Hyperledger Indy 네트워크 가용성 향상을 위한 모니터링 시스템 설계

최규현, 김근형
동의대학교 게임공학
giry8647@gmail.com, geunkim@deu.ac.kr

A Design of Monitoring System for Availability Improvement of Hyperledger Indy Network

Gyu-Hyun Choi, Geun-Hyung Kim
Game Engineering Major, Dong-eui University

요 약

Hyperledger Indy는 탈중앙 신원증명을 위한 블록체인 기반 분산원장으로 디지털 신원증명을 위한 여러 기능을 제공한다. 디지털 신원증명은 디지털 생태계의 핵심 요소이기 때문에 디지털 신원증명 시스템의 가용성 확보는 끊김이 없는 서비스를 제공한다는 관점에서 중요하다. 본 논문에서는 Hyperledger Indy 기반 신원증명 시스템의 가용성을 높이는 핵심 아이디어의 검증 결과를 근거로 모니터링 시스템의 설계 내용을 기술한다.

1. 서론

최근 블록체인 기술을 활용한 서비스에 대한 연구가 활발하게 이루어지고 있다. 탈중앙 신원증명을 위한 분산 원장인 Hyperledger Indy(이하 Indy)는 디지털 신원증명을 위한 여러 가지 기능을 제공한다. Indy를 사용하면 특정 중앙 기관에 개인 정보가 밀집되는 현상을 방지할 수 있으며 자신의 개인 정보를 직접 관리 및 증명하는 것이 가능하다[1].

디지털 신원증명은 온라인 생태계의 핵심 요소로 탈중앙 신원증명 시스템인 Indy의 서비스 가용성을 보장하는 기술이 요구된다. 또한 Indy 네트워크 문제로 인해 가용성이 떨어지는 상황이 발생할 수 있으며, 최악의 경우 서비스 이용이 불가능할 수도 있다. 본 논문에서는 Indy 네트워크을 구성하는 노드에 문제가 발생하여 Indy의 기능이 중단되는 상황을 방지하는 방법을 검증하고 Indy 네트워크의 가용성을 높이는 모니터링 시스템을 제안한다.

2. Hyperledger Indy

Indy는 앞서 설명한 바와 같이 블록체인을 활용하며, 분산 네트워크 환경을 지니고 있다. Indy의 분산 네트워크는 중앙의 서버에서만 데이터를 저장하는 것이 아닌 여러 노드가 같은 정보를 저장한다. Indy의 네트워크는 노드를 통해 참여할 수 있으며 많은 수의 노드가 모인 네트워크를 'Pool'(이하 풀)이라 한다. Indy는 풀 내의 노드들의 합의를 통해 트랜잭션 데이터를 블록체인에 추가한다. Indy의 풀에 트랜잭션 데이터의 저장은 Redundant Byzantine Fault Tolerance (RBFT) 합의 알고리즘을 기반으로 동작한다. RBFT는 소수의 문제 노드가 있어도 합의가 이루어져 트랜잭션 데이터를 네트워크에 저장하는 것을 가능하게 한다. RBFT 합의 알고리즘은 문제 노드의 수가 f이면 정상적인 합의가 이루어지기 위한 총 노드 수 N은 다음 수식을 만족하여야 한다.

$$3f + 1 < N \qquad (1)$$

풀을 구성하는 노드 중 f개의 노드에서 문제가 발생하더라도 전체 노드의 수가 3f + 1 개 보다 많다면 Indy 풀의 동작은 멈추지 않는다. 위 수식에 따르면 문제 노드가 1개 발생하여도 최소 3개의 노드가 정상적으로 동작을 한다면 합의가 이루어진다[2][3].

Indy는 오픈소스로 노드 생성 및 실행 기능을 제공하는 indy-node와 이러한 노드 이용을 위한 Application Programming Interface(API)를 제공하는 indy-sdk가 있다. indy-node는 노드 시작을 위한 기능을 제공하며 이를 사용해 새로운 노드를 만들거나 동작시킬 수 있다. indy-sdk는 풀에 새로운 노드 추

가 및 현재 풀에 있는 노드 정보 확인에 사용된다.

3. 제안 방안

Indy는 각각의 노드가 서로의 정보를 복제하여 공유하기 때문에 풀이 동작한다면 일부의 노드에 문제가 발생하여 서비스를 제공하지 못하는 경우가 발생하기 전에 풀에 노드를 추가하여 이전 노드의 정보를 복제할 수 있다. 또한 일부 노드에 문제가 발생하여도 다수의 노드가 제대로 동작한다면 일부 노드의 문제에도 불구하고 풀의 상태는 정상 상태를 유지한다. 그러나 풀이 동작을 멈추는 상황이 발생하면 클라이언트가 서비스 이용을 하지 못하게 된다[4]. 이를 해결하기 위해 풀의 동작을 멈추기 전에 노드를 추가하는 방법을 제안한다. 위에서 언급한 바와 같이 풀 내의 모든 노드들은 같은 정보를 가지도록 동작하며 새로운 노드가 추가되면 새로운 노드에 이전 노드가 가지는 정보가 복제된다. 이러한 점을 이용하여 풀의 노드의 수가 합의에 필요한 최소의 수 이하로 줄기 전에 새로운 노드를 추가하여 풀이 멈추는 것을 막는다.

<표 1> Indy의 RBFT 알고리즘 테스트

No.	전체 노드 수	동작 노드 수	추가 노드 수	문제 노드 수	풀 동작 상태
1	10	10	0	0	O
2	10	7	0	3	O
3	13	10	3	3	O
4	13	9	3	4	O
5	19	15	9	4	O
6	21	15	11	6	O
7	21	14	11	7	X

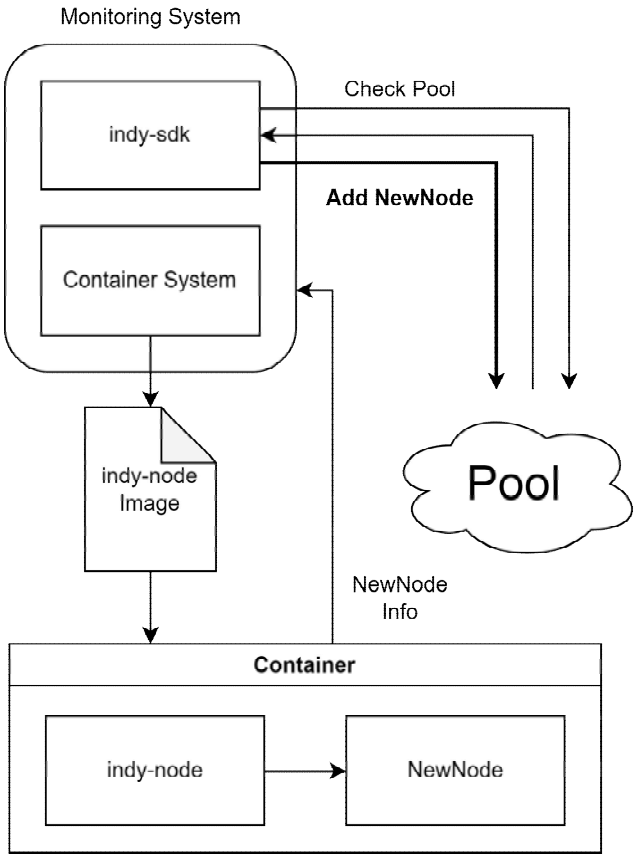
<표 1>은 풀 내의 일부 노드에서 문제가 발생하여도 합의 알고리즘이 동작할 수 있는 최소 노드 수를 유지하여 풀의 가용성을 높일 수 있는 방법을 테스트 한 결과이다. 처음 풀에는 노드가 10개로 구성하여 동작을 하였다. 수식 (1)에 따르면 10개의 노드 중 3개의 노드에서 문제가 발생하여도 풀은 정상적으로 동작한다. 그러나 그 이상의 노드에서 문제가 발생하면 풀은 정상적인 동작을 하지 못한다. 3단계에서 새로운 노드 3개를 풀에 추가하면 4단계처럼 한 노드에서 추가로 문제가 발생하여도 이미 3단계에서 새로운 노드 3개를 더 추가하여 합의 알고리즘을 구동할 수 있는 최소 노드 수가 9가 되어 풀은 정상적으로 동작됨을 확인하였다. 5단계에서는 풀의 가용성을 높이기 위해 새로운 노드 6개를 추가하여 풀에는 19개의 노드가 존재하며 문제가 발생한 노드가 4개, 정상적인 노드가 15개로 동작한다. 풀의 노드가 19개이므로 합의에 참여하는 노드의 수가 13개로 증가하여 2개의 노드에서 문제가 발생하여도 풀은 정성적으로 동작을 한다. 6단계에서는 새로운 노드를 2개 추가하여 풀의 총 노드 수를 21개가 된 상태에서 2개의 노드에서 문제가 발생하여 풀 내의 문제 노드가 총 6개가 되었다. 총 노드의 수가 21개인 경우에도 수식 (1)에 따라 합의에 필요한 최소 노드 수가 15개로 풀은 정상 동작하였다. 7단계는 노드 한 개에서 문제가 발생하여 정상 동작하는 노드의 수가 14개로 합의에 필요한 최소 노드 수보다 적어 풀이 동작하지 않음을 확인하였다. <표 1>의 실험 결과를 바탕으로 Indy 네트워크에서 동작에 문제가 있는 노드의 수를 감지하여 새로운 노드를 추가하여 Indy 네트워크의 가용성을 높이는 모니터링 시스템을 제안한다.

모니터링 시스템은 Indy 네트워크의 전체 노드 수와 정상적으로 동작이 되지 않는 노드 수 정보를 확인하는 방법이 필요하다. 이 정보는 Indy의 두 기능 중 indy-sdk의 Restful API 인 Validator_info를 사용하여 얻는다. ‘Validator_info’ API는 풀의 전체 노드 수와 현재 동작 중인 노드 수를 반환한다[5].

문제 노드 수가 일정 개수 이상 증가하여 정상적으로 동작 중인 노드의 수가 합의에 참여하는 최소 노드 수 보다 적게 되지 않도록 새로운 노드를 추가한다. 새로운 노드는 indy-node 도커 이미지를 컨테이너화 하여 풀에 추가된다. 새로운 노드 추가는 노드 생성 및 실행 기능을 제공하는 indy-node와 풀에 노드 추가 요청 기능을 제공하는 indy-sdk가 사용된다. 먼저 indy-node를 통해 새로운 노드를 생성 및 실행시킨 후, indy-sdk를 사용해 실행된 새로운 노드를 기존 풀에 추가한다. 마지막으로 ‘Validator_info’를 요청하여 노드가 제대로 추가되었는지 확인한다.

모니터링 시스템은 Indy 네트워크를 실시간으로 감시하며 네트워크의 노드 수를 확인한다. 이때 정상적으로 동작하는 노드의 수가 합의에 필요한 최소

노드 수의 차이가 일정 수 이하가 되면 새로운 노드를 추가한다. 풀의 상황에 따라 새로운 노드를 추가하여 풀의 서비스가 중단되지 않도록 한다. (그림 1)은 설계한 모니터링 시스템의 진행도이다.



(그림 1) 모니터링 시스템 진행도

모니터링 시스템은 풀의 노드 수를 확인하기 위해 indy-sdk를 사용하여 'Vaildator_info'를 요청한다. 'Vaildator_info'받아 노드 수를 확인하여 합의에 필요한 최소 노드 수가 충족되는지를 확인한다. 만약 정상적인 노드 수가 합의에 필요한 최소 노드 수에 가까울 경우 Container System을 사용해 미리 만들어둔 indy-node Image로 Container를 만든다. Container 내부에 있는 indy-node를 사용하여 New Node를 생성 및 실행하고 새로 만들어진 NewNode 정보를 모니터링 시스템에 전달한다. 모니터링 시스템은 받은 NewNode 정보를 사용해 풀에 노드 추가 요청을 전송한다. 마지막으로 'Vaildator_info'를 요청하여 노드가 정상적으로 추가되었는지를 확인한다.

4. 결론

본 논문은 Indy 풀을 구성하는 노드 중 일부 노드에 문제가 발생하여 풀의 서비스가 중단되지 않도록 풀의 상태를 자동적으로 체크하며 관리하기 위한 모니터링 시스템 설계내용을 기술하였다. 향후 해당 설계를 바탕으로 실제 모니터링 시스템을 구현하여 가용성 테스트를 진행하며 문제점을 보완해 나갈 것이다.

Acknowledgement

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2021R1F1A1047573).

참고문헌

[1] Tracy Kuhrt. hyperledger wiki. "Hyperledger Indy". January 3. 2019. https://wiki.hyperledger.org/display/Indy/Hyperledger+Indy.

[2] P. -L. Aublin, S. B. Mokhtar and V. Quéma, "RBFT: Redundant Byzantine Fault Tolerance," 2013 IEEE 33rd International Conference on Distributed Computing Systems, 2013, pp. 297-306, doi: 10.1109/ICDCS.2013.53.

[3] 윤대근 "자기주권 신원증명 구조 분석서" 제이펍 2020

[4] Sergey Khoroshavin. indy-node. "Indy Node troubleshooting guide". March 10. 2020. https://github.com/hyperledger/indy-node/blob/main/docs/source/troubleshooting.md.

[5] Alexander Shcherbakov. indy-node. "Extension of validator_info". May 24. 2018. https://github.com/hyperledger/indy-node/blob/main/design/validator_info.md.