이더리움 데이터를 읽기 위한 API 와 노드활용 방식 비교분석

김민석 O1 문수묵 1
[1] 서울대학교 전기정보공학부
minseokk1m@snu.ac.kr, smoon@snu.ac.kr

Analysis and comparison of API and node based methods to read Ethereum data

Minseok Kim O1 Soo-Mook Moon 1
[1]Electrical & Computer Engineering, Seoul National University

요 약

본 연구는 이더리움(Ethereum) 블록체인 데이터에 접근하는 방식을 서버 API 를 활용한 방식과 블록체인 노드를 활용한 방식으로 나누어 그 특징과 장단점을 살펴본다. 이어서 각 방식별로 클라이언트(Client)가 이더리움 주소의 잔고 정보를 요청하고 받는 데에 까지 소요되는 반응시간(response time)을 측정 및 비교한다. 이를 통해 이더리움 정보를 이용하고자 하는 클라이언트가 목적에 따라 어떤 방식으로 블록체인 데이터에 접근하는 것이 좋을지에 대한 정보를 제공하고자 한다.

1. 서 론

이더리움(Ethereum)은 튜링 완전 스마트 컨트랙트 기능을 도입한 스테이트 머신(state-machine)으로 데이터 저장과 연산을 수행할 수 있다[1]. 저장과 연산 능력을 갖춘 이더리움 등의 블록체인을 활용하여 서버 백엔드(Back-end)를 대체하는 새로운 웹 아키텍처가 등장하였다. 이는 클라이언트-서버 통신 구조가 지배적인 웹 구조에서 클라이언트-블록체인 통신 구조를 제시했다는 데 큰 의미가 있다.

블록체인을 활용한 새로운 웹 아키텍처 상의 어플리케이션을 DApp (Decentralized Application)이라 한다. 2022 년 4 월을 기준으로 이더리움에는 약 3000 개의 DApp 이 있는 것으로 추산되며[2], 이더리움의 성장추이를 감안했을 때 앞으로 그 수는 더욱 늘어날 것으로 보인다.

블록체인 기반 웹과 DApp 에서는 블록체인의 데이터를 쉽게 읽고 활용할 수 있어야 한다. 그러나, 확장성 문제 등 현재의 블록체인이 가진 여러 한계[3]로 인해, 많은 블록체인 기반 서비스들이 블록체인만을 백엔드로 활용하지는 않고 있다. 즉, 블록체인 데이터를 읽고 활용함에 있어서 블록체인과 직접 통신하기도 하지만 클라이언트-서버 기반 웹과 같이 블록체인의 정보를 저장한 서버를 두고 서버와 통신하는 경우도 빈번하게 있다.

*** 이 논문은 2021 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00180, 다양한 산업 분야 활용성 증대를 위한 분산 저장된 대규모 데이터 고속 분석 기술 개발)

본 연구에서는 클라이언트가 블록체인 데이터를 얻는 방법을 크게 서버 API 와 통신하는 방법과 블록체인 노드와 통신하는 방법으로 나눈다. 각 방식의 장단점이 무엇인지 정량적으로 분석해보기 위하여 클라이언트가 해당 방법들로 이더리움 블록체인의 데이터를 얻는 데까지 걸린 시간을 측정하여 비교 분석한다.

이는 블록체인의 데이터를 활용해야만 하는 웹-블록체인 기반의 새로운 웹 아키텍처에서 향후 어떤 방식으로 블록체인 데이터를 얻어오는 것이 좋을지에 대한 방향성을 제시할 것으로 기대한다.

2. 배경지식

2.1 웹 아키텍처

기존의 웹 아키텍처(Web-Architecture)는 클라이언트-서버 구조로 이루어져 있다. 백엔드(Back-end)를 처리하는 중앙화된 웹서버를 두며, 다량의 데이터를 저장하기 위한 데이터베이스를 두기도 한다. 웹 서비스 사용자의 증가에 대처하기 위해, 서버와 데이터베이스를 수직,수평적으로 확장하는데, 수직적 확장은 서버의 사양을 높이는 방법이며, 수평적 확장은 서버를 추가하여 확장하는 방법이다[4].

반면 클라이언트-블록체인 웹 구조는 기존의 백엔드를 블록체인으로 대체한 탈중앙화 인터넷 환경이다. 여기서 구동되는 앱은 DApp 이다. 오늘날 많은 블록체인은

확장성 문제로 인해 중앙서버 기반의 웹에 비해 느린 연산 속도와 비싼 수수료를 발생시킨다는 단점이 있다.

2.2 DApp 에서 이더리움 데이터의 사용

DApp 에서는 블록체인 상의 정보를 이용한 서비스를 제공한다. DApp 으로 구현한 금융인 디파이(DeFi) 분야를 예로 들자면, 디파이에서 사용하는 토큰의 수량, 가격 등을 열람해야 하고, 토큰과 토큰 사이의 교환비율 등을 계산해야 한다. NFT 를 이용하는 DApp 에서는 사용자의 지갑을 연결하여, 사용자가 보유한 블록체인 상에서 소유한 NFT 가 무엇인지 확인해야 한다. 이러한 정보는 블록체인의 상태를 진행시키는 블록체인 노드가 가지고 있으므로, 블록체인 노드와 직간접적으로 연결하여 해당 데이터를 이용할 수 있게 된다.

3. 이더리움 데이터를 읽는 두가지 법

본 연구에서는 웹의 구현과 연동될 수 있는 스테이트 기반 블록체인 중 2022 년 5 월의 시가총액을 기준으로 규모가 가장 큰 이더리움 블록체인의 데이터를 읽어오는 법을 조사한다.

이더리움의 데이터를 읽는 방법은 크게 중앙서버의 API 와 통신하는 방식과 이더리움 노드와 직접 통신하는 방식이 있다. 전자의 경우, 서버의 API 를 이용하여 서버에서 가공된 정보를 쉽게 가져올 수 있다는 장점이 있다. 후자의 경우, 노드와의 RPC 통신으로 정보를 가져오는데, 이는 블록체인의 무신뢰 가정을 준수하기에 탈중앙성과 보안이 강화된다는 장점이 있다.

3.1 서버의 API 이용

이는 클라이언트가 중앙서버의 API 와 통신하여 서버에 저장되어 있는 이더리움과 관련한 각종 정보를 받아오는 방법이다. 클라이언트는 중앙서버를 직접 구축할 수도 있고, API 를 제공하는 기타 서비스를 이용할 수도 있다.

이더리움 API 를 제공하는 유명한 서비스로는 Alchemy, Infura, Opensea, Etherscan 등이 있다. 블록체인 지갑 DApp 인 Metamask 는 Infura 를 이용해 이더리움 정보를 가져오고 있으며, 모바일 버전에서는 NFT 와 관련한 정보에 접근하기 위해서 Opensea API 를 추가로 활용하고 있다. 또한 이더리움 기반 디파이 서비스인 Uniswap, Compound 등은 Infura API 를 이용하고 있다.

중앙서버를 구축하는 주체는 이더리움 풀노드(Full node) 등을 통해 받아온 정보로 IndexedDB 등의 서버 구조를 마련해놓고, 가공된 연산을 적용한 정보를 저장할 수 있다. 대부분의 중앙서버 API 는 이더리움 노드로의 단순 메소드 콜로 얻을 수 없는 정보를 가지고 있어서, 하나의 콜을 통해 원하는 정보를 쉽게 받아낼 수 있다는 장점이 있다. 따라서 특정 주소의 NFT 잔고, Dapp 에서 취급하는 토큰들의 가격정보 등의 정보를 얻어오기 적합할 것이다.

또한 Proxy 서버 등을 통해, 반복되는 데이터 읽기 요청 내용을 캐시(Cache)할 수 있다. 블록체인과 통신하여 소모되는 추가적인 시간을 절약할 수 있고, 반복되는 공통의 요청에 대해서 미리 연산을 하여 결과를 돌려줌으로써 시스템의 효율성을 높일 수 있다.

다만, 서버의 API 만을 의존하는 방식은 중앙화된 서버에 의존하기 때문에 탈중앙성과 보안성의 장점을 누리기 힘들다는 단점이 있다.

3.2 이더리움 노드와 RPC 콜 이용

이는 Geth 등 이더리움 클라이언트를 이용하여 이더리움 노드를 실행하고, RPC 콜을 이용해 이더리움 정보에 직접 접근하는 방법이다.

이더리움에서 토큰의 정보나 계좌 잔고 등을 확인하고 싶을 때는 이 방법을 활용하여 실시간성 있는 데이터를 가져올 수 있다. 하지만 이더리움 클라이언트가 제공하는 명령어가 제한적이어서, 이더리움 노드에서 직접적으로 열람할 수 있는 정보 역시 다소 한정된다는 단점이 있다. 예를 들어, 특정 이더리움의 주소가 보유한 ERC721 NFT 를 확인하고자 해도, 노드를 통해 이를 직접적으로 확인할 수 있는 함수는 제공되어 있지 않다.

해당 작업을 위해서는 ERC721 컨트랙트 주소의 모든 transfer 이벤트를 인덱싱해서 소유주소 정보를 분별해내는 작업을 거쳐야한다. 이는 이더리움의 자료구조가 유저에 대한 모든 인덱스 데이터 정보를 가지고 있는 것이 아니라, 트랜잭션 정보만을 가지고 있기 때문이다. 이더리움의 자료구조와 노드 명령어의 제한으로 인한 어려움이다.

이러한 이유로 서비스 개발단계에서부터 관심의 대상이 될만한 정보를 읽을 수 있도록 스마트 컨트랙트 내에 read 와 write 함수를 써 놓는 경우도 많다. 이 경우, 클라이언트는 RPC 콜을 통해 이더리움 노드로 스마트 컨트랙트의 함수를 호출하여 정보를 얻어올 수 있게 된다.

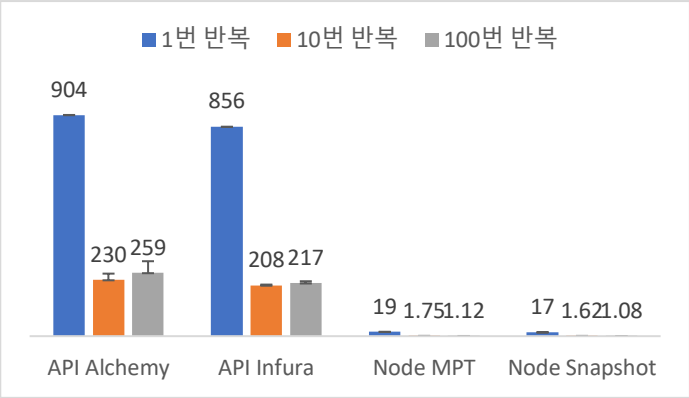
4. eth.getBalance() 반응시간 측정 및 평가

본 연구는 서버와 노드를 이용한 이더리움 정보 읽기의 반응시간(response time)을 비교 측정하였다. 서버 기반 방식의 실험을 위해 이더리움 블록체인 데이터를 제공하는 중앙서버 API 서비스인 Alchemy[5]와 Infura[6]를 선택하였다. 노드 기반 방식의 실험을 위해 GO 언어로 구현된 이더리움 클라이언트 Geth 를

이용하였고, 블록체인 정보를 MPT(Merkle Patricia Trie)와 Snapshot 자료구조[7]로 snap sync[8] 받았다.

모든 실험조건에서 동일하게 한 개 이더리움 어카운트(EoA)의 Ether 잔고 정보(eth.getBalance())를 데이터로 요청했으며, 데이터를 요청하기 직전과 받은 직후의 시간차를 반응시간으로 정의하였다.

각 실험군에서 최초 1 번만 데이터를 호출하는 경우, 10 번 반복하여 호출하는 경우, 100 번 반복하여 호출하는 경우의 1 회당 데이터 호출 반응시간의 평균과 표준편차를 구했다. 실험결과는 아래와 같으며 실험의 결과 단위는 Millisecond 이다.



그래프 1. 이더리움 getBalance() 응답속도(단위:msec)

실험 결과 Alchemy 와 Infura 중앙서버를 이용하여 정보를 얻어오는 반응시간이 노드를 이용한 나머지 방식들에 비해 크다는 것을 확인할 수 있었다. 중앙서버 기반 방식과 노드 기반 방식의 반응시간 차이는 표준편차 면에서도 확인되었다. 10 번 반복한 실험의 반응시간 표준편차는 Alchemy, Infura 를 이용한 경우 각각 26.305, 3.6154 였지만, MPT 노드와 Snapshot 노드를 이용한 경우 0.462, 0.517 이었다. 100 번 반복한 실험의 반응시간 표준편차는 Alchemy, Infura 를 이용한 경우 각각 47.97, 8.2148 였지만, MPT 노드와 Snapshot 노드를 이용한 경우 0.662, 0.49 이었다. 위 결과를 종합하자면 중앙서버를 이용한 이더리움 데이터 읽기 요청의 반응시간은 노드 기반 방식에 비해 평균과 표준편차가 모두 매우 크다. 데이터 읽기의 시간과 일관성 측면에서는 노드를 이용하여 이더리움 정보를 가져오는 것이 우월한 방식일 수 있음을 시사한다.

하지만 이더리움 노드를 운영하기 위해서는 일정한 수준의 하드웨어 수준을 맞추어야 한다는 제약이 있다. 이더리움 클라이언트를 위해서는 최소 2 개 이상의 CPU, 4GB RAM 이상의 SSD 혹은 8GB RAM 이상의 HDD 성능을 요하며, 저장공간을 가장 덜 차지하는 Snap sync 모드의 경우도 이더리움 블록체인을 저장하기 위해 최소 약 20GB 이상의 Storage 를 필요로 한다[9]. 이러한 성능 제약을 맞추기 위한 추가적인 컴퓨팅 자원을 고려하면 비용 측면에서는 서버 API 를 이용한 방식이 노드를 이용한 방식에 비해 우월할 수 있음을 시사한다.

5. 결론 및 향후 연구

탈중앙화의 장점을 제시한 새로운 프레임워크인 블록체인은 기존의 웹 백엔드를 대체하는 클라이언트-블록체인 구조의 웹 아키텍처를 제시한다. 블록체인 데이터를 읽는 방법은 크게 두가지로 나뉘는데, 첫째는 블록체인 노드와 통신하여 정보를 가져오는 방식이고 둘째는 중앙서버를 사이에 두고 서버 API 를 활용하여 정보를 가져오는 방식이다.

본 연구에서는 두 방식을 기반으로 이더리움 어카운트의 잔고 정보를 읽어오는 반응시간을 측정하는 실험을 진행했다. 서버 API 를 기반으로 하는 방식은 반응시간의 평균과 표준편차가 모두 크지만, 비용이 적다는 장점이 있다. 노드를 기반으로 하는 방식은 그 반대의 결과를 가진다.

향후 연구에서는 더 다양하고 복잡한 데이터를 처리하는 실험을 진행할 수 있을 것이다. 이를 통해 각 경우에 이더리움의 데이터를 읽기 위한 최적 웹 환경을 알아낼 수 있을 것으로 기대한다.

참 고 문 헌

[1] Buterin, Vitalik. "A next-generation smart contract and decentralized application platform." *white paper* 3, no. 37 (2014).

[2] https://www.stateofthedapps.com/platforms/ethereum)

[3] M. Bez, G. Fornari and T. Vardanega, "The scalability challenge of ethereum: An initial quantitative analysis," 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), 2019, pp. 167-176, doi: 10.1109/SOSE.2019.00031.

[4] Alex Xu, System Design Interview: an insider's guide

[5] https://dashboard.alchemyapi.io/

[6] https://infura.io/dashboard

[7] 이준모,이준하,문수묵. *이더리움 스냅샷 구조의 상태 데이터 접근 효율성 분석,* 한국정보과학회, 학술발표논문집, 2021, Vol.2021(12)

[8] https://ethereum.org/en/developers/docs/nodes-and-clients/#:~:text=under%203%20days-,Synchronization%20modes,building%20a%20local%20blockchain%20database.

[9] https://blog.ethereum.org/2021/03/03/geth-v1-10-0/