Design and Implementation of a Docker-Based Ipv6 Blockchain Application Framework

Wei Zhang[1], Lan Pan[2,*], Zhihao Ma[1] and Tianrong Cao[1]

[1]Zhejiang University of Finance&Economics, Hangzhou, Zhejiang, China

[2]Tourism College of Zhejiang, Hangzhou, Zhejiang, China

*Corresponding author

Keywords: Docker, Blockchain, Ipv6

Abstract: With the development of financial technology, people hope that more blockchain applications can be used on IPv6 networks. However, the complexity of technology makes the transplantation of blockchain to IPv6 network hard. Containers based on Docker can change this situation. The paper designed a container-based IPv6 blockchain application framework, using the installation of blockchain nodes in Dockers, one-click deployment, and data collection and displaying in the visualization service Docker, making blockchain applications very convenient transplant to IPv6 network. This article also uses Ethereum as an example to show the details of implementing the framework. This framework can greatly promote the deployment of blockchain applications on IPv6 networks and enrich the scenarios of financial technology.

1. Introduction

With the development of financial technology, blockchain, as a new type of technology based on distributed storage and smart contracts, has become widely used[1]. The development of the Internet is also changing these days, and IPv4 protocol can no longer meet the needs of massive networked devices[2]. As a new network protocol, IPv6 has huge address resources, faster network transmission rates, higher network service quality, better mobile network support and higher security[3]. The development of IPv6 is the foundation of the future network[4].

The combination of blockchain technology and IPv6 network can make full use of the advantages of both. IPv6 can solve the bottleneck of blockchain in P2P network transmission, data transmission, security authentication and other aspects from the basic transmission technology level, greatly improving The transmission speed and security of the blockchain.

2. Current Shortcomings

Blockchain, as an innovative application model of financial technology that integrates encryption algorithms, distributed data storage, and consensus mechanisms, is difficult to deploy. Although there exist IPv6-based blockchain applications, such as Xuexin Chain[5], technical difficulties limit the richness of applications deployed in IPv6 network. To run the blockchain in IPv6 mode, for the address type is different from IPv4, not only the source code needs to be modified, but the installation and operation are also very complicated. Despite the potential value of blockchain technology, the IPv6 network does not provide enough support for blockchain. It is necessary to find a way to build a variety of applications in the IPv6 environment easily and quickly. We consider using virtual machine or container technology to accelerate the popularization of blockchain applications in IPv6.

3. Designing the Framework

Docker is an open source container development tool that appeared in 2013. It can automatically package and deploy applications, and create a lightweight and private application environment. With its characteristics, it can be installed at the application level and can run applications on Linux or

  DOI: 10.25236/iccem.2021.042

Windows machines or virtual machines. Compared with virtual machines, Docker is a lighter and more flexible virtualization processing method, which is independent, and can ensure that the container and host environment do not affect each other [6,7]. Users can use Docker to develop and deploy applications more quickly; to achieve faster upgrades and expansions of applications; and get simpler system operation and maintenance and more efficient use of computing resources [8]. As a result, the container technique has been applied in many fields [9].

Docker can also support the configuration of IPV6. IPV6 option can be enabled in a Docker, and then restart the Docker engine to realize the interconnection between containers through IPV6 addresses. The bridge mode is used in Docker. Therefore, we choose to use Docker to implement the IPV6 blockchain application framework.

Docker needs management scheduling tools. When running applications such as building clusters that require multiple environments, Docker-Compose or K8s is needed. Docker-Compose is a tool used to define and run multi-container Docker applications[10]. Through it, we can use YAML files to configure all the services required by the application. Then you can create and start all services from the YAML file configuration with a single command.

Therefore, we design our blockchain application architecture based on the Docker environment, as shown in Figure 1.

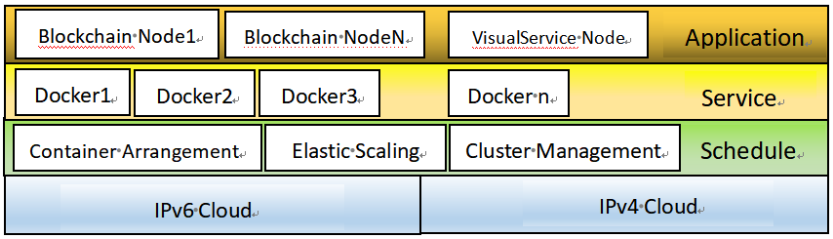

Fig.1 Docker-Based Ipv6/Ipv4 Blockchain Application Architecture

Figure 1 shows that the bottom layer of the architecture is an IPv6 or IPv4-based network, which can run on public clouds such as Ali Cloud and Tencent Cloud, or it can be set up on private servers. The scheduling layer above it helps users to manage Docker containers, such as using Docker Compose, K8s, etc. for cluster management, container arrangement, etc., to simplify Docker container operations. The Docker service layer above it specifically uses multiple Docker containers as carriers to provide services for various applications. These Docker containers are loaded with related applications and stored in the container repository in the form of images, and many Dockers are released as needed. This can flexibly deal with the uncertain activities of financial applications. The top layer is the application layer dominated by blockchain nodes, and each blockchain node is installed in a Docker. Among them, the blockchain node is composed of the data layer and the application layer, as shown in Figure 2.

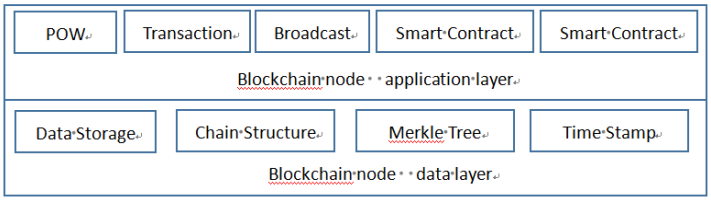

Fig.2 Blockchain Node Composition

Figure 2 shows a blockchain node's structure runs on a Docker, which has a data layer and an application layer.The distributed storage based on a consensus algorithm. In addition to proof of work, transaction recording, data synchronization between nodes, it is also has smart contracts. The data stored or sent by the blockchain application also needs to interact with users, which is the function of the visual service node in Figure 1. It can be a DAPP or some kind of website service to display the data and transactions in the blockchain application.

4. Development

The paper takes the development of Ethereum application as an example to observe the convenience of deploying Docker-based applications on IPv6.

First we set up Docker Engine to support IPV6, start Docker Engine after setting up, its bridge network will automatically support IPV6 network, and then build the container, Docker will automatically assign an IPV6 address to the container. Of course, we can also use a custom setting, which will not be detailed here.

After Docker supporting IPV6, we only need to create the Ethereum node in Docker according to normal routines, and package the visualization server into another Docker, so that we can directly access the visualization web site by obtaining the IPV6 address assigned to the Docker. As for how the blockchain can run on IPV6, you can deploy Ethereum in Dockers by configuring the virtual machine of the Ubuntu18.04 system, installing Dockers, geth, and Firefox browser (which can support the Metamask plug-in). After the start of Ethereum, the visualization serer can access the nodes' information through IPV6.

This blockchain application deployment is specifically divided into these steps:

1) Docker pulls the blockchain node images and the visualization service (visualserver) image.

2) Create a blockchain node network.

3) Start the container of the visualserver node (automatically run the visualserver node).

4) Then run multiple Node nodes on this machine or other hosts with public or local IP addresses (note: you need to start the corresponding blockchain nodes first).

5) After the blockchain nodes are connected, perform block operations such as mining.

6) The visualserver node receives the updated block information of the Nodes and plots them.

7) Browse the corresponding visualserver port through the browser to view the trajectory graph formed by the node block.

To specifically configure an Ethereum network, we first create a master node, use a persistent container to read the genesis.json file, and initialize the Ethereum network.

Then we create miner by creating the file: /workspace/dapp/miner.sh:

#!/bin/sh

cp -r /workspace/dapp/miner/data/keystore/ ~/data/keystore/*

geth --datadir ~/data/ --networkid 622 --port 8200 --http --http.addr "[::]"

--rpccorsdomain "" --rpcport 8545 --allow-insecure-unlock --unlock "0x5464bece7148e1d5c45618f5603389851c30f485" --password /workspace/dapp/password --miner.etherbase "0x5464bece7148e1d5c45618f5603389851c30f485" console*

#--rpcapi admin,eth,miner,web3,personal,net,txpool

The cp command above is to copy the private key file of the generated account to the home directory of the container. The second command line starts the command of the Ethereum node.

We start creating the master node container by creating the file: workspace/miner_docker.sh:

docker stop miner

docker rm miner

docker run -it --name miner -v /workspace:/workspace -p 8545:8545 -p 30303:30303 -p 8200:8200 --entrypoint /workspace/dapp/init.sh ethereum/client-go /workspace/dapp/miner.sh

Next, we will create the slave node container.

We create an autorun script by creating the file: /workspace/dapp/node1.sh:

#!/bin/sh

cp -r /workspace/dapp/miner/data/keystore/ ~/data/keystore/*

geth -datadir ~/data/ --networkid 622 --miner.etherbase "0x2dd36c57fbc1ba30406e5fe1e307572465a7288b" console

Create a container by creating the file: /workspace/dapp/node1_docker.sh:

docker stop node1

docker rm node1

docker run -it --name=node1 -v /workspace:/workspace --entrypoint /workspace/dapp/init.sh ethereum/client-go /workspace/dapp/node1.sh

In the same way, multiple node containers can be created. After a node is created, it needs to be added to the Ethereum network where the master node is located.

Record the node “enode://<node public key>@<node IP address>:<node port>“.

Then we add the node by using admin.addPeer(“enode://<node public key>@<node IP address>:<node port>“), to connect all nodes to the Ethereum network of the master node to perform mining.

From the foregoing operations, we can configure the Ethereum application in containers, pull them from the container repository when necessary, configure multiple blockchain nodes as needed, and then deploy them on the public cloud or private servers on the IPv6 network with just one click. You can easily get the running Ethereum application. When an application needs to be upgraded, simply replace the version of the application in the container and deploy it according to the original process.

5. Visualization Service Implementation

After the Ethereum network is set up, you can run the visualization service website, display blockchain data and transactions on a web page’s interface. The web-side design is written in Node.js and uses the Echarts plug-in.

First, by defining the required global variables, we use the get( ) method to obtain data from the blockchain node, assign it to the corresponding variable, and use the refresh() method to update these variables every 10 seconds. These variables include changes in nodes, changes in transactions and changes in account balances. These are finally displayed in webpages using the Echarts canvas.

We initialize the balance of the account by deploying a smart contract. Then Web3.js was called to communicate with the established Ethereum node through RPC. Then the Metamask plug-in or web page was used to call a MetaCoin.sol smart contract for transferring virtual money. After the transaction is completed, the webpage interface will display the flight graph trajectory of the transaction node, as shown in Figure 3.

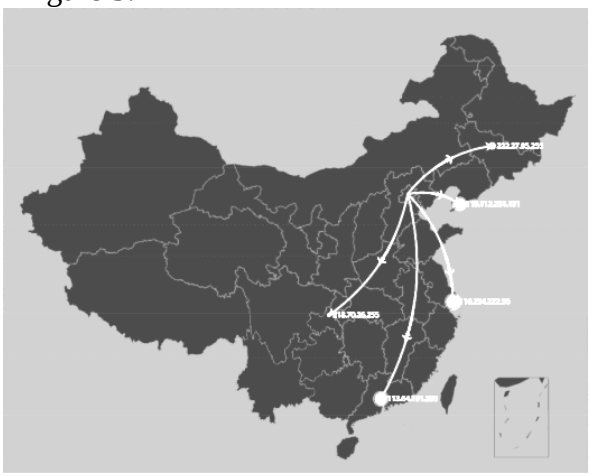

Fig.3 Blockchain Visualization Service

In Figure 3, 6 blockchain nodes were created in different dockers, and sent or receive virtual currency asynchronously.

The Ethereum applications and visualization services above can all be developed in the IPv4 network environment and placed in Docker, and the YAML file is used to make a one-click deployment mode. When needed, it also can be pulled from the container repository and deployed to the IPv6 network environment, which greatly improves the deployment efficiency. The ease of the application installation provides a good way to promote the application of complex systems such as blockchain to IPv6 network.

6. Summary

We use an example of Ethereum application deployment to an IPv6 network to explain our IPv6 blockchain application framework based on Docker containers, and use Docker's powerful rapid

application development and functions to achieve rapid application deployment on IPv6 networks. The framework provides easy delivery; faster application upgrade and expansion; simpler system operation and maintenance; more efficient use of computing resources. This is particularly important for developing IPv6-based financial technology applications and enriching financial scenarios. It can dynamically increase or decrease blockchain nodes according to security requirements, and flexibly respond to the uncertainty of financial applications. We hope this framework will be used to cross IPv4 and IPv6 platform to integrate traditional IPv4 applications and IPv6 applications together, and provides financial services for various industries.

7. Acknowledgement

This research was financially supported by CERNET Innovation Project(NGII20170903)and the Huaneng Group Headquarters Science and Technology Project “HNKJ20-HXX Blockchain Applied Technology Research in the Bidding”.

References

[1] Li Wei, Blockchain Empowers Innovative Supply Chain Financial Services, The Banker, vol.5, pp. 49-50, 2021.

[2] He Jinsong, Peng Zhichao, He Wenhua, Jiang Xuejun. Comparative study of IPv4,IPv6 and IPv9, Software Engineering, vol.19, pp. 18-20, 2016.

[3] Yu Kun, Wu Xiaojin. IPv6 Technology and Application, Tsinghua University Press, pp. 4-5, 2020

[4] Tian Hui;Wei Zheng. IPv6+ innovation of the NGI, Telecommunications Science, vol.36, pp. 3-10, 2020

[5] Lin Jiaqin, Xia Jiaoxiong, Xu Zhao. Xuexinchain platform based on blockchain and ipv6, Yangtze River Information and Communication, vol.196(4), pp. 152-153, 2019.

[6] G. M. Tihfon, S. Park, J. Kim, et al. An efficient multi-task PaaS cloud infrastructure based on docker and AWS ECS for application deployment, Cluster Computing, vol.19, pp. 1585–1597, 2016.

[7] Liu Yuliang, He Lu, Analysis of container cloud platform based on Docker technology, Popular Science and Technology, vol.23, pp. 15-17, 2021.

[8] Wenbo, Liu Shasha, Gong Yufei, The application of Docker technology in the rapid deployment of enterprise system software, Jiangsu Science and Technology Information, vol.38, pp. 57-59, 2021.

[9] Zhao Yufeng, Lei Sheng, Zhang Guogang, Geng Yingsan, Design and development of a container-based power equipment simulation cloud platform, Computer Engineering, 2020.9. https://doi.org/10.19678/j.issn.1000-3428.0058324

[10] Jiang Anguo;Wang Jinquan;Bian Jinwei. Docker-Compose research on Docker technology, Modern Information Technology, vol.2, pp. 75-77, 2018.