2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)

EtherShare: Share Information in JointCloud Environment Using Blockchain-based Smart Contracts

Peilin Zheng
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China
zhengpl3@mail2.sysu.edu.cn

Zibin Zheng
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China
zhzibin@mail.sysu.edu.cn

Weili Chen*
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China
chenwli9@mail2.sysu.edu.cn

Jing Bian
The Guangdong Province Key Laboratory of Computational Science
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China
mcsbj@mail.sysu.edu.cn

Jianxun Eileen Yang*
Shenzhen Research Institute
Sun Yat-Sen University
Shenzhen, China
eileenyjx@163.com

***Abstract*—With the development of cloud computing, a great amount of information is stored on cloud, such as Facebook, Twitter, and so on. Single cloud or company is not reliable enough for permanent information storage. Once the cloud or company is in downtime, users could lost their information. Multiple clouds can store the information with more security. However, in JointCloud environment, it is a big problem for multiple clouds or companies to maintain the common rights of information for each user. In this paper, we propose EtherShare, a blockchain-based application in JointCloud environment. EtherShare enables users to share information with permanent storage and open access. It keeps the rights of users through smart contracts to make consensus between multiple clouds. And this application is open source on Github.**

***Index Terms*—blockchain, smart contract, decentralized application, JointCloud**

I. Introduction

Blockchain is proposed as the underlying data structure of Bitcoin [1]. It is a continuously list of blocks. Each block is linked and secured using cryptography. Each peer in the P2P transaction network records the transactions and packages them into a block in a period of time. Finally, all the peers in the P2P network reach a consensus of the blockchain.

Blockchain technology is decentralized, tamper-resistant, and traceable [2]. A blockchain-based smart contract [3] is an event-driven promise defined by the program. This program is run independently on each peer in the blockchain network. The result of the program is confirmed by the blockchain so that nobody can tamper with it. Therefore, the execution of the blockchain-based smart contract can be trusted by everyone.

The applications using blockchain-based smart contracts are called decentralized application (*DApp*). In recent years, since blockchain-based decentralized applications can enhance trustworthy, decrease the cost of central trusted authority, they have gained a lot of attentions from both industry and academia [4].

JointCloud is a new generation of cloud computing model which facilitates developers to customize cloud services and create values among clouds by the way of software definition [5]. Since sharing information on one single cloud is unsafe for both storage and usage due to the downtime of the cloud, to share information in JointCloud environment with multiple cloud entities is important to keep the safety of the information. However, it is difficult for multiple cloud entities to trust each other and do the same operation of the information (e.g., update, delete, etc.).

To address this problem, we propose an blockchain-based application in JointCloud computing to enhance the trust between different cloud entities. As Figure 1 shows, different entities in the JointCloud environment maintain a blockchain running with the smart contracts. Then the users interact with the JointCloud entities to share information or do other operations. Since the storage is kept by all the entities which are maintainning the blockchain, it is reliable enough and can be considerd as permanent storage in some cases. The blockchain-based smart contracts are used to save the rights of the users. Each JointCloud entity validates the users and operates the storage through the contracts so that they can trust each other.

The contribution of EtherShare can be divided into three folds:

- EtherShare enables users to share information with permanent storage and open access to any entry.
- EtherShare keeps the rights of information for users so that users have the same rights in different entities in

978-1-7281-1442-2/19/$31.00 ©2019 IEEE
DOI 10.1109/SOSE.2019.00040

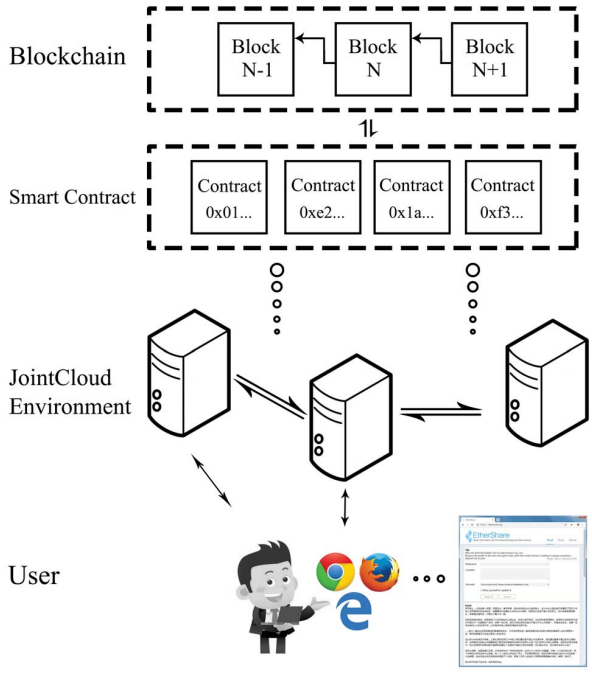


Fig. 1. Overview of Entities of EtherShare in JointCloud Environment

JointCloud.

- EtherShare has been implemented on Ethereum and open source on Github.

The rest of this paper is organized as follows. Section II describes the basic concepts of blockchain, smart contract and JointCloud computing. Section III proposes the framework of EtherShare. Section IV introduces the specific implement and Section V concludes the paper.

II. Basic Concepts

This section introduces the basic concepts of the blockchain, smart contract, and JointCloud computing.

A. Blockchain

The concept of blockchain was firstly proposed in Bitcoin [1]. At first, it is a kind of data structure as the underlying storage for peer-to-peer payments. Figure 2 shows the structure of blockchain. Every block contains the transactions in a period of time. And, every block is linked into a chain-like data structure. Thus it is called blockchain. Each block has a hash value of itself and the hash value is contained in the next block, so that the content in the block (e.g., transactions, timestamp, etc.) is tamper-resistant and traceable.

Each peer in the peer-to-peer network called miner maintains a blockchain by itself. And they reach a consensus of the local blockchain with other peers by consensus protocols.

Take blockchain as a distributed ledger, the basic concepts of it are listed as follows:

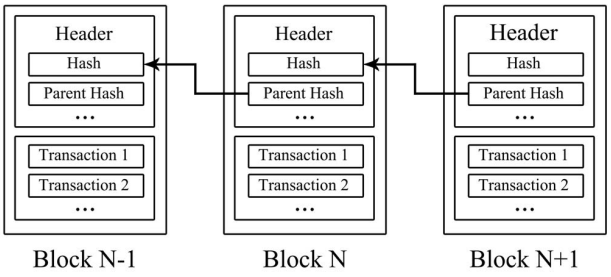


Fig. 2. Data Structure of Blockchain [6]

- Transaction: A message to change the ledger, such as a payment. If someone wants to send Bitcoin to other, he should send a message with the amount and the receiver's address. This kind of data structure is so-called transaction. After that, this transaction is signed by his private key and broadcast to the p2p network. When every Bitcoin peer receives the transaction, they will validate the signature before execution.
- Block: A set of the transactions in a period of time. Every block consists of the block header and the block content (all the transactions at the time). The block header is used to record the basic information of the block (e.g., hash of previous block, timestamp, etc.).
- Chain: In the Bitcoin blockchain [1], every block is generated after the previous one so that they record the hash of the previous block. All the blocks are linked by their hash values. This chain-like structure is similar to the list structure, shown in Figure 2.

In this paper, blockchain is used as a decentralized and traceable storage for every entries of EtherShare, so that it is hard for someone to temper with the information.

B. Smart Contract

Smart contract is a promise defined by digital form [3]. A blockchain-based smart contract is defined by the code stored with the blockchain system. It can be considered as an event-driven program. As shown in Figure 3, the execution of smart contract on the blockchain is distributed and independent. The users who want to invoke a contract should send the peers a transaction including the address of the contract, the calling function, and the parameters. After that, every peer will invoke the function of the contract. It is executed independently by every validating peer [7]. Finally the peers will save the result back to the blockchain no matter it is successful or not. The basic concepts are as follows.

- World State: A key-value database that records all the state of the accounts and contracts in the blockchain system. To store information on blockchain is limited, so smart contracts need this database to store information of each account (e.g., balance). In order to make the world state the same, most blockchain systems put the world state into a data structure of tree to record the root onto

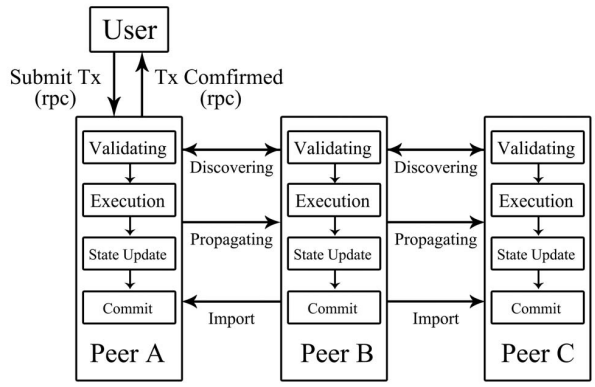


Fig. 3. An Execution of Smart Contract Invocation on Blockchain [6]

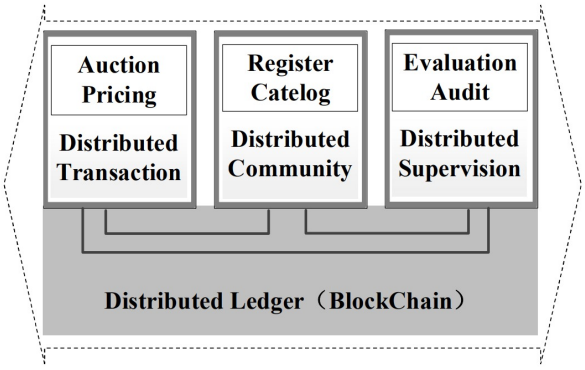


Fig. 4. JointCloud Collaboration Environment [5]

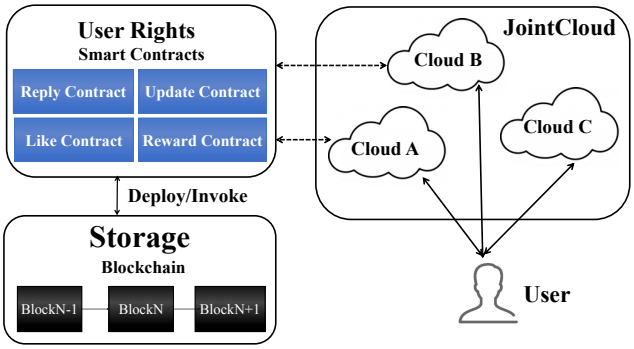


Fig. 5. Framework of EtherShare in JointCloud environment

the blockchain (e.g., Merkle Patricia Tree in Ethereum [8], Bucket Tree in Fabric [9]).

- Contract Code: The code stored on the blockchain with the world state. The contract code can be read from the world state and executed in the virtual machine (e.g.Ethereum Virtual Machine). It is also called "chaincode". It defines the main execution logic of the smart contract.
- Validating Peer: The peers that maintain the blockchain. The validating peers will validate transactions, execute the smart contract, update the state, and finally reach the consensus. In this paper, each JointCloud entity should hold at least one validating peer.
- Non-validating Peer: The peers that do not commit the block. Non-validating peer only respond to REST requests from clients and send the transaction to the validating peers.

In this paper, smart contracts are used to define the user rights of EtherShare. The smart contracts enable that the user rights cannot be maliciously interfered by others. Thus, using blockchain-based smart contract to store and update information is more reliable than traditional software.

C. JointCloud Computing

JointCloud is a new generation of cloud computing model [5]. It aims at empowering the cooperation among multiple Cloud Service Providers to provide cross-cloud services [10]. It can be considered as a large-scale, flexible, and elastic computing resource platform [11].

In order to achieve borderless resource sharing, JointCloud computing integrate the multiple cloud resources and services deeply. As shown in Figure 4, JointCloud Collaboration Environment enables the cooperation among independent clouds based on the transaction service, community service, and supervision services. These services are conducted with blockchain as a distributed ledger so that each independent cloud can trust on it.

III. Framework

In this section, the framework of EtherShare is given. As shown in Fig 5, the framework of EtherShare can be divided into three layers: storage, user rights and entries. Specifically, the storage is designed on blockchain. And the user rights including the rights to reply, update, like, reward the information is defined as the smart contracts. The independent cloud entities in the JointCloud environment are the entries of EtherShare. These three layers would be described in details in the next subsections.

A. Storage: Blockchain

The storage of EtherShare is on blockchain. In this way, the data storage could be considered as permanent storage. Different from traditional centralized system, the data storage in JointCloud environment should be fully distributed to make it reliable enough and trusted by each cloud entities

Public blockchain could be chosen as the blockchain layer, such as Ethereum [12], Neo and so on. In some cases (e.g., military scene), in order to keep the data privacy, consortium blockchain should be chosen.

B. User Rights: Smart Contracts

The user rights are the operation access to the information shared in the system. For example, if a user wants to update

Data: nickname, content, AllowUpdated, msg.sender

Result: Information Shared

Load the smart contract;

ShareID := count ;

allShare[ShareID][0].nickname := nickname ;

allShare[ShareID][0].content := content ;

allShare[ShareID][0].AllowUpdated := AllowUpdated ;

allShare[ShareID][0].sender := msg.sender ;

count += 1 ;

EventNotification(ShareID, 0);

Algorithm 1: Basic Contract

the information that is shared previously, he should have the right. In a centralized system such as Facebook and Twitter, it is easy to validate the user. But in JointCloud environment, each entities trust little with each other. Thus, EtherShare uses smart contracts to define and validate the user rights.

Actually, to store information on blockchain is to invoke a basic contract to record the structured data of information. In EtherShare, a tuple of $< \quad nickname, content, AllowUpdated, msg.sender \quad >$ is defined to record the information shared. As shown in Algorithm 1, this smart contract firstly generate a ShareID of an information, then record the tuple into the state of the blockchain. In this smart contract, any entity (or users) has the right to share information into the system. Although this contract does not validate any right of the users, it records the address of the sender and other information that will help the rights validation for other contracts.

The user rights are divided into four parts: Update, Reply, Like (or so-called thumbs-up), and Reward.

Update Contract is used to validate and execute the right to update information. As shown in Algorithm 1, when a user share the information in the system, the smart contract will record the address of the author, and whether the author allow the information to be updated next time. And, as shown in Algorithm 2 if the user wants to update the information stored in the system, he should have the right. The right to update information consists of two parts: the sender of the message should be the author and the author should allow himself to update the information.

Reply Contract is used to validate whether the information exists and add the reply attached to it. As shown in Algorithm 3, the Reply Contract requires the parameters of ShareID, nickname, AllowUpdated, and msg.sender which are similar to the Basic Contract except for ShareID. Actually, the Reply Contract and Basic Contract hold the same data structure to store the information. But Reply Contract is only able to write the information attached to a existed "Share". Thus the contract will firstly validate that whether the information corresponding to the ShareID exists. Then the "Reply" will be attached to the "Share". Finally the event will notice all

Data: ShareID, ReplyID, content, msg.sender

Result: Information Updated

Load the smart contract;

if *msg.sender==allShare[ShareID][ReplyID].sender* **then**

  if *allShare[ShareID][ReplyID].allowUpdated* **then**

    allShare[ShareID][ReplyID].content := content;

    ;

    EventNotification(ShareID, ReplyID);

  else

    Revert and Exit;

  end

else

  Revert and Exit;

end

Algorithm 2: Update Contract

the entities in JointCloud that the corresponding information is added.

Data: ShareID, nickname, content, AllowUpdated, msg.sender

Result: Information Replied

Load the smart contract;

if *0* $\leq$ *ShareID* $<$ *Count* **then**

  ReplyID := Reply(ShareID, nickname, content, AllowUpdated, msg.sender) ;

  EventNotification(ShareID, ReplyID);

else

  Revert and Exit;

end

Algorithm 3: Reply Contract

Like Contract is used to validate and record the "Thumbs-up" (or so-called "Like") of the information. Algorithm 4 shows the pseudocode of the Like Contract. After received a message of "Like", the Like Contract will firstly validate whether the user has already sent a "Like" to this information. If the user has, the contract will revert the whole transaction and exit. If the user has not, the Like Contract will add the "Like" into the counter of the information, then record this "Like" for the user.

Reward Contract is used to validate the reward and transfer the money from the user to the author of the information. As shown in Algorithm 5, the Reward Contract will do complex validation for the reward due to the security requirement of money transferring. The first validation is to check whether the corresponding information is existed. If so, the Reward Contract will fetch the address of the author of the information

Data: ShareID, ReplyID, msg.sender

Result: Information Thumbs-up

Load the smart contract;

if *hasLiked[ShareID][ReplyID][msg.sender] is null* **then**

    allLike[ShareID][ReplyID] += 1 ;

    hasLiked[ShareID][ReplyID][msg.sender] := True ;

else

    Revert and Exit;

end

Algorithm 4: Like Contract

as the target address to send the money. Then the contract will send the money of the value of the message to the target address and check whether it is successful. In some cases, the author of the information could be a contract with no function to receive money so that the reward would be refunded. Finally, after all the validation, the reward will be recorded into the state of the blockchain.

Data: ShareID, ReplyID, msg.sender, msg.value

Result: Author Rewarded

Load the smart contract;

if *allShare[ShareID][ReplyID] is not null* **then**

    targetAddress = allShare[ShareID][ReplyID].sender;

    if *SendMoney(targetAddress, msg.value)* **then**

        RecordReward(ShareID, ReplyID, msg.sender,

        msg.value) ;

    else

        Revert and Exit;

    end

else

end

Algorithm 5: Reward Contract

C. Entries: JointCloud Entities

Different from the centralized information system that holds all the entries, the entries of EtherShare are held by all the entities in the JointCloud environment, even the user himself. The entry could be a website, a mobile application or just an explorer of the blockchain. Each JointCloud entity will run at least one peer in the blockchain system to run the smart contract and keep storage synchronization.

As for the user, in most of the cases, the user should hold the private key by itself. When the user wants to interact with the blockchain and the smart contracts of EtherShare, he should send a message to an entry signatured by his private key. In this way, the user will keep the user rights by itself, represented by the private key.

Fig. 6. Example of Graph User Interface of EtherShare

In some cases, if the user trusts much on the JointCloud entity, it can store the private key on the JointCloud entity. However, if the entity is down, the rights of user could be lost.

In summary, EtherShare is designed in three layers: storage in blockchain, user rights kept by several smart contracts, and entries held by the JointCloud entities. This framework provides a permanent storage, open access and secure user rights to the users in JointCloud environment.

IV. Implement

We implement EtherShare as a decentralized application on Ethereum (one of the most popular public blockchain). And we also create an example website[1] as the graph user interface of one of the entry, shown in Figure 6. All the source code has been open on Github[2]. In this section, some detailed implement will be proposed.

A. Blockchain and Smart Contracts

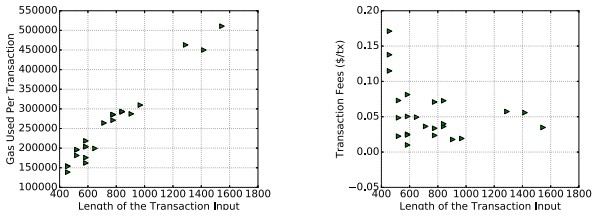
Because public blockchain is more open than consortium blockchain, we implement the EtherShare on Ethereum. Four smart contracts are already deployed on the mainnet of Ethereum, corresponding to the contracts given in Section III. The smart contracts are written in Solidity and deployed as follows:

EtherShare[3]: The contract to record the information shared and control the user rights to update and reply.

[1]http://EtherShare.org

[2]http://github.com/ethershare/

[3]https://etherscan.io/address/0xc86bdf9661c62646194ef29b1b8f5fe226e8c97e



(a) Gas Used Per Transaction (b) Transaction Fees

Fig. 7. Cost Evaluation on Ethereum Mainnet

EtherShareReward[4]: The contract to validate, transfer and record the reward from the user to the author of the information.

EtherShareLike[5]: The contract to validate and record the "Thumbs-up" to the information.

EtherShareDonation[6]: The contract to receive and use the donation of EtherShare.

B. Graph User Interface

We found an example website to show the graph user interface to EtherShare. It is conducted with HTML and JavaScript. The HTML files define the web page style and the JavaScript files interact with the blockchain peers to run the smart contract. Figure 6 shows the screenshot of the website.

The website is not the only one entry to EtherShare. EtherShare is a decentralized application which can be run by any personal entities or JointCloud entities. Thus users can download code to their own server or access the public blockchain and smart contracts in other ways.

C. Cost Evaluation

Sincerely, the more reliable the storage is, the more it costs. As for consortium blockchain, every JointCloud entity holds at least one server as a node so that the cost is the fees of maintaining the server. As for public blockchain, it costs more to send transaction to the public blockchain. Thus, we conduct real-word evaluation to see how much the fees are. The result is shown in Figure 7.

Gas is a unit that measures the resource consumption of Ethereum. Figure 7(a) shows that the consumption of gas is proportional to the length of the information. However, Figure 7(b) shows that the transaction fees is not significantly related to the length of the information. Because the Gas Price that the user choose to paid for the transaction could be different. The higher the Gas Price given, the higher the transaction cost will be and the quicker the transaction will be confirmed. And, in most cases, at the current price of ETH of $123.97, the transaction fees are generally less than $0.10.

V. Conclusion and Future Work

In this paper, a framework and implement to share information in JointCloud environment is proposed. The contribution is concluded as follows:

- EtherShare provides permanent storage and open access to share information to any JointCloud entity.
- EtherShare uses smart contracts to keep the user rights of information to ensure the same rights in different entities in JointCloud.
- EtherShare is open source on Github and developed on Ethereum public blockchain.

There are also some future work to follow up: **(1)Improve throughput:** The throughput of public blockchain is too low to meet the requirement of performance to share information. Consortium blockchain can be considered. **(2)Information Filtering or Management:** There could be some harmful information shared on the blockchain, thus a way of information filtering or management in JointCloud is needed to avoid them.

[4]https://etherscan.io/address/0x28daa51dc3d80a951af9c451d174f0c7156c6876

[5]https://etherscan.io/address/0x43820f75f021c34ce13ded1595633ed39b79ab47

[6]https://etherscan.io/address/0x475de1f3e1ba5aeefc9fc694852c8fce59b353a1

Acknowledgment

The work described in this paper was supported by the National Key Research and Development Program (2016YFB1000101), the Science and Technology Plan of Guangdong Science and Technology Department(2015A030401004) and the Fundamental Research Funds for the Committee of Science and Technology in Shenzhen (JSGG20160607161350293). Weili Chen and Jianxun Eileen Yang are the corresponding author.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] Z. Zheng, S. Xie, H.-N. Dai, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services, accepted*, 2016.

[3] N. Szabo, "The idea of smart contracts," 1997.

[4] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Big Data (BigData Congress), 2017 IEEE International Congress on*. IEEE, 2017, pp. 557–564.

[5] H. Wang, P. Shi, and Y. Zhang, "Jointcloud: A cross-cloud cooperation architecture for integrated internet service customization," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 1846–1855.

[6] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*. ACM, 2018, pp. 134–143.

[7] M. Murthy, *Life Cycle of an Ethereum Transaction*, https://medium.com/blockchannel/life-cycle-of-an-ethereum-transaction-e5c66bae0f6e, 2017.

[8] W. Chen, Z. Zheng, M. Ma, P. He, P. Zheng, and Y. Zhou, "Efficient blockchain-based software systems via hierarchical bucket tree," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*. ACM, 2018, pp. 360–361.

[9] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*.

[10] F. Xiang, H. Wang, P. Shi, Y. Fu, and Y. Wang, "Jcledger: A blockchain based distributed ledger for jointcloud computing," in *IEEE International Conference on Distributed Computing Systems Workshops*, 2017.

[11] Y. Zheng, L. Xu, W. Wei, Z. Wei, and D. Ying, "A reliability benchmark for big data systems on jointcloud," in *IEEE International Conference on Distributed Computing Systems Workshops*, 2017.

[12] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.