대량의 트랜잭션을 처리하는 블록체인을 위한 분산처리 시스템

고혁준*, 한성수**, 정창성***
*고려대학교 영상정보처리협동과정
**강원대학교 자유전공학부
***고려대학교 전기전자공학부
e-mail : doltwo@hanmail.net*, sshan1@kangwon.ac.kr**, csjeong@korea.ac.kr***

Distributed processing system for blockchain processing a large number of transactions

Hyug-Jun Ko*, Seong-Soo Han**, Chang-Sung Jeong***
*Dept. of Visual Information Processing, Korea University
**Dept. of Division of Liberal Studies, Kangwon National University
***Dept. of Electrical Engineering, Korea University

요 약

최근 비트코인(Bitcoin)과 이더리움(Etherium)과 같은 퍼블릭 블록체인(Public Blockchain) 사용자의 급격한 증가로 인하여 블록체인 지갑 사용자가 늘어나고 있다. 또한, 암호화폐 거래소의 거래량이 증가와 이로 인한 지갑의 잔액 조회와 코인 이체를 위한 트랜잭션이 빈번하게 이루어 지고 있다. 한편, 최신의 잔액 조회와 빠른 이체를 위하여 마이닝 풀(Mining Pool)에서 사용되는 노드(Node)를 사용하는 것 같이 트래픽이 일부 노드에 집중되는 현상이 발생하여 시스템의 성능이 저하되는 문제가 있다. 이러한 문제를 해결하기 위하여 본 연구에서는 아파치 카프카(Apache Kafka)를 이용하여 트래픽 분산처리를 통한 효율적인 시스템을 제안한다. 또한, 시스템의 구조 설계 및 상세 모듈 설계를 제안한다. 제안 시스템은 기존 블록체인 시스템과의 연계가 가능하며, 기존 시스템의 변경 없이 구축할 수 있다. 또한, 주키퍼(ZooKeeper)의 분산처리를 통해 고성능과 가용성 및 안정성을 확보할 수 있다.

1. 서론

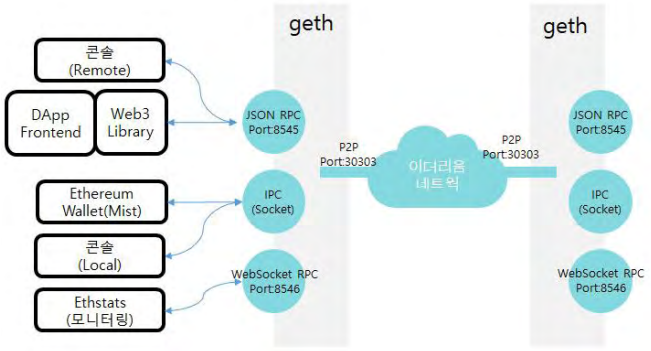
최근 비트코인과 이더리움등 퍼블릭 블록체인의 발전을 바탕으로 실 사용을 위한 암호화폐의 지갑이 시린랩스(Sirin Labs)의 스마트폰 핀니(Finney)에 장착되고, 삼성전자의 갤럭시 S10 에 장착이 되는 등 전 세계적인 보급이 예정되어 스마트폰 지갑(Wallet)을 이용한 지불 결제 및, 스마트 컨트랙트(Smart Contract)를 기반으로 작성된 댑(DApp)을 통하여 다양한 블록체인 서비스가 예상되는 시점이다. 이에 대응하는 블록체인 노드(Node)들은 설계 시점부터 현재까지 분산 시스템으로 안정성은 높지만 고성능을 내기에는 부족한 시스템들이다. 즉, 주요 노드들의 수는 2019 년 3 월 비트코인 약 1 만개, 이더리움 약 8 천개이며 이 노드들은 로드 밸런싱(Load Balancing) 되어 서비스가 되는 것이 아니라 노드 개별적으로 서비스를 하기 때문에 특정 노드로 트래픽이 집중될 가능성이 높다. 또한 노드에 집중된 트래픽은 P2P(Peer to Peer)를 통하여 각 노드에 전송되어 블록으로 저장되는 과정을 거치므로 하나의 노드에 집중된 트래픽은 전체 블록체인 네트워크에 위협요소가 된다. 예를 들어 2017 년 12 월 이더리움의 댑(Dapp)인 세계 최초의 블록체인 기술 기반의 온라인 게임 크립토 키티는 한때 1 만명이 넘는 DAU(Daily Active User)를 기록하며, 대량의 트래픽을 발생 시켰고, 특정 노드가 마비되는 경우가 있었으며, 전체 트래픽의 급격한 상승으로 네트워크의 과부하와 트랜잭션의 수수료가 급상승하는 문제를 발생되어, 블록체인의 스케일링(Scaling)에 대한 문제를 현실적으로 부각 시켰다. 블록체인 하나의 노드에서 대량의 트랜잭션 서비스를 한다는 것은 무척 어려운 일이다. 그러므로 서비스를 주 목적으로 하는 대규모 트래픽을 담당하는 노드의 설비가 필요하며, 본 논문에서는 이 노드를 구현하는 방법을 설명하려 한다.

대규모 트래픽을 이를 한번에 처리할 수가 없으므로 분산시스템을 사용해야 하며, 분산시스템을 가장 효과적으로 프로세스간의 협력과 경합을 조절하기 위해 주키퍼(ZooKeeper)[1]를 사용하고, 동시에 많은 트래픽을 순차적으로 처리하기 위해 분산 스트리밍 플랫폼(Distributed Streaming Platform)인 아파치 카프카[2]를 사용하는 방식을 제안하려고 한다.

2. 관련 연구

블록체인은 분산 원장(Distributed Ledger)을 생성하기 위해 사용되는 데이터 구조를 가지며, 블록들은 순차적인 방식으로 구성되어 있다. 블록은 일련의 트랜잭션(Transacton) 목록, 이전 블록의 해시(Block Hash), 블록생성시점인 타임스탬프(Timestamp), 블록 논스(Nonce), 블록 높이(Block Height)등으로 구성되어 있고, 모든 블록은 이전 블록의 고유 해시 값을 가지고 있으므로 이 값을 기준으로 일렬로 연결된 블록체인을 생성한다. 네트워크내의 모든 노드는 블록체인의 사본을 보유하고 있으며, 비트코인의 노드는 2019년 3 월 27 일 기준으로 약 10,326 개가 운영중이며[3], 이더리움의 노드는 8,577 개가 운영중이다[4].

비트코인은 UTXO(Unspent Transaction Output)를 기반으로 하는 분산원장을 기록하는 블록체인으로, P2PKH(Pay to Public Key Hash)를 기반으로 트랜잭션의 유효성을 검증하고, 기록한다[5]. 반면, 이더리움은 트랜잭션 기반의 상태머신(Transaction-based State Machine)이라고도 불리며, 이더리움상의 모든 계정(Account)의 정보를 포함하는 순간의 스냅샷(Snapshot)을 저장한 것을 상태(Status)라하고 이 상태를 기반으로 블록체인을 구성하며, 특별히 튜링 완전(Turing Completeness)한 언어인 솔리디티(Solidity)를 사용하여 스마트 컨트랙트(Smart Contract)를 가능하게 하고, 분산 애플리케이션을 구현하며, <그림 1>과 같이 이더리움 네트워크를 통해 스마트 컨트랙트를 실행하는 플랫폼(Platform)이다[6].

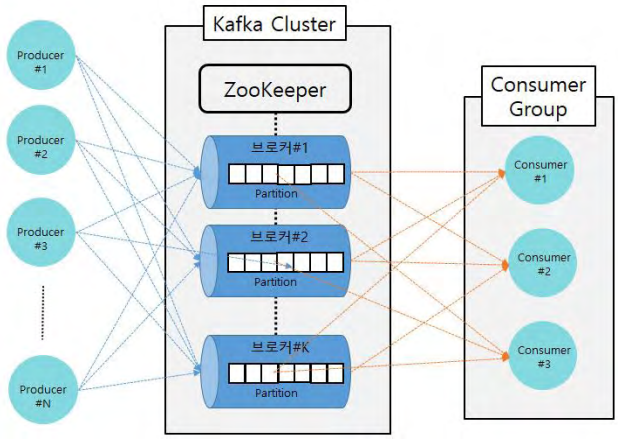


<그림 1. 이더리움 네트웍 구성도>

분산 애플리케이션인 댑(DApp)은 사용자가 분산 환경에서 비즈니스 거래 등을 편리하게 해주기 위해 사용되는 애플리케이션을 통칭하며, 비트코인도 하나의 DApp 으로 설명할 수 있으며, 반면 이더리움은 DApp 을 실행할 수 있는 탈 중앙화된 플랫폼으로 설명할 수 있다. 이 DApp 들은 스마트 컨트랙트를 사용해 작성되고, 하나 이상의 스마트 컨트랙트가 DApp 을 구성할 수 있다. 이더리움의 스마트 컨트랙트는 이더리움상에서 동작하는 프로그램이며, 다운타임(Downtime), 검열, 사기행위, 제 3 자의 간섭 없이 프로그래밍된 대로 정확히 실행된다.

카프카(Kafka)는 미국의 대표적인 비즈니스 인맥 소셜 네트워크 서비스인 링크드인[7]에서 개발한 대규모 메시지 데이터를 빠르게 처리하도록 개발된 메시징 플랫폼으로 클러스터로 구성이 가능하며, 클러스터를 통해서 단일 시스템 보다 더 높은 고성능을 가지며 장애 발생시 다른 노드가 대신 처리하는 고 가용성과 시스템 확장이 용이한 확장성(Scability)를 가진다[8].

주키퍼(ZooKeeper)는 YAHOO! Research[9]에서 개발되었으며, 클러스터를 구성한 분산 시스템의 코디네이션 작업을 가능케 하고, 프로세스간 협력이나 경합을 조절하는 분산 프로세스를 관리를 한다. <그림 2>와 같이 클러스터 적용시에 카프카는 발행(Publish)/구독(Subscribe) 기반의 분산 메시징 시스템(Distributed Messasing System)으로 동작하며, 장애감지(Detect crash), 토픽발견(topic discovery), 토픽에 대한 생성과 소모 상태를 관리하는 데는 주키퍼를 사용한다[10].



<그림 2. 카프카 클러스터 구성도>

3. 시스템 구성

이더리움이나 비트코인은 JSON RPC 를 지원하며, 이더리움의 경우 디폴트 포트는 8545 이고, 비트코인의 JSON RPC 의 포트는 8332 에 해당한다. 이 포트에 원격 접속을 통하여 비트코인을 송금을 할 수 있으며, 예를 들어 비트코인에서 company 라는 이름의 코인베이스에서 비트코인 주소

"39XPTV8V3e619uPjJ381xrHsVxtxJaUkcr"로 0.1 BTC 를 보내는 명령어는 <그림 3 >과 같다.

```
$ curl --user myusername --data-binary '{"jsonrpc": "1.0", "id":producer",₩
 "method": "sendfrom", "params": ["company", ₩
 "39XPTV8V3e619uPjJ381xrHsVxtxJaUkcr", 0.1, 6, " comment ", " comment_to"] }' ₩
 -H 'content-type: text/plain;' http://1.234.51.91:8332/
```

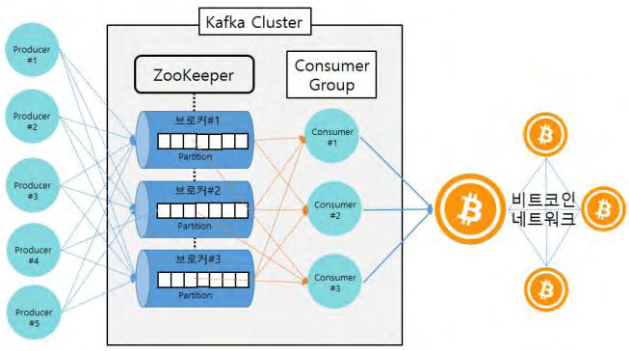
<그림 3. Curl 을 이용하여 JSON RPC 비트코인송금>

이에 대한 응답으로 트랜잭션 아이디(txid)를 받는다. 이는 동기방식(synchronous)이며, 비트코인의 노드가 해당 명령을 수행하여 트랜잭션을 생성하고, 응답이 되어야 연결이 종료된다. 대량의 트랜잭션이 발생하는 경우에는 명령당 소요시간이 있으므로, 명령수* 응답시간 만큼의 시간이 소요된다. 예를 들면 암호화폐의 코인의 에어드랍(Airdrop)처럼 해당 코인을 보유한 사용자들에게 코인을 보유한 비율대로 나눠주는

경우나 월 말 월급을 지급하는 것처럼 1 천명의 직원에게 일시에 송금할 경우 하나의 JSON RPC 를 통해 소켓 동기방식으로 송금하게 되면 각각 응답을 받아야 함으로 1 개의 JSON RPC 수행시간에 1,000 배의 장시간 소요된다. 소요되는 시간을 줄이기 위하여 병렬로 처리하기 위해 수십 개의 클라이언트에서 생성한 다수의 쓰레드로 JSON RPC 를 연결하여 비트코인의 노드에 전송하게 하면 해당 노드는 과부하로 인해 트랜잭션을 모두 처리하지 못하거나, 연결 수 제한을 인해 일부 접속은 원활하지 못하게 된다. 이렇게 동시에 접속되어 전송되는 대량의 트랜잭션을 처리하기 위해 본 논문에서 사용한 시스템은 아래 <표 1>과 같다.

용도	수량	사양
Broker /Zookeeper /Consumer	3	Xeon E3-1246V3(3.4GHz/4Core), 8G RAM SSD 250Gb OS : CentOS 7.3 Apache Kafka: 2.1.0 버전 ZooKeeper: 3.4.13 버전 openjdk version "1.8.0_201" Go v. 1.11.2
비트코인서버	1	Xeon E3-1246V3(3.4GHz/4Core), 8G RAM SSD 250Gb OS : CentOS 7.3 비트코인노드: 0.9.1 버전
Producer	5	G3930 (2.9GHz/2Core), 4G RAM, SSD128G OS : CentOS 7.3 Go v. 1.11.2

<표 1. Kafka Broker 하드웨어 사양>



<그림 4. 전체 시스템 구성도>

<그림 4>에서 5 대의 프로듀서는 각각 10 개의 Go 루틴(Routine)을 생성하여 20 회 반복해서 비트코인의 API 에 따른 JSON 형식의 데이터 생성하여 Kafka Cluster 에 연결하여 미리 정의한 토픽(Topic)인 “sendto”로 전송하게 되는 역할을 수행하고 프로세스를 종료하고, Kafka Cluster 의 각 브로커가 분산 처리하여 파티션 큐(Partition Queue)에 이를 보관하고 컨슈머(Consumer)는 해당 토픽의 내용을 정확히 한번(Exactly-once) 구독(Subscribe)하여 순차적으로 블록체인에 API 를 전송하여 안정적으로 블록체인에 전송하는 구조를 구성하게 된다.

5 개의 프로듀서가 생성한 대량의 트래픽을 카프카 클러스터는 하나의 거대한 메시지 큐가 되어 트래픽을 보관하는 기능을 수행하게 되고, 3 개의 컨슈머는 비트코인의 노드에 부하가 되지 않는 범위 내에서 하나씩 큐에서 꺼내어 비트코인의 노드와 통신하여 해당하는 트랜잭션 명령을 수행하게 된다. 트랜잭션명령외에 이더리움의 스마트 컨트랙트나 하이퍼 레저의 체인코드등 노드내에서 가상머신을 대량으로 실행하는 경우에는 블록체인의 처리를 가속화할 필요가 있고, 이때는 각각의 컨슈머가 하나의 노드에 접속하는 하는 대신 다수의 노드에 접속하여 각 노드의 스마트 컨트랙트 또는 체인코드를 실행하여 대량의 실행 명령을 분산 처리하여 짧은 시간 내에 처리할 수 있었다.

본 논문에서는 카프카 클러스터로 트래픽을 지연시켜서 대량의 비트코인 트랜잭션을 컨트롤하여 하나의 노드에 집중되는 순간적인 트래픽 폭주를 막을 수 있었고, 이더리움의 스마트 컨트랙트나 하이퍼 레저의 체인코드에서는 각각의 노드에 트랜잭션을 분산하여 블록체인의 처리속도를 가속화 할 수 있다.

4. 결론 및 향후 연구

4 차 산업혁명 기반기술로서 블록체인은 전 세계적으로 수천 개의 프로젝트가 진행되고 있다. 각 블록체인들은 각각의 필요에 따라 이용자를 위한 지갑 서비스를 사용하거나 사용될 것이다. 평상시에는 트래픽이 평이하게 발생되나, 특정 시점에 대규모 트래픽이 발생하여 블록체인의 전체 네트워크의 과부하와 특정 노드가 다운되어 원활한 서비스가 어려워질 수 있다. 원활한 블록체인의 운영을 위해서 트래픽을 감당할 수 있는 특정 노드는 구축되어야 하며 실험을 통해 카프카/주키퍼 분산처리를 통해 처리할 수 있는 시스템을 구현하였다.

향후에는 블록체인 노드에서 발생된 대규모 트래픽을 전체 노드에 P2P 를 통한 브로트캐스트(broadcast)하는 로직을 개선하여 카프카 클러스터를 이용한 분산 토폴로지(Topology)인 스타형 블록체인을 구현할 예정이다.

참고문헌

[1] Apache ZooKeeper, https://zookeeper.apache.org/
[2] Apache Kafka, http://kafka.apache.org/.
[3] BitNodes, https://bitnodes.earn.com/
[4] Etherscan, https://etherscan.io/nodetracker
[5] BitcoinCore, http://bircoincore.org
[6] Ethereum Project, https://www.ethereum.org
[7] Linked in, https://www.linkedin.com/
[8] 고승범, 공용준, “카프카, 데이터 플랫폼의 최강자” 책만, 2018 년 4 월
[9] YAHOO! Research, https://research.yahoo.com/
[10] Flavio Junqueira, Benjamin Reed, “주키퍼: 고가용성 서버를 위한 분산 프로세스 코디네이션” 에이콘, August 2014.