블록체인 분산원장 구현을 위한 Docker 컨테이너 활용 사례연구*

A Docker Container Case Study for Implementing
Block Chain Distributed General Ledger

이종기 (Lee, Jong - Ki) **

〈요 약〉

4차산업혁명시대에 블록체인기술이 각광 받고 있다. 블록체인은 분산원장처리기술로 원장의 세부 내역을 블록 단위로 네트워크상에서 분산으로 저장하고 검증하는 보안에 뛰어난 기술 중 하나이다.

블록체인은 분산 환경에서의 프로그램 실행과 배포가 중요하므로, 이에 부합하여, 오픈소스인 리눅스에 기반한 도커(Docker)를 이용하여, 블록체인 분산원장처리 네트워크를 구축한 사례를 살펴보았다. 블록체인을 처리하기 위한 도커 컨테이너(Container) 사용 확산 추세에 맞추어, 본 연구에서는 도커 컨테이너를 사용하고, IBM 등이 주도하는 프라이빗 블록체인 하이퍼레저패브릭(Hyperledgerfabric) 비즈니스 네트워크 구현을 실행하였다. 도커 컨테이너의 설치, 환경설정, 이미지 구축, 커밋, 그리고 웹브라우저를 통해 블록체인 하이퍼레저패브릭 컴포저(Composer)에 접속하여 블록체인 구축의 세부내용을 살펴보았다. 본 연구는 도커 컨테이너의 여러 기능과 처리절차를 숙지함으로써, 블록체인 하이퍼레저 패브릭 컴포저에 접속하여, 분산원장처리 참여자, 자산, 거래를 생성하는 방법을 탐색하고, 실제로 블록을 만들고 블록을 배포 처리하는데 유용한 시사점을 제공한다. 또한 블랙체인에 접속한 분산원장 정보처리를 통해, 회계정보 생성과 보관에 대한 중앙집중식 회계정보처리를, 분산시스템으로 관리하게 됨으로 회계정보 보안 문제를 근본적으로 해결할 수 있을 것으로 본다.

한글검색어 : 블록체인, 분산원장, 도커, 컨테이너, 하이퍼레저패브릭

• 접수일자 : 2018.5.9 • 1차수정일 : 2018.6.9 • 2차수정일 : 2018.6.20 • 게재확정일 : 2018.6.27

* 본 논문은 2018학년도 제주국제대학교 학술연구비 지원에 의하여 연구되었음

** 제주국제대학교 교양학부 교수, e-mail : mirae910@jeju.ac.kr

I. 서론

블록체인에 의한 분산형 원장 처리 기술은 4차 산업혁명 분야의 인공지능, 사물인터넷 등과 함께 인프라 기술로 그 활용 가능성이 매우 높은 기술이다.

구글 회장인 에릭 슈미트는 4차 산업혁명의 도래로 인해 가상 세계에서 더 많은 일들을 할 수 있게 됨으로서 현실세계의 메카니즘은 보다 효율적으로 변하며, 디지털 연결성이 전 세계 구석구석에 까지 영향을 미치면서, 새로운 기회와 더 많은 혁신과 기회를 제공함으로써 효율성과 가치를 극대화할 것으로 예측하고 있다(Schmidt, 2013). 또한 2016년 다보스포럼에서 슈밥이 “세계적으로 일어나는 많은 변화 현상들 중에서 사물인터넷, 인공지능, 가상현실 등과 같은 기술들이 기업의 생산과 소비 프로세스와 패러다임을 변화시키는 일련의 기술 융복합 현상”으로 발표하였다(Schwab, 2016). 또한 빅 데이터 시대의 선도를 위해 주목해야할 미래 인프라 및 소프트웨어 기술은 클라우드, 모바일, 소셜, 보안 및 분석을 꼽을 수 있으며(이종기, 2015), 클라우드 기술 사용을 위한 요소 기술 중 하나가 도커(Docker) 기술이라 할 수 있다.

이러한 융복합 기술의 핵심으로 부상한 블록체인 기술은 암호화 보안, 분산된 합의 및 적절하게 통제되고 권한을 가진 공유된 공공원장 덕분에 우리의 경제적 사회적 제 활동을 체계화하는 방식을 근본적으로 바꿀 수 있다.

본 연구에서는 비즈니스 기록 시스템인 원장(Ledger)을 살펴보고, 가상운영체제를 사용하여 분산원장 처리기술인 블록체인을 구현하는 것을 절차적으로 실행시켜 보고자 한다.

본 연구의 목적은 가상운영체제를 구현하는 도커 컨테이너 시스템 사용 사례를 통하여, 보편화에 근접한 클라우드 기반의 하이퍼레저패브릭 블록체인으로, 분산원장 처리를 위한 기술 기반을 제공하고자 한다.

II. 블록체인 분산원장과 도커 컨테이너 기술 선행연구

2.1 분산 원장(distributed ledger)

회계학의 거래는 부기의 분개를 거쳐 계정구좌에 기록되는 것이나 원장(Ledger, 元帳)

은 이 계정 구조를 모두 집성한 것으로 분개장이나 일기장 등에 대응하는 회계장부의 중심적인 것이다. 반면에 전체 비즈니스 관점에서 원장을 살펴보면, 통상적인 경제활동은 연결되고 통합된 세계에서 국가, 지리적 및 관할 경계를 넘어서는 비즈니스 네트워크에서 이루어진다. 비즈니스 네트워크는 생산자, 소비자, 공급업체, 파트너 및 기타 이해 관계자가 자산에 대하여 권리, 특권 및 권한을 소유, 통제 및 행사하는 시장에서 함께 사용된다. 여기서 자산은 유형, 무형 및 가상 자산 일 수 있다. 비즈니스 네트워크상 이루어지는 자산의 소유 및 이전은 거래(transaction)로 불려진다.

일반적으로 비즈니스는 여러 가지 원장을 사용하여 다양한 원장의 자산 소유 및 자산 이전을 추적하는데, 원장은 기업의 경제적 활동과 이해관계에 대한 데이터베이스(systems of record)라 할 수 있다.

Simon Stone(2017)은 블록체인에서 공유된 분산 원장들을 [그림 1]과 같이 소개하고 있다. 블록체인 분산 원장이 4차 산업혁명 시대에 많은 기업 현장에서 활용될 것으로 예상되고 있다.

[그림 1] 블록체인에서 공유된 분산 원장

분산 원장의 이해를 위해 엑셀의 ‘공유 통합 문서’ 기능을 보면, 공유 통합 문서는 여러 사람이 같은 엑셀 문서 파일을 편집할 수 있는 기능이다. 이는 여러 사람이 동일한 파일을 편집할 때, 한 명에게만 편집을 허용하고 다른 사람에게는 참조 권한만 부여하거나,

같은 부분을 편집하는 것이 아니라면, 여러 사람에게 동시에 편집할 수 있는 권한을 부여하는 등의 권한 관리를 해주는 것이 통합 문서 기능이다. 마찬가지로 DB 시스템에서의 권한 관리를 'DB서버'가 수행하고, 동시에 다수 사용자가 쓰기 요청이 오더라도 한 사람이 쓰기를 하고 있을 때 다른 사용자가 쓸 수 없도록 데이터를 잠근다. 분산 원장의 경우 동일한 원장 데이터가 여러 시스템에 배치돼 있다. 사용자로부터 쓰기 요청(블록 생성)이 있는 경우 그 상태를 모든 시스템에 공유해 각 시스템이 그 상태를 저장하도록 만들어 동기화한다. 이 경우 각 시스템 간에 존재하는 합의 규칙을 따르며, 합의된 내용은 모든 시스템이 따르게 된다. 참여하고 있는 모든 시스템이 각자 원장 데이터를 보유하고 항상 동기화를 하는 것이 분산 원장이며, 이를 위한 상태 공유와 합의 형성 방법 중 하나가 블록체인 기술이다. 각 시스템에서는 블록체인 안에서 일어나고 있는 것을 볼 수 없기 때문에 마치 하나의 원장에 각 시스템이 접근하고 있는 것처럼 생각할 수 있다(아카하네 요시 하루 등, 2016).

〈표 1〉 시기별 산업혁명의 특징 분류

구 분	시 기	주요 기반기술	특 징
1차산업혁명 -동력혁명	18세기 후반	증기기관, 기계식 생산설비 등	영국이 공업대국으로 부상
2차산업혁명 -자동화 혁명	19세기~20세기 초, 중반	전기동력, 대량생산 체계(컨베이어벨트) 등	미국이 세계 최고 제조대국으로 지위 구축
3차산업혁명 -디지털혁명	20세기 후반	전자기기, 정보통신, 자동화 생산 등	한국 제조업의 국제경쟁력 부상
4차산업혁명 -초연결혁명	2015년~	사물인터넷, 빅데이터, 인공지능, 블록체인	- 산업간 경계가 사라짐 - 기술간 융합 활성화

한편 이러한 분산 원장 개념의 발전과 함께 최근의 4차 산업혁명 특징을 살펴보는 것은 의미가 있는데, 바로 4차 산업혁명의 기반기술이 블록체인 기술로 볼 수 있기 때문이다. 〈표 1〉에서 보는 바와 같이 4차 산업혁명을 초 연결 혁명으로 구분되고, 4차산업혁명은 사물인터넷, 빅데이터, 인공지능 등의 기술을 활용한 기술 간 융합의 활성화가 가능한(김미정 등, 2017) 새로운 유형의 비즈니스 플랫폼 운영의 필요성이 대두되고 있다. 새 유형의 비즈니스 플랫폼의 핵심 아이템으로 블록체인 기술을 모두가 주목하고 있는 것이다.

2.2 블록체인

블록체인은 퍼블릭 또는 프라이빗 P2P 네트워크에서 일어나는 트랜잭션의 정보가 네트워크 참여자간 공유된 디지털 원장이다. 네트워크 참여자에 의해 합의되고 유효성이 확인된 모든 거래 블록은 체인의 시작에서부터 가장 최근 블록으로 연결됨으로, 블록체인이라는 이름으로 불리며, 참여 구성원은 자신과 관련이 있는 트랜잭션만 볼 수 있다. 트랜잭션을 위해 제3의 기관을 이용하는 대신, 블록체인 네트워크의 구성원은 합의 프로토콜을 사용하여 원장 내용의 동의과정을 거치며, 거래 무결성을 보장하기 위해 해시 알고리즘으로 암호화하거나 디지털 서명을 사용한다, 합의는 공유 원장이 임의로 변형되지 않은 무결한 사본임을 확인하여 조작의 위험을 낮춘다(이종기, 2017). 블록체인 시스템 엔진에는 모든 노드가 선택한 합의 모델을 사용하여 모든 노드에서 합의된 트랜잭션이 시간 순서대로 기록된다. 그 결과 블록체인에 저장된 트랜잭션의 기록은 블록체인 네트워크의 모든 노드에서 뒤집을 수 없는 합의된 거래만 저장하게 된다. 서버 기능을 분산화 하는 기술인 블록체인 기술을 비교하면 [그림 2]와 같다(이종기, 2017). IBM 블록체인이 주도하는 하이퍼레저(Hyperledger)는 합의알고리즘이 PBFT라는 특징이 있다.

[그림 2] 블록체인 기술비교

Comparison Matrix

	Hyperledger	Ethereum	Ripple	Bitcoin
Description	General purpose Blockchain	General purpose Blockchain	Payments Blockchain	Payments Blockchain
Governance	Linux Foundation	Ethereum Developers	Ripple Labs	Bitcoin Developers
Currency	None	Ether	XRP	BTC
Mining Reward	N/A	Yes	No	Yes
State	Key-value database	Account data	None	Transaction data
Consensus Network	Pluggable : PBFT	Mining	Ripple Protocol	Mining
Network	Private or Public	Public or Private	Public	Public
Privacy	Open to Private	Open	Open	Open
Smart Contracts	Multiple programming languages	'Solidity' programming language	None	Possible, but not obvious

2.3 운영체제 레벨 가상화

운영체제(OS) 레벨 가상화란 [그림 3]과 같이, 하나의 CPU에 하나의 OS만 수행되고, 가상화 계층(Virtualization Layer)이 호스트 OS 윗부분에 존재하여 가상화 환경을 제공하는 방법을 말한다. 단일 호스트 OS 위에 다중 애플리케이션 전용 컨테이너를 생성하여 애플리케이션 단위의 가상화 환경을 제공한다. 각각의 애플리케이션 컨테이너 안에는 가상 OS와 네트워크, 프로세서 등을 생성한다. 이 방법은 크게 두 가지 방식으로 분리하는데 하나는 컨테이너 방식이고 또 다른 하나는 하드웨어 에뮬레이터(Emulator)방식이다.

컨테이너 방식은 가상화 계층이 호스트 OS 내에 임베디드 형태로 구성되고, 각각의 애플리케이션별로 추가적인 컨테이너를 생성하는 방식이다. 각각의 컨테이너 안에 생성된 가상 OS와 호스트 OS에 있는 공통 커널을 공유하고 사용한다. 대표적인 제품이 오픈 소스인 OpenVZ, Docker 등이 있다(배유미 등, 2014).

또 다른 방법인 하드웨어 에뮬레이터방식은 애플리케이션 중심의 가상 OS를 생성하는 과정에서 소프트웨어적으로 하드웨어를 가상 에뮬레이팅 하는 방식이다. 대표적인 제품으로는 마이크로소프트사의 Virtual PC for Mac, Bochs 등이 있다.

[그림 3] 운영체제 레벨 가상화

2.4 도커

도커(Docker)는 2013년 3월 Docker Inc.(구 dotCloud)에서 출시한 오픈 소스로 컨테이너(Container) 생성·관리 시스템이다. 2010년을 지나며 컴퓨터 서버 시장은 급속히 클라우드 환경으로 옮겨왔다. 클라우드 환경에서는 1대가 되었던 1,000대가 되었던 클릭 몇 번 만으로 가상 서버를 만들어낼 수 있게 되었다. 생성된 가상 서버에 각종 소프트웨어를 설치하고 설정해야 하는데, 서버 개수가 많아짐으로 클라우드 환경에서의 각종 소프트웨어 설치와 배포가 큰 어려움으로 다가왔다. 이런 상황에서 변경 불가능 구조(Immutable Infrastructure)라는 패러다임이 등장하게 되었는데, 이는 호스트 OS와 서비스 운영 환경(서버 프로그램, 소스코드, 컴파일된 바이너리)을 분리하고, 한번 설정한 운영 환경은 변경하지 않는다는 개념이다. 즉 서비스 운영 환경을 이미지로 생성한 뒤 서버에 배포하여 실행한다. 이때 서비스가 업데이트 되면 운영 환경 자체를 변경 하지 않고, 이미지를 새로 생성하여 배포하는 것이다. 클라우드 플랫폼에서 서버를 쓰고 버리는 것처럼, 변경 불가능 구조(Immutable Infrastructure) 역시 서비스 운영 환경 이미지를 한 번 쓰고 버린다. 이것은 편리한 관리, 쉬운 확장, 테스트의 손쉬움 그리고 가벼운 시스템으로 어디서든 실행 가능한 환경을 장점으로 가지며, 도커는 바로 변경 불가능 구조를 구현한 프로젝트라고 할 수 있다(이재홍, 2014).

한편 도커를 기능적 측면에서 보면, 서버에서 여러 개의 컨테이너(이미지)를 실행하고 저장, 배포가 가능함을 생각할 수 있다. 컨테이너에 오픈소스 소프트웨어는 물론이고, 직접 만든 프로그램도 넣을 수 있다.

도커는 지금까지 사용해왔던 VMware, Microsoft Hyper-V(Virtual PC), Xen, 리눅스 KVM 등의 가상 머신과 유사하다. 가상머신에 리눅스를 설치한 뒤 각종 서버 프로그램과 DB를 설치하고, 개발한 애플리케이션이나 웹사이트를 실행했다. 이렇게 구축된 가상 머신 이미지를 여러 서버에 복사해서 실행하면 이미지 하나로 서버를 계속 만들어낼 수 있다. 보통 가상 머신 서버를 독자적으로 운영하거나, 서비스 형태로 제공되는 AWS, Azure, GCP를 사용하기도 한다.

도커는 가상 머신과는 달리 이미지 생성과 배포에 특화된 기능을 제공하고, 도커 허브를 통해 이미지를 공유할 수 있다.

컨테이너는 가상 머신처럼 작동하지만 VM 관련 오버헤드나 과도한 앱 크기 등 VM의

단점을 넘어 애플리케이션 워크로드를 쉽게 이동할 수 있다. 앱과 서비스를 패키징화하기 때문에 물리, 가상 또는 클라우드 환경으로 자유롭게 옮길 수 있다. 리눅스의 컨테이너 기능을 가져와 명령줄 인터페이스(CLI : Command Line Interface), API를 통해 사용할 수 있도록 지원한다.

특히 여러 가지 공통 애플리케이션 요소를 사전 구성해 도커 컨테이너로 사용할 수 있어, 소프트웨어 스택을 분리된 구성요소, 즉 마이크로서비스(Microservices) 모델로 배포할 수 있다. 도커를 100% 활용하기 위해서는 각 요소가 세부적으로 어떻게 결합하는지 이해하는 것이 중요하다.

도커 허브는 다운로드할 수 있는 막대한 양의 컨테이너 이미지를 제공하여, 다른 사람들이 이미 작업해 둔 내용을 그대로 이용할 수 있는 장점을 제공한다. 나아가 도커가 개발한 도구들도 제공되는데, 클러스터링 관리자인 스웜(Swarm), 컨테이너를 이용한 작업을 할 수 있는 GUI를 제공하는 카이트매틱(Kitematic), 도커 호스트를 생성하기 위한 명령 유틸리티인 머신(Machine) 등이 있다. 도커 엔진을 오픈 소스화 함으로써, 도커는 대형 커뮤니티로 성장할 수 있게 되었으며, 버그 수정과 기능 향상 등과 관련되어서 많은 이의 도움이 가능하다. 도커의 여러 장점으로 인한 급성장은, 이 분야의 사실상 표준으로 자리 잡게 되었음을 의미한다(아드리안 모트, 2016).

III. 도커 이미지 구축을 통한 블록체인 접속 및 거래 구현

3.1 도커 시스템 구성 방안

도커 이미지 구축을 위하여, 도커 컨테이너를 오라클 가상 머신과 함께 설치하여야 한다. 가상 머신 및 컨테이너 안에 실행되는 OS의 기준은 여러 가지가 있으나, 본 연구에서는 오라클 가상 머신과 함께 윈도우 8.1(64bit)에서 사용 가능한 도커 18.3버전을 다운로드 하여 설치한다.

시스템 권장 사항을 기준으로 하드웨어 구성 환경은 x86_64 계열의 하드웨어를 사용하고, 메모리는 2GB를 할당하고 하드디스크 용량은 10GB를 할당하며, 세부 사양은 <표 2>와 같다.

〈표 2〉 도커 설치 하드웨어 구성

CPU	Intel Pentium 3558U 64bit Processor
Chipset	1.70 GHz, 2MB L3 Cache
Memory	4GB DDR3L System Memory at 1600MHz
HDD	Samsung 500GB S-ATA III Hard Drive
Network	Fast Ethernet [10/100], 802.11 b/g/n 1X1
VGA	Intel HD Graphics

3.2 도커 설치 및 이미지 구축

Git 허브에서 도커 18.3버전의 도커 툴박스를 임의의 폴더로 내려 받은 후, 실행하고 환경설정을 마친다.

바탕화면에서 오라클 가상 머신과 도커 퀵스타트 단축메뉴를 확인한 다음, 가상 머신의 설정값을 본인의 시스템 사양에 최적화시켜 준다. 바탕화면의 퀵스타트를 실행하여 'hello world'를 입력하고 정상설치 여부를 확인한다. 정상설치와 접속이 이루어 지면 [그림 4]의 화면을 확인할 수 있다. 컨테이너 작동은 다음 명령을 입력한다.

\$docker ps

또한 작동 중인 것과 작동되지 않는 모든 컨테이너 확인은 다음 명령을 입력한다.

\$docker ps -a

[그림 4] 도커 컨테이너 초기 접속 화면

한편 도커 허브에 mirairo라는 아이디 소유자의 하이퍼레저패브릭(Hyperledgerfabric), 레포지토리(Repository)의 이미지를 jk라는 이름을 부여하여 작동시키고자 하면 다음 명령을 입력하며, 그 결과를 [그림 5]에서 확인할 수 있다.

\$docker run -name jk -d mirairo/hyperledgerfabric

아울러 작동 중인 컨테이너의 여러 환경변수 값을 확인하기 위해서는 다음과 같이 inspect 명령을 사용한다.

\$docker inspect <컨테이너ID>

[그림 5] 도커 허브 레포지토리 ID와 이름이 부여된 컨테이너 작동

```

mirae@leejo MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name jk -d mirairo/hyperledgerfabric
939853bc135a5ca4d5a7a45ebf991dcf853e94790cecbca284453023ab07e7f9

mirae@leejo MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
939853bc135a   mirairo/hyperledgerfabric           "pn2-docker composer..." 24 seconds ago Up 20 seconds 8080/tcp     jk
dca4f5a4380b   mirairo/hyperledgerfabric           "pn2-docker composer..." 35 minutes ago Up 35 minutes 8080/tcp     jkhyper
  
```

또한 가상 머신에 접속한 IP 주소가 중요한데 이를 찾는 명령어는 다음과 같으며, 결과는 [그림 6]과 같다.

\$docker - machine ip

[그림 6] 가상 머신 IP 주소 찾기

```

mirae@leejo MINGW64 /c/Program Files/Docker Toolbox
$ docker-machine ip
192.168.99.100

mirae@leejo MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
dca4f5a4380b   mirairo/hyperledgerfabric           "pn2-docker composer..." 7 minutes ago Up 7 minutes 8080/tcp     jkhyper
  
```

\$docker inspect <컨테이너ID>로 확인된 IP주소를 이용하여, 브라우저 상에서 블록체인 하이퍼레저패브릭 접속 결과 화면을 [그림 7]에서 확인할 수 있다. 이때 사용한 IP주소는 192.168.99.100이며 포트 번호는 7000으로 연결하였고, 블록체인의 하이퍼레저패브릭 컴포즈 플레이그라운드에 접속된 것이다.

[그림 7] 도커 가상 머신을 이용한 하이퍼레저패브릭 접속

이제 [그림 8]에서 보는 것과 같이, 블록체인 시스템이 이용가능 상태가 되었으므로, 사용자는 원하는 비즈니스 네트워크를 구성하여 직접 사용할 수 있다. 즉, 참여자, 자산, 거래 및 이벤트를 차례대로 생성하여 비즈니스 목적에 부합한 체인을 구성하여 운영할 수 있으며, 이를 통하여 구체적인 회계장부의 처리, 자산의 이동 등과 관련한 회계 거래 내역을 기록하고 분류 정리할 수 있다.

[그림 8] 블록체인 비즈니스 네트워크 구성 시작

IV. 결 론

블록체인 시스템을 효과적으로 사용하기 위해, 도커 컨테이너(Docker Container)를 활용하는 추세에 맞추어, 본 연구에서는 하이퍼레저패브릭 컴포즈에 접속하고 블록체인 참여자 등의 비즈니스 네트워크 생성을 위하여, 도커 컨테이너를 설치하고, 환경을 설정하여 도커 이미지를 통한 가상 접속을 실행하였다.

도커 컨테이너의 사용 목적이 리눅스 가상 머신에 기초하여, 효과적인 이미지 프로그램의 생성, 실행 및 배포를 통하여, 가상 및 클라우드 환경에서의 효율적 시스템 관리에 있음을 확인하였다.

본 연구에서는 분산원장처리와 네트워크 구축을 위한, 하이퍼레저패브릭 컴포즈 실행을 위하여 가장 많이 활용되고 있는 도커 컨테이너의 기초를 살펴보았다.

구체적으로 도커 컨테이너의 설치, 환경설정, 이미지 구축, 가상 머신의 IP 구성, 생성된 도커 허브에 레포지토리(Repository) 접속 그리고 웹 브라우저를 통한 블록체인 하이퍼레저 패브릭 컴포저에 접속하기까지의 세부적 절차를 실행하고 결과를 확인할 수 있었다.

본 연구의 시사점은 가상 환경에서의 블록체인 구현을 위하여, 도커 컨테이너의 여러 기능과 처리절차를 수행하여, 블록체인 하이퍼레저패브릭 컴포저에 접속하고, 분산원장처리 초기 실행 단계까지의 여러 처리 과정을 탐색하는데 유용한 시사점을 제공한다. 아울러 블록체인에 접속한 분산원장 정보처리를 통해, 회계정보 생성과 보관에 대한 중앙집중식 회계정보처리를, 분산시스템으로 관리하게 됨으로 회계정보 보안 문제를 근본적으로 해결할 수 있을 것으로 본다.

향후 연구에서는 도커 컨테이너와 하이퍼레저패브릭 컴포즈를 이용하여 회계 장부의 자산을 등록하고 다양한 거래 처리를 수행하는 전 과정을 블록체인으로 구현하여 블록체인의 사용 확산을 기하고자 한다.

참고문헌

〈국내문헌〉

- 김미정, 이수전(2017), “4차산업혁명 기반 사물인터넷 비즈니스 플랫폼 개발 연구,” **전산회계연구**, 15(2) : 1-18.
- 배유미, 정성재, 소우영(2014), “웹서버 구성을 통한 가상머신과 컨테이너 방식 비교 분석,” **한국정보통신학회논문지**, 18(11) : 2670-2677.
- 아드리안 모트(2016), **제대로 배우는 도커**, 비제이퍼블릭.
- 아카하네 요시하루, 아이케이 마나부(2017), **블록체인 구조와 이론**, 위키북스 출판
- 이재홍(2014), **가장 빨리 만나는 Docker**, 길벗
- 이종기(2015), “회계자료 빅 데이터 분석을 위한 R 프로그래밍 실무 처리 차례,” **전산회계연구**, 13(1) : 1-22.
- 이종기(2017), “블록체인에 의한 분산형 원장 처리. 기법의 탐색적 사례연구,” **전산회계연구**, 15(1) : 25-38.

〈외국문헌〉

- Schmidt, E. and Cohen, J.(2013) “The New Digital Age, Reshaping the Future of People,” Nations and Business : 1-315.
- Schwab, K.(2016), *The Fourth Industrial Revolution*, World Economic Forum.
- Simon Stone(2017), “Building Blockchain Apps with Hyperledger Computer,” *Node Summit 2017*, San Francisco.

〈웹사이트〉

- 연구용 도커 가상머신, 2018, <http://192.168.99.100:7000/editor>
- 위키북스, 2018, <http://wikibook.co.kr/blockchain>
- 한국전산회계학회, 2018, <http://kaoca.net>
- IBM, 2018, <http://ibm.com/developerWorks>
- GIT, 2018, <https://github.com>
- Youtube, 2018, <https://www.youtube.com/watch?v=gAxK6zYrfxI>

“본 논문은 학회윤리규정을 준수하였으며 논문에 대한 판권을 학회에 위임합니다.”

<Abstract>

A Docker Container Case Study for Implementing Block Chain Distributed General Ledger

Lee, Jong - Ki*

In the era of the Fourth Industrial Revolution, blockchain technology is attracting more and more attention. Blockchain is one of the best security techniques to store and verify the details of a ledger in a distributed system over a network, using distributed processing technology. Blockchain technology is important for program execution and distribution in a distributed environment. We have therefore dealt with an example of building a blockchain distributed branch processing network using an open source Linux-based docker system. In keeping with the proliferating trends of using a docker container to handle blockchains, this study implemented a hyperledger fabric business network, which is a private block led by IBM and others, using a docker container. We examined the details involved in building a blockchain by installing a virtual machine container, configuring an image, building an image, committing it, and connecting to a blockchain hyperledger fabric composer through a web browser. This study explores the various functions and processing procedures of the docker container to connect to the blockchain hyperledger fabric composer to explore how to create distributed ledger participants, assets, and transactions, and to actually create and distribute blocks. In addition, it is possible to fundamentally solve the accounting information security problem by managing the generation and storage of accounting information by the distributed system through the distributed general ledger process connected to the blockchain.

Key words : Blockchain, Distributed ledger, Docker, Container, Hyperledger fabric

* Professor, Ph.D., Faculty Liberal Arts, Jeju International University, e-mail : mirae910@ jeju.ac.kr