Distributed Reinforcement Learning based Code Offloading in Mobile Fog

Md. Golam Rabiul Alam, Nguyen Hoang Tran, Cuong T. Do, Chuan Pham, Sarder Fakhrul Abedin, Anupam Kumar Bairagi, Rim Haw, Choong Seon Hong*
Department of Computer Engineering, Kyung Hee University
{robi, nguyenth, dtcuong, pchuan, saab0015, anupam, rhaw, cshong}@khu.ac.kr

Abstract

Fog computing is the horizontal and distributed expansion of centralized cloud computing. The highly virtualized and densely distributed network edges become the fog nodes and provide compute, storage and networking services as the complementary unit of centralized cloud. Mobile Fog is a special purpose computing prototype, which leverages the mobile computing to deliver seamless and latency-aware mobile services. In this paper, we propose distributed reinforcement learning based autonomic code offloading mechanism to ensure low-latency service delivery towards mobile service consumers. We use ACE ("ACtion Estimation"), the distributed reinforcement learning algorithm to offload basic blocks in a decentralized fashion on geographically distributed *mobile fogs*.

1. Introduction

Fog computing introduced by Cisco Systems Inc. to extend the cloud computing paradigm to the edge of the network especially for Internet of Things (IoT) services [1]. Offloading computation in *mobile fog* is challenging because of the spatiotemporal resource requirements of heterogeneous mobile devices. The authors' of [2], proposed an effective way to offload useful heap objects and partial stack in run time of application. ThinkAir [3] proposed, a method level computation offloading mechanism for mobile cloud computing.

In contrast of above methods, we propose reinforcement learning based basic block offloading mechanism in mobile fog. The proposed scheme is multi-agent based distributed method for offloading computation in dispersed mobile fogs.

2. Mobile Fog Architecture

In Fig. 1, we introduce a model of mobile cloud computing based on the fog computing [1] model of Cisco Systems, Inc. In this mobile fog computing model we use the hierarchical architecture of LTE (long term evolution) and Wi-Fi internetworking reference model [4]. In this architecture, the mobile fog is created on the edge of the networking modules. We consider the access point (AP) and the access point controller (APC) units as the fog nodes of mobile fog. The fog enabled AP and APC are symbolized as F-AP and F-APC.

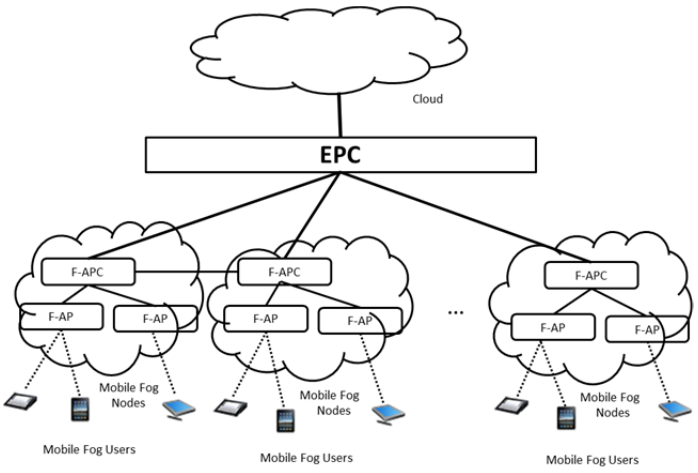

Figure 1. Proposed Mobile Fog Architecture

3. Reinforcement Learning based Code Offloading

In response to the compute service request from mobile fog users, according to the flow graph [5] of the generated codes the basic blocks are executed on various fog nodes, where compatible basic blocks are executed in parallel fashion as shown in Fig. 2. We have three different options to offload codes i) the mobile fog in close proximity of mobile stations, L_1 ii) the adjacent mobile fog (or distant mobile fog) to handle mobility and load balancing issues, L_2 iii) the remote public cloud to manage huge traffic and computing requirements, and archiving, L_3. We deploy multiple mobile agents to find the best suitable options to offload basic blocks of mobile codes.

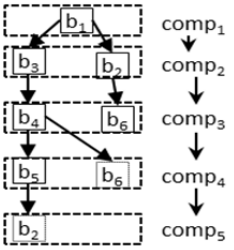

Figure 2. The flow-graph of N-queen problem with 6 basic blocks. The blocks are grouped into 5 compatible sets which we can easily find out by applying BFS algorithm.

The mobile agents action space A in case of basic block offloading in mobile fog is i) $a_{1,k}$: offload at fog node k in location L_1 ii) a_2: offload in location L_2 iii) a_3: offload in location L_3. Each individual agent G_i is responsible for a specific action in our multi agent system. We consider the state space S of mobile agents as the vector of storage, processing, time schedule, and networking bandwidth capability of fog nodes. So, the state space can be represented as $S=\{s_1, s_2, \ldots, s_n\}$ where $s_i=(mem_i, cpu_i, time_i, band_i)$, $i = 1 \ldots n$. The mobile agents will learn from environment and experience by following ACE, the distributed reinforcement learning algorithm through its action selection and credit assignment policy [6].

Algorithm 1: Active_Agents_Set_ Selection ( )

1. Determine the candidate agents set $A^{cand}[S_j]$ for state space S_j. Set the active agents set $A^{act}[S_j] = \emptyset$ to offload compatible basic blocks set $B = \{b_1, b_2, \ldots, b_t\}$.
2. While B ≠ {∅} do step 3 to 6
3. Select the agent $A_i \in A^{cand}[S_j]$ with $B_i^j > B_k^j$ for all $A_k \in A^{cand}[S_j]$ to offload the block b_i.
4. Update the active agents set as $A^{act}[S_j] = A^{act}[S_j] \cup \{A_i\}$.
5. Update the candidate agents set as $A^{cand}[S_j] = A^{cand}[S_j] \setminus (A_i \cup \{A_k \in A^{cand}[S_j]\})$, where A_i and A_k are incompatible.
6. Update the basic block set as $B = B \setminus \{b_i\}$.
7. Return $A^{act}[S_j]$ as the active set to offload the compatible basic blocks set B.

A. Action Selection

The F-AP receives the service request of offload computation from fog users. Then the agent of fog node selects the appropriate actions as presented in *Algorithm 1* to deploy the code blocks. The multiple agents of the fog node announces their bids B_i^j to deploy the compatible set of basic blocks in respected location considering the estimated capability E_i^j of host fog nodes.

$$B_i^j = \begin{cases} E_i^j + \alpha * F_i^j + \beta * F_i^j & If\ E_i^j > 0 \\ 0 & Otherwise \end{cases} \quad (1)$$

Here, i and j represent the agent and state respectively, α is a small constant called the risk factor of the estimation and β is a small random number called the noise factor to escape from local minima. We consider the estimated response time to process the block as the estimated capability. We assume *M/M/1/K, M/M/c/K* and *M/M/c/α* queuing model to estimate the response time of location L_1, L_2, L_3 respectively. Considering the above queuing disciplines the expected response time R_i^j can be formulated as in (2).

$$R_i^j = \begin{cases} \frac{E[N]}{\varphi(1-P_K)}, & If\ a_{1,k} \in A\ and\ s_j \in S \\ \frac{E[Q]+\rho(1-P_K)}{\varphi(1-P_K)} + L_{1,2}, & If\ a_2 \epsilon A\ and\ s_j \in S \\ \frac{1}{\mu} + \frac{1}{\mu(c-\rho)} \sum_{l=n}^{\infty} P_0 \frac{\rho^l}{c!(c^{K-c})} + L_{1,3}, & If\ a_3 \in A\ and\ s_j \in S \end{cases} \quad (2)$$

We also consider the cost of processing code blocks in cloud instances and in fog nodes. Let, the price vector of different cloud and fog instances is **p**= (p_1, p_2, ⋯ , p_n). Now we can determine the demand of a cloud and fog node instance through the demand function $d_i=d_i(\mathbf{p})$. Here, we assume the generalized *logit* demand model (3) to determine the demand $d_i \in [0, 1]$ of different cloud and fog instances.

$$d_i(\boldsymbol{p}) = \frac{\exp(-\varphi_i p_i)}{\sum_{j=1}^{n} \exp(-\varphi_j p_j) + \tau} \quad (3)$$

Thus the estimated capability E_i^j of host fog nodes can be formulated as (4), and initially we consider $F_i^j = E_i^j$ and $\gamma_1 + \gamma_2 = 1$.

$$E_i^j = \gamma_1 * \frac{1}{R_i^j} + \gamma_2 * d_i(\boldsymbol{p}) \quad (4)$$

B. Credit Assignment

Credit assignment is the policy to pass the rewards to former winner agents to set up the environment. The bucket-brigade model is followed to assign credits to the predecessor. Let the agent i is the currently active (winner) on state space s_j and agent k is its immediate predecessor, which was active at

state space s_l. According to bucket-brigade policy, agent i reduces it's E_i^j by a fraction of the risk factor $\alpha * F_i^j$ as in (5). Finally, agent k receives credit from agents i according to (6).

$$E_i^j = E_i^j - \alpha * F_i^j \quad (5)$$

$$E_k^l = E_k^l + \frac{\Sigma_{A_i \epsilon A^{act}[S_j]} \alpha * F_i^j}{|A^{act}[S_j]|} \quad (6)$$

The distributed reinforcement learning algorithm of code offloading is presented in *Algorithm 2*.

Algorithm 2: DRL_Fog_Offloading( )

1. Offload request receives by mobile fog nodes (F-AP, F-APC) to offload computation.
2. Read the basic blocks and flow graphs of the generated code.
3. For each set of compatible basic block *B*, perform the following steps.
4. Agents of fog nodes check their actual environmental state (*mem, cpu, time, band)* of deployable options.
5. Determine set of active agents to offload the basic blocks set *B* using *Algorithm 1.*
6. Assign credits to each agent according to (5) and (6).
7. End for

4. Performance Evaluation

The performance of mobile fog is evaluated through simulation using OMNeT++. We offload the basic blocks of benchmarked N-queen problem in three different computing nodes. Fig. 3 shows that smart phone takes longer time to process the benchmark application, whereas the mobile fog takes shorter time to place the Queens on the board. The cloud also takes less time to process the solution because of the execution power of cloud servers. Another important aspect of our proposed basic block offloading mechanism is the ability of parallel execution. ThinkAir [3] also executed different methods concurrently, but basic blocks have more parallelism options, which lead lower response time as shown in Fig. 4.

5. Conclusion

We propose Mobile Fog architecture to leverage mobile cloud computing and then we proposed a distributed reinforcement learning based code offloading mechanism in mobile fog. The experimental results show the improved performance of the proposed offloading method in respect to execution time and latency.

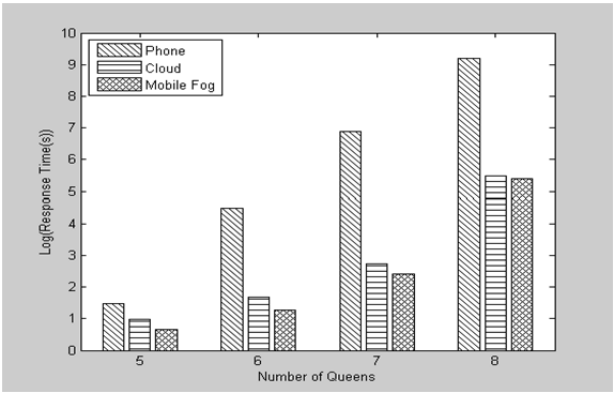

Figure 3. The log-normal response times of benchmark N-queen problem

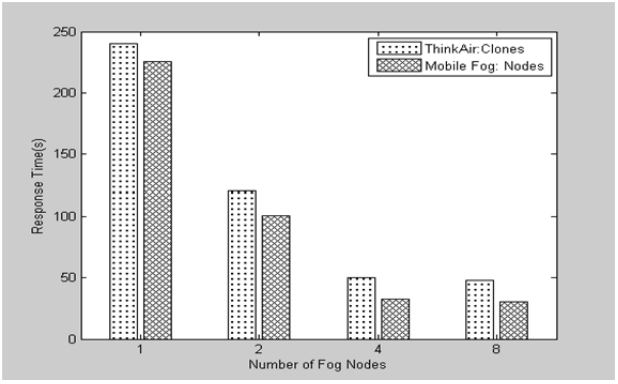

Figure 4. Comparative study with existing benchmark solution of mobile cloud computing. ThinkAir deployed up to 8 colons to solve the 8-Queen problem, whereas Mobile Fog deployed 8 fog nodes to solve 8-Queen puzzle.

Acknowledgement

This work was supported by the ICT R&D program of MSIP/IITP, Republic of Korea. (2014-044-011-003, Open control based on distributed mobile core network) *Dr. CS Hong is the corresponding author.

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. "Fog computing and its role in the internet of things." In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16. ACM, 2012.

[2] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, and Y. Paek. "Techniques to Minimize State Transfer Costs for Dynamic Execution Offloading in Mobile Cloud Computing.", IEEE Transaction on mobile computing, Vol. 13, No. 11, 2014.

[3] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang. "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading." In INFOCOM, 2012 Proceedings IEEE, pp. 945-953. IEEE, 2012.

[4] Chris Yoo, "Network Architecture for LTE and Wi-Fi Internetworking", NMC consulting group, August 16, 2012.

[5] L. Monica, R. Sethi, J. D. Ullman, and A. Aho. "Compilers: Principles, Techniques, and Tools.", 2006.

G. Weifi, "Learning to coordinate actions in multi-agent systems.", Readings in agents 481, 1998.