Real-time Multi-resource Allocation via a Structured Policy Table

Arslan Qadeer, Myung J. Lee and Kazuya Tsukamoto

Abstract Mobile Edge Cloud endures limited computational resources as compared to back-end cloud. Semi-Markov Decision Process (SMDP) based Multi-Resource Allocation (MRA) work [6] introduces optimal resource allocation for mobile requests in the resource constrained edge cloud environments. In this study, we scale existing SMDP MRA work for real-world scenarios. First, we structure the policy tables in a two dimensional matrix such that columns represent states of the system and rows for the actions. Second, we propose an index based search technique over structured policy tables. Simulation results demonstrate that our approach outperforms the legacy method and retrieves an optimal action from the policy tables in the order of microseconds, which meets the delay criteria of real-time applications in edge cloud based systems.

1 Introduction

With the ever-increasing need of IT infrastructure for computing and storage in tremendously growing mobile communications technologies, the emerging Internet of Things (IoT) and Machine Type Communication (MTC) are expected to instigate a huge number of device connections, which undergo low storage capacity, high energy consumption, low bandwidth and high latency [1]. Mobile Edge Cloud (MEC), leveraging the traits of both cloud computing and mobile computing, has provided considerable capabilities to mobile devices to alleviate the above inherent network limitations [2, 3].

As compared to back-end clouds, where unlimited pools of computational and storage resources are provided, edge clouds are supposed to posses limited amount of resources [4]. Therefore, efficient resource allocation techniques are required to improve the overall

Arslan Qadeer, Myung J. Lee
A. Qadeer and M.J. Lee, Dept. of EE, City College of CUNY, New York, USA, e-mail: aqadeer000@citymail.cuny.edu, mlee@ccny.cuny.edu

Kazuya Tsukamoto
K. Tsukamoto, Dept. of ECE, Kyutech, Japan e-mail: tsukamoto@cse.kyutech.ac.jp

system capacity. Several solutions including the partitioning approach [5–7] are proposed to determine which module of a mobile application should be offloaded and how efficiently allocated resources in the edge cloud. A Semi-Markov Decision Process (SMDP) based Multi-Resource Allocation scheme is devised [6], which works on average reward criterion and proposes a strategy to determine whether to offload a service request on edge cloud or back-end cloud. However, aforementioned works are purely theoretical and lack any practical implementations in the real-world scenarios. We scale MRA (Multi-Resource Allocation) [6] work in order to implement it in practical use cases.

A Resource Manager (RM) is introduced at the Edge Cloud (EC) to help manage all computing and bandwidth resources, which will be responsible for meeting the QoS requirements inherent to IoT applications and services, and acts as a coordinator for the end-devices, EC, and back-end Cloud (BC). This work seeks to aid the RM to find an optimal action for resource allocation either in the EC or BC in real-time to minimize the latency for delay sensitive IoT and real-time mobile applications. The policy tables calculated and stored at EC [5, 6] grow exponentially which contain optimal resource allocation actions responding to a range of service requests. As such, it is impractical to search large size policy tables in linear fashion due to the time limitation of delay sensitive applications. Therefore, how to search the policy tables while complying the end-to-end delay constraint should be carefully considered by the RM.

The rest of the paper is organized as follows: Section 2 describes our system model for policy tables, while index based policy table search techniques are described in Section 3. The performance evaluation is discussed in Section 4, and conclusions are presented in Section 5.

2 System Model

For our study, we consider the COSMOS test-bed environment which consists of edge and core cloud computing infrastructure in which multiple mobile or IoT devices can connect to the EC through wireless access point [8, 9]. These devices can run applications locally, or offload some modules of the application to EC or to the BC cloud for faster execution and better energy conservation.

Upon arrival of a new service request, it can be decided whether to run it on the native device or to be offloaded to the edge cloud based on the network performance and application characteristics [7]. For the offloaded module, the policy table is searched for optimal action, which has already been calculated by MRA algorithm [6]. MRA strategy not only adaptively determines the location (EC or BC) for the execution of service request but also determines the optimal amount of wireless bandwidth i and computing resource j to allocate to the accepted service request. This MRA strategy achieves the maximum system benefits (Eq. (11) in [6]) in terms of throughput and blocking probability while maintaining required latency requirement. This MRA problem is formulated as SMDP and solved as a linear programming problem to calculate an optimal policy which is composed of all the probabilities of randomly selecting the actions (Eq. (13) in [6]). This approach has a predictive ability of future state that lies in the transition probabilities (p_i^j)

from the current state x_i^j to all potential next states if action a_i^j is chosen upon receiving a new service request. These policy tables are stored at the EC to be consumed by the Resource Manager (RM). The provisioning of the resources can be sped-up if tailored approaches are applied while arranging and searching the policy tables.

In case of real-world 5G environments, the linear search for large size policy tables is an inappropriate approach as it violates the latency requirement. Therefore, fast and efficient mechanisms have to be devised to meet the latency requirement for real-time applications. Another aspect could be to support massive machine type communication (mMTC), where hundreds of thousands of devices are expected to be connected with the EC [10]. Resource provisioning for such large number of devices should be fast enough to minimize the blocking probability of service requests.

The formulation of policy tables is classified into two different ways. The first one is comparatively large table and the second one is significantly smaller and contains adequate information to satisfy an incoming service request. The size of the policy tables is proportional to the available resources in terms of the total number of VM units (M) at edge cloud and total number of bandwidth units (B) on the wireless channel[1]. This also corresponds to the number of total possible states (S_T) of the system. The notations used in this paper are given in Table 1.

The flow of policy table search during resource provisioning is illustrated in Fig. 1. Upon arrival of a new service request the RM searches the policy table and finds an optimal action. Here, 0 means the chosen action was a reject. On the other hand, if chosen actions were$\{a_i^j, a_i\}$, then the RM allocates resources on the EC or BC accordingly. The objective is to provide a mechanism to construct and search huge size policy table for the RM such that, an optimal action is retrieved in minimal possible time.

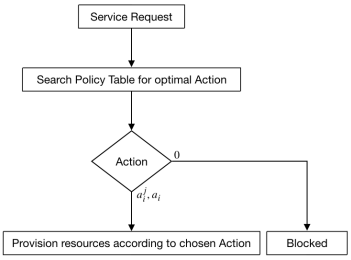

Fig. 1 Policy Table Search-Flow for resource provisioning by Resource Manager in Edge based Cloud Computing System

3 Index Based Policy Table Search

Aforementioned Policy Tables are stored as a database which contains, to name a few, all the possible states of a system, occurrences (values) of those states and their corresponding actions. The searching time of a value is proportional to the growth of such databases. Efficient and effective ways of amalgamating sporadic data and retrieving use-

[1] The wireless bandwidth and VM units are defined in [6]. Where *"bandwidth refers to the wireless connections between the end devices and the EC, and one wireless bandwidth unit refers to the minimum bandwidth required to support mobile computing offloading, for example, 50, 100 Kbps, etc. Similarly, VM refers to the minimum computational resource required to execute a service request, e.g. 1 core of CPU with 1 GB memory. Then, the total bandwidth/VM available can be expressed as the integer multiple of the bandwidth and VM unit. For the simplicity of computation, we assume a single service request requires at least one basic unit of wireless bandwidth and VM units, and only the integral numbers of basic bandwidth units and VM units are allocated"*.

Table 1 Notations

Name	Description
B	The total number of wireless bandwidth units available
M	The total number of VM units available on the edge cloud
W	The maximal number of wireless bandwidth units that the system provides to one service request
T	The maximal number of VM units that the system provides to one service request
α_W	Limiting factor for W
α_T	Limiting factor for T
x_i^j	The number of ongoing services that are allocated i units of wireless bandwidth and j units of VM on edge cloud
y_i	The number of ongoing services that are allocated i units of wireless bandwidth and T units of VM on back-end cloud
a_i^j	The action to accept the request by allocating i units of wireless bandwidth and j units of VM on edge cloud
a_i	The action to accept the request by allocating i units of wireless bandwidth and T units of VM on back-end cloud
S_T	Total number of states
S_{Ec}	Edge Cloud based states
S_{Bc}	Back-end Cloud based states
A_s	Set of allowable actions at state s
M_x	Maximum occurrences (value) of state x_i^j
M_y	Maximum occurrences (value) of state y_i
X_i^j	Size of Policy Table of state x_i^j
Y_i	Size of Policy Table of state y_i

ful information are indispensable in any databases. This assertion is even more critical for real-time applications. The standard linear search is not adequate as the time complexity of such search is $O(m \times n)$, where m and n are the number of rows and the number of columns respectively of the policy table matrix [11]. In our case, we see that the total number of elements approaches to 70 millions which takes search time on the order of milliseconds (Section 4). Therefore, index based search like [12, 13] is devised which drastically diminishes the search time.

3.1 Apply Limit to W and T

A limit can be set with an integer variable (α) to define the maximum number of bandwidth units W and maximum number of VM units T which can be assigned to one service request. In order to accommodate multiple instances of a state which require

maximum allowed bandwidth and VM units W and T respectively, we limit W and T by an integer variable α such that:

$$W \leq \frac{B}{\alpha_W} \tag{1}$$

$$T \leq \frac{M}{\alpha_T} \tag{2}$$

where B is total available wireless bandwidth units and M is total available VM units. This also allows us to generate dynamic state models according to a desired environment instead of having a fixed upper limit for both bandwidth and VM units as suggested in previous study [6].

3.2 Total Possible Number of States (S_T)

The states for an EC is given by

$$\{x_1^1, x_1^2, x_1^3, ..., x_2^1, x_2^2, x_2^3, ..., x_W^T\}$$

and BC is given by

$$\{y_1, y_2, y_3, ..., y_W\}.$$

The sizes of both EC and BC states are different and calculated separately:

$$S_{Ec} = W \times T \tag{3}$$

$$S_{Bc} = W \tag{4}$$

where W and T represent the maximum bandwidth units and VM units allowed for a service request, respectively. Thereafter, the total number of states are:

$$S_T = S_{Ec} + S_{Bc} \tag{5}$$

Thus, the set of allowable actions A_s at EC for any given state x_i^j becomes

$$\{a_1^1, a_1^2, a_1^3, ..., a_2^1, a_2^2, a_2^3, ..., a_W^T\}$$

and at BC for any given state y_i becomes

$$\{a_1, a_2, a_3, ..., a_W\}.$$

Note that, for certain occurrences of a given state there may be some actions which are prohibited. For example, if $M = 10, B = 10$, the current state of the system is x_2^2, and occurrence of such state is 5 then further resource allocations cannot be done. Hence, such actions are replaced with 0, meaning the action of reject. Accordingly, it makes the total

number of possible actions for any given state as:

$$A = S_T + 1 \tag{6}$$

where one extra action represents the reject action i.e. 0 in Fig. 1.

3.3 Policy Tables

Policy tables (PT) are composed in the form of two dimensional matrix of variable size, which contains the states arranged by occurrences as columns and corresponding actions as rows. Upon arrival of a new service request, RM consults the policy table for an optimal action and provisions resources at EC or BC. The size of PT depends upon Eqs. (5-6) and the maximum values (occurrences) of all the states which are calculated later. We propose two kinds of policy tables which are concatenated differently from small policy tables calculated for all the individual states at each possible occurrence [6].

3.3.1 Policy Table-I

This is a two dimensional matrix where rows demonstrate the actions A_s, and columns show the occurrences of a state x_i^j and their corresponding probabilities of actions. For any given state x_i^j the occurrences range from 0 to M_x which represent the maximum possible occurrence of the state. Here, it is important to calculate M_x, which is an integer value, to make sure that the resource demand by a state x_i^j does not exceed the available edge cloud VM units M and bandwidth units B. Therefore, M_x is given by:

$$M_x \leq \begin{cases} \frac{M}{j}, & \forall M < B \\ \frac{B}{i}, & \forall B < M \\ \frac{M+B}{i+j}, & \forall M = B \end{cases} \tag{7}$$

where $i \leq W$ and $j \leq T$. Similarly, for any given state y_i the values range from 0 to M_y. Here, we assume that the available number of VM units on back-end cloud are unlimited and occurrence of state y_i is only constrained by available bandwidth units B. Therefore, M_y is given by:

$$M_y \leq \frac{B}{i} \tag{8}$$

where $i \leq W$. Therefore, the size (X_i^j) of a matrix for any given state x_i^j becomes $R \times M_x$ and size (Y_i) of a matrix for any given state y_i becomes $R \times M_y$, where R represents the number of rows of matrix of policy table and corresponds to the actions A Eq. (6). We construct Policy Table-I by concatenating these tiny matrices of all states. Thus, the size of PT-I becomes $(R \times N)$, where $R = A$ and N is given as:

$$N = \sum_{j=1}^{T} \sum_{i=1}^{W} X_i^j + \sum_{i=1}^{W} Y_i \tag{9}$$

Note that rows R of PT-I remain the same, which makes it a perfect rectangular matrix. From Fig. 2 we can see that if the current state and its value in the system are x_1^2 and 2 respectively, the RM retrieves the whole column (circled in red) and decides an optimal action among all the probabilistic actions.

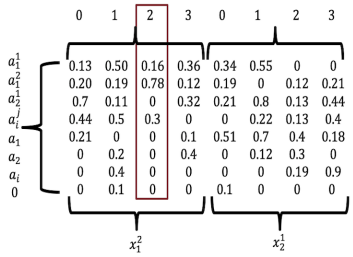

Fig. 2 Policy Table I for service requests. Elements depict the probability of actions with respect to current state and value of the state.

3.3.2 Policy Table-II

Just like PT-I, PT-II is also a two dimensional matrix. This time states are placed in columns and rows represent the occurrence (value) of each state. As compared to PT-I in which a range of possible actions are extracted and then RM decides which action to choose, PT-II is given only with single action. We can say that PT-II is structured in more of a normalized [14] form and debars the redundant data of PT-I. The size of PT-II is $R \times N$ where R ranges from 0 to $Max(M_x, M_y)$. The reason to choose the maximum among the maximum occurrences of all the states is to create a matrix of a regular shape. In this case, the maximum occurrence constraints (M_x, M_y) are not applied. Therefore, actions for such prohibited occurrences are replaced by 0 as explained in Section 3.2. Here, $N = S_T$, this makes the size of PT-II remarkably small as compared to PT-I.

Fig. 3 illustrates an example of PT-II in which states are arranged column wise and rows represent occurrences of states. For example, If the current state is x_2^2 and its occurrence in the system is 2, then a_1^1 (circled in red) is taken with corresponding probability, and other actions are also possible with their corresponding probabilities.

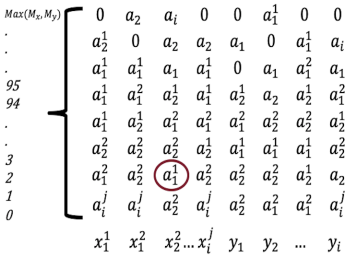

Fig. 3 Policy Table II for service requests. Elements depict the optimal actions with respect to current state and value of the state.

3.4 Index Based Search on Policy Tables

In order to search the policy table and find an optimal action in real-time, we propose an index based search on the policy tables. We assume that system have information about the current state and its occurrence. For index based search we store the indices of states, actions and max occurrences of states as $1 \times N$ vectors. The number of operations that

are executed to find an optimal action differ for both policy tables. We also apply linear search for both policy tables and compare the results in Section 4.

3.4.1 Searching Through PT-I

The following operations involved while searching PT-I:

- Get the index of currently known state: To find that we search through the vector of states.
- Get the maximum possible occurrence of that state: use above index to find the maximum possible occurrence of given state from the vector of the maximum values.
- Consume the currently known occurrence of the state to calculate index of exact column in PT-I: add all occurrences of previous states and the current occurrence of the state.
- Retrieve the column using column index found in above step.
- RM decides actions depending on the probabilities of such actions of service requests.
- Find the index of chosen probability.
- Retrieve action using above index: search through vector of actions.

For a particular state, only relevant column is retrieved and rest of the columns are eliminated during search.

3.4.2 Searching Through PT-II

As mentioned before that PT-II includes only normalized data and discards the redundant information. Thus, the reduced number of steps required to fetch the optimal action are listed below:

- Get the index of currently known state: To find that we search through the vector of states.
- Consume the currently known occurrence of the state and above found index to find action (e.g. $Action =$ PT-II$(2,1)$)

PT-I is large, which provides flexibility over choosing an action but takes more search time. Conversely, PT-II takes less search time with no flexibility.

4 Evaluation

In this section, we evaluate the performance of proposed techniques. The performance is calculated under various values of available wireless bandwidth units (B) and VM units (M). For the manageability of the model computation, we use $B = M$. However, the model works for other cases ($B < M, M < B$) as well (Eq. (7)). The advantages of proposed techniques are clearly revealed by comparing with legacy linear search method.

The simulations are written in MATLAB and ran on Windows PC (Dell, Intel Core i7 (7th Gen) 7700T / 2.9 GHz Quad-Core, DDR3 16 GB SDRAM). The system values (B,M) are scaled up with the purpose of replicating real-world scenarios depending on the size of edge cloud [2]. Other system parameters (α_W,α_T) are set to default values (i.e. 4). The limiting factors α_W and α_T can be modified to replicate different traffic models according to the need. For example, α_W and α_T can be set closer to W and T to support massive number of users requiring less resources (a few Kbps and few MB memory), while they can be set closer to 1 for broadband users [10].

No. of States: As we can see in Fig. 4, number of states grow linearly upon increasing the edge cloud resources. If we double the number of bandwidth and VM units from 200 to 400, the number of states increase approximately four times, which causes growth of policy tables likewise.

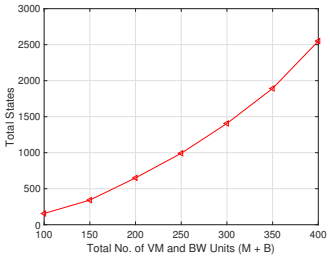

Fig. 4 No. of States for $\alpha_W = 4$, $\alpha_T = 4$ and $M = B$

Policy Tables: Fig. 5 shows that PT-I grows exponentially and reaches up to 72 million (2551×28318) elements in the matrix. Whereas PT-II hardly crosses half million (201×2550) elements. Both have their own advantages over the other which are discussed at the end of Section 3.4.2.

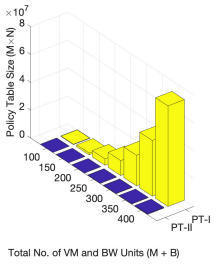

Fig. 5 Size of Policy Tables for $\alpha_W = 4$, $\alpha_T = 4$ and $M = B$

Search Time: The simulations are run 1000 times and average time for both linear and indexed search of PT-I and PT-II is shown in Fig. 6. We observe that linear search time for PT-I approaches to $90ms(milliseconds)$ which is beyond the end-to-end delay criteria (in the order of $1ms$) of ultra-reliable low-latency Machine Type Communication (uMTC) [10]. Linear search time for PT-II remains under $10ms$, which is due to the remarkably reduced size of the table. However, this also does not meet the minimum delay requirement of uMTC as network and application processing delays are yet to be added which further increase the end-to-end latency. Whereas, index based search time for PT-I and PT-II remains under $70\mu s$ and $10\mu s$ respectively. This can be clearly seen that index based search time for both policy tables qualify the end-to-end delay criteria of uMTC and real-time applications.

5 Conclusion

In this paper, we scale SMDP Multi-Resource Allocation work and present an index based search approach on large size policy tables to speed up admission control of service requests in the edge

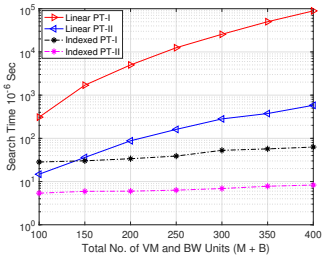

Fig. 6 Linear and indexed Search Time comparison between PT-I and PT-II

cloud based systems. In deriving the optimal action from a policy table, we consider end-to-end delay constraints of uMTC and real-time applications. Our proposed technique for structuring the policy tables and searching through them, not only outperforms the legacy linear search method, but also meets the delay requirement of real-time applications in the growing edge cloud systems.

The index based search technique can help Resource Manager (RM) in EC based systems to search an optimal action for a service request from large size policy tables in order of microseconds(μs). We intend to continue to study further and implement our proposed work for real traffic in the real-world experiments like COSMOS test-bed.

Acknowledgements This work is partially supported by USA Grant number NSF Award 181884.

References

1. T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," in IEEE Communications Surveys and Tutorials, vol. 19, no. 3, pp. 1657-1681, thirdquarter 2017.
2. N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, "Mobile Edge Computing: A Survey," in IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450-465, Feb. 2018.
3. E. Borgia, R. Bruno, M. Conti, D. Mascitti, and A. Passarella, "Mobile edge clouds for information-centric IoT services," in Proc. IEEE Symp. Comput. Commun. (ISCC), Messina, Italy, Jun. 2016, pp. 422428.
4. Y. Jararweh et al., The future of mobile cloud computing: Integrating cloudlets and mobile edge computing, in Proc. 23rd Int. Conf. Telecommun. (ICT), Thessaloniki, Greece, May 2016, pp. 15.
5. L. Yang, B. Liu, J. Cao, Y. Sahni and Z. Wang, "Joint Computation Partitioning and Resource Allocation for Latency Sensitive Applications in Mobile Edge Clouds," 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, CA, 2017, pp. 246-253.
6. Y. Liu, M. J. Lee and Y. Zheng, "Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System," in IEEE Transactions on Mobile Computing, vol. 15, no. 10, pp. 2398-2410, 1 Oct. 2016.
7. Y. Liu and M. J. Lee, "An Adaptive Resource Allocation Algorithm for Partitioned Services in Mobile Cloud Computing," 2015 IEEE Symposium on Service-Oriented System Engineering, San Francisco Bay, CA, 2015, pp. 209-215.
8. E. Piri et al., "5GTN: A test network for 5G application development and testing," 2016 European Conference on Networks and Communications (EuCNC), Athens, 2016, pp. 313-318.
9. NSF, PAWR "COSMOS: Cloud Enhanced Open Software Defined Mobile Wireless Testbed for City-Scale Deployment" https://cosmos-lab.org.
10. A. Osseiran, Jose F. Monserrat., "5G Mobile and Wireless Communications Technology" Cambridge University Press 2016, pp. 240-270.
11. S. Shah and A. Shaikh, "Hash based optimization for faster access to inverted index," 2016 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, 2016, pp. 1-5.
12. C. Lin, C. Hsu and S. Hsieh, "A Multi-Index Hybrid Trie for Lookup and Updates," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 10, pp. 2486-2498, Oct. 2014.
13. L. Bulysheva, A. Bulyshev and M. Kataev, "Visual Database Design: Indexing Methods," 2018 Sixth International Conference on Enterprise Systems (ES), Limassol, 2018, pp. 25-29.
14. Online, "Database Normalization:" https://support.microsoft.com/en-us/help/283878/description-of-the-database-normalization-basics.