JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 26 No. 12, pp. 143-150, December 2021
https://doi.org/10.9708/jksci.2021.26.12.143

Design and Implement of Power-Data Processing System with Optimal Sharding Method in Ethereum Blockchain Environments

Taeyoung Lee*, Jaehyung Park**

*Student, Dept. of Electronics and Computer Engineering, Chonnam National University, Gwangju, Korea
**Professor, Dept. of ICT Convergence System Engineering and Dept. of Electronics and Computer Engineering, Chonnam National University, Gwangju, Korea

[Abstract]

In the recent power industry, a change is taking place from manual meter reading to remote meter reading using AMI(Advanced Metering Infrastructure). If such the power data generated from the AMI is recorded on the blockchain, integrity is guaranteed by preventing forgery and tampering. As data sharing becomes transparent, new business can be created. However, Ethereum blockchain is not suitable for processing large amounts of transactions due to the limitation of processing speed. As a solution to overcome such the limitation, various On/Off-Chain methods are being investigated. In this paper, we propose a interface server using data sharding as a solution for storing large amounts of power data in Etherium blockchain environments. Experimental results show that our power-data processing system with sharding method lessen the data omission rate to 0% that occurs when the transactions are transmitted to Ethereum and enhance the processing speed approximately 9 times.

▸ **Key words**: Blockchain, Smart Contract, Ethereum, Sharding, AMI, Off-Chain

[요 약]

최근 전력산업에서는 검침원에 의한 수기 검침에서 AMI(Advanced Metering Infrastructure)를 활용한 원격검침으로 변화가 일어나고 있다. 원격검침 인프라에서 발생하는 전력 데이터가 블록체인에 기록된다면 위변조 방지로 무결성이 보장되고, 데이터 공유가 투명해짐에 따라 새로운 비즈니스가 창출될 수 있다. 하지만 기존의 이더리움 블록체인은 처리속도의 한계로 인해 대량의 트랜잭션 처리에 적합하지 않다. 이에 대한 해결책으로 다양한 On/Off-Chain 솔루션들이 연구되고 있다. 본 논문에서는 대량의 전력 데이터를 블록체인에 저장하기 위한 해결책으로 데이터 샤드(Shard) 처리를 활용한 연계 인터페이스 서버를 제안한다. 데이터를 샤드 단위로 처리하는 기법을 적용하여 대량의 트랜잭션을 이더리움에 전송했을 때 데이터 누락률이 0%가 되고, 동시에 처리속도가 대략 9배 향상되는 것을 실험을 통해 검증하였다.

▸ **주제어**: 블록체인, 스마트 컨트랙트, 이더리움, 샤딩, 지능형 전력계량 인프라, 오프체인

• First Author: Taeyoung Lee, Corresponding Author: Jaehyung Park
*Taeyoung Lee (leety1930@kakao.com), Dept. of Electronics and Computer Engineering, Chonnam National University
**Jaehyung Park (hyeoung@jnu.ac.kr), Dept. of ICT Convergence System Engineering and Dept. of Electronics and Computer Engineering, Chonnam National University
• Received: 2021. 10. 21, Revised: 2021. 12. 13, Accepted: 2021. 12. 13.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

블록체인은 2008년 사토시 나카모토의 논문에서 '비트코인'이라는 가상화폐의 기반 기술로써 소개되었다[1]. 해당 논문에는 '블록체인'이라는 용어가 직접 사용되지는 않았지만 블록(Block)이라는 단위에 다수의 거래내역(Transactions)이 저장되고, 각 블록은 이전 블록헤더의 해시값을 포함하여 자신의 블록 해시를 생성하는 구조를 보며 후대에 '블록체인'이라 불리게 되었다. 그 이후 2015년에 등장한 '이더리움'은 단순히 가상화폐로서의 기능을 가진 비트코인을 넘어 스마트 컨트랙트라는 개념을 도입하여 글로벌 네트워크에서 어플리케이션을 탑재할 수 있는 플랫폼으로 도약했다[2]. 스마트 컨트랙트는 프로그래밍 코드로 쓰여진 계약서이며, 계약서 내 코딩된 조건이 충족되면 자동 이행되는 개념을 말한다[3]. 이러한 스마트 컨트랙트를 기반으로 한 이더리움은 블록체인 네트워크에 프로그램을 배포하고, 사용할 수 있는 장점 덕분에 다양한 비즈니스 모델이 탄생하였다. 블록체인 상에서 동작하는 프로그램을 가리켜 DApp(Decentralized Application)이라고 하며 현재 이더리움에만 2,778개의 DApp이 배포되어 있다[4]. 이렇게 많은 DApp은 다양한 비즈니스를 가능하게 해준다. 게임, 금융, 소셜미디어, 보안, 에너지 등의 분야로 구성되어 있으며 해마다 그 수는 늘어가고 있다. 여러 비즈니스 중 에너지 분야에서 블록체인을 활용하기 위해서는 대량의 전력 데이터를 처리할 수 있는 성능이 중요하다. 전력 데이터는 전국에 설치된 AMI(Advanced Metering Infrastructure)에서 수집되는 데이터를 의미하고, 한국전력은 2024년까지 2,320만 호 보급을 목표로 하고 있다[5].

하지만 이더리움의 TPS(Transaction Per Second)는 4~38정도로 측정된다[6]. 이는 초당 4~38개의 거래를 처리할 수 있다는 의미이며 점점 늘어가는 전력 데이터를 처리하기에는 턱없이 부족하다. 따라서 최근에는 이더리움의 확장성 문제를 해결하기 위해 블록체인 내부에서 해결하는 On-Chain 솔루션과 외부에서 해결하는 Off-Chain 솔루션 등 다양한 연구와 솔루션들이 제안되고 있다.

이러한 이더리움의 한계를 해결하기 위해 본 연구에서는 다양한 비즈니스 중 에너지 분야에서 대량의 전력 데이터 처리에 적합한 샤딩 기반의 이더리움 연계 인터페이스를 제안한다.

본 논문의 구성은 다음과 같다. 2장은 이더리움의 낮은 성능과 개선을 위한 관련 선행연구에 대해 알아보고, 3장에서는 본 연구에서 제안하는 전체적인 시스템 구조와 설계에 대해 설명한다. 그리고 4장에서는 제안 시스템의 성능 평가를 위한 실험을 설계하고, 5장에서 이에 대한 결과를 분석한다. 마지막으로 6장에서는 결론과 향후 연구 방향에 대해 서술하고 마무리한다.

II. Preliminaries

1. Related works

1.1 Ethereum

이더리움을 운영하기 위해 Geth를 설치할 때 'POW', 'POA' 합의 방법을 선택할 수 있다. 먼저 POW(Proof of Work)는 블록 생성을 하고자 하는 노드들이 특정 해시(Hash)값을 찾는 연산을 수행하여 특정한 난이도의 작업을 수행했음을 증명하는 것이다[7]. 또한, POW합의 방식을 사용하는 이더리움의 경우에는 초당 약 6~8개의 트랜잭션을 처리할 수 있는 성능을 가지고 있다[8]. 반면 POA(Proof of Authority)는 사전 승인된 유효성 검증자에 의해 블록이 생성된다. 블록생성을 위한 연산량이 POW에 비해 월등히 낮고, 설정된 시간에 정확하게 블록을 생성한다. 따라서 이더리움을 폐쇄형 블록체인으로 구성할 때 주로 사용된다.

이더리움에서 실제 트랜잭션을 처리하는 것은 EVM(Ethereum Virtual Machine)에서 실행된다. EVM은 튜링완전한 가상머신이며, 스택 기반의 아키텍처로 되어있다[9]. EVM구조는 256비트 아이템이 1024개로 이루어진 스택, 함수 호출 및 메모리 연산을 수행할 때 사용하는 휘발성인 메모리 영역(Memory), 영구 저장을 위한 비휘발성인 스토리지 영역(Storage)을 가지고 있다[9].

이더리움 2.0(Serenity)은 이더리움 네트워크의 확장성을 증진시키기 위해 오래전부터 준비되어 온 프로젝트다. PoS 합의로의 대체, 샤딩(Sharding), 비콘체인, 영지식증명 등의 기능을 여러 단계로 구현함으로써 보안 및 탈중앙화 특성을 해치지 않으며, 속도, 효율성, 확장성을 개선하는 것이 목표다[10].

1.2 On-Chain & Off-Chain Work

이더리움의 낮은 성능을 극복하기 위해서 다양한 On/Off-Chain이 연구되고 있다. 먼저 블록체인 내부에서 처리하는 On-Chain에서 대표하는 샤딩은 이더리움 네트워크를 샤드(Shard)라는 여러 개의 조각으로 나눈 것이다. 샤드는 자신이 관리하는 계정과 연관된 트랜잭션들을 처리하며 이더리움 네트워크에서 발생한 트랜잭션을 각 샤드가 나누어서 병렬적으로 처리함으로써 이더리움의 트랜잭션 처리량을 상승시킨다[11].

다음으로 블록체인의 외부에서 성능향상을 도모하는 Off-Chain 연구 중 Raiden Network는 두 사용자가 블록체인에 미리 예금해둔 금액 내에서 블록체인 외부에서 거래를 할 수 있도록 해 주는 시스템이다[12].

또 다른 솔루션인 플라즈마는 이더리움에 플라즈마 컨트랙트를 배포하고 Child chain의 채널을 생성하여 이더리움의 확장성을 향상시킨다[13]. 마지막으로 Truebit은 이더리움 메인체인에서 처리하기 복잡하고 높은 수수료가 발생하는 컨트랙트 연산을 블록체인 외부에서 처리하는 솔루션이다[14].

III. The Proposed System Design

1. Overall Proposed System Structure

본 연구에서 제안하는 전체적인 시스템 구조는 그림 1과 같다. 시스템의 전력 데이터 저장소인 데이터베이스, 최적 데이터 처리를 위한 샤딩 기능과 큐, 전력 데이터 처리를 위한 스마트 컨트랙트로 구성되어 있다. 블록체인에 배포된 스마트 컨트랙트의 구성요소는 데이터를 저장할 변수들과 데이터 저장 및 조회용 함수로 되어있다.

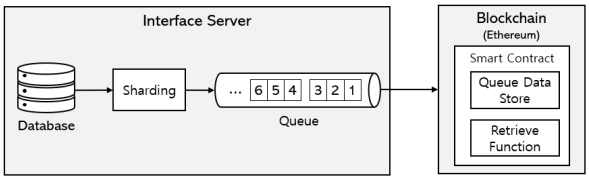

Fig. 1. System Architecture

2. Database

인터페이스 서버의 데이터베이스로 MongoDB를 사용했다. MongoDB는 'Key-Value' 형식으로 된 Document Database이다[15]. 표 1은 테스트용 전력 데이터의 구조를 나타내며, 본 연구에서는 여러 경우를 실험하기 위해 최대 10만 개까지 생성하였다. 여기서 Meter ID는 전력 계량기의 ID를 의미하며, 데이터 식별을 위한 Key로 지정했다. 이는 실험 후 블록체인에 데이터가 정상적으로 저장되었는지 검증할 때 사용된다. DCU(Data Concentration Unit)는 전력 계량기의 데이터를 수집하는 장치이며 Active Power는 유효 전력값을 의미한다. 그리고 모든 실험 데이터의 값은 +1씩 증가하게 구성하였다.

Table 1. Power Data for Simulation Test

Data	Meaning
0+N	Meter ID
0+N	DCU ID
0+N	Active Power(kW)
ddmmyy 00:00:00	Registration Date Time

3. Interface Server

본 연구에서 제안하는 인터페이스 서버는 그림 2과 같이 여러 모듈로 구성되어 있다. 먼저 Node.js의 Express 프레임워크를 적용하여 REST API를 구축하였다. Express의 라우터를 통해 블록체인 저장 및 누락 데이터 검증의 3가지 요청을 각각의 함수로 라우팅된다. 샤딩 모듈은 전체 데이터를 설정된 크기의 샤드로 구성하고 이를 큐로 이동시킨다. 수신한 URI에 따라 샤드로 구성되어 있는 데이터를 처리하는 큐는 NPM(Node Package Manager)에서 제공하는 패키지 중 Queue-fifo v0.2.6을 사용하였다. 이더리움 블록체인과의 연계는 Web3 v1.3.4를 통해 수행하게 되며 Promise객체를 통해 비동기 방식으로 처리한다. 비동기 방식을 사용하는 이유는 이더리움의 트랜잭션 처리 방법 때문이다. 블록 단위로 다수의 트랜잭션을 처리하는 구조이므로 블록이 생성되기 전까지는 그 트랜잭션이 정상적으로 처리되었다고 볼 수는 없다. 동기 방식을 사용하게 되면 전송된 트랜잭션이 블록에 저장될 때까지 서버는 대기 상태가 되기 때문에 매우 비효율적이다. 또한, 블록체인의 응답을 기다리면서 타임아웃이 발생할 수도 있다. 하지만 비동기 방식에서 서버는 이더리움의 트랜잭션 풀에 계속해서 트랜잭션들을 전송하면 된다.

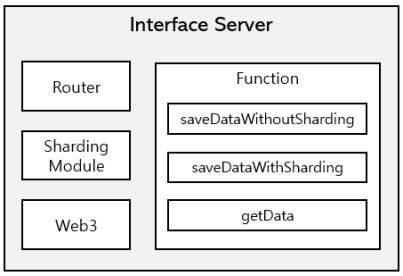

Fig. 2. Interface Server Structure

4. Smart Contract

이더리움 블록체인에 배포된 스마트 컨트랙트는 그림 3과 같다. Data 필드는 'activePower', 'dateTime', 'dcuId', 'meterId' 로 되어 있으며, 모두 String Type으로 구성했다.

함수 필드에서 ‘getData(string, string)’은 매개변수와 일치하는 전력 데이터를 반환하는 함수이다. 여기서 매개변수는 전력 데이터를 식별할 수 있는 계량기 ID와 DCU ID를 의미한다. 이 함수는 블록체인에 데이터 저장이 완료된 후 누락 데이터 확인을 위해 사용된다. 그리고 ‘saveData()’, ‘saveDataArray()’는 모두 매개변수의 전력 데이터를 블록체인에 저장하는 역할을 수행하지만, 차이점은 ‘saveData()’는 매개변수로 들어오는 데이터 1건 마다 트랜잭션을 발생시켜 처리하고, ‘saveDataArray()’는 샤드 형태로 된 데이터를 하나의 트랜잭션으로 처리하게 된다.

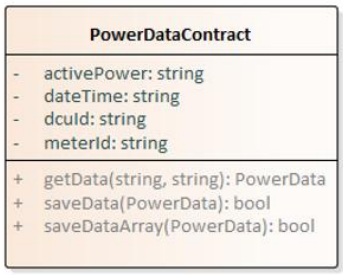

Fig. 3. Smart Contract Structure

5. Sequence Diagram

그림 4는 제안하는 방법을 실험하기 위한 전체적인 시스템간 동작에 관해 나타내고 있다. 실험은 Request를 생성해주는 소프트웨어인 Postman에서 실험에 대한 URI을 전송함으로써 시작된다. 수신한 URI에 따라 인터페이스 서버는 DB로부터 전력 데이터를 가져오게 되고, 설정된 샤드 크기(Shard Size)에 맞게 큐에 담는 Enqueue 과정을 수행한다. 샤드 크기는 그림 5와 같이 큐 내부의 저장 단위를 의미한다. 즉 샤드 크기가 3인 경우는 Enqueue나 Dequeue과정에서 한 번에 데이터를 3개씩 처리한다. Enqueue과정이 끝나면 반복적인 Dequeue 과정을 통해 큐에 저장된 데이터를 인출해서 이더리움 블록체인에 배포된 컨트랙트의 ‘saveDataArray()’함수로 전달한다.

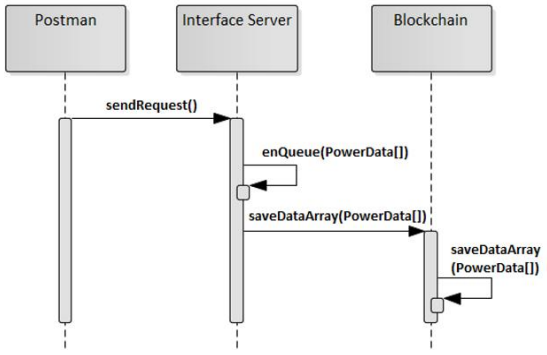

Fig. 4. Sequence Diagram

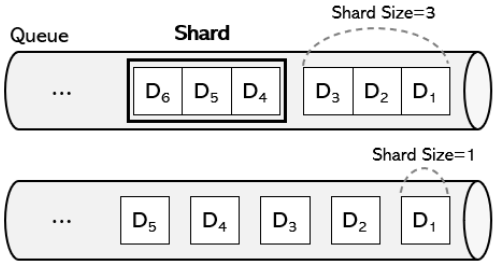

Fig. 5. Shard Size Example

6. Function Mapping

시작 Request가 블록체인의 컨트랙트까지 도달하는 과정에 대해 그림 6에 도식화하였다. Postman에서 ‘/insert’ URI을 호출하게 되면 인터페이스 서버에 ‘saveDataWithoutSharding’함수를 호출하게 되고, 이는 샤드 처리를 거치지 않고 바로 컨트랙트로 이어지게 된다. 즉 이 프로세스는 데이터를 어떠한 처리 없이 블록체인으로 전달하게 되는 기존의 방식(AS-IS)을 나타내고, 제안하는 방법(TO-BE)과의 성능 비교를 위해 추가한 기능이다. 반면 ‘/enqueue’ URI은 모든 데이터가 샤드로 구성되어 큐로 이동하게 되고 ‘saveDataWithSharding’함수를 통해 순차적으로 블록체인의 컨트랙트로 전송된다.

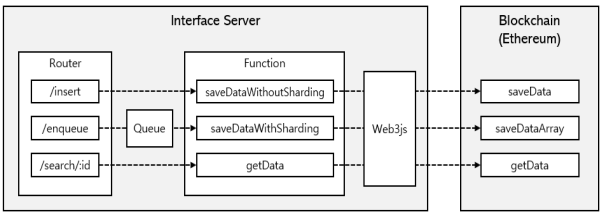

Fig. 6. Function Mapping

IV. Experimental Setup

1. Experiment environment

실험은 총 2대의 기기를 사용했고, 각 시스템에 따른 하드웨어 사양은 표 2와 같고, 네트워크 구성은 그림 7과 같이 구성된다. 모든 하드웨어 기기들은 성능측정 시 네트워크 상태에 영향을 받지 않기 위해 폐쇄망으로 구성하였다.

표 3은 실험을 구성하기 위해 사용한 소프트웨어와 그 버전을 나타내고 있으며 모든 시스템의 운영체제는 Windows 10이다. 이더리움은 Go-Ethereum을 설치했으며, 합의 방식은 POA로 선택했다. 블록 생성시간은 6초로 설정하였는데, 이는 POW합의에서 실험 시 채굴에 따른 하드웨어 사용량과 네트워크의 상태에 따라 실험결과에

영향을 미쳐 일관된 결과를 얻을 수가 없었다. 따라서 본 연구는 이더리움의 연계 인터페이스 서버의 구조에 따른 성능 평가를 객관적으로 하기 위해 일정한 시간으로 블록을 생성하는 POA합의를 선택했다. 컨트랙트는 Truffle프레임워크에서 개발하고 배포하였다. Truffle은 개발자가 스마트 컨트랙트 개발과 컴파일을 보다 쉽게 할 수 있는 환경을 제공해주고, 이더리움에 편리하게 배포할 수 있는 프레임워크다[16].

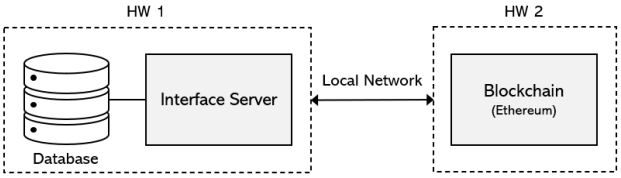

Fig. 7. Network Deployment

Table 2. H/W Specification

No	System	CPU	Memory
HW1	Interface Server, Database	Intel i7 9750 2.60GHz	16GB
HW2	Blockchain Node	AMD Ryzen 3 3200G 3.60GHz	9.95GB

Table 3. S/W Version

System	S/W
Database	MongoDB 4.4.5
Interface Server	Node.js 14.16.0, Express 4.17.1, Web3 1.3.4
Blockchain	Geth 1.10
Smart Contract	Solidity 0.7.6

2. Definition of Measuring Time

그림 8은 실험 시 측정한 시간에 대해 정의하고 있다. 시스템이 시작되면 시나리오의 설정대로 DB로부터 전력 데이터를 가져오게 되는 데 데이터 1만 건 기준으로 0.003초 이하가 소요되었다. 전체 실험은 초 단위로 소수점 둘째 자리까지 표현했기 때문에 실험결과에 포함하지 않았다. 그 후 인터페이스 서버는 데이터를 샤드 처리하는 과정을 거치며, 본 실험에서는 이 시간을 '샤딩시간'으로 정의했다. 그리고 인터페이스 서버에서 블록체인으로 데이터를 모두 전송하기까지 걸리는 시간을 '전송시간'으로 정의한다. '처리시간'은 이더리움에서 모든 트랜잭션을 블록에 저장하는 데까지 걸린 시간을 의미한다. 이더리움은 각 블록마다 생성된 시간인 'Timestamp'값이 존재하며 식(1)에서 m은 마지막 블록, n은 시작 블록을 의미한다. 최초에 트랜잭션은 시작 블록 이전에 이미 수신했기 때문에 n-1번째 블록에서부터 시간을 측정해야 한다. 마지막으로 '총 소요시간(TE_t)'은 식(2)처럼 '샤딩시간 (S_t)'과 '전송시간(T_t)', '처리시간(P_t)'의 합으로 이루어져 있다.

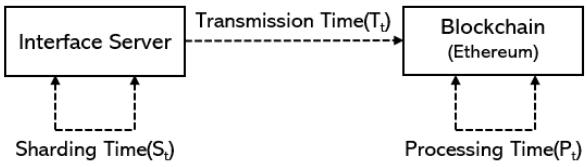

Fig. 8. Definition of Measurement

$$P_t = Timestamp(m) - Timestamp(n-1) \quad (1)$$

$$TE_t = S_t + T_t + P_t \quad (2)$$

3. Simulation Test Scenario

전체 실험은 표 4와 표 5에 나와 있는 것처럼 두 종류의 시나리오에 대해 진행하였으며 먼저 시나리오-I은 샤드처리 여부가 처리성능과 데이터 누락률에 미치는 영향을 확인하기 위한 시나리오이다. Case1은 데이터의 수를 2천 개로 설정하여 이더리움의 '트랜잭션 풀'보다 적은 경우는 데이터 누락이 발생하지 않음을 확인한다. 그 외의 Case는 모두 2만 개를 기준으로 했고, 샤딩이 적용된 경우는 샤드 크기를 100으로 설정하였다.

다음으로 시나리오-II는 시나리오-I에서 샤딩을 적용하면 성능이 향상되는 결과를 바탕으로 샤드 크기에 따른 성능변화를 측정하기 위함이다. 각 Case는 모두 샤딩이 적용된 경우이며 전력 데이터 10만 개에 대해 샤드 크기를 변경시켜 실험을 진행했다.

Table 4. Scenario-I

Case	Data	Shard Y/N	Shard Size
Case1	2×10^3	N	-
Case2	2×10^4	N	-
Case3	2×10^4	Y	100

Table 5. Scenario-II

Case	Data	Shard Size
S5	10^5	5
S10	10^5	10
S25	10^5	25
S50	10^5	50
S100	10^5	100
S150	10^5	150
S200	10^5	200
S250	10^5	250

V. Performance Evaluation

1. Scenario-I Result

표 6은 시나리오-I의 실험결과를 보여준다. 먼저 Case1의 결과는 데이터 2천 개가 모두 잘 저장되고, 처리속도도 빠른 것을 볼 수 있다. 하지만 Case2에서 보듯 데이터가 많아질수록 성능이 하락하고, 누락률이 급격하게 증가하는 것을 볼 수 있다. 이는 이더리움의 트랜잭션 풀과 관련이 있으며 풀의 디폴트 크기는 4,096이기 때문에 그 이상의 트랜잭션이 들어오면 이더리움은 수신할 수 없게 된다.

두 번째로 Case2와 3은 데이터 2만 개에 대해 샤드처리 여부에 따른 성능변화와 누락률을 확인할 수 있다. 이 경우 샤딩이 적용된 Case3은 Case2에 비해 전체 시간은 8.37배 향상되었고, 데이터 누락률은 0%로 확인되었다.

시나리오-I를 통해 알 수 있는 것은 샤딩을 적용한 경우가 그렇지 않은 경우에 비해 처리속도가 높고, 데이터 누락 현상이 전혀 발생하지 않은 것을 확인할 수 있었다.

Table 6. Result of Scenario-I

Case	S_t	T_t	P_t	TE_t	Loss
Case1	0	0.50	96	96.50	0
Case2	0	10.12	606	616.12	3794
Case3	0.20	1.49	72	73.62	0

2. Scenario-II Result

표 7와 그림 9은 샤드 크기 변화에 따른 실험결과를 나타낸다. 이번 실험에서는 샤드 크기가 증가할수록 '총 소요시간'이 줄어드는 것을 볼 수 있다. 샤드 크기를 증가하게 되면 한 블록 내에 저장되는 블록의 수도 줄어들었다. 왜냐하면, 샤드 크기를 가진 전력 데이터의 배열이 컨트랙트로 넘어오면 내부에서 반복문을 통해 각각 따로 저장할 변수에 담고, 트랜잭션으로 만든다. 따라서 샤드 크기가 커질수록 샤드를 처리하는 반복문의 횟수도 늘어나게 되고 그로 인해 트랜잭션의 생성시간은 길어지게 된다. 예를 들어 실험에서 샤드 크기가 50과 200인 경우에는 블록마다 조금의 차이는 있었지만 저장된 트랜잭션의 수가 각각 약 21개와 5개였다. 이러한 이유로 샤드 크기가 α배 커지더라도 내부적으로 데이터의 수만큼 반복문을 실행해야 하기 때문에 성능은 α배로 향상되지 않는 것이다.

다음으로 S5를 보면 샤딩을 적용하더라도 데이터 손실이 발생할 수 있다는 것이다. 너무 작은 샤드로 묶인 전력 데이터 배열은 그 수가 트랜잭션 풀의 크기를 월등히 넘는다. 따라서 이러한 경우도 시나리오-I과 같이 이더리움의 트랜잭션 풀에 도달하지 못하는 트랜잭션이 많이 발생한다.

그리고 샤드 크기를 293이상으로 설정하고 실험한 경우에는 오류가 발생하였다. 그 이유는 큐에서 인출된 배열이 컨트랙트의 매개변수로 넘어오는데 이때 메모리 타입 변수로 넘어온다. 이더리움 EVM의 메모리 영역은 가변적이며 제한적이기 때문에 한 번에 받을 수 있는 크기가 정해져 있어 샤드 크기를 계속 늘릴 수 없다.

표 8과 그림 10은 전체 실험에 대한 DPS(Data Per Second)를 나타내고 있다. DPS는 '총 데이터 수'를 '총 소요시간'으로 나눈 값으로 초당 데이터 처리에 대한 수치이다. 샤딩이 적용된 경우가 적용되지 않은 경우에 비해 DPS가 높았으며, 샤드 크기가 증가할수록 DPS도 같이 증가하는 것을 볼 수 있다.

Table 7. Result of Scenario-II

Case	S_t	T_t	P_t	TE_t	Loss	Tx
S5	0.93	25.95	1374	1400.88	18155	20000
S10	0.76	10.42	1302	1313.18	0	10000
S25	0.79	7.33	960	968.12	0	4000
S50	0.92	6.08	762	769.0	0	2000
S100	0.84	6.76	708	715.6	0	1000
S150	0.83	5.78	684	690.61	0	667
S200	0.77	6.93	654	661.7	0	500
S250	0.81	5.30	600	606.11	0	400

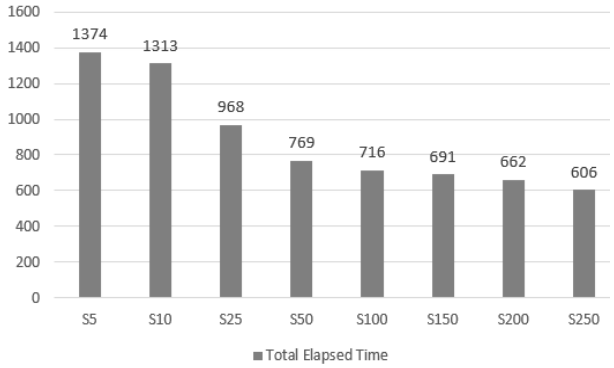

Fig. 9. Graph of Scenario-II Results

Table 8. DPS of Scenarios

Scenario	Case	Data	Shard Size	DPS
Scenario-I	Case1	2×10^3	-	20.72
	Case2	2×10^4	-	32.46
	Case3	2×10^4	100	271.68
Scenario-II	S5	10^5	5	72.78
	S10	10^5	10	76.15
	S25	10^5	25	103.29
	S50	10^5	50	130.03
	S100	10^5	100	139.74
	S150	10^5	150	144.80
	S200	10^5	200	151.12
	S250	10^5	250	164.98

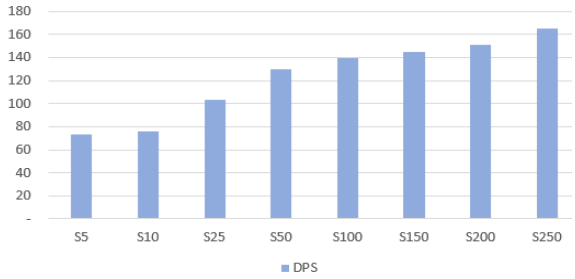

Fig. 10. DPS Graph of Scenario-II Results

3. Performance Analysis

시나리오-I을 통해 샤딩이 블록체인의 처리성능에 영향을 미친다는 것을 확인할 수 있었다. 또한, 데이터 손실 개수도 샤딩을 적용한 경우에는 발생하지 않는 결과도 도출되었다. 그 이유에 대해 먼저 그림 11은 샤딩이 적용된 경우에 블록체인에서 어떻게 처리되는지를 설명하고 있다. 데이터는 샤드 크기만큼 큐에 순차적으로 저장되고 블록체인으로 전송될 때 이 단위만큼 한 번에 전송된다. 컨트랙트의 매개변수로 데이터 배열이 전송되고, 컨트랙트 내부에서 배열을 순회하며 트랜잭션을 생성한다. 식(3)은 그림 11을 수식으로 표현한 결과이며 트랜잭션의 순서(n)에 따라 포함되는 데이터를 의미한다. 예를 들어 SD_S(Shard Size)가 100이면 TX_1에는 D_1~D_{100}, TX_2는 D_{101}~D_{200}이 포함되어 있음을 의미한다.

$$TX_n = \sum_{k=(n-1)SD_s+1}^{n \times SD_s} D_k \qquad (3)$$

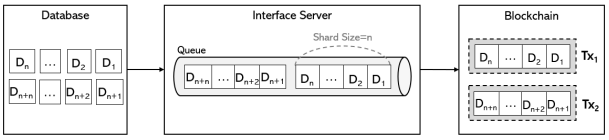

Fig. 11. Data Processing using Sharding

반면 샤딩을 사용하지 않고 데이터를 블록체인에 전송하게 되면 그림 12와 같이 각각의 데이터마다 트랜잭션이 생성되기 때문에 전체 데이터의 수와 전체 트랜잭션의 수는 같게 된다. 식(4)에서 전체 데이터의 수(D_T)는 전체 트랜잭션의 수(TX_T)와 손실된 데이터의 수(D_L)의 합으로 표현된다. 식(5)는 전체 트랜잭션의 수가 샤드 크기(SD_S)에 따라 생성되는 규칙을 나타낸다. 전체 데이터 수를 샤드 크기로 나누어 나머지가 0인 경우에 발생하는 총 트랜잭션의 수는 총 데이터 수를 샤드 크기로 나눈 값이 되고, 그렇지 않은 경우는 정수 몫에 +1한 값이 된다.

$$D_T = TX_T + D_L \text{(T:Total, L:Loss)} \qquad (4)$$

$$\text{if } D_T \% SD_s \begin{cases} TX_T = D_T / SD_s \\ TX_T = [D_T / SD_s] + 1 \end{cases} \qquad (5)$$

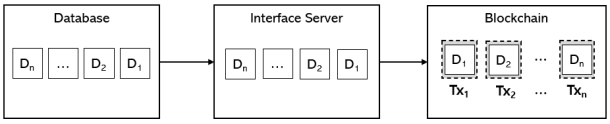

Fig. 12. Data Processing without using Sharding

데이터 누락 현상은 3.3장에서 언급한 '트랜잭션 풀'과 연관이 있다. 많은 트랜잭션으로 인해 트랜잭션 풀이 가득 차면 이더리움은 더 이상의 트랜잭션은 수신하지 않는 특성 때문에 데이터 누락 현상이 발생하게 된다. 또 다른 원인으로는 대량의 데이터를 어떠한 처리 없이 빠른 속도로 전송할 때 데이터가 소실되었고, 실험을 거듭하거나 일정 시간의 간격을 두고 실험한 경우에는 누락률이 계속 바뀌었다. 이는 곧 이더리움의 데이터 수신 시 하드웨어와 네트워크 환경이 처리속도에 영향을 미친다는 것을 의미한다.

시나리오-II에서는 샤드 크기의 증가량만큼 블록체인의 성능향상이 되지는 않았지만, 선형적으로 처리성능이 향상하는 것을 알 수 있었다. 시나리오-I에서 확인한 것처럼 블록체인은 샤드를 하나의 트랜잭션 단위로 처리한다. 따라서 샤드 크기가 증가할수록 하나의 트랜잭션에 포함되는 데이터가 많아져 전체적인 트랜잭션의 수는 줄어들게 된다. 하지만 5.2절의 실험에서 확인했듯이 EVM 메모리 영역의 한계 때문에 샤드 크기를 계속 증가할 수는 없다. 현재 이더리움의 트랜잭션 풀의 크기는 4096으로 소스코드에 설정되어 있다. 실험결과를 통해 이론적으로 데이터의 수에 따라 최적 샤드 크기를 도출할 수 있다. 식(6)은 총 데이터 수(D_T)를 트랜잭션 풀 크기(TXPS)로 나눈 몫이 최적 샤드 크기(OSD_S)가 됨을 의미한다. 예를 들어 본 실험환경의 경우에는 샤드 크기가 최대 292였고, 트랜잭션 풀 크기는 4,096이었다. 식을 적용해 보면 '292×4096 = 1,196,032'이 되고, 즉 누락 없이 처리 가능한 데이터의 수는 1,196,032개가 됨을 의미한다. 만약 그 이상의 데이터를 처리해야 한다면 하드웨어 성능과 네트워크 상태에 따라 차이가 있겠지만 누락 현상이 발생할 가능성이 크다.

$$OSD_S = D_T / TXPS \qquad (6)$$

VI. Conclusions

블록체인에 데이터를 저장하면 데이터 공유의 편리성과 DApp개발을 통한 새로운 비즈니스를 창출할 수 있다. 하지만 전력 데이터와 같이 대량의 데이터를 블록체인에 저장하기 위해서는 블록체인의 성능이 뒷받침되어야 한다. 본 연구에서는 샤딩이 미적용된 경우에 데이터의 누락 현상이 발생

하는 것을 실험을 통해 확인할 수 있었다. 그에 대한 해결책으로 블록체인에 직접 전송하는 방식이 아닌 전력 데이터를 샤드 단위로 묶어 전달하는 인터페이스 서버를 제안하였다. 실험결과를 통해 샤딩이 적용되면 데이터 누락 현상을 방지할 수 있고, 샤드 단위로 처리됨으로써 트랜잭션의 수가 줄어들어 처리속도 또한 월등하게 향상된 것을 확인할 수 있었다. 이때 샤드를 구성하는 시간은 전체 처리시간에 미미한 영향을 주었다. 더불어 샤드의 크기에 따라 성능도 향상된다는 결과와 더불어 최적 샤드 크기를 도출할 수 있었다.

2020년까지 우리나라에 구축된 AMI는 대략 1,000만 호가 넘으며 실험에 사용된 데이터보다 월등하게 많다. 이는 15분마다 생성되는 전력 데이터가 1,000만 건이 이상이고, 15분 내 모두 처리되어야 함을 의미한다. 5.3절에서 도출된 결론은 본 연구와 같은 환경에서 누락 없이 최대 1,196,032개의 전력 데이터를 처리할 수 있을 것으로 예상되었다. 따라서 향후 연구는 1,000만건 이상의 전력 데이터 처리를 위해 성능을 더욱 높일 수 있는 방법을 연구하고, 보다 많은 수의 전력 데이터를 실험하고 결과를 분석할 예정이다.

ACKNOWLEDGEMENT

This research was supported by the BK21 FOUR Program(Fostering Outstanding Universities for Research, 5199991714138) funded by the Ministry of Education(MOE, Korea) and National Research Foundation of Korea(NRF).

REFERENCES

[1] S. Nakamoto, "*Bitcoin: A Peer-to-peer Electronic Cash System,*" [Online]. https://bitcoin.org/bitcoin, 2008.

[2] V. Buterin, "*A Next-generation Smart Contract and Decentralized Application Platform,*" white paper, 2014.

[3] N. Szabo, "*Smart Contracts: Building Blocks for Digital Markets,*" EXTROPY: The Journal of Transhumanist Thought,(16), 1996.

[4] *https://www.stateofthedapps.com/platforms/ethereum.*

[5] *https://www.etnews.com/20201223000179.*

[6] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance Analysis of Private Blockchain Platforms in Varying Workloads," IEEE ICCCN, pp. 1-6, Vancouver, Canada, 2017 Jul. 10.1109/ICCCN.2017.8038517

[7] J.C. Yim, and et al, "*Blockchain and Consensus Algorithm,*" ETRI Electronics and Telecommunications Trends, 2018.

[8] H. Malik and et al, "Performance Analysis of Blockchain based Smart Grids with Ethereum and Hyperledger Implementations," IEEE ANTS, pp. 1-5, Goa, India, 2019 Dec. 10.1109/ANTS47819.2019.9118072

[9] G. Wood, "*Ethereum: A Secure Decentralised Generalised Transaction Ledger,*" Ethereum yellow paper, 2014.

[10] *https://ethereum.org/ko/eth2/.*

[11] S. Kim, J. Kim, S. Woo, and S. Park, "Account Relocation Algorithm for Load Balancing between Shards on Ethereum Sharding," Proceeding of KIISE, pp. 1331-1333, Korea, 2018 Dec.

[12] J. Lee, and C. Park, "Analysis and Evaluation of The Raiden Network in Ethereum Blockchain," Proceeding of KIISE, pp. 1484-1486, Korea, 2017 Dec.

[13] S.H. Kim, J.Y. Jeong, and I.R. Jeong. "Implement and Experiment of Efficient Off-Chain," Journal of the Korea Institute of Information Security & Cryptology, Vol. 29, No. 6, pp. 1413-1424, Dec, 2019. 10.13089/JKIISC.2019.29.6.1413

[14] C. Choi, Y. Lim, Y. Song, and J. Lee, "Analysis of Off-Chain Solutions for Ethereum Scalability Issues," Korea Institute Of Communication Sciences, pp. 208-209, Korea, 2019 Jan.

[15] *https://docs.mongodb.com/manual/introduction/.*

[16] *https://www.trufflesuite.com/docs/truffle/overview.*

Authors

Taeyoung Lee received the B.S. in Computer Engineering from Kumoh National Institute of Technology in 2014, M.S. degrees in Electronics and Computer Engineering from Chonnam National University, Gwanju, Korea, in 2018. Taeyoung Lee is currently a Ph.D student at Chonnam National University and a researcher in Kepco KDN. He is interested in Blockchain, Interface Server and Computer Architecture.

Jaehyung Park received his B.S. in Computer Science from Yonsei University, Korea, in 1991, and his M.S. and Ph.D, in Computer Science from Korea Advanced Institue of Science and Technology(KAIST), Korea, in 1993 and 1997, respectively. Since 2002, Dr. Park has been with the faculty of Chonnam National University, Korea, where he is currently a professor in Dept. of ICT Convergence System Engineering and Dept. of Electronics and Computer Engineering. He is interested in network security and cryptography, distributed and cloud computings, internet routings and protocols, and wireless and vehicular networks.