# 하이퍼레저 패브릭의 실행과 순서화 단계에 대한 성능 분석I,II

권민수*, 명노영*, 유헌창*
*고려대학교 대학원 컴퓨터학과
e-mail:{minsuft19, mry1811, yuhc}@korea.ac.kr

# Performance analysis of execution and ordering phases of Hyperledger Fabric

Minsu Kwon*, Rohyoung Myung*, Heonchang Yu*
*Dept of Computer Science and Engineering, Korea University

요 약

4차 산업 혁명에 들어서며 블록체인 기술이 크게 주목받고 있다. 블록체인의 간단한 정의는 분산 환경에서 모든 노드들이 수정할 수 없는 원장을 관리 하는 것이다. 블록체인은 크게 public 블록체인과 private 블록체인으로 구분할 수 있다. 이중 private 블록체인에 대한 연구가 활발히 이뤄지고 있고, 실제 많은 기업이나 연구소에서 실제 환경이나 시스템에 적용 시키려는 노력을 하고 있다. 많은 블록체인 플랫폼 중 하이퍼레저는 가장 활발히 개발이 진행되고 있는 오픈 소스 프로젝트 중 하나이다. 하이퍼레저는 Linux Foundation에서 주관하는 프로젝트로 그 중 가장 널리 알려져 있는 플랫폼으로 패브릭이 있다. 패브릭은 private 블록체인 기술로 비즈니스 상황에서 조직간 블록체인 환경을 구성하는 것에 초점이 맞춰져 있는 플랫폼이다. 패브릭은 클라이언트가 트랜잭션을 제출한 후 원장에 저장될 때까지의 과정을 독립적으로 실행되는 3 단계로 구분해 놓고 있다. 3단계는 실행, 순서화, 검증 단계로 이루어져 있다. 본 논문에서는 실행과 순서화 단계에 대해 처리시간 분석을 진행한다. 분석한 데이터를 기반으로 어떤 단계의 영향이 가장 큰지 살펴보고 차후 연구에 이용할 것이다. 차후 연구는 실행과 순서화 단계에 대한 처리시간을 줄일 수 있는 연구를 진행할 것이다.

## 1. 서론

최근 4차 산업 혁명의 중심에 핵심 기술로 IoT, AI, 블록체인 등이 있다. 블록체인의 경우 중앙 집중형 시스템에서 탈 중앙 시스템의 성공 사례를 보여주는 비트코인의 성공으로 인해 관심이 증가하게 되었다. 이후 이더리움, 하이퍼레저 등 여러 오픈소스 프로젝트가 진행되고 있다. 그 중 하이퍼레저는 리눅스 재단의 주도 하에 산업간 블록체인 기술 발전을 위해 진행 중인 오픈소스 프로젝트이다. 하이퍼레저의 목적은 금융, 뱅킹, 사물인터넷, 공급망, 제조 및 기술 분야에 적용시키는 것이며 현재 8개 프로젝트가 진행 중이다.[1] 블록체인의 주요 이슈중 하나는 트랙잭션을 발생시키고 원장에 저장되는데 까지 걸리는 시간이다. 블록체인 시스템이 가용성, 일관성 등 여러 분산 데이터 관리에 필요한 요소들은 충족하지만, 금융과 같이 많은 트랜잭션이 발생하는 환경에 처리속도가 느려 적용하기 힘든 상황이다.[3] 이와 같은 문제점으로 인해 블록체인 플랫폼들에 대해 트랜잭션 처리속도를 증가시키기 위한 여러 연구가 진행중에 있다. 본 논문에서는 하이퍼레저 프로젝트 패브릭에 대해 원장 추가 과정의 특정 단계의 처리시간을 분석한다.

패브릭은 트랜잭션 발생에서부터 원장이 모든 노드에 저장될 때까지 독립적으로 실행되는 3단계를 거쳐 처리된다.[2] 패브릭으로 구성된 블록체인 네트워크 내에 클라이언트가 트랜잭션을 발생시키고 그 트랜잭션이 모든 원장에 저장될 때까지의 총 처리시간을 측정하고, 코드 레벨에서 주고받는 메시지를 분석하고 추적하여 실행 단계와 순서화 단계에 해당하는 처리시간을 세부적으로 분석을 진행한다.

## 2. 관련 연구

### 2.1 블록체인

블록체인은 다수의 트랜잭션이 모여 하나의 블록으로 만들어지고 이 블록이 해시 함수를 통해 암호화 된 후 체인처럼 순차적으로 이어졌다고 하여 블록체인이라고 말한다. 블록체인은 분산된 환경에서 사용되며 탈 중앙, 안정성, 보안성, 가용성과 일관성 등 분산 환경에서 중요시하는 부분을 만족하는 특징을 가지고 있다. 블록체인의 종류로는 크게 public과 private 블록체인으로 구분할 수 있다.[3] 블록체

I "본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT 연구센터지원사업의 연구결과로 수행되었음" (IITP-2018-0-01405)
II 이 논문은 2019년도 정보(과학기술정보통신부)의 재원으로 정보 통신기획평가원의 지원을 받아 수행된 연구임 (No.2018-0-00480)

인은 수정 불가능한 분산 원장을 안전하고 투명하게 관리하는 것이 핵심인 기술이다.

public 블록체인은 비트코인과 같이 누구나 참여 가능한 분산 네트워크 환경이다. public 블록체인은 보통 가상화폐를 가지고 네트워크가 유지되며 화폐를 얻기 위해서 PoW(proof of work) 작업 증명과 같은 합의 알고리즘을 통해 합의에 다다랐을 때 화폐를 얻고 트랜잭션을 추가시키게 되는 작업을 하게 된다. public 블록체인은 데이터가 투명하게 관리되며 모든 유저가 접근가능하다.

private 블록체인은 public과 다르게 신원이 확인된 노드만 참여할 수 있는 폐쇄형 블록체인이다. private 블록체인 네트워크에 참여하기 위해서는 네트워크의 관리자 승인이 필요하다. private 블록체인을 사용할 때 굳이 public 블록체인에서 사용하는 암호 화폐 개념이 필요하지 않다. 신원이 확인된 인원들끼리의 합의를 통해 저장되기 때문에 작업 증명과 같은 합의 알고리즘은 사용하지 않아도 된다. private 블록체인의 경우 스마트 컨트랙트[3] 개념을 사용하여 특정 클라이언트가 발생시키는 트랜잭션에 대해 자동화 처리를 할 수 있다.

## 2.2 분산 원장

분산 원장은 블록체인 네트워크에 속해 있는 모든 노드들이 가지고 있고 모든 원장이 일관성을 가지고 유지된다. 해시 체인으로 이어져 있는 블록들은 링크드 리스트 형태를 갖는다. 새로운 블록이 추가될 때 원장의 끝에 새로운 블록이 덧붙여지며 블록이 추가된다. 블록 전체에 대한 데이터를 해시 처리하여 블록의 데이터 하나라도 변하게 된다면 일관성이 깨지는 것을 확인할 수 있다. 그렇게 수정이 불가능한 원장을 각 노드들이 관리하게 된다.

패브릭에서 원장들을 관리할 때 가장 최신의 블록은 원장에 업데이트시키지 않고 데이터베이스 형태로 따로 관리한다. 그 최신 블록을 이용해 실행단계에서 시뮬레이션을 수행할 때 사용하고, 보증 단계에서 모든 피어가 추가할 새로운 블록에 대해 보증 과정을 거칠 때 사용된다.[5]

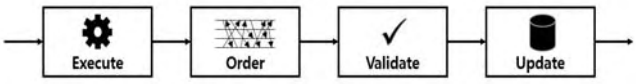
## 2.3 하이퍼레저 패브릭

하이퍼레저 패브릭은 Linux Foundation에서 진행하는 비즈니스 모델 블록체인 오픈 소스 프로젝트이다. 하이퍼레저 패브릭은 가장 왕성하게 활동 중인 하이퍼레저 프로젝트로 초기에 IBM이 제공한 44,000여 줄의 코드를 바탕으로 현재 전 세계 개발자들이 참여 중이다.[1] 합의 및 회원 서비스와 같은 플러그 앤 플레이 구성 요소를 허용하고, 컨테이너가 시스템의 응용프로그램 논리를 구성하는 체인코드라는 스마트 계약을 호스트 할 수 있게 설계되어있다. 패브릭에서는 원장을 관리하는 노드를 피어라고 부르며 트랜잭션을 시간에 따라 정리하는 순서화 노드를 따로 만든다. 패브릭은 트랜잭션이 원장에 업데이트가 완료될 때까지 3단계를 거치게 된다.[2] 3단계는 각각 독립적으로 수행되는 과정이다.

실행 단계에서는 클라이언트에서 발생된 트랜잭션에 대해 endorsing peer라고 부르는 보증 받을 피어를 미리 정하고 그 피어에게 트랜잭션을 보낸다. 트랜잭션을 받은 보증 피어들은 트랜잭션을 독립적으로 시뮬레이션하고 결과 값을 다시 클라이언트에게 리턴 한다.

순서화 단계는 클라이언트가 미리 정해놓은 정책에 맞춰 보증피어들에게 시뮬레이션 결과 값을 받게 되면 순서화 노드에게 보내게 된다. 순서화 노드는 시뮬레이션 값들을 순서에 맞춰 순서화한 후 블록으로 패키징 하는 작업을 한다. 만들어진 블록은 모든 피어에게 전달된다.

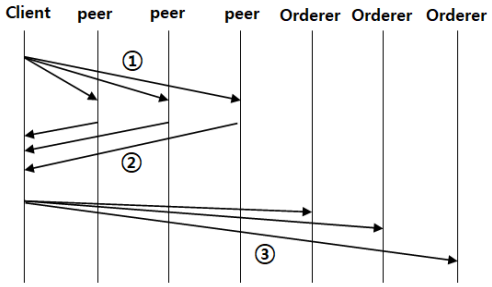
보증 단계는 순서화 노드에게 받은 블록을 트랜잭션 단위로 자신이 가지고 있는 원장에서 실행을 시켜보고, 다른 피어들과 확인을 하며 보증을 완료하고 블록을 추가시키게 된다.




(그림 1) 패브릭 트랜잭션 업데이트 단계

## 3. 분석 모델

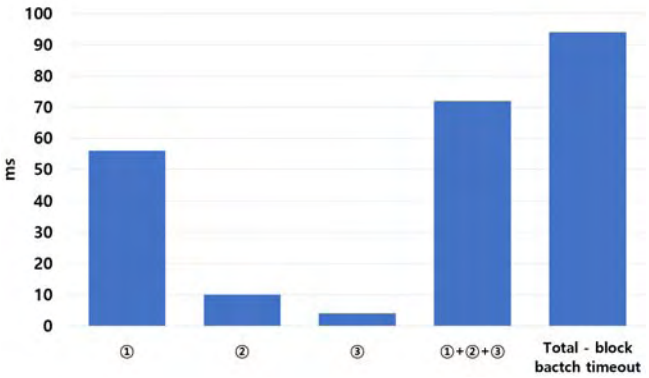
본 논문에서는 패브릭이 동작하는 독립적인 3단계 중 실행 단계와 순서화 단계 일부에 대해서 성능 분석을 진행한다. 실행 단계와 순서화 단계에서 원장에 업데이트가 되는 시간을 전체 시간으로 보고, 그 중 실행 단계와 순서화 단계가 차지하는 처리시간을 분석한다. (그림 2)에서 표현한 ①~③의 처리시간을 측정한다.[4] ①과 ②의 과정은 실행 단계로 클라이언트가 보증 피어에게 보내고 시뮬레이션 결과 값들을 돌려받는 과정이다. ③ 과정은 시뮬레이션 결과 값을 순서화 노드들에게 브로드캐스트 하는 과정을 나타내고 있다. 순서화 노드들이 시뮬레이션 결과 값을 받은 후 순서화 단계를 계속 수행하지만 ③ 과정 이후의 단계는 세세하게 처리시간을 측정하지 않는다. 실험에서 클라이언트가 발생시키는 트랜잭션으로는 각 피어들이 가지고 있는 원장의 상태를 수정시킬 수 있는 트랜잭션을 발생시켜 실험을 진행한다. 3가지를 모두 측정한 후 클라이언트가 트랜잭션을 발생시킨 후부터 모든 피어에 원장이 업데이트될 때까지의 시간을 측정하여 측정한 3가지의 처리시간이 전체 대비 어느 정도의 영향이 있는지를 분석한다.




(그림 2) 주요 처리시간 측정

## 4. 실험

모든 컴포넌트들은 VMware 위의 가상머신으로 구성된다. 호스트 PC의 스펙은 프로세서 Intel Core i5-7500 CPU 3.40GHz를 사용하고 RAM은 32GB이다. 각 네트워크 구성은 peer 4, client 1, kafka-zookeeper 1, orderer 1로 구성된다. 4가지 부분의 처리시간을 측정하게 되는데 첫 번째, 클라이언트에서 트랜잭션을 발생시킨 후 endorsing peer의 핸들러에서 받은 후 시뮬레이션 까지 마친 후 클라이언트에게 다시 보내기 전까지 시간을 측정한다. 두 번째, endorsing peer가 시뮬레이션 결과 값을 클라이언트에게 보내고 클라이언트가 만족하는 수의 peer로부터 받아 순서화 노드에게 보내기 전까지 시간을 측정한다. 세 번째, 클라이언트가 순서화 노드에게 데이터를 보낸 메시지가 확인되는 시간을 측정한다. 네 번째, 클라이언트가 트랜잭션을 발생시키고 모든 노드에 업데이트가 완료되는 시간을 측정한다. 순서화 노드가 블록 패키징을 위해 트랜잭션을 기다리는 시간 설정은 1s로 설정하고 진행한다. 발생시키는 트랜잭션은 invoke로 “A가 B에게 10원을 준다”라는 업데이트 트랜잭션을 발생시켜 실험을 진행한다.




(그림 3) 주요 처리시간 실험 결과

(그림 3)에서는 클라이언트에서 하나의 트랜잭션을 발생시키고 각 단계의 처리시간을 보여주는 그림이다. ①은 한 명의 클라이언트가 트랜잭션을 발생시키고 endorsing peer에게 트랜잭션을 보내는 작업이다. ①~③의 작업 중 약 54ms로 시뮬레이션 실행을 포함하여 가장 긴 처리시간을 가지고 있다. ②는 endorsing peer에서 트랜잭션에 대한 시뮬레이션을 한 결과 값을 endorsing peer의 서명과 함께 보내주는 작업이다. 클라이언트는 만족하는 수의 endorsing peer에게 값을 받을 때 까지 대기시간을 가진다. 대기시간과 처리시간을 포함하여 약 10ms 걸렸다. ③은 클라이언트에서 순서화 노드들로 브로드캐스트를 하는 과정인데 실험에서는 클라이언트 하나에서 하나의 순서화 노드로 보내는 실험으로 약 5ms가 나왔다. 이 과정은 단순 통신 시간만 나타난 것으로 보여 지고, 향후에 여러 클라이언트로부터 결과 값을 받는 순서화 노드에 대해 분석할 것이다.

클라이언트가 트랜잭션을 발생시킨 이후 원장에 블록이 추가될 때 까지 전체 처리시간을 살펴보면 약 94ms로 나타나는데, 이중 ①~③의 합이 약 70ms로 큰 비중을 차지하는 것으로 볼 수 있다. (전체 처리시간을 계산할 때 블록 패키징을 위해 트랜잭션을 대기하는 시간은 제외) 향후 연구는 분석한 비중이 높은 과정의 처리시간을 단축시키는 연구를 진행하여 높은 효율을 얻을 것이다.

## 5. 결론

하나의 트랜잭션 발생에 대해 전체 처리시간과 각 단계의 처리시간 비중에 대해 분석을 진행하였다. 향후 연구로는 분석한 단계의 최적화를 진행하며 처리시간을 줄이는 연구를 진행할 것이다. 첫 번째, 실행단계에서 endorsing peer가 핸들러를 통해 트랜잭션을 받고 그 안에서 바로 시뮬레이션을 수행하는 형태로 코드가 진행되는데, 그 부분을 스레드 풀을 만들어 처리하여 여러 클라이언트의 요청을 효율적으로 처리 할 것이다. 두 번째, 본 논문에서는 실험이 순서화 단계에서 하나의 클라이언트가 하나의 순서화노드에게 보내는 작업으로 나타났는데, 실제 시스템에서는 다수의 클라이언트와 다수의 순서화노드들이 구성되므로 그 부분에 대한 처리시간도 증가할 것이다. 패브릭에서는 클라이언트가 모든 순서화노드에게 브로드캐스트 하도록 되어 있는데, 클라이언트가 여러 순서화노드 중 일부에게만 보내는 멀티캐스트 하도록 처리하며 연구를 진행할 것이다.

## 참고문헌

[1] https://www.hyperledger.org/

[2] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukoli´c, S. Weed Cocco, and J. Yellick. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. ArXiv e-prints, Jan. 2018.

[3] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, “Untangling blockchain: A data processing view of blockchain systems,” IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 7, pp. 1366 - 1385, 2018.

[4] Hyperledger Blockchain Performance Metrics White Paper, v1.0 ed. The Linux Foundation, Oct. 2018. [Online]. Available: https://www. hyperledger.org/resources/publications/blockchain-performance-metrics

[5] https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf