Performance Optimization of the Endorsement Phase Using Channels in Hyperledger Fabric

Minsu Kwon, Junebeom Jang, Jaehak Lee, Heonchang Yu
Dept of Computer Science and Engineering, Korea University

요약
블록체인은 4차 산업혁명 핵심 기술로 각광받고 있다. 하지만 현재 블록체인 기술은 문제점이 많아 사용가는 것이 적었다. 블록체인의 문제점 중 하나는 블록체인의 성능 저하가 있다. 효율적이고 안정적인 블록체인 시스템을 구축하기 위해서는 블록체인의 성능을 향상시키는 것이 중요하다.

1. 서론
본 논문에서는 블록체인의 성능 향상에 대한 연구를 진행하였다. 블록체인의 성능 향상에 대한 연구는 블록체인의 성능을 향상시키기 위한 방법론을 개발하고자 한다. 블록체인의 성능을 향상시키기 위한 방법론은 블록체인의 성능을 향상시키기 위한 방법론을 개발하고자 한다.

1.1 관련 연구
1.1.1 페브릭의 트랜잭션 처리 속도에 영향을 미친 요인
페브릭에서의 트랜잭션 처리 속도에 영향을 미치는 요인으로는 트랜잭션의 크기, 트랜잭션의 처리 순서, 트랜잭션의 처리 횟수 등이 있다. 이러한 요인들에 따라 트랜잭션 처리 속도가 달라질 수 있다.

1.1.2 트랜잭션 처리 속도의 향상
페브릭에서의 트랜잭션 처리 속도를 향상시키기 위한 방법론은 트랜잭션 처리 속도를 향상시키기 위한 방법론을 개발하고자 한다.

2. 관련 연구
2.1 페브릭의 트랜잭션 처리 과정
페브릭의 트랜잭션 처리 과정은 트랜잭션의 생성, 트랜잭션의 검증, 트랜잭션의 저장, 트랜잭션의 비교 등이 있다. 트랜잭션의 처리 과정은 트랜잭션의 처리 속도를 개선하기 위한 방법론을 개발하고자 한다.

2.1.1 트랜잭션 처리 속도의 향상
페브릭에서의 트랜잭션 처리 속도를 향상시키기 위한 방법론은 트랜잭션 처리 속도를 향상시키기 위한 방법론을 개발하고자 한다.

1 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구 지원사업의 연구결과로 수행하였음
2 이 논문은 2019년도 정부(과학기술정보통신부)의 제도로 정보통신기획평가원의 지원을 받아 수행된 연구임
3 이 논문은 블록체인 이슈 및 시뮬레이션 형식으로 수행됨
(그림 1) 블록을 추가하기 위한 과정

1단계에서 시뮬레이션을 통해 나온 결과값은 수행된 파이어에 의해 보증하는 작업을 거치게 된다. 이 보증은 블록에 추가하기 전 여러 파이어로부터 보증된 값이 모두 같은지 확인할 때 사용된다. 이 보증값은 시뮬레이션 결과값에 추가한 후에 다시 클라이언트에게 보내게 된다.


3단계는 블록을 받은 파이어들이 블록의 유효성을 판단하는 과정을 거친다. 유효성 판단은 보증 파이어들 사이의 서명 값을 확인하고, 가지고 있는 원장과의 비교 값 확인한다. 확인이 완료되면 블록은 원장에 추가된다.

2.2 보증 단계에서 사용되는 구성요소

파이어는 브로커에서 원장의 관리를 하는 노드이다. 파이어들이 채널을 그리워하게 각각 개별로 원장 관리를 하며 기밀성을 보장할 수 있다. 파이어는 작업에 따라 구분할 수 있는 데, 1) 보증 파이어 - 트랜잭션에 대한 채인코드를 수행하고 보증을 진행하는 파이어이다. 보증에 대한 보증의 수와 역할을 나누고 하며 작업을 진행한다. 보증 파이어는 1단계에서 작업을 진행하며 본 논문에서 말한 파이어이다. 2) 커밋 파이어 - 순서화 노드에서 순서화가 완료된 블록을 받은 파이어이다. 블록을 공유하는 퀘직론의 파이어를 모두 커밋 파이어라고 한다. 3) 업데이트 피어 - 다른 조작과의 통신을 담당하는 블록이다. 4) 레이터 파이어 - 순서화 노드의 연결되어 채널 내의 레이터를 전달받는 파이어이다. 레이터 파이어가 대표로 블록을 받아 가시 프로토콜[6]을 이용해 채널 내에 다른 파이어에게 전달한다.

채인코드는 원장에 데이터를 기록하거나 임의로 채인코드를 생성되며 블록을 생성하는 블록이다. 채인코드는 네트워크 토큰에 맞는 분산 이더리움과 함께 개발된 다. 파이어에는 계발자가 개발하는 일반적인 채인코드와 시스템 채인코드로 나뉜다. 시스템 채인코드는 이더리움에 결합되는 일반적인 채인코드와는 달리 시스템 네트워크에서 수행되는 채인코드이다. 시스템 채인코드는 기존의 URI를 이용해 블록 번호, 트래픽 ID 등을 사용하여 데이터를 받아올 수 있다. 2) ESCC(Configuration System ChainCode)는 클라이언트의 트랜잭션 요청을 보낸 보증 파이어가 자신의 시그너처시뮬레이션 결과값에 적용해 보증된다. 본 논문에서 주로 다루는 1단계에서 사용되는 시스템 채인코드이다. 3) VSCC(Validation System ChainCode)는 3단계 단계에서 보증 정책에 맞춰 보증된 서명이 있는지 확인할 때 사용된다. 4) CSCC(Configuration System ChainCode)는 채널 설정할 때 사용되는 채인코드로 채널에 파이어를 참가시키는 등의 작업을 수행할 때 사용된다. 5) LCSM(Lifecycle System ChainCode)는 채인코드의 설치부터 인스턴스화 하는 과정에서 사용된다.

클라이언트는 트랜잭션을 발생시키는 주체이다. 보증 파이어로부터 보증이 포함된 결과 값을 받기 위해 보증 정책에 선정되어 있는 하나 이상의 보증 파이어들에게 트랜잭션 요청을 한다. 그리고 클라이언트는 보증파이어가 시뮬레이션값과 보증 서명을 확인하고, 보증 정책의 조건과 만족할 때까지 보증 파이어로부터 결과값을 오길 기다린다. 보증 정책에 맞게 모든 보증 파이어의 결과값과 보증 서명들이 모두 모이면 모든 순서화 노드들에게 보낸다.

(그림 2) 보증 단계 체결화

3. 보증 단계 체결화

우리는 트랜잭션을 블록에 추가하기 위한 3단계의 체결화 단계에 대한 체결화 방법을 제안한다. (그림 2)에서 기본적인 설정은 클라이언트가 트랜잭션 요청을 보낸 노드 확인을 우선 수행한 후, 노드 확인은 요청을 보낸 노드가 네트워크 내에 속해 있는지 보증 파이어 확인하는 것이다. 이 과정은 동일한 클라이언트가 반복적인 요청을 보증 파이어에게 하더라도 동일한 확인하는 과정이다.

본 논문에서는 클라이언트가 반복적인 트랜잭션을 발생시킬 때 수행하는 노드 확인 작업의 반복성을 줄이기 위해 채널 생성을 시작했다. 이 채널을 클라이언트와 보증 파이어 간 보증된 채널로써, 보증 파이어에서 채널을 통해 받으면, 노드 확인을 생략하고 바로 채인코드를 수행하도록 수정하였다. 이 방법론은 보증 파이어가 반복적인 노드 확인 과정을 피할 수 있어서 성능 향상을 이룰 수 있다.
5. 결론

본 논문은 하이퍼레지 페브릭의 수행단계 중 단계별로 보증 단계의 성능 향상을 제안한다. 우리는 수행단계를 코드 단위로 분석하여 수행효율이 높은 트랜잭션을 발행시킬 때 보증 피어에서 수행하는 반복적인 과정을 최소화하는 방법이다. 이는 단련되지만, 적응하기 위해서 의존성 있는 코드를 분석하고 수정하는 과정을 가졌다. 제안한 방법은 하이퍼레지 페브릭에서 성능향상을 위해 사용할 수 있을 것이다. 실험에서는 보증 단계에서의 성능향상을 보이지만, 전체적인 성능에도 영향을 줄 것이다.

참고문헌