2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom) | 978-1-6654-1485-2/20/$31.00 ©2020 IEEE | DOI: 10.1109/ISPA-BDCLOUD-SOCIALCOM-SUSTAINCOM51426.2020.00148

Container Scheduling in Blockchain-based Cloud Service Platform

Yu Lei [1], Philip S. Yu [2]

[1] Department of Computer Science, Inner Mongolia University, Hohhot, China
[2] Department of Computer Science, University of Illinois at Chicago, Chicago, USA

***Abstract*—When massive mobile and Internet of Things applications emerge, the cloud computing models relying on traditional centralized data centers will encounter many constraints. Unlike traditional centralized cloud computing, Mobile Edge Computing integrates cloud computing, mobile computing and wireless network functions, enabling Cloud to handle large amounts of storage and computing problems of mobile devices in real time. We established a Blockchain-based service platform for QoS and QoE in mobile edge cloud ecosystem. The platform user can invoke all API services in the platform without redirecting to the third-party website. The platform users can filter and accept the services recommended by the platform, and also freely compose the services which are Real-world services. In addition, data collected from the platform can be stored on Blockchain, and smart contract is proposed for assuring the QoS and QoE.**

Keywords- Recommendation, Composition, Cloud Services, Blockchain, Smart Contract

I. Introduction

When massive mobile and Internet of Things applications emerge, the cloud computing models relying on traditional centralized data centers will encounter many constraints. Firstly, moving all computing data to the cloud is a huge overhead for network bandwidth resources. For example, traditional cloud computing cannot guarantee real-time processing requirements for all audio and video surveillance data in cities or mass data generated by IoT applications. Secondly, for other time-sensitive applications, traditional cloud computing cannot guarantee low latency requirements. For example, some mobile games require fast response (less than 100ms). Furthermore, some interactive Internet applications, such as real-time voice translation, require less latency (less than 10ms).

Unlike traditional centralized cloud computing, MEC (Mobile Edge Computing) integrates cloud computing, mobile computing and wireless network functions, enabling Cloud to handle large amounts of storage and computing problems of mobile devices in real time.

MEC servers will be deployed on the wireless access network side (network edge) and provide computing and storage resources, thus they can minimize the service delay of terminal devices, save network bandwidth and achieve better positioning. Meanwhile, network operators can also provide additional services in MEC services to enhance user experiences with mobile terminals. In addition, based on MEC hardware platform, network operators may allow third parties (application service providers and content providers) to access their infrastructure for application development.

A service platform for QoS and QoE in Mobile Edge Cloud ecosystem (QMEC) is built by us. The platform is a standalone platform that connects API service providers and service users (including secondary developers and service users). The platform is dedicated to providing users with the most comprehensive and convenient services, as well as helping service providers register services and increase their API invocation times. The platform has brought together more than 80 services required for application development. This service mashup platform has exclusive services such as image processing, speech synthesis, ancient poetry synthesis, movie inquiry and song inquiry. After registering the account, the platform user can invoke all API services in the platform without redirecting to the third-party website. The platform users can filter and accept the services recommended by the platform, and also freely composite the services. This paper has below contributions:

(1) QMEC enable service users compose and invoke services in the mashup platform after a service provider provides the service mashup platform with the URL address of the service, the necessary service parameters and a service key.

978-1-6654-1485-2/20/$31.00 ©2020 IEEE
DOI 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00148

(2) Based on the service recommendation algorithm and service composition algorithm in academia, QMEC provides convenient function to service users.

(3) In addition to the image recognition, speech synthesis, intelligent translation and other services provided by the mashup platform itself, QMEC is also responsible for providing services such as the publish, display and promotion of APIs of other service providers, but does not provide an API running server, which improves system stability and scalability.

This paper has below innovations:

(1) A Blockchain-based mashup platform is proposed and implemented for MEC. A mechanism of heuristic-based container scheduling for service is proposed to support the mashup platform in MEC environment.

(2) Trusted data collected from the platform are collected on Blockchain, and a smart contract is proposed for assuring QoS and QoE.

II. Related Work

Researches on commercial and non-profit service orchestration platforms are discussed: OpenStack Heat, Windows Azure AppFabric / MarketPlace (including AppMarket), Amazon AWS Lambda and Google App Engine.

The combination of Windows Server and AppFabric [2] provides an easy-to-manage platform for developing, deploying, and reliably hosting middle-tier WCF/WF services. Malawski [3] have developed prototype workflow executor functions using AWS Lambda. Villamizar [4] presents a cost comparison of a web application developed and deployed using the same scalable scenarios by AWS Lambda. Abrahao [5] proposed a platform-independent monitoring middleware for cloud services. This middleware was implemented in both Microsoft Azure and Google App Engine to monitor the quality of cloud services. Using Google App Engine as a platform, Nishida [6] proposes a modeling and simulation based framework to predict the cloud performance. Basu [7] presents a performance case-study on implementing the building blocks of a privacy preserving collabo-rative filtering scheme in Java on the Google App Engine (GAE/J) cloud platform. Prodan [8] employs the Google App Engine (GAE) for high-performance parallel computing. Prodan [9] designed a generic master-slave framework that enables implementation and integration of new algorithms by instantiating one interface and two abstract classes.

Researches on service processes and Blockchains are discussed below. Business process models with business rules that represent business constraints have been studied widely. In order to restore business process model, hidden transition is discovered from process model [10]. An approach [11] on how to automatically generate dynamic business process simulation model is presented. The approach discovers belief network of the process from an event log and uses it to generate a simulation model automatically. Such model then can be further customized to facilitate analysis. Cho M [12] proposes a business process assessment framework focused on the process redesign lifecycle phase and tightly coupled with process mining as an operational framework to calculate indicators. For Business process model discovery, Sun [13] proposes a technique that is able to locate the process behaviors recorded in an event log which cannot be expressed by the Heuristics Miner and then transform them into expressible behaviors so that a high fitness model can be built. Jlailaty [14] uses an unsupervised mining technique accompanied by semantic similarity measurement methods. Two representative similarity measurement methods are examined: Latent Semantic Indexing and Word2vec. These methods are compared to prove that Word2vec provides a better performance than LSA in grouping emails. Nguyen [15] proposes a technique that, given an event log, discovers a stage decomposition that maximizes a measure of modularity borrowed from the field of social network analysis. The hidden knowledge [16] is considered as business rules. The business rules extracted from process models are significant and valid sequential correlations among business activities belonging to a particular organization. Bala [17] uses a mining technique to generate models that visualize the work history as GANTT charts. He formally defines the notion of a project-oriented business process and a corresponding mining algorithm. A relation mining method [18] is proposed for obtaining activity dependence relations. The formal description of these relations is defined in control flow perspective [19], which is expressed as serial-dependence relations and parallel-dependence relations in the form of three tuples after analyzing all the control flow patterns.

Blockchain technology enables the execution of collaborative business processes involving untrusted parties without requiring a central authority. Weber [20] proposes a Blockchain-based approach to support data accountability and provenance tracking. The approach relies on the use of publicly auditable contracts deployed in a Blockchain that increase the transparency with respect to the access and usage of data. Several Blockchain-based approaches providing security services are compared thoroughly [21]. Challenges associated with using Blockchain-based security services are also discussed to spur further research in this area. Mendling [22] analyzes Blockchain's impact on business process management (BPM), and provides research directions for investigating the application of Blockchain technology to BPM. An automated BPM solution is investigated [23] to select and compose services in open business environment, Blockchain technology is explored and proposed to transfer and verify the trustiness of businesses and partners. García [24] proposes an optimized method for executing business processes on top of commodity Blockchain technology. Schulte et al [25] identify the state of the art of elastic Business Process Management with a focus on infrastructural challenges. An architecture is conceptualize for an elastic Business Process Management System and discuss existing work on scheduling, resource allocation, monitoring, decentralized coordination, and state management for processes [26, 27].

Research and industrial products are investigated as mentioned in the related work, but they do not consider service scheduling mechanism that QMEC did.

III. Platform Architecture

To achieve high availability, QMEC is deployed in a distributed way. Service users on cloud can access services in QMEC easily anywhere anytime by any devise. Based on cloud computing, QMEC (Fig. 1) includes a distributed coordination service for horizontal scaling, multiple container clusters, multiple volumes, a central API server for service invocations, a customized container scheduler for MEC scheduling, and multiple Blockchain peers for QoS and QoE storage. Below is the deployment diagram.

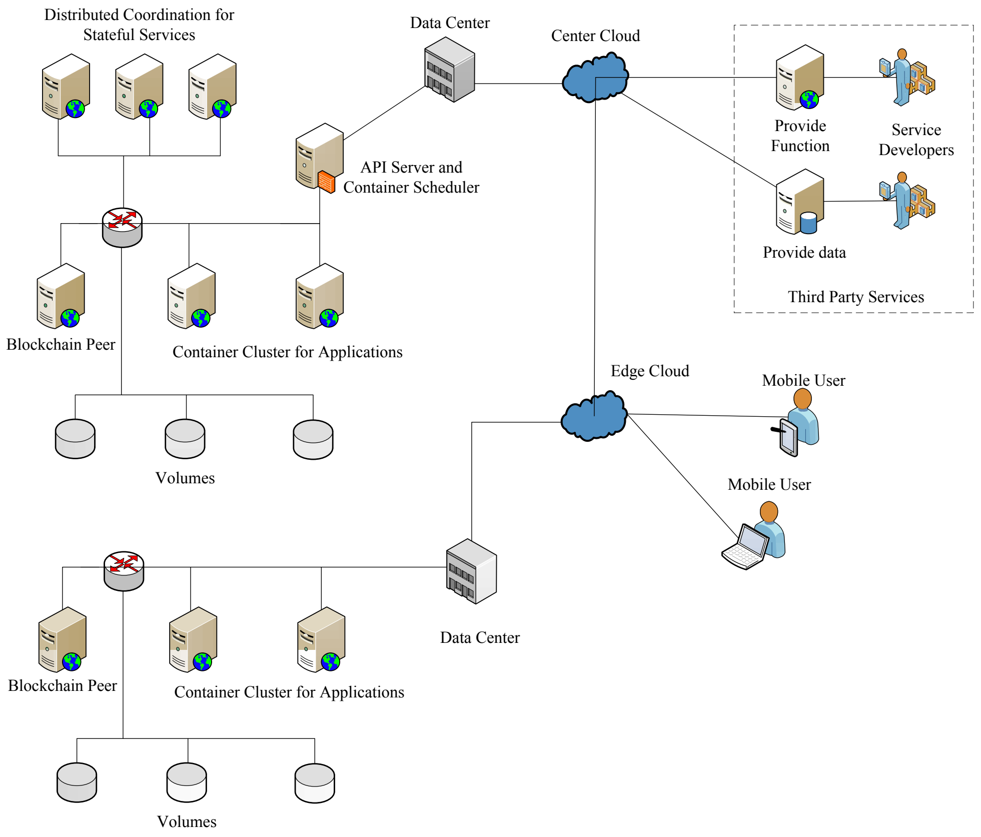

Fig. 1. Platform Architecture

In order to adapt to the mobile edge computing environment, our platform aggregates a large number of services, and these services should be adaptively put into virtual machines or containers to ensure SLA of services and enhance user experiences. An intelligent service scheduling mechanism is needed when a service with multiple identical instances is dynamically deployed to the vicinity of the user access network.

IV. Container Scheduling for Services

The main function of our container platform is to allocate computing, storage and network resources for containers. The container scheduling system is responsible for selecting the most suitable host to start containers and binding them. It must be able to automatically handle container failures and automatically restart more containers on suitable hosts to cope with more application access.

Container schedulers transfer containers to work nodes according to specific algorithms and strategies. In general, container schedulers can meet most of the requirements, such as scheduling containers to run on nodes with sufficient resources, or dispatching containers to different nodes to balance the workload of cluster nodes. However, in some special scenarios, default scheduling algorithms and strategies do not meet the actual needs, such as allowing users to schedule certain containers to specific hardware nodes on demand (deploying database services to SSD hard disk nodes, deploying CPUs and memory-intensive services to high-allocation CPUs and memory nodes) or deploying containers to the place where containers have frequent interactions with each other (such as the same machine, the same room, or the same network segment, etc.). QMEC implements a custom schedulers which is based on a heuristic scheduling algorithm by extending kubernetes, therefore we can place containers in edge Cloud.

A. Scheduler rules for resourses

Resources managed by QMEC scheduler can be divided into two attributes. One is compressible resources (for example, CPU, Disk I/O, and bandwidth), which can be limited and recycled, and the scheduler can reduce the use of these resources by a container without shutting down the container. Another is incompressible resources (such as memory, hard disk space), which are generally not recoverable without shutting down containers.

QMEC scheduler evaluates the task's need for future resources based on current resource consumptions. It will recycle resources to tasks that can tolerate low SLA (Service Level Agreement). There are two kinds of tasks, one is real-time task, and the other is non-real-time task. The real-time tasks never receive recycled resources by the scheduler. However, non-real-time tasks can use reservation resources and recycled resources. When there is no available resources in a host, the scheduler will firstly limit or terminate non-real-time tasks.

The scheduler determines which node a container should run on by below rules. First, mandatory rules are used to match the resources required by the container. If there is no host satisfies the requirement, then the container will be suspended until the host satisfies.

The mandatory rules available include followings:

(1) There are no port conflicts.

(2) There are enough resources to run containers.

(3) There are enough space for container-linked volumes.

(4) Nodes selected by the requirement is available.

If the scheduler finds that more than one host satisfies the above rules, the priority rules are used to determine which host is the most suitable for running the container.

The available priority rules include followings:

(1) CPU-inefficient and memory-inefficient nodes gain more values.

(2) Nodes with the most balanced utilization of resources (CPU, memory) gain more values.

(3) Nodes with different containers gain more values.

(4) Give all cluster nodes the same values.

(5) Nodes that already have container running environment gain more values.

(6) Affinity nodes gain more values.

(7) CPU-efficient and memory-efficient nodes gain more values so that when the cluster scales out, busy machines are closed.

Each rule above indicates a fitting value in QMEC heuristic algorithm, and this algorithm outputs a schedule plan for services in edge cloud. In order to limit the resources required by the container at runtime, the scheduler can set resource quotas and limits through YAML files to manage the resource requirements of all containers. When a task is submitted, the scheduler automatically deploys multiple service instances to the host based on system resources.

V. Blockchain for QoS and QoE

A. Blockchain for Composite Service

QMEC acts like a transportation hub of services, and it automatically obtains and records QoS (delay, availability, etc.) during execution of each service. Meanwhile, QoE (user ratings, comments, etc.) is easily obtained from service users who invoked the services. Information of QoS and QoE can also be observed by the service users in the platform

If data provenance existed for distributed and exchanged data, then detecting attacks and identifying the exact source of intrusions are achievable. However, the current data provenance (e.g. logs) in cloud does not address this challenge. Logs only provide a history of sequential actions, while the provenance data provides origins of all changes to a data object. Logs have two limitations: cost of provenance is high and lack of transparency.

We are using Blockchain technology to improve QMEC. Blockchain peers are integrated to record QoS and QoE, which is shown in Fig.1. Blockchain can enhance trustworthiness, provide data provenance, and decrease the cost of trusted central authority in QMEC. Peers (service providers) cooperate for a block chain for a composite service, where peers package transactions (results of invoking services) together into a block. An event-driven smart contract on a block chain is a programming code snippet, which nobody controls and therefore everybody can trust. The validating peers (core peers) execute the smart contract independently, and this execution includes the calling function, the parameters, and the address of the contract. Based on a consensus protocol, each peer reaches the consensus for the execution result. We show important concepts and QMEC framework of Blockchain-based data provenance below.

Transaction (refer to a service output in QMEC): A transaction is an item of a ledger. The owner of the transaction (a service provider) uses its public key to generate an address, and a private key to generate a corresponding signature. After that, the transaction is broadcast to the network and will be validated by every peer that receives the transaction.

Blockchain: Block is used to record a set of transactions, which consists of two parts: a header (hash, Merkle-tree root, timestamp, etc.) and a content (transaction details). Block Chain includes a number of blocks which are linked together, where each block is linked after the previous one and the link is called parent hash.

Consensus: When network latency exists, consensus protocol is a peer-to-peer protocol that is run by the peers to make sure each block in a correct sequence.

Smart Contracts: Smart contracts is an event-driven program which is called Chaincode in Hyperledger Fabric implementation. Chaincode defines the execution logic of a smart contract, and it can be read and executed in a virtual machine or a docker container in the peer's network. The execution of a chaincode is also called deployment (after a transaction was propagated, validated and committed, this transaction is confirmed, the chaincode will be stored on the Blockchain with the world state.). World state is a key-value database to store states of transactions and the information of each account balance.

Provenance Database: It is optional for off-chain mode. For every data object belonging to cloud users, Provenance Database records all provenance information. The Peers should validate and anonymize provenance information in the Blockchain to protect users' privacy.

Provenance Auditor: Provenance Auditor is used for detecting malicious behaviors and transfer all the provenance data from the Blockchain into the provenance database. Users are permitted to query for provenance records from Provenance Auditor so that users only keep the block headers and store detailed records in Provenance Database.

Based on our trusted and adaptive service discovery method, main steps of data provenance in a composite service is shown below:

(1) Trusted service discovery method is used for a composite services. All candidate services will run in order.

(2) When running candidate services, a smart contract that monitors untrusted services will be deployed in peers. If an untrusted service in a composite service is discovered, it will be substituted by the smart contract.

(3) During the execution of services, each output of the services will be recorded in a block chain so that others cannot tamper the output. Moreover, this data link can be analyzed to find out more association patterns of trusted services in future.

B. Smart Contract for Trusted data

In a scenario of a service user and a service provider, below figure presents a sequence diagram for storing trusted service data.

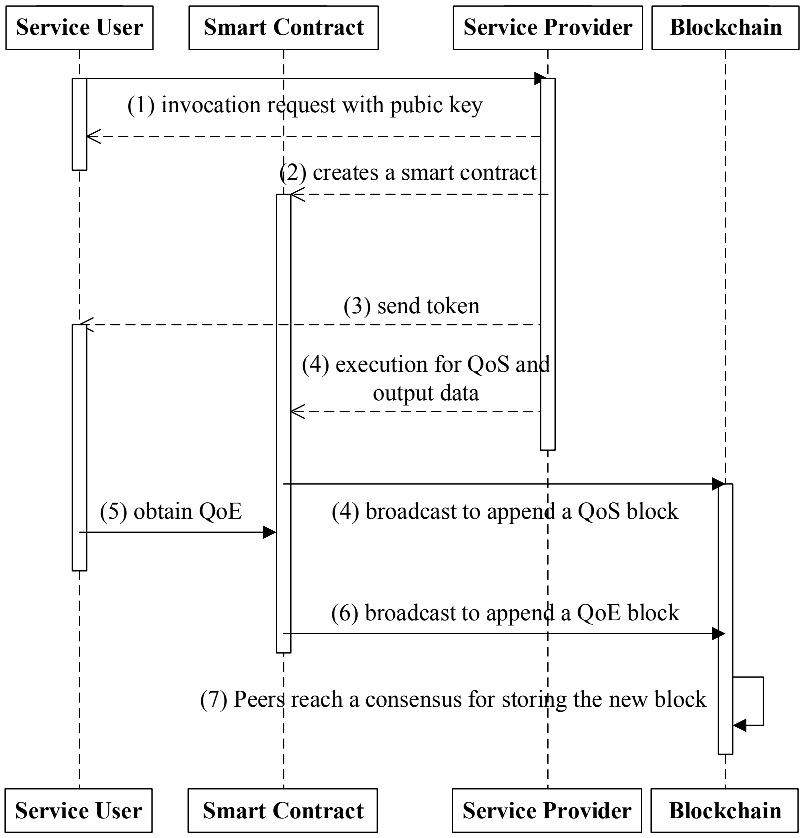
Service User
Smart Contract
Service Provider
Blockchain
(1) invocation request with pubic key
(2) creates a smart contract
(3) send token
(4) execution for QoS and output data
(4) broadcast to append a QoS block
(5) obtain QoE
(6) broadcast to append a QoE block
(7) Peers reach a consensus for storing the new block
Service User
Smart Contract
Service Provider
Blockchain

Fig. 3. Sequence diagram for storing trusted service data

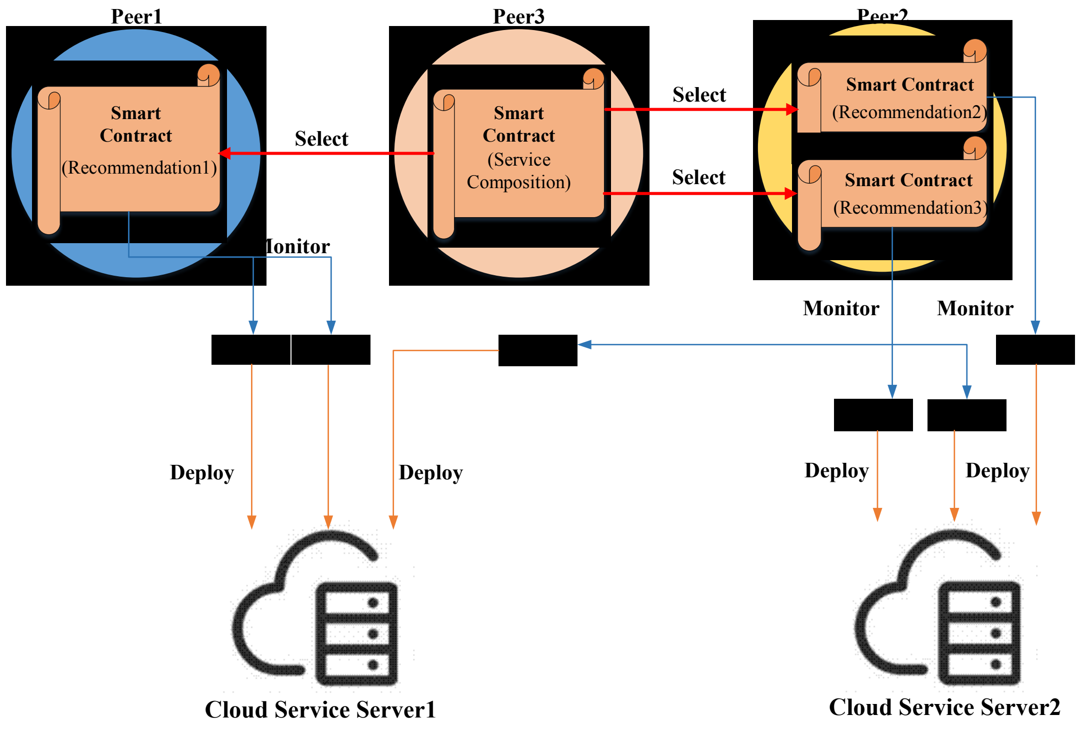
Peer1
Peer3
Peer2
Smart Contract (Recommendation1)
Smart Contract (Service Composition)
Smart Contract (Recommendation2)
Smart Contract (Recommendation3)
Select
Select
Select
Monitor
Monitor
Monitor
Deploy
Deploy
Deploy
Deploy
Cloud Service Server1
Cloud Service Server2

Fig. 4. Mechanism of Blockchain-based Service Composition and Recommendation

The detailed descriptions for storing trusted service data presented in above figure is shown below.

(1) A service user selects a service provider, and the service user sends its Sig_u (Public key of the service user) to the service provider.

(2) The service provider creates a smart contract for data provenance. The service provider generates a secret token to this contract to identify the service user in the following steps. The service provider uses the token to compute a hash: H = MD5 (token, Pub_u, Pub_p).

(3) The token is sent to the service user so that the service user verifies the correctness of a transaction. The token is used as a part of credentials to authenticate the service user.

(4) The service provider computes a digital signature, Sig_p = (Pub_u, Pub_p, H). Then, the provider instantiates the smart contract by executing the function Record_QoS() to output data and record service QoS. The result of this procedure is that a transaction is created and appended into the Blockchain. After that, a message that the consumer is invoking the service is broadcast to other peers.

(5) The service user can verify the block by Sig_p. To give QoE (quality of experience), user comments, and ratings to that service, the service user executes function Record_QoE(). After the execution, the service user sends a message containing token, Pub_p, H, QoE, and computes a signature, Sig_u = (token, Pub_p, H, QoE).

(6) The service user executes function Termination(). This function verifies that the transaction indexed by H in the Blockchain is the same with H computed by MD5 (token, Pub_u, Pub_p). Then it appends the transaction into Blockchain. Other peers can check its signature Sig_u.

For illustrating the relation between Blockchain related concept and service related concept, an architecture is shown in Fig.4. On the node of Peer1, a smart contract with the service recommendation function monitors the running status information of Service11 and Service12, which are deployed on one cloud server: Cloud Service Server1. On the node of Peer2, there are two smart contracts: one smart contract with the service recommendation function monitors the running status information of Service21 which is deployed on another cloud server: Cloud Service Server2. The other smart contract with service recommendation function monitors the running status information of Service31, Service32, and Service33. The Service31 and Service32 are both deployed on the Cloud Service Server2, while Service33 is deployed on the Cloud Service Server1. On the node of Peer3, the smart contract, which has the function of composing multiple services, is able to select the suitable services which are monitored and deployed by the smart contract on the node of Peer1 and Peer2.

Solo.io [28] developed Gloo, a high-performance, platform-agnostic function gateway built on Envoy proxy (a high-availability and load-balance proxy). Gloo is able to integrate legacy API from monolithic applications, micro-service API, and serverless functions. In addition, they developed an open sourcing Qloo which allows user query data without any coding to existing applications. Qloo is based on a GraphQL and Gloo. GraphQL is a data query language capable for multi-invocations in one query. However, Gloo does not add Blockchain to enhance service data security.

VI. CONCLUSION

Existing PaaS platforms are investigated. We established a Blockchain-based service platform for QoS and QoE in mobile edge cloud ecosystem. The platform user can invoke all API services in the platform without redirecting to the third-party website. The platform users can filter and accept the services recommended by the platform, and also freely compose the services which are Real-world services. In addition, data collected from the platform are stored on Blockchain, and smart contract is proposed for assuring the QoS and QoE.

Kubernetes and Docker swarm have been studied to extend our adaptive platform. In future, QMEC platform will be able to automatic binpacking (automatically places containers based on service resource requirements), self-healing (restarts failed containers, replaces and reschedules containers), and horizontal scaling (scale services up and down).

ACKNOWLEDGMENTS

This work was supported by grants from NSFC under Grant (No. 61962040).

REFERENCES

[1] Couto, Rodrigo S; Sadok, Hugo; Cruz, Pedro; etc. Building an IaaS cloud with droplets: a collaborative experience with OpenStack. Journal of Network and Computer Applications, 2018(117): 59-71

[2] Kaufman, Stephen; Garber, Danny. Pro Windows Server AppFabric, Stephen Kaufman and Danny Garber.Pro windows server AppFabric, 2010:1-310

[3] Malawski, Maciej; Gajek, Adam; Zima, Adam. etc.Serverless execution of scientific workflows:Experiments with HyperFlow, AWS Lambda and Google Cloud Functions. Future Generation Computer Systems, 2017.

[4] Villamizar, Mario; Garc&eacute;s, Oscar; Ochoa, Lina.Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures. Service Oriented Computing and Applications, 2017(11):233-247

[5] Abrahao, Silvia; Insfran, Emilio.Models@runtime for Monitoring Cloud Services in Google App Engine. IEEE 13th World Congress on Services, 2017: 30-35

[6] Nishida, Sachi ; Shinkawa, Yoshiyuki.A Performance Prediction Model for Google App Engine. 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2015: 134-140

[7] Basu, Anirban ; Vaidya, Jaideep; Dimitrakos, Theo.Feasibility of a privacy preserving collaborative filtering scheme on the Google App Engine - A performance case study. 27th Annual ACM Symposium on Applied Computing, 2012: 447-452

[8] Prodan, Radu; Sperk, Michael; Ostermann.Evaluating high-performance computing on google app engine. IEEE Software, 2012(29): 52-58

[9] Prodan, Radu ; Sperk, Michael.Scientific computing with Google App Engine. Future Generation Computer Systems, 2013(29): 1851-1859

[10] Fang, Xianwen, et al. A Method of Mining Hidden Transition of Business Process Based on Region. IEEE Access, 2018 (6): 25543-25550.

[11] Savickas T, Vasilecas O. An approach to business process simulation using mined probabilistic models. Computer Science & Information Systems, 2017:45-45.

[12] Cho M, Song M, Comuzzi M, et al. Evaluating the effect of best practices for business process redesign: An evidence-based approach based on process mining techniques. Decision Support Systems, 2017(104): 92-103.

[13] Sun Y, Bauer B. A Novel Heuristic Method for Improving the Fitness of Mined Business Process Models. Service-Oriented Computing, 2016:537-546.

[14] D, Grigori D, Belhajjame K. Mining Business Process Activities from Email Logs. IEEE International Conference on Cognitive Computing, 2017:112-119.

[15] Nguyen H, Dumas M, Hofstede A H M T, et al. Mining business process stages from event logs. 29th International Conference on Advanced Information Systems Engineering (CAiSE), 2017:12-16.

[16] Polpinij J, Ghose A, Dam H K. Mining business rules from business process model repositories. Business Process Management Journal, 2015, 21(4):1367-1372.

[17] Bala S, Cabanillas C, Mendling J, et al. Mining Project-Oriented Business Processes. International Conference on Business Process Management. Springer International Publishing, 2015:425-440.

[18] Hu G, Wu B, Chen J. The Mining of Activity Dependence Relation Based on Business Process Models. IEEE International Conference on Services Computing. IEEE Computer Society, 2017:450-458.

[19] Martin N, Depaire B, Caris A. The Use of Process Mining in Business Process Simulation Model Construction. Business & Information Systems Engineering, 2016, 58(1):1-15.

[20] Weber, Ingo, et al. Untrusted Business Process Monitoring and Execution Using Blockchain. International Conference on Business Process Management Springer International Publishing, 2016:329-347.

[21] Salman, Tara, et al. Security Services Using Blockchains: A State of the Art Survey. IEEE Communications Surveys & Tutorials, 2018:1-12.

[22] Mendling, Jan, et al. Blockchains for Business Process Management - Challenges and Opportunities. ACM Transactions on Management Information Systems. 2018(9):1.

[23] Viryasitavat, Wattana, et al. Blockchain-based business process management (BPM) framework for service composition in industry 4.0. Journal of Intelligent Manufacturing, 2018:1-12.

[24] García-Bañuelos, Luciano, et al. Optimized Execution of Business Processes on Blockchain. Business Process Management. 2017.

[25] Schulte, Stefan, et al. Elastic Business Process Management: State of the art and open challenges for BPM in the cloud. Future Generation Computer Systems, 2014 (46):36-50.

[26] Neisse, Ricardo, G. Steri, and I. Nai-Fovino. A Blockchain-based Approach for Data Accountability and Provenance Tracking. ACM International Conference on Availability, Reliability and Security, 2017:14.

[27] Ramachandran, Aravind, and M. Kantarcioglu. Using Blockchain and smart contracts for secure data provenance management. 2017.

[28] https://gloo.solo.io