체인링크 기반 블록체인 오라클 시스템 온-체인 비용 분석

주하영[1], 김재훈[2]
[1]아주대학교 소프트웨어학과
[2]아주대학교 사이버보안학과
fpzkgkdud@ajou.ac.kr jaikim@ajou.ac.kr

On-chain Cost Analysis of Chainlink-based Blockchain Oracle System

HaYeong Joo[1], Jai-Hoon Kim[2]
[1]Dept. of Software Engineering, Ajou University
[2]Dept. of Cyber Security, Ajou University

요 약

Smart contract는 Off-chain 데이터를 On-chain에서 조회해보지 못하는 "Blockchain Oracle Problem"을 갖고 있다. Oracle Problem을 적용할 때 실제 사용자 측면에서 어떠한 비용이 발생하는지 연구가 부족하다고 느껴, Chainlink 오라클 아키텍처 기반 시스템의 On-chain 비용을 측정할 수 있는 식을 제안한다. 그 결과 각 아키텍처마다 gas fee, oracle fee, latency의 차이를 확인한다.

1. 서론

블록체인 기술은 암호화폐, 스마트 계약 및 DID(Distributed Identification)까지 다양한 분야에서 활용되고 있다. 특히, Ethereum은 Smart contract를 만들기 위한 Blockchain Platform으로써 고안되었다. 하지만, Smart contract는 "Blockchain Oracle Problem"이라는 태생적인 문제가 존재한다.

Blockchain oracle 문제를 해결하기 위한 아키텍처 제안 및 성능 평가 등 다양한 연구가 존재하지만, 실제 사용자 측 비용 면에서 분석한 연구가 부족해 본 연구를 수행하게 되었다.

2. 배경지식

2.1 EVM과 Smart contract

EVM(Ethereum Virtual Machine)은 Ethereum 블록체인에서 Smart contract를 실행하기 위한 런타임 환경이다. EVM은 quasi-turing-complete state machine이며, 연산 비용으로 "gas fee"를 소모한다[1]. Smart contract는 블록체인상에서 소프트웨어처럼 작동하는 가상의 계약이다. 정의된 기준이 충족되면 자동으로 트랜잭션을 실행하는데, 이때 gas fee를 소모한다.

2.2 Oracle Problem

Oracle Problem이란 Smart contract가 외부 데이터에 직접 접근할 수 없어서 발생하는 문제이다. Oracle은 블록체인 외부 환경인 Off-chain과 블록체인 자체 환경인 On-Chain을 연결하는 역할을 한다[2].

Chainlink는 On-Chain Roll-Up Smart Contract를 두고, 필요한 기능을 Off-chain에서 수행해 Roll-up smart contract에 그 결과를 전달하는 Layer 2 솔루션의 대표적인 예시이다. Roll-up smart contract(Oracle contract)가 Message broker의 역할을 한다.

2.3 Chainlink Oracle 아키텍처

Chainlink의 Oracle 아키텍처는 3가지로 나뉜다. Basic Request Model(BRM), Decentralized Oracle Model(DOM)과 Off-Chain Reporting(OCR)이다[3]. BRM은 request/reply model 기반, DOM과 OCR은 publish/subscribe model을 기반으로 동작한다.

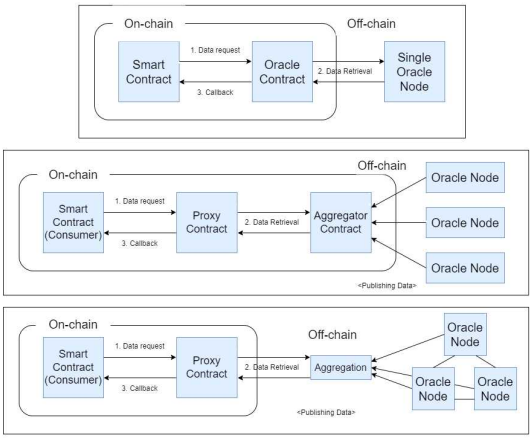
BRM은 다음과 같은 순서로 동작한다.

1. 사용자(Smart contract)는 Oracle contract에 데이터를 요청한다.
2. Oracle contract는 Oracle node에 데이터를 요청한다. Oracle node에서 데이터를 찾으면, Oracle

Contract로 데이터를 push한다.
3. Oracle contract는 다시 User contract로 데이터를 callback해 원래 과정을 수행한다.

DOM과 OCR은 Off-chain 데이터를 On-chain 환경에 push하는 aggregation을 수행한다.
1. 데이터를 제공하는 Oracle Node는 데이터를 Aggregator Contract에 publish한다.
2. Aggregator Contract는 Voting 방식을 사용해 data를 update 한다. 사용자는 subscribe를 통해 update 때마다 데이터를 받는다.

이때, DOM에서는 Voting 과정을 On-chain인 Aggregator Contract에서 수행하고, OCR에서는 Voting 과정을 Off-chain에서 별도로 수행해 On-chain으로 publish한다.



<그림 1 - Chainlink Oracle 아키텍처>

3. 비용 분석

위 Chainlink Oracle 아키텍처마다 사용되는 gas fee, oracle fee와 latency 측면에서 분석해보았다. 사용자 smart contract의 deploy 비용은 고려하지 않았다.

3.1 사용자 비용 분석

BRM에서 사용자는 request gas fee와 callback 될 때 비용인 response gas fee, oracle node에게 지불하는 oracle fee를 지불해야 한다. oracle fee는 Chainlink의 LINK를 사용한다. Smart contract는 요청을 보내는 시간, 데이터를 찾는 시간, 요청받는 시간 이후 실행된다.

DOM에서 사용자는 subscribe를 하는데 register gas fee, register oracle fee를 지불한다. 이후, k번의 요청이 도착하면, 사용자는 각 요청이 실행될 때마다 별도의 gas fee와 oracle fee를 낸다. Smart contract는 데이터를 받는 즉시 실행된다.

OCR에서도 DOM과 유사한 방식으로 비용을 지불한다. Smart contract 또한 데이터를 받는 즉시 실행된다.

DOM과 OCR의 oracle fee는 데이터를 aggregation하는 방식의 차이로 다르다. 하지만, 둘 다 Oracle Node의 수에 비례하고, subscriber의 수에 반비례한다.

- $\mathrm{cost}_{\mathrm{BRM}} = \mathrm{gas}_{\mathrm{req}} + \mathrm{fee}_{single} + gas_{res}$
- $t_{BRM} = t_{req} + t_{ret} + t_{res}$
- $\mathrm{cost}_{\mathrm{DOM}} = (\mathrm{gas}_{\mathrm{reg}} + \mathrm{fee}_{\mathrm{reg}}) + \mathrm{k}^{*}(\mathrm{gas}_{\mathrm{sub}} + \mathrm{fee}_{\mathrm{DOM}})$
- $\mathrm{cost}_{\mathrm{OCR}} = (\mathrm{gas}_{\mathrm{reg}} + \mathrm{fee}_{\mathrm{reg}}) + \mathrm{k}^{*}(\mathrm{gas}_{\mathrm{sub}} + \mathrm{fee}_{\mathrm{OCR}})$
- $t_{DOM}, t_{OCR} = t_{exe}$
- $fee_{DOM}, fee_{OCR} \propto \frac{M}{N}$
  (M은 $Oracle Node$의 수, N은 subscribr의 수)

3.2 Publish 비용 분석

DOM은 aggregation을 수행하는 과정에서, M개의 Oracle Node가 데이터를 각각 On-chain에 push한 후, On-chain 상에서 데이터에 대한 합의를 이룬다.

OCR은 Off-chain에서 Oracle Node끼리 합의를 이룬 후, 단일한 결과를 On-chain에 전달한다. 따라서 OCR의 publish 비용이 더 저렴하다.

- $\mathrm{cost}_{\mathrm{DOMpub}} = \mathrm{M}^{*}\mathrm{gas}_{\mathrm{pub}}{}^{*}\mathrm{T} + \mathrm{gas}_{\mathrm{consensus}}$
- $\mathrm{cost}_{OCRpub} = gas_{pub} * T$
  (M은 $Node$의 수, T는 $publish$주기)

4. 결론 및 향후 계획

본 연구에서는 Chainlink Oracle 아키텍처에 따라 사용자가 내야 하는 비용이 다름을 표현했다. 하지만, smart contract의 deploy 비용, 별도의 oracle 수수료, Ethereum 네트워크 상황 등을 고려했을 때 더 다양한 요인을 포함한 분석이 필요하다. 향후, 다양한 요인을 포함한 분석을 수행하고 실제로 환경도 구축해 그 분석을 뒷받침할 예정이다.

참고문헌

[1] Ethereum, https://ethereum.org/en/, 2023.
[2] Adbeljalil Beniiche, A Study of Blockchain Oracles, arXiv, 2020.
[3] ChainLink, https://docs.chain.link/, 2023.