(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2023-0116692
(43) 공개일자 2023년08월04일

(51) 국제특허분류(Int. Cl.)
G06F 11/30 (2006.01) *G06F 16/182* (2019.01)
G06F 9/46 (2006.01) *H04L 67/1097* (2022.01)
(52) CPC특허분류
G06F 11/3006 (2013.01)
G06F 16/1834 (2019.01)
(21) 출원번호 10-2023-0006766
(22) 출원일자 2023년01월17일
심사청구일자 2023년01월17일
(30) 우선권주장
1020220012165 2022년01월27일 대한민국(KR)

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
정익래
서울특별시 광진구 아차산로 69길 19, 904동 2205호
이연주
충청남도 천안시 서북구 백석로 136, 113동 1102호 (백석동, 백석현대아파트)
최재현
서울특별시 성북구 안암로9가길 55, 305호
(74) 대리인
김등용

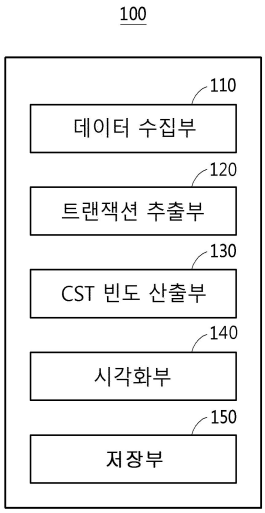
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 **이더리움 샤딩에서 CST 측정 장치 및 방법**

(57) 요 약

샤딩 환경이 적용된 블록체인에서의 CST(Cross Shard Transaction) 측정 장치 및 방법이 개시된다. 상기 CST 측정 장치는 샤딩 환경을 적용한 블록체인에 포함된 복수의 블록들 중 적어도 일부를 수집하는 데이터 수집부, 수집된 블록들로부터 트랜잭션 정보를 추출하는 트랜잭션 추출부, 및 추출된 트랜잭션 정보를 이용하여, CST(Cross Shard Transaction) 빈도를 산출하는 CST 빈도 산출부를 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 9/466 (2013.01)
H04L 67/1097 (2022.05)

명 세 서

청구범위

청구항 1

샤딩 환경을 적용한 블록체인에 포함된 복수의 블록들 중 적어도 일부를 수집하는 데이터 수집부;

수집된 블록들로부터 트랜잭션 정보를 추출하는 트랜잭션 추출부; 및

추출된 트랜잭션 정보를 이용하여, CST(Cross Shard Transaction) 빈도를 산출하는 CST 빈도 산출부를 포함하는 CST 측정 장치.

청구항 2

제1항에 있어서,

CST 빈도 산출부는 송신자의 어카운트 주소가 속하는 샤드와 수신자의 어카운트 주소가 속하는 샤드가 상이한 경우의 트랜잭션을 CST로 결정하는,

CST 측정 장치.

청구항 3

제2항에 있어서,

CST 빈도 산출부에 의해 산출된 결과를 디스플레이 장치를 통해 출력하는 시각화부를 더 포함하는,

CST 측정 장치.

청구항 4

제3항에 있어서,

상기 시각화부는 샤드별 CST의 빈도를 그래프로 표시하는,

CST 측정 장치.

청구항 5

제4항에 있어서,

트랜잭션 추출부는 트랜잭션을 식별할 수 있는 식별 정보, 송신자의 어카운트 주소, 및 수신자의 어카운트 주소를 상기 트랜잭션 정보로 추출하는,

CST 측정 장치.

청구항 6

적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는 CST 측정 방법에 있어서,

샤딩 환경을 적용한 블록체인에 포함된 복수의 블록들 중 적어도 일부를 수집하는 단계;

수집된 블록들로부터 트랜잭션 정보를 추출하는 단계; 및

추출된 트랜잭션 정보를 이용하여, CST 빈도를 산출하는 단계를 포함하는 CST 측정 방법.

청구항 7

제6항에 있어서,

상기 CST 빈도를 산출하는 단계는, 송신자의 어카운트 주소가 속하는 샤드와 수신자의 어카운트 주소가 속하는 샤드가 상이한 경우의 트랜잭션을 CST로 결정하고,

상기 트랜잭션 정보를 추출하는 단계는, 트랜잭션을 식별할 수 있는 식별 정보, 송신자의 어카운트 주소, 및 수신자의 어카운트 주소를 상기 트랜잭션 정보로 추출하는,

CST 측정 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 이더리움 샤딩에서 CST 측정 방법에 관한 것으로, 특히 서로 다른 샤드(shard) 상에서 존재하는 어카운트(account)들 간의 거래 트랜잭션인 CST(Cross Shard Transaction) 생성 빈도를 측정하는 장치 및 방법에 관한 것이다.

배 경 기 술

[0002] 블록체인은 P2P 방식의 분산 컴퓨팅 기반 원장 관리 기술로 탈중앙화된 구조를 가진다. 블록체인에 참여한 각 노드들 간의 거래는 트랜잭션 형태로 블록체인 상에서 블록에 저장되며, 암호학적 해시함수로 해시하여 저장됨으로써 데이터 무결성 및 불변성을 보장한다는 점에서 많은 분야에서 사용중에 있다. 그러나 이러한 블록체인 기술은 서로 해결할 수 없는 3가지 문제인 트릴레마가 존재한다. 블록체인의 트릴레마로는 전송 처리 용량의 한계로 인한 확장성(Scalability) 문제, 탈중앙화(Decentralization) 문제, 보안성(Security) 문제가 존재한다.

[0003] 샤딩(sharding)은 데이터베이스에서 사용되던 개념으로 데이터를 다중 스토리지에 분산시켜 저장하는 기술이다. 블록체인의 확장성 문제를 개선할 수 있는 주요 방법으로는 블록체인에 샤딩 기술을 적용하는 방법이 존재한다. 샤딩의 주요 아이디어는 블록체인에 참여한 노드들끼리 샤드라고 불리는 그룹으로 나누어 블록체인 상의 트랜잭션 일부를 병렬적으로 처리하는 것이다. 각 샤딩은 collation chain을 이루고 있으며, 각 샤드에서 생성된 블록의 헤더값 만을 메인 체인에 올림으로써 블록체인의 처리량을 높일 수 있다. 이러한 샤딩의 트랜잭션 종류로는 IST(Intra Shard Transaction)와 CST(Cross Shard Transaction)가 존재한다. IST는 동일 샤드에 존재하는 어카운트 간의 트랜잭션으로 샤드 내에서 합의를 통하여 트랜잭션을 처리함으로써 비교적 값싼 트랜잭션 비용이 발생한다. CST는 다른 샤드에 존재하는 어카운트 간의 트랜잭션으로 동기식/비동기식 방법에 따라 다른 샤드 간의 트랜잭션 처리 절차가 복잡하고 많은 트랜잭션 비용이 든다.

[0004] 샤딩 환경에서 CST의 빈도가 높아질수록 낮은 처리량과 길어진 대기시간, 그리고 서로 다른 샤드간에 복잡한 처리 과정으로 인한 증가된 트랜잭션 비용 등 다양한 문제를 발생시킨다. 또한 Sonnino 등의 연구에 따르면, CST에서 발생하는 크로스 샤드 합의(Cross Shard Consensus)로 인하여 재전송 공격(replay attack)이 발생할 수도 있다.

[0005] 이러한 블록체인 상에서의 CST가 발생시키는 문제들을 사전에 방지하기 위해서는 시스템상에서 발생하는 CST를 측정하는 기술이 필요하다. 그러나 현재까지 제시된 기술들 중에서 CST만을 측정하는 기술은 존재하지 않는다. 이에 본 발명은 블록체인의 종류 중 이더리움 샤딩에서 CST(Cross Shard Transaction) 측정 기술을 제안한다.

선행기술문헌

비특허문헌

[0006] (비특허문헌 0001) Sonnino, Alberto, et al. "Replay attacks and defenses against cross-shard consensus in sharded distributed ledgers." 2020 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2020.

발명의 내용

해결하려는 과제

[0007] 샤딩 환경을 적용한 블록체인 연구와 이러한 환경을 적용한 시스템이 증가함에 따라 블록체인 상에 발생하는 CST를 측정하는 기술이 필요하다. 그러나 현재까지 블록체인 상의 CST를 측정하는 기술은 나와 있지 않아 그에 대한 필요성이 증대되었다. 이에 본 발명을 통하여 샤딩 환경에서의 블록체인과 관련된 모든 연구 및 시스템에

기반이 되는 이더리움 샤딩에서 CST(Cross Shard Transaction) 측정 기술을 제안한다.

과제의 해결 수단

[0008] 본 발명의 일 실시예에 따른 CST 측정 장치는 샤딩 환경을 적용한 블록체인에 포함된 복수의 블록들 중 적어도 일부를 수집하는 데이터 수집부, 수집된 블록들로부터 트랜잭션 정보를 추출하는 트랜잭션 추출부, 및 추출된 트랜잭션 정보를 이용하여, CST(Cross Shard Transaction) 빈도를 산출하는 CST 빈도 산출부를 포함한다.

[0009] 또한, 본 발명의 일 실시예에 따른 CST 측정 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되고, 샤딩 환경을 적용한 블록체인에 포함된 복수의 블록들 중 적어도 일부를 수집하는 단계, 수집된 블록들로부터 트랜잭션 정보를 추출하는 단계, 및 추출된 트랜잭션 정보를 이용하여, CST 빈도를 산출하는 단계를 포함한다.

발명의 효과

[0010] 본 발명에서는 샤딩 환경(예컨대, 이더리움 2.0 등)을 적용한 블록체인을 이용하는 시스템상에서 CST 발생 빈도를 측정할 수 있는 기술을 제공한다. 이러한 기술은 CST 증가로 인한 시스템 처리량, 대기시간 및 트랜잭션 비용의 문제를 사전에 방지할 수 있으며 시스템을 보다 효율적으로 운영할 수 있다. 또한, 이는 향후 샤딩 환경을 적용한 블록체인을 연구하는 모든 학문에 기반이 될 수 있으며, 상업적으로 이용할 시 사전에 CST 측정하여 시스템을 관리 함으로써 효율적으로 시스템을 사용할 수 있다.

도면의 간단한 설명

[0011] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.

도 1은 본 발명의 일 실시예에 따른 CST 측정 장치의 기능 블럭도이다.

도 2는 도 1에 도시된 CST 측정 장치에 의해 수행되는 CST 측정 방법을 설명하기 위한 흐름도이다.

도 3a와 도 3b는 CST를 측정하기 위한 소스코드를 도시한다.

도 4는 도 1에 도시된 시각화부에 의해 출력된 예시 화면을 도시한다.

발명을 실시하기 위한 구체적인 내용

[0012] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태들로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0013] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.

[0014] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.

[0015] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0016] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구

성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0017] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0018] 이하, 본 명세서에 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세히 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0019] 본 발명은 이더리움 2.0의 샤딩 기술 도입에 따라 서로 다른 샤드(shard) 상에 존재하는 어카운트(account)들 간의 거래 트랜잭션인 CST(Cross Shard Transaction) 생성 빈도를 측정하는 기술이다. CST는 블록체인의 샤딩 환경에서 발생하는 트랜잭션으로 블록체인의 처리량(Throughput), 대기시간(latency), 비용(Fee) 등에 영향을 끼친다. 이에 본 발명은 향후 이더리움 연구 및 관련 시스템 관리에 기반이 되는 기술로 블록체인의 종류 중 이더리움의 샤딩 환경에서 발생하는 CST 발생 빈도를 측정하는 기술을 제안한다. 그러나, 본 발명의 권리범위가 암호 화폐의 종류에 제한되는 것은 아니다.

[0020] 도 1은 본 발명의 일 실시예에 따른 CST 측정 장치의 기능 블럭도이고, 도 2는 도 1에 도시된 CST 측정 장치에 의해 수행되는 CST 측정 방법을 설명하기 위한 흐름도이다.

[0021] CST 측정 장치(100)는 적어도 프로세서(processor) 및/또는 메모리(memory)를 포함하는 컴퓨팅 장치로 구현될 수 있다. 따라서, CST 측정 방법을 이루는 단계들 중 적어도 일부는 컴퓨팅 장치의 프로세서의 동작으로 이해될 수도 있다. 컴퓨팅 장치는 PC(Personal Computer), 서버(server), 랩탑 컴퓨터, 노트북 등을 포함할 수 있다.

[0022] CST 측정 장치(100)는 데이터 수집부(110), 트랜잭션 추출부(120), 및 CST 빈도 산출부(130)를 포함한다. CST 측정 장치(100)는 시각화부(140)와 저장부(150) 중 적어도 하나를 더 포함할 수도 있다.

[0023] 데이터 수집부(110)는 블록체인, 블록체인의 적어도 일부(예컨대, 블록체인에 포함된 블록들 중 적어도 일부)를 유무선 통신망을 통하여 수집할 수 있다(S110). 실시예에 따라, 데이터 수집부(110)는 USB 메모리 장치와 같은 저장 장치로부터 소정의 입출력 인터페이스를 통하여 블록체인 등을 수신할 수도 있다. 또 다른 실시예로, 블록체인 등은 저장부(150)에 미리 저장되어 있을 수도 있다.

[0024] 트랜잭션 추출부(120)는 수신된 적어도 하나의 블록에 포함된 트랜잭션 정보를 추출할 수 있다(S120). 트랜잭션 정보는 트랜잭션을 식별하기 위한 식별 정보, 송신자의 어카운트 주소, 및 수신자의 어카운트 주소 중 적어도 하나를 포함할 수 있다.

[0025] CST 빈도 산출부(130)는 추출된 트랜잭션 정보를 이용하여 CST 빈도 및/또는 IST(Intra Shard Transaction) 빈도를 산출할 수 있다(S130). CST는 송신자의 어카운트 주소와 수신자의 어카운트 주조에 기초하여 결정될 수 있다. 예컨대, 송신자의 어카운트 주소가 속하는 샤드와 수신자의 어카운트 주소가 속하는 샤드가 상이할 경우, 해당 트랜잭션은 CST가 된다.

[0026] 시각화부(140)는 CST 빈도 산출부(130)에 의해 산출된 결과를 소정의 추력 장치(예컨대, 디스플레이 장치)를 통해 출력할 수 있다(S140). 일 예로, 시각화부(140)는 각 샤드 별 CST의 빈도를 그래프로 출력할 수 있다.

[0027] 저장부(150)에는 CST 측정 장치(100)의 동작을 위한 OS(Operating Systme), 프로그램, 어플리케이션 등이 저장되어 있을 수 있다. 또한, 저장부(150)에는 데이터 수집부(110)에 의해 수집된 데이터, 트랜잭션 추출부(120)에 의해 추출된 정보, CST 빈도 산출부(130)에 의한 산출 결과 등이 저장될 수 있다.

[0028] 도 3a와 도 3b는 CST를 측정하기 위한 소스코드를 도시한다.

[0029] 해당 기술은 visual studio 2017 v15.9.10 상에서 구현되었으며, 코드는 C++로 작성되었다. 이더리움의 트랜잭션은 이더리움 블록체인에서 일어나고 있는 모든 활동과 정보를 쉽게 검색할 수 있는 이더스캔을 활용하였다.

[0030] 도 3a와 도 3b에서, data.txt는 추출한 데이터(예컨대, 트랜잭션 정보)가 저장된 파일이며, datafinal.txt는 CST 측정 알고리즘을 통하여 출력된 결괏값이 저장된 파일이다. Shard 구조체에는 사용자가 지정한 샤드와 그에 대한 CST를 셀 수 있는 변수 CST이다.(이는 향후 사용자의 설정애 따라 샤드 수 조정이 가능하다.) 본 발명에서

는 16개의 샤드를 기준으로 CST를 측정하였다. IntraST는 같은 샤드에 존재하는 어카운트 간의 트랜잭션으로 초기 설정 시 0으로 초기화한다.

[0031] if (datafile.is_open()) : data.txt 파일에 저장된 데이터 속의 CST를 측정한다. data.txt로부터 추출된 데이터는 array[i]에 순차적으로 저장된다.

[0032] array[i] = array[i+1] : 트랜잭션의 각 어카운트가 포함된 샤드가 다를 경우, Shard 구조체에 저장된 x번째 샤드의 CST 변수에 1이 추가된다. 어카운트의 샤드가 동일할 경우, IntraST 변수에 1이 추가된다.

[0033] struct nshard : data.txt에 저장된 데이터로부터 CST 발생 빈도가 측정된 결괏값은 datafinal.txt에 저장된다. struct nshard는 datafinal.txt에 저장된 CST 발생 빈도 출력 결과를 보여준다.

[0034] 마지막으로 datfinal.txt.의 수정을 닫는 메시지를 보낸다.

[0035] 도 4는 도 1에 도시된 시각화부에 의해 출력된 예시 화면을 도시한다.

[0036] 도 3을 통해 설명한 CST 측정 알고리즘을 통하여 측정된 데이터 결괏값 datafinal.txt를 시각적으로 보이기 위하여 2차원이나 3차원 함수나 자료를 그래프로 그려주는 소프트웨어 GNUPlot를 사용할 수 있다.

[0037] GNU.exe 실행 후 datafinal.txt가 저장된 파일을 선택한다.

[0038] set yrange [0:50] : y축의 범위가 0부터 50이 되도록 설정한다.

[0039] p ‘datafinal.txt’ using 1:20 : datafinal.txt에 저장된 자료를 그래프로 시각화하여 출력한다.

[0040] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 콘트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable 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(Operation System, 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 기술 분야에서 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.

[0041] 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, 구성 요소(Component), 물리적 장치, 가상 장치(Virtual Equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(Signal Wave)에 영구적으로, 또는 일시적으로 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.

[0042] 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), 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. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수

있으며, 그 역도 마찬가지이다.

[0043] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성 요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

부호의 설명

[0044] 100 : CST 측정 장치

110 : 데이터 수집부

120 : 트랜잭션 추출부

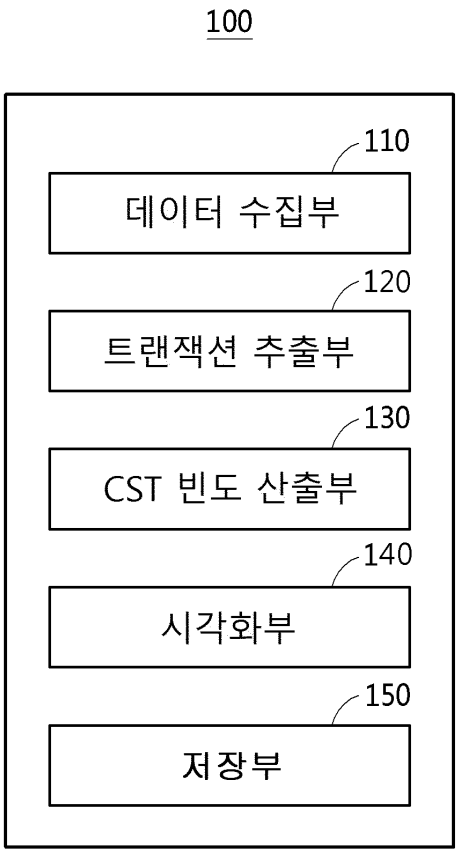
130 : CST 빈도 산출부

140 : 시각화부

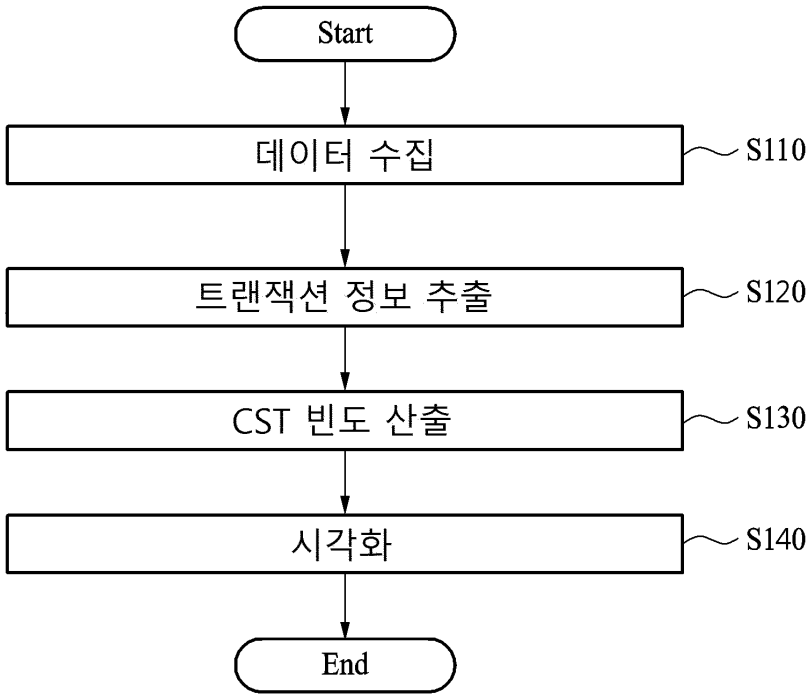
150 : 저장부

도면

도면1



도면2



도면3a

```
#include <iostream>
#include <string>
#include <fstream>
constexpr auto arraysize = 1024;;
using namespace std;

typedef struct Shard{
    char nshard;
    int CST;
};

int main()
{
    struct Shard shard[16]{
        {'a',0},{'b',0}, {'c',0}, {'d',0}, {'e',0}, {'f',0}, {'0',0}, {'1',0}, {'2',0}, {'3',
    {'6',0}, {'7',0}, {'8',0}, {'9',0}
    };
    char s_shard[16];
    char * array = new char[arraysize];
    int data = 0;
    int IntraST = 0;

    ifstream datafile("data.txt");
    ofstream datafinal("datafinal.txt");

    if (datafile.is_open())
    {
        while (!datafile.eof() && data < arraysize)
        {
            datafile.get(array[data]);
            data++;
        }
```

도면3b

```
for (int i = 0; array[i] != 'WO'; i++) {
    if (array[i] == array[i + 1])
    {
        cout << "same shard TxWn";
        cout << array[i] << array[i + 1] << "+##" << i << "()" << "Wn";
        IntraST++;
        cout << "WnWnIntraST : " << IntraST << "Wn";
        cout << "=====================================Wn";
    }
    else if(array[i] != array[i + 1] && array[i]!= 'Wn' && array[i+1] != 'Wn' &
        cout << "different shard TxWn";
        cout << array[i] << array[i + 1] << "+##" << i << "()" << "Wn";
        cout << "-------------------------------Wn";

        for (int j = 0;j < 16;j++) {
            if (array[i] == shard[j].nshard && array[i] != 'Wn' && array[i + 1]
                cout << shard[j].nshard << " shard " << j << " num" << "Wn";
                shard[j].CST++;
                cout << "WnWnCST : " << shard[j].CST;
                cout << "Wn=====================================Wn";
            }
        }
    }
```

도면4