2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW) | 978-1-6654-8879-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDCSW56584.2022.00012

A Blockchain-based Profiling System for Exploring Human Factors in Cloud-Edge-End Orchestration

Minghao Li and Wei Cai*
School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China
email: minghaoli1@link.cuhk.edu.cn, caiwei@cuhk.edu.cn

***Abstract*—In mobile edge computing (MEC), application partitioning is one of the most effective measures to leverage computing resources. Due to the user's unpredictable behavior pattern, which is an indispensable factor affecting the performance of an offloading system, traditional partitioning algorithms, considering only purely technical QoS, are no longer enough to meet the increasing concern for the user experience of mobile applications. In this paper, in order to explore human factors in modeling partitioning algorithms for cloud-edge-end orchestration under a safe and trusted environment, we present a blockchain-based profiling system to collect behavioral data from several invited subjects. For discovering user-driven relations of method-level components, we propose a clustering algorithm framework to process each subject's data. Based on the disparate results, we illustrate a case study to prove the usefulness of the system and the data for the orchestration by analyzing the variance of user behavior and the feasibility of applying human factors to the partitioning algorithm.**

***Index Terms*—Mobile Edge Computing, Blockchain, Partitioning Algorithm, Orchestration, Reverse Engineering**

I. Introduction

Nowadays, mobile multimedia applications, such as intelligent editing of mobile videos, intelligent auditing of live broadcasts on mobile, and mobile games, often require assistance from cloud computing in terms of data and computing resources [1]. However, using Mobile Cloud Computing (MCC) often implies unavoidable network latency limitations. As an extension to MCC, mobile edge computing (MEC) utilizes hardware facilities at the edge of the network and minimizes the response latency perceived by mobile devices (MDs) by avoiding backbone network communication as much as possible [2]. However, migrating the entire computing task to edge servers is not necessary and effective due to its far less resourcefulness than cloud servers'. For some complex mobile applications, a cloud-edge-end orchestration is required to make partitioning strategies that determine which components are going to be offloaded.

Current partition algorithms are mainly based on the call graph of the program, which means that program execution can naturally be described as a graph. Some orchestrations utilize algorithms, like [3], that incorporate several external factors into the modeling of the program. However, the current orchestrations only focus on pure technical Quality of Service (QoS) and lack consideration of the impact of different user interaction habits on the use of components in the actual operation of programs, and cannot work for the Quality of Experience (QoE), which is undoubtedly insufficient for current mobile applications that value user experience [4]. Therefore, for a high-performance offloading system, human factors play indispensable roles in constructing cloud-edge-end orchestration.

The acquisition and maintenance of user data are fundamental to the study of human factors in partitioning algorithms. For instance, by collecting in-program data from MDs, like [5], we can get the precise fingerprint of each user. However, the collecting and storing of personal information by centralized organizations may risk users' privacy since there is little transparency regarding what studies do with this data [6]. Even if storing private data in centralized databases managed by trusted individuals or research teams is not secure, since there are a bunch of injection attacks for both SQL [7] and NoSQL [8] databases. In order for subjects to be completely free from the fear of having their behavioral data exposed, we need to provide a secure and controlled database.

For purpose of transmitting private data, a reliable and secure network is required for delivery. Though some popular Internet services, like Google, Apple, and Microsoft, enable subjects to share their files with the experimenter remotely, subjects may be worried about the invasion of their privacy. Centralized servers are one of the main tools used by these Internet giants, which have been under the surveillance of organizations and governments for a period of time [9]. In other words, a decentralized and distributed public network service is needed for subjects and experimenters trading private behavioral data.

In order to address the above issues, we construct a blockchain-based profiling system for exploring human factors in the partitioning algorithm of cloud-edge-end orchestration, which provides subjects with security and privacy. Blockchain-based properties allow subjects to securely and selectively trade their data to experimenters. Besides the blockchain, we adopt IPFS [10] as our database for storage, which enables subjects to take control of their own behavioral data without manipulation and hacking. In this paper, we first introduce the underlying techniques of the system, after beginning with related work. Then, we describe the architecture of the profiling system and its implementation. Moreover, to validate the system and the data, we present the description of the subjects' data and an approach for clustering. Finally, by analyzing the variance of user behavior and the feasibility of

*corresponding author

2332-5666/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDCSW56584.2022.00012

applying human factors to partition algorithms in cloud-edge-end orchestration, we prove the usefulness of the profiling system and the collected data.

II. Related Work

A. Partition Algorithms of Cloud-Edge-End Orchestrations

Effective orchestrations working for collaborative regulation are significant for the exponential growth in the pressure on cloud and edge servers. Modeling applications and specifying the partitioning algorithm is the cornerstone of this kind of orchestration. There are many researchers engaging in the field of application partitioning in MEC. A framework in [11] was designed for runtime computation repartitioning in dynamic mobile cloud environments to solve the performance degradation issue arising from the dynamic network and device status. Venus Haghighi *et al.* [12] modeled offloading strategy via a mathematical graph where both Wi-Fi and 3G links are topics of concern. A study [13] on how to dynamically partition a given application effectively, considered a wide range of factors, including response time, energy consumption, and other types of consumption. However, current studies in application partitioning and related industrial orchestration do not take into account the human impact on the offloading system. There is no suitable cloud-edge-end orchestration that aims at applying partitioning algorithms with human factors, which can actually improve the QoE of users.

B. Application Analyzers

In the research of application partitioning, it is necessary to decompose the program into numerous components and analyze their relations. There are two mainstream and widely varying program analysis schemes, one for static analysis of the code itself, and the other for dynamic analysis of application performance running on devices. For static code analysis, the mainstream solution is to retrieve the call graph. Soot[1] is a Java bytecode analysis tool, which provides a variety of bytecode analysis and transformation functions. FlowDroid is a context-sensitive, flow-sensitive, domain-sensitive static stain analysis tool developed by Soot and Heros[2]. Those tools are is mainly based on static bytecode call relationships, without considering the impacts of user behavior. For dynamic performance analysis, the Android Profiler is one of the most powerful analyzers, which provides real-time data to visualize how an application uses CPU, memory, network, and other resources. Besides, Appetizer[3] is able to collect comprehensive data during the app runtime, including crashes, uncaught exceptions, lengthy operations, etc., by instrumenting Android package (APK) binary. Though the Android Profiler is professional and Appetizer is capable of remote profiling, there is no analyzer that can reflect user behavior patterns based on the sequence of component states, let alone being able to guarantee the security, autonomy, and ownership of the gathered user data.

[1]https://github.com/soot-oss/soot
[2]http://sable.github.io/heros/
[3]https://www.appetizer.io/

III. Architecture and Implementation

A. System Overview and Techniques

Since the system needs to both protect the subjects' private data from disclosure and satisfy the experimenter's need to incorporate human factors into the partitioning algorithm and construct an orchestration for QoE improvement. The architecture design of the system is divided into two parts, one is the blockchain module, which is designed to provide secure and private communication. The other part is the profiling module, which is deployed to retrieve behavioral data from running APKs. The architecture design of the blockchain-based profiling system is shown in Fig. 1.

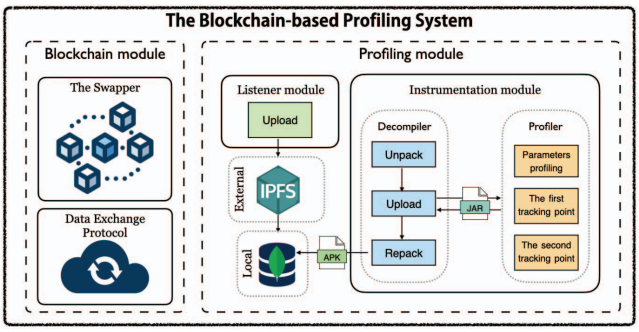


Fig. 1. System Architecture

1) Profiling module: In order to add user behavior impact factors to the software modeling in the future partitioning algorithm, it is necessary to collect massive state data of the computation components when users use the app. Thus, the system has the following features that meet the requirements: 1. applicable to method-level computing components; 2. able to instrument a variety of real mobile apps; 3. capable of remotely connecting multiple devices at the same time; 4. easy for users to upload their own apps for profiling.

In order to get the activity of the method-level computing components in the java layer of APKs, the profiling system utilizes Javassist[4] and Apktool[5] for Java bytecode manipulation and APK reverse engineering. The instrumentation module includes two sub-modules, which are the Decompiler and the Profiler. For the Decompiler, it can be divided into the Unpacking unit, Uploading unit, and Repacking unit. The Unpacking unit utilizes the aforementioned tools to extract the Java Archive (JAR) file. Then the Uploading unit sends the JAR file to the Profiler for instrumentation. The processed JAR file will be received and repacked as an executable APK by the Repacking unit. Finally, the Repacking unit stores the address of the processed APK in Local MongoDB for Subjects to reinstall. For the Profiler, after receiving the pending JAR file, through Javassist, three units of the Profiler will be inserted into the JAR, which will execute in the manipulated APK. Those three units are the Parameters Fetcher, the First-tracking Spot, and the Second-tracking Spot. The Parameters

[4]https://www.javassist.org/
[5]https://ibotpeaches.github.io/Apktool/

Fetcher is used for obtaining device information, such as brand, system version, memory, CPU frequency, etc. When the user starts to call one of the instrumented methods, the First-tracking Spot will firstly be activated and record the current timestamp. At the end of the life cycle of the instrumented component, the Second-tracking Spot will be executed to get method parameters and calculate the execution time. Through the Profiler, the user device will automatically send the above data to the Listener module, when the manipulated application is running. The Listener module contains an uploading unit, which automatically uploads the received pieces of component call data at the end of the subject's operation and then gets a returned hash value. The hash value will be stored in local MongoDB for swapping to experimenters.

2) Blockchain module: In order that subjects can be protected from data leakage, it is necessary to build a secure and private web service. Also for faster and easier data transition, the blockchain module needs to provide the following features to the system: 1. Capable of delivering a large volume of data files; 2. Not permitted for subjects to modify the uploaded data; 3. All transaction records are publicly available; 4. Only the participants of the transaction can decrypt the content of the data.

For the first two requirements, the InterPlanetary File System (IPFS) is an ideal option to provide a related high-bandwidth and permanent database. IPFS is a peer-to-peer distributed file system, where a large volume of subjects' behavioral data can be stored efficiently. With the distributed feature, IPFS protects subjects' private behavioral data from being attacked by hackers. Moreover, the uploaded behavioral data in IPFS cannot be falsified, since it is content-addressed. With the hash value returned after uploading, subjects can retrieve the original data file at any time. The other two requirements can be met by blockchain technology. The blockchain here provides a decentralized network for use as a publicly distributed ledger, where nodes (subjects and experimenters) collectively adhere to a protocol for sharing private behavioral data. Without going through a trading system with intermediaries and transmission software with centralized servers, the access of nodes in the blockchain to broadcast messages can be easily controlled by different encryption schemes.

Supported by the secure, stable, and distributed IPFS and blockchain technology, the blockchain module includes two parts, the Swapper and Data Exchange protocol. The profiling system utilizes blockchain's broadcasting mechanism to build the Swapper, where the experimenter needs to clarify their public key generated by RSA, target devices' UID, and a regular expression for data filtering, including time, name and arguments of methods, and a minimum amount of its volume. Both the applications of experimenters and the acceptance of subjects are broadcast to each other in unencrypted JSON string via the Swapper. Under this circumstance, all nodes can know how many times experimenters have asked which subjects for data and each subject's willingness. Therefore, the transparency for subjects and convenience for experimenters of the system is guaranteed. The Data Exchange Protocol is a mechanism for the transmission of large data from subjects to experimenters. The basic idea is therefore to encrypt the address hash of the data file returned by IPFS with the public key and then broadcast it in the blockchain via this protocol. Since the experimenter holds the private key, the experimenter can get the address of the subjects' data files. Through the list of addresses, the experimenter is able to download data files from IPFS. In this way, blockchain, which is supposed to be inefficient in transmission, can still handle huge amounts of behavioral data to experimenters safely and quickly.

Through the blockchain module, subjects deliver untampered behavioral data to experimenters, which will be used for incorporating human factors in the establishment of cloud-edge-end orchestration.

B. Logic Flow

The logic flow of subjects and experimenters, as shown in Fig. 2, describes how the system works at each step when the experimenter obtains data from the subject. Through this process, the subjects safely collect and share their own behavioral data, and the experimenters can also receive the experimental data remotely for the modeling of the partitioning algorithm and the construction of the orchestration for cloud-edge-end.

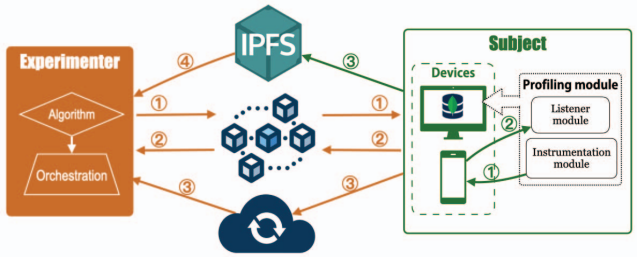


Fig. 2. Flow chart of the blockchain-based profiling system

The flowchart will be interpreted in two parts, which are related to the experimenter and the subject, respectively. The green flow representing the subject is described firstly as follows: 1. Subjects use the Instrumentation module to add the Profiler in Fig. 1 to the target APK and install the processed version of APK in the personal MD; 2. The Listener module continuously forces MD to send component data information to the local server in the personal computer; 3. The local server uploads the collected data to IPFS and stores the returned file addresses in the local MongoDB.

Then the interpretations of the experimenter's orange flow are presented following: 1. When the experimenter wants to obtain the subject's behavioral data, the experimenter needs to create a new data application, which includes the subjects' UID, minimum data volume, and period range. Then, through the Swapper in the blockchain module, this application broadcasts to all participants of the blockchain. All the experimenter can do is wait for responses from the target subjects; 2. Subjects send a response to the application via the Swapper to

the experimenter; 3. If the subject accepts the application, then all the subject's eligible data file addresses will be sent to the experimenter through the Data Exchange Protocol, otherwise the application is closed; 4. The experimenter pulls the files from IPFS with the obtained addresses.

After the experimenter has obtained the required data, the construction of partitioning algorithms of the orchestration can be launched soon.

IV. Data Collection and Analysis

A. Data Collection

We invited seven volunteers as our subjects and collected their behavioral data from four of them. Specifically, the behavioral data is made up of the component data that was called. We decomposed the APK into, components, which are the methods of the java layer in the APK. Where the component data contains, the name of the method, the data type of the method, the arguments of the method, the state type, and the timestamp of the method lifecycle. The component data can be interpreted as when a method starts or ends.

We want to obtain the relationship between different components through subjects' sequential data and develop a partitioning algorithm to build a high-performance cloud-edge-end orchestration.

B. Clustering Algorithm

In this section, in order to classify components in the mobile application, we proposed a clustering algorithm, with its framework shown in Fig. 3 , which consists of the following three steps: 1. Defining the similarity function between the two components based on the transition probability and dependencies; 2. Applying node2vec algorithm to generate component vectors; 3. Using the DBSCAN algorithm to cluster the generated component vectors.

By adjusting parameters to a suitable value, we obtain the clustering result, as shown in Fig. 4.

1) Similarity Function Definition: Since we want to migrate components of close execution to the same edge server to improve QoE, we need to define the similarity between components based on the transition probability between component states. By calculating the similarity matrix, we can migrate the components with high similarity together.

By aggregating all the component states in a program we can get a collection of component states

$$S = \{S_i | i = 1, 2, \ldots, 2n\}$$

, where $S_{\text{start}} = \{S_i | i = 1, 2, \ldots, n\}$represents the start state of all components in the program, and $S_{\text{end}} = \{S_i | i = n+1, n+2, \ldots, 2n\}$represents the end state of all components in the program. From sequence of random variables $X = X_0, X_1, \ldots, X_t, \ldots, t = 0, 1, 2, \ldots$,the stochastic matrix can be calculated as

$$\mathrm{P}_u = \begin{bmatrix} p_{11} & \cdots & p_{1n} & p_{1,n+1} & \cdots & p_{1,2n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} & p_{n,n+1} & \cdots & p_{n,2n} \\ p_{n+1,1} & \cdots & p_{n+1,n} & p_{n+1,n+1} & \cdots & p_{n+1,2n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ p_{2n,1} & \cdots & p_{2n,n} & p_{2n,n+1} & \cdots & p_{2n,2n} \end{bmatrix}$$

Thus, the similarity function is defined as

$$w_{ij} = \begin{cases} 1, & \text{if i and j are dependent or identical} \\ \frac{p_{ij}+p_{i,j+n}+p_{i+n,j}+p_{i+n,j+n}}{2}, & \text{else} \end{cases}$$

2) Graph Embedding: We use node2vec [14] for unsupervised clustering of the components, an improved version of DeepWalk [15], which defines a policy generation sequence of bias random-walk and still uses skip-gram to train. After the nodes of the weighted Graph are converted into high-dimensional vectors through node2vec and it will be visualized into 2-dimensional by t-SNE. Then, components can be clustered by a density clustering algorithm in the next step. One of the important parameters of node2vec is $walk\_length$.

3) Density-based Clustering: We decide to use one of the most classic density clustering algorithms, DBSCAN [16], as its parameter is easy for a domain expert to set. Those two hyperparameters are ε that represents the neighborhood radius of defined density, and $minPts$ is defined as the neighborhood density threshold of clustering.

C. Clustering Result

In order to explore the influence of user operating habits on the clustering results, we selected two subjects', Subject A (UID:448583767ECDF7052DA8A22DCB64101C) and Subject B (UID:59423D91ED4D72F30CF7B376C9CB0B05), component data of an same video editing commercial app, Androvid, as examples. The components were clustered and analyzed by the aforementioned algorithm process. The clustering results are shown in Fig. 4(a) and Fig. 4(b), where node color represents category.

V. Case Study

In order to validate the profiling system for exploring human factors in partition algorithm and cloud-edge-end orchestration in offloading systems, we first analyze the variance of user behavior. Based on the quantification results, we further interpret the feasibility of applying human factors.

A. Variance of User Behavior

In Fig. 4(a) and Fig. 4(b), clustering results of component data vary greatly among different users when using the same app with the same parameters ($\varepsilon = 1.5$, $walk\_length = 4$). This difference is divided into two categories, one difference is the component set and one difference is the component state sequence feature.

We use Jaccard similarity to quantify the difference between these two users. We get the result that the Jaccard similarity

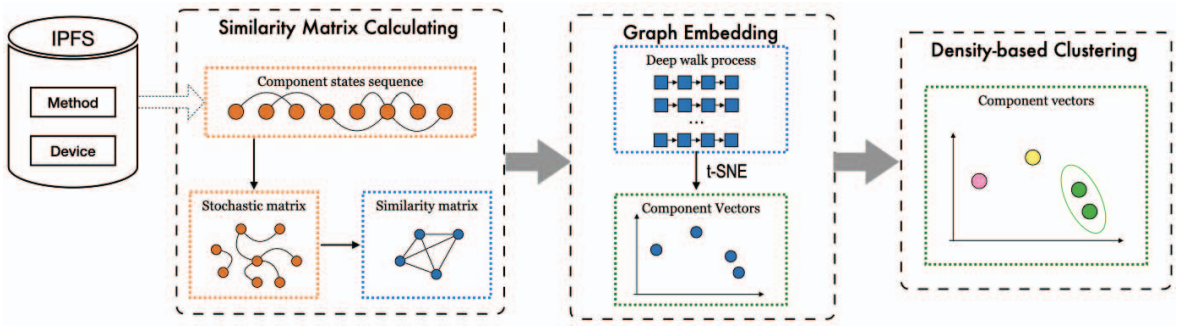
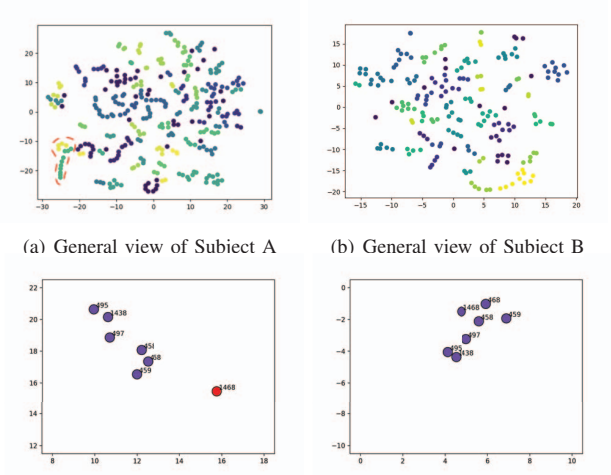


Fig. 3. Framework of the Clustering Algorithm



(a) General view of Subject A (b) General view of Subject B

(c) One cluster of Subject A (d) One cluster of Subject B

Fig. 4. Clustering Result

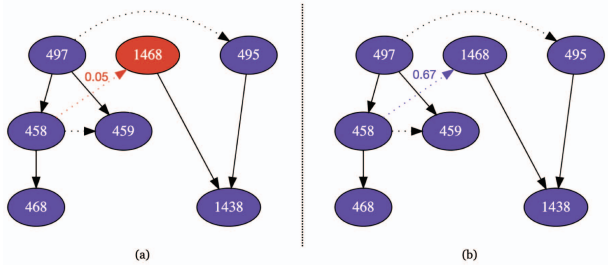


Fig. 5. Fig. 5(a) and Fig. 5(b) are the corresponding Markov Chains and Call Graphs in Fig. 4(c) and Fig. 4(d). The dotted line represents Markov Chain, and the solid line represents Call Graph generated by Flowdroid and Graphviz.

between the set of components invoked by Subject A and Subject B components is only 0.43. This represents a very large difference between the set of components of the two users. In particular, the method clustering community of the class `VideoCropActivity` is considerable in the clustering result of Subject A, illustrated by the red circle in Fig. 4(a), but this class does not appear in Subject B's dataset. A large part of the clustering results are different because subjects have different preferences for features, so it is easy to infer that Subject A likes to use Androvid for video crop while Subject B never uses it for cropping propose.

In addition, the comparison between several similar groups of the two users is shown in Fig. 4(c) and Fig. 4(d). The red dot represents `method(1468)`, which is not classified in the purple cluster in Subject A's result. With the same algorithmic treatment of the same parameters, there are still distinctions in the composition of the members of these similar groups and the distance. Except for the subtle changes due to the randomness of t-SNE, this kind of difference arises mainly because the transition probability between component states varies widely under different users' operations, and since the component similarity is defined by this transition probability, the clustering results will also be influenced.

Then, by checking the transition probabilities related to the start and end states of `method(458)` in the datasets of Subject A and Subject B, the main distinction is that Subject B is more likely to call `method(1468)` after calling `method(458)`, which leads to different clustering results for both. Fig. 5 illustrates that under Subject A's operation, the transition probability from the end state of `method(458)` to the beginning state of `method(1468)` is 0.05, which is much lower than 0.67 for Subject B. Thus, even though the call graph of Androvid used by both Subject A and Subject B are identical, the differences in Markov transition probability will ultimately affect the relationship between the two components.

The above results demonstrate that the sequence of component states caused by the user's operation habit also has a great influence on the component clustering. This variance between the two subjects' behaviors proves that the human factor is a major influence in the partitioning algorithm, which has to be considered while framing the orchestration in MEC scenarios.

B. Feasibility of Applying human factors to Partitioning Algorithm

The above discussion of the variance of the component set and the component state sequence feature, allows us to model the vertices and edges of the program structure graph for each user. From the above preferences of Subject A and

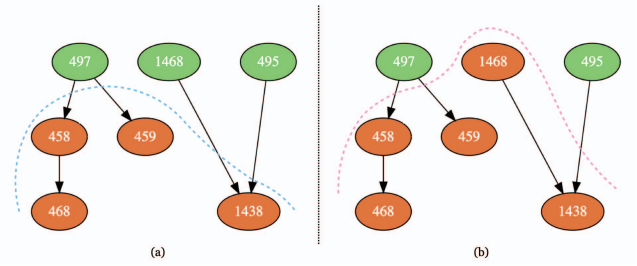


Fig. 6. Fig. 6(a) and Fig. 6(b) are possible partitioning strategies provided by the future orchestration for Subject A and Subject B, respectively.

Subject B for using different functions in the app, we can know that the frequency of calling methods varies from user to user. Therefore, when designing the partitioning algorithm, in addition to incorporating the component complexity, the content size of the component return value, and other factors into the modeling of the vertices, we can also incorporate the user's preference level for the component into the process of modeling the points. For example, if Subject B has never called the methods of `VideoCropActivity`, then it can be excluded as a migration target, even if it is determined to be in need of migration by the algorithms described in [12], [13].

In the process of modeling the edges, in addition to data dependencies, the order of execution of the components needs to be considered. When the orchestration formulates the partitioning algorithm, the distance of different components can be calculated as one of the modeling weights of the edges based on the results of the user's components after the embedding process. If two components are often executed together in succession although there is no call relationship between them, then migrating these two components to one edge server in a unified manner will eliminate the transmission consumption and network latency. Thus, in the future cloud-edge-end orchestration, possible offloading strategies for Subject A and Subject B are distinct, as shown in Fig. 6, where green components are decided to be executed locally and orange components are going to be offloaded to edge or cloud. In Fig. 6(b), the pink dotted line represents the offloading solution of Subject B, which decides to offload `method(1468)` with the other three orange components to the same edge server. On the contrary, suggested as in the blue dotted line Fig. 6(a), the orchestration determines to execute `method(1468)` in the local environment of Subject A's MD and only offload the orange components to edge/cloud servers. Thus, the orchestration can theoretically achieve QoE enhancement by developing adaptive partitioning algorithms for making variable offloading decisions for users.

VI. Conclusions

In this paper, we constructed a system and invited several subjects to use it with us, thus participating in the data collection effort. The data collected by the system allows us to know the gaps in various habits of using the application, which leads to different distances between components via the aforementioned clustering process. The components are no longer in a call relationship but have potential human-based associations. Through the associations, we further explored the feasibility of adding human factors' weight to the partitioning algorithm of a cloud-edge-end orchestration.

However, the user's preference is not constant. The time-sensitive feature, which we didn't take into account in this paper, is crucial for orchestration to provide a desirable QoE for users. Therefore, in future studies, we plan to deploy forgetting factors to our human-centered cloud-edge-end orchestration for achieving higher QoE.

Acknowledgment

This paper is funded by Project 6190070637 from the National Natural Science Foundation of China (NFSC) Young Scholars Program.

References

[1] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V. C. Leung, and C.-H. Hsu, "The future of cloud gaming [point of view]," *Proceedings of the IEEE*, vol. 104, no. 4, pp. 687–691, 2016.

[2] K. Bilal and A. Erbad, "Edge computing for interactive media and video streaming," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2017, pp. 68–73.

[3] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, 2010, pp. 49–62.

[4] W. Cai, Z. Hong, X. Wang, H. C. Chan, and V. C. Leung, "Quality-of-experience optimization for a cloud gaming system with ad hoc cloudlet assistance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 2092–2104, 2015.

[5] Z. Tu, R. Li, Y. Li, G. Wang, D. Wu, P. Hui, L. Su, and D. Jin, "Your apps give you away: distinguishing mobile users by their app usage fingerprints," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–23, 2018.

[6] A. Ghosh and A. Roth, "Selling privacy at auction," in *Proceedings of the 12th ACM conference on Electronic commerce*, 2011, pp. 199–208.

[7] R. Dorai and V. Kannan, "Sql injection-database attack revolution and prevention," *J. Int'l Com. L. & Tech.*, vol. 6, p. 224, 2011.

[8] J. Kumar and V. Garg, "Security analysis of unstructured data in nosql mongodb database," in *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*. IEEE, 2017, pp. 300–305.

[9] G. Greenwald and E. MacAskill, "Nsa prism program taps into user data of apple, google and others," *The Guardian*, vol. 7, no. 6, pp. 1–43, 2013.

[10] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[11] L. Yang, J. Cao, S. Tang, D. Han, and N. Suri, "Run time application repartitioning in dynamic mobile cloud environments," *IEEE Transactions on Cloud Computing*, vol. 4, no. 3, pp. 336–348, 2014.

[12] V. Haghighi and N. S. Moayedian, "An offloading strategy in mobile cloud computing considering energy and delay constraints," *IEEE Access*, vol. 6, pp. 11 849–11 861, 2018.

[13] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[14] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[15] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[16] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.