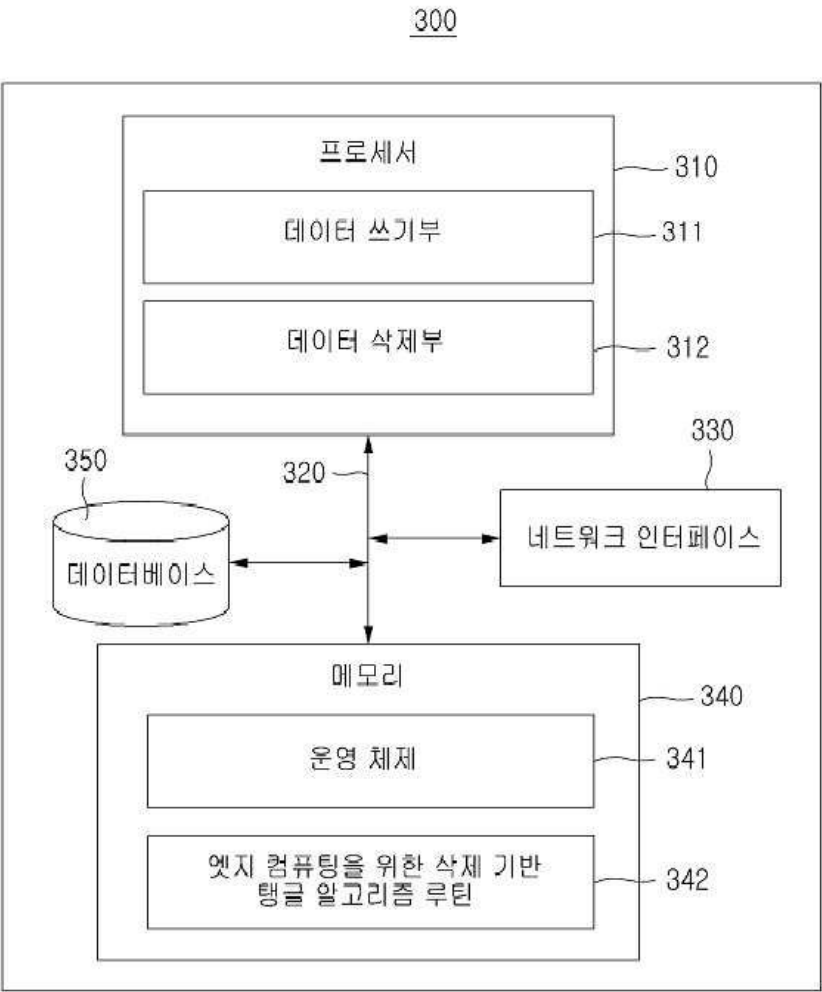
도면2a



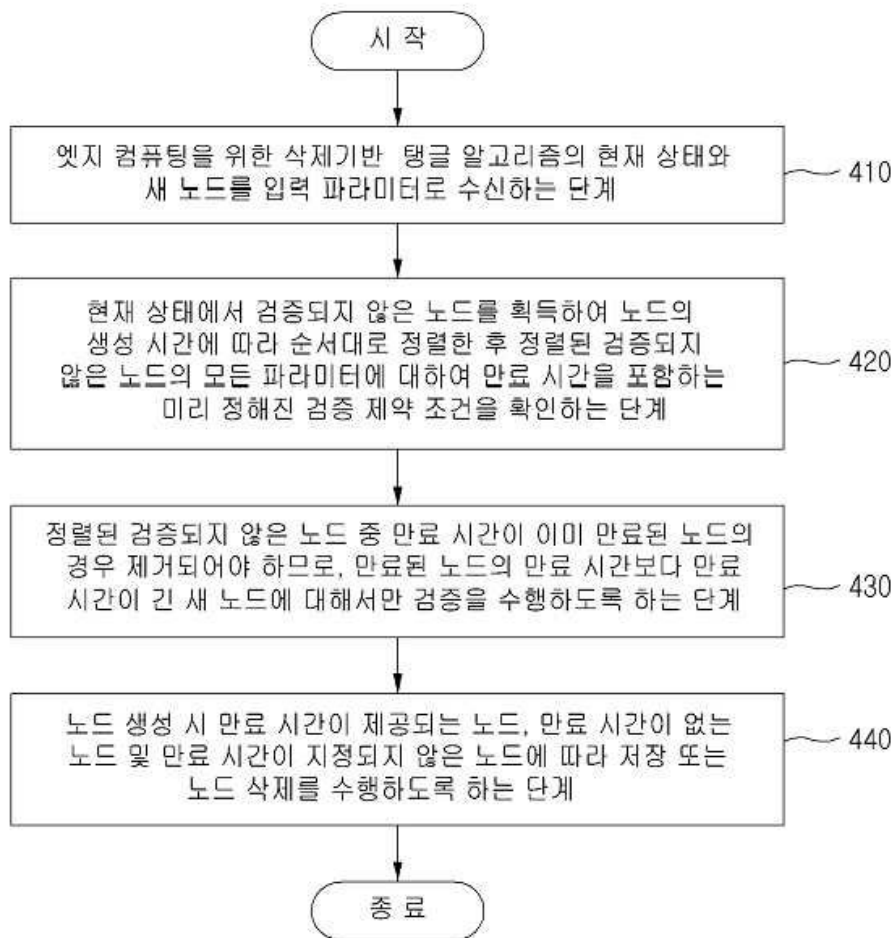
도면2b



도면3



도면4



도면6

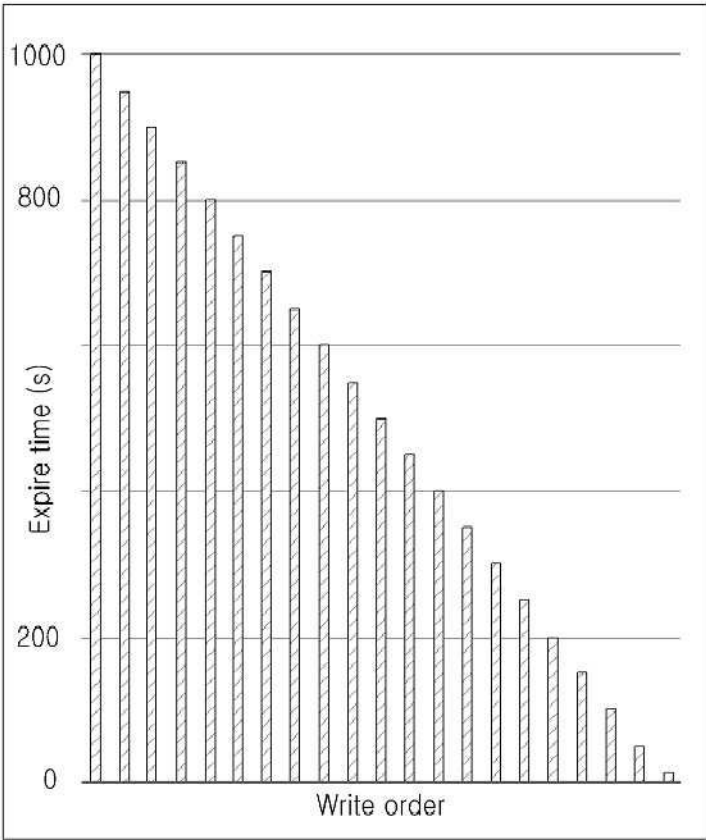
Algorithm 1 Climb-up write technique

```

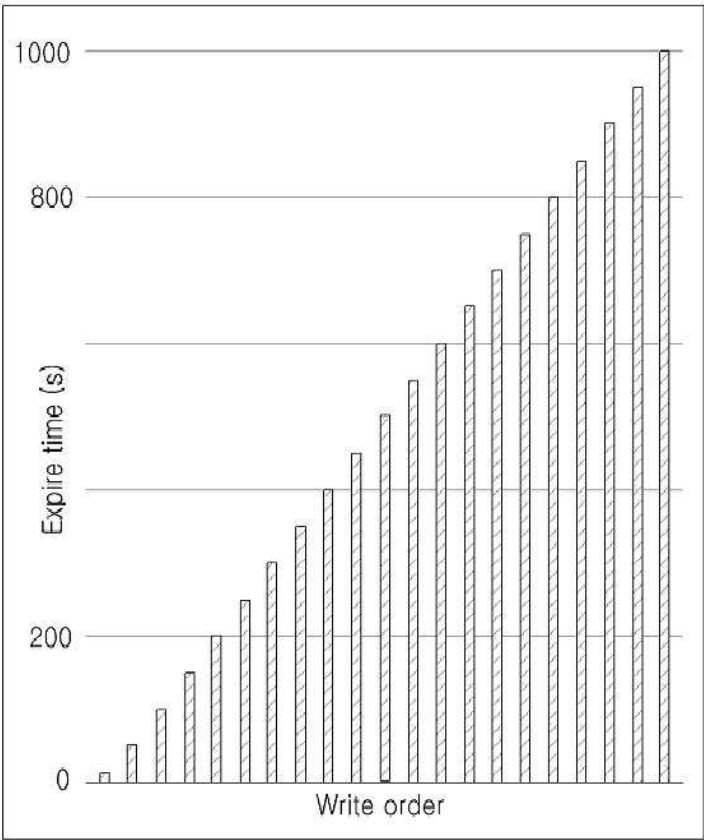
1: Input:    tangle → current state of Tangle
2:          data → new Data to write
3: Output:   out → new Tangle including new Data
4:
5: procedure
6:
7:   list ← tangle.getUnverifiedList()
8:   list.sort(item => item.timeA, ASC)
9:   tD ← data.timeE
10:
11:  for inti = 0; i < list.size(); i++ do
12:    tN ← list.get(i).timeE
13:    if tN ≤ CURRENT then
14:      tangle.remove(list.get(i))
15:    if tD < t then
16:      if data.parent == NULL then
17:        data.parent1 = list.get(i)
18:      else
19:        data.parent2 = list.get(i)
20:      continue
21:
22:  while data.parent1 == NULL & data.parent2 == NULL do
23:    node ← list.next()
24:    while tC > node.parent.timeE do
25:      node ← node.parent
26:    if data.parent1 == NULL then
27:      data.parent1 = node
28:    else
29:      data.parent2 = node
30:
31:  tangle.getUnverifiedList().append(data)
32:  return tangle as out

```

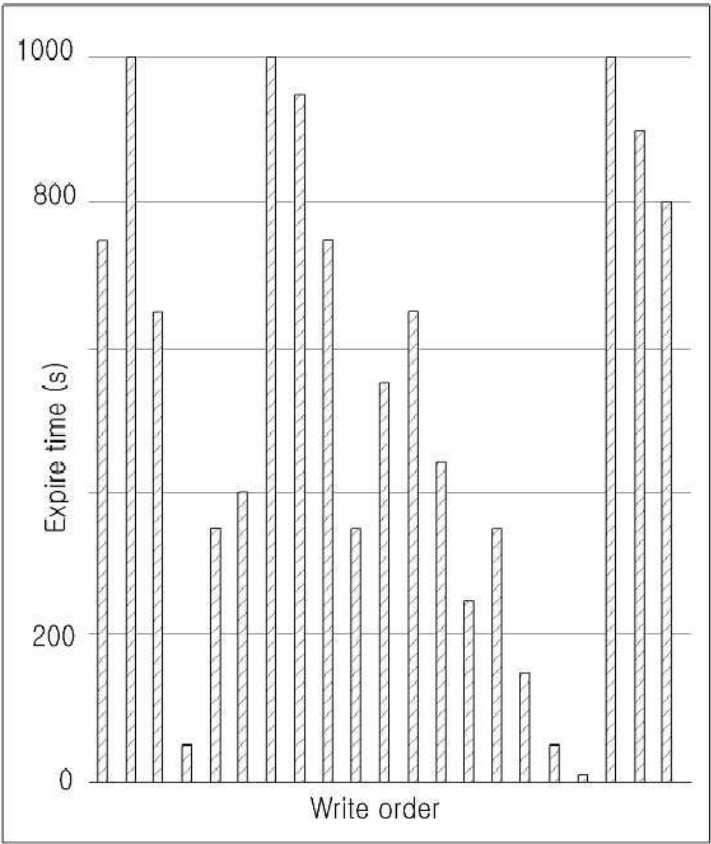
도면7a



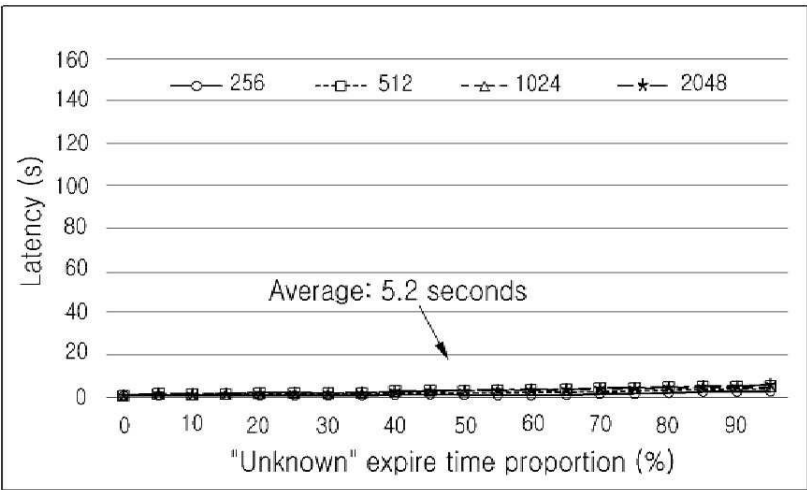
도면7b



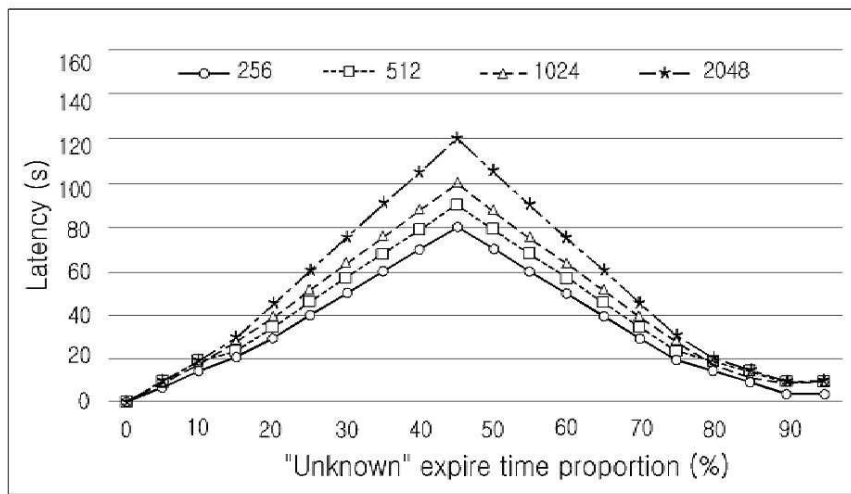
도면7c



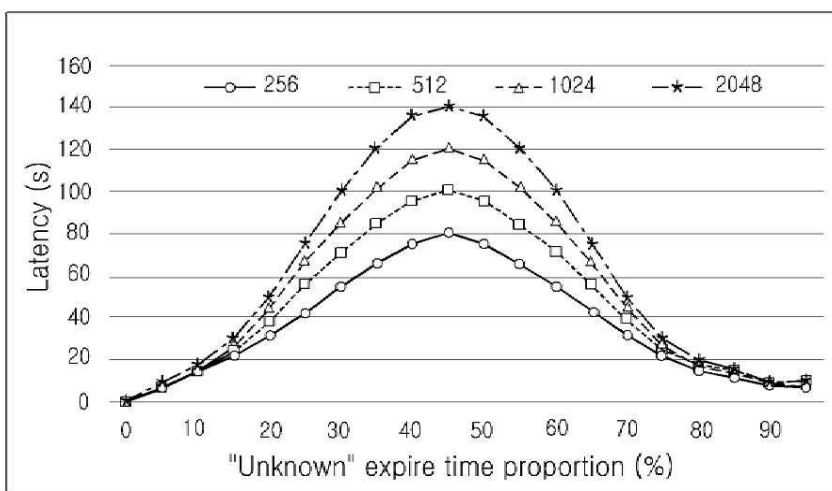
도면8a



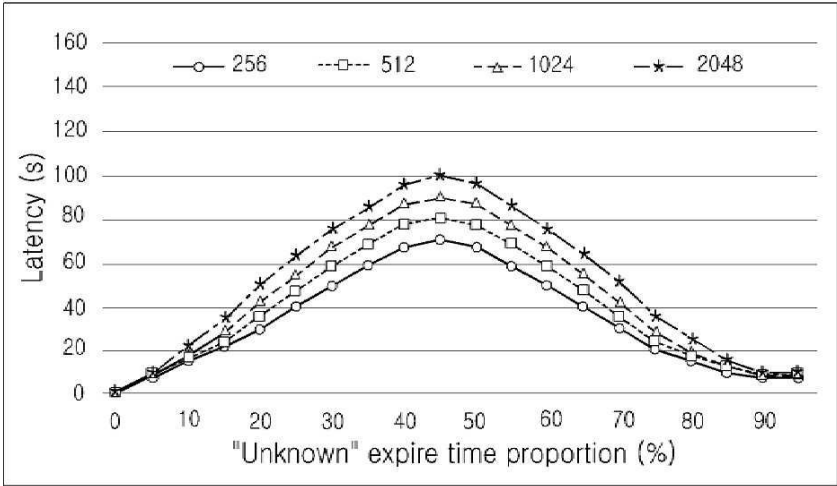
도면8b



도면8c



도면8d



도면9

