FEC 환경에서 로컬과 에지 서버 간의 협업 실행경로 추출

백재석*, 남광우*, 장민석*, 이연식O
*군산대학교 컴퓨터정보통신공학부,
O군산대학교 컴퓨터정보통신공학부
e-mail: gwings@naver.com*, {kwnam*, msjang*, yslee O}@kunsan.ac.kr

Extraction of Collaborative Execution Path between Local and Edge Server in an FEC Environment

Jae-seok Baik*, Kwang-Woo Nam*, Min-seok Jang*, Yon-sik Lee O
*School of Computer Information & Comm. Engineering, Kunsan National University,
OSchool of Computer Information & Comm. Engineering, Kunsan National University

● 요 약 ●

FEC (Fog/Edge Computing) 환경에서 지연시간 최소화는 로컬과 에지 서버 간의 효율적인 협력을 보장하기 위한 최적의 계산 오프로딩 방법 결정을 통해 실현될 수 있다. 본 논문은 임의의 응용 서비스 실행모듈에 대한 부분 오프로딩 기반의 로컬(모바일 장치)과 에지 서버 간의 협업 경로를 추출하는 방법을 제안한다. 제안 방법은 다중 분기구조를 포함하는 응용 서비스 실행모듈에 대한 부분 오프로딩 기반의 최적 협업 실행 경로 추출 방법을 제안한다. 제안 방법은 각 부분 모듈들의 실행위치에 따라 변화되는 지연시간 측정 및 분석에 적용가능하다.

키워드: 포그/에지 컴퓨팅(Fog/Edge Computing), 부분 오프로딩(partial offloading), 지연시간(latency time), 협업 경로(collaboration path)

I. Introduction

FEC 환경에서 지연시간 최소화를 위하여 컴퓨팅 리소스를 모바일 네트워크의 에지에 배치하여 모바일 장치와 FEC 서버의 리소스를 모두 활용하는 방법이 요구된다. 이를 위하여, 리소스가 제한된 모바일 장치에서 프로그램 분석 및 프로파일링을 통해 FEC 서버로 부분 오프로딩을 수행하여 처리를 분할하고 프로그램 가능을 처리할 위치를 결정한다. 본 논문은 이러한 부분 오프로딩 기반의 로컬(모바일 장치)과 에지 서버 간의 최적 협업 경로 추출 방법을 제안한다.

II. Partial offloading

모바일 장치는 서비스 실행모듈의 파티션을 생성하고, 오프로딩 정책에 따라 부분 오프로딩을 수행하여 에지 서버를 통하여 작업을 완료한다. m개의 구현 모듈을 각각 파티션 대상으로 간주하여 파티션 집합 DAG = {DAG_1, DAG_2, · · ··, DAG_m}을 정의한다. 각 DAG_i는 오프로딩 정책에 의해 로컬에서 처리되는 DAG^l_i와 에지 서버에서 처리되는 DAG^e_i로 구분한다.

부분 오프로딩의 목표는 응용 서비스를 위한 연산의 총 대기 시간을 최소화하는 것으로, 이는 로컬과 에지 서버 간의 효율적인 협력 기반의 최적 오프로딩 결정을 통해 실현될 수 있다. 작업의 총 처리 지연시간(T)은 다음 식 (1)과 같으며, 이 시간은 해당 기지국 내에서의 체류시간보다 짧아야 하고 또한 클라우드 서버 이용 시간보다 짧아야 함을 보장해야 한다.

$$T = T_l + T_e + T_{tt} = \sum_{i=1}^{m} (t_i^l + t_i^e) + \sum_{i=1}^{n} tt_i \quad (1)$$

식 (1)에서 T_l은 모든 DAG^l_i 처리시간의 합으로 로컬 처리시간이며, T_e는 모든 DAG^e_i 처리시간의 합으로 에지 서버 처리시간이고, T_{tt}는 오프로딩으로 인한 전송시간으로 로컬과 에지 서버 간 각 모듈의 출력 전송시간들의 합이다.

III. Extraction of Local and Edge Server Collaboration Paths

로컬에서 수행되는 서비스의 실행모듈 토폴로지에 대한 부분 오프로딩 여부를 결정하기 위하여, DAG 토폴로지의 각 모듈의 분기구조들을 체인 토폴로지로 변환하여 실행위치를 결정하고, 이를 이용하여 로컬과 에지 서버의 협업 실행경로 생성을 위하여 DAG를 재구성한다. 그림 1은 실행모듈에 대한 부분 오프로딩을 적용한 DAG 토폴로지를 나타낸다.

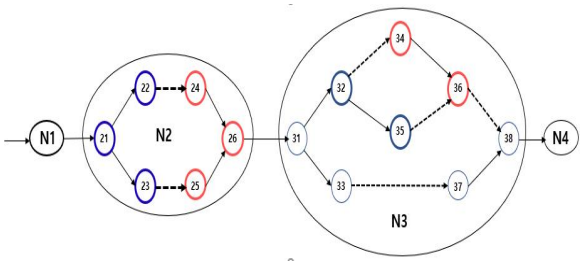

Fig. 1. DAG Topology for Executable code

그림 1의 N2와 N3는 로컬과 에지 서버에서 처리되는 부분으로 구분되었으며, 최소-컷 문제에 의해 잘려진 간선(점선 표시)들은 출력 전달시간을 나타낸다. N1은 로컬에서 N4는 에지에서 처리되는 것으로 가정하였으나, 부분 오프로딩으로 인한 처리위치에 따라 N2와 N3의 처리시간이 변할 수 있다.

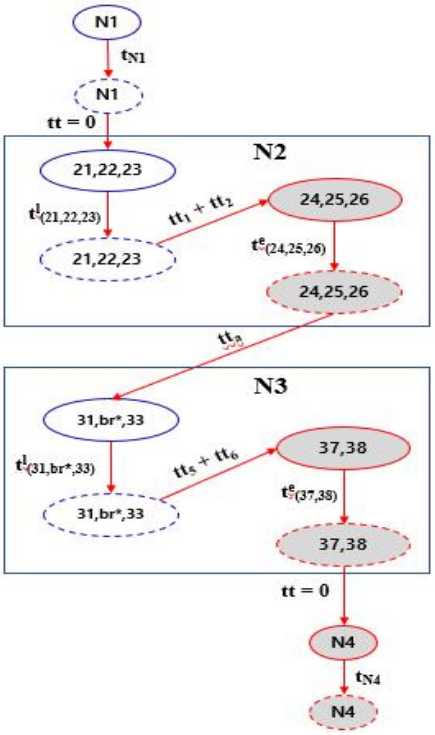

Fig. 2. Optimal collaboration execution path between local and edge servers

그림 2는 그림 1의 DAG 토폴로지에 대하여 Dijkstra 최단 경로 알고리즘을 적용하여 로컬과 에지 서버 간의 협업 실행경로를 추출한 것이다. 실선 표시 경로가 DAG에서 최적 실행경로이며, 실선 표시 노드는 실행 전, 점선 표시 노드는 실행 후를 나타내며, 좌측 노드들은 로컬에서 우측 노드들은 에지 서버에서 처리됨을 나타낸다.

전체 처리시간(T)은 식 (2)와 같이 산출된다.

$$T = \sum_{i=1}^{4} t_i + \sum_{j=1}^{6} tt_j + tt_a \qquad (2)$$

(여기서, i = node 수, j = 출력 전달 수, $t_{N2} = t^l_{(21,22,23)} + t^e_{(24,25,26)} + tt_1 + tt_2$, $t_{N3} = t^l_{(31,br*,33)} + t^e_{(37,38)} + tt_5 + tt_6$, $t_{br*} = t^l_{(32,35)} + t^e_{(34,36)} + tt_3 + tt_4$, tt_a = N2의 출력이 에지 서버에서 로컬로 전달되는 시간임)

N1의 뒷부분과 N4의 앞부분의 처리위치에 따라 전송 오버헤드 유무가 결정되므로, 실제 최소 처리시간은 이전 노드의 처리위치 결정과 밀접한 관련이 있다. 따라서 서비스 실행모듈에 대한 모든 노드에 대한 최적 오프로딩 결정 방법이 중요하다. 이와 같은 방법은 사용자가 단일 기지국 내에 있을 경우 최적의 오프로딩을 결정하지만, 장치 이동성으로 인한 기지국 핸드오버가 발생하면 시간 손실이 발생하므로 제안 방법으로 결정한 경로는 최적 실행경로라 할 수 없다. 이 경우 매 노드 i의 처리시간($t_i = t^l_i + t^e_i + tt_i$)이 기지국 사용자의 거주 시간보다 작아야 한다.

IV. Conclusions

본 논문은 FEC 환경에서 부분 오프로딩을 위한 기반연구이다. FEC 환경에서 지연시간 최소화는 로컬과 에지 서버 간의 효율적 협력을 보장하는 최적의 부분 오프로딩 결정을 통해 실현될 수 있다. 본 논문은 임의의 응용 서비스 실행모듈에 대한 부분 오프로딩 기반의 로컬(모바일 장치)과 에지 서버 간의 협업 경로를 추출하는 방법을 제안하였다. 제안 방법은 다중 분기구조를 포함하는 응용 서비스 실행모듈에 대한 부분 오프로딩을 적용하였으며, 각 부분 모듈들의 실행위치에 따른 지연시간 변화에 대한 분석이 가능하다. 제안 방법은 단일 기지국 내에서의 부분 오프로딩 적용 기반이므로, 기지국 핸드오버에 따른 동적 환경에서의 지연시간 최소화 연구가 요구된다.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A1047768) and a grant (22RITD-C161698-02) from Regional Innovation Technology Development Program funded by Ministry of Land, Infrastructure and Transport of Korean government.

REFERENCES

[1] M. Chen, B. Liang, M. Dong, “Multi-user multi-task offloading and resource allocation in mobile cloud systems,” IEEE Transaction on Wireless Communication. pp. 6790-6805, 2018

[2] Gao, G., Xiao, M., “Opportunistic Mobile Data Offloading with Deadline Constraints,” IEEE Transaction on Parallel Distributed System, 28, pp. 3584 -3599, 2017

[3] H. Jeong, H. Lee, C. Shin, S. Moon, “Ionn; Incremental offloading of neural network computations from mobile devices to edge servers,” Proceedings of the ACM Symposium on Cloud Computing, pp. 10-13, 2018

[4] Y. Lee, K. Nam, M. Jang, “Extracting optimal moving patterns of edge devices for efficient resource placement in an FEC environment,” Journal of the KIICE, 26(1), pp. 162-169, 2022

[5] L. Lin, X. Liao, H. Jin, P. Li, “Computation Offloading Toward Edge Computing,” Proceedings of IEEE 2019, 107, pp. 1584-1607, 2019

[6] A. Ndikumana, S. Ullah, T. LeAnh, N. Tran, C. Hong, “Collaborative Cache Allocation and Computation Offloading in Mobile Edge Computing.” Proceedings of the 19th APNOMS, pp. 366-369, 2017