부분 오프로딩을 위한 DAG의 분기구조 분할

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

Branch Structure Partitioning of DAG for Partial Offloading

Jae-seok Baik*, Min-seok Jang*, Yon-sik LeeO
*School of Computer Information & Comm. Engineering, Kunsan National University,
OSchool of Computer Information & Comm. Engineering, Kunsan National University

● 요 약 ●

본 논문은 FEC (Fog Edge Computing) 환경의 모바일 장치에서 요구되는 서비스의 구현 모듈을 에지 서버에 부분 오프로딩하기 위하여, 서비스 구현 모듈의 DAG 토폴로지에 포함된 분기구조의 분할 방법을 제안한다. 제안 방법은 최소-컷 문제를 적용하여 분기구조들의 오프로딩 여부 결정, 부분 모듈들의 실행위치 결정 및 최적 실행경로 추출에 유용하게 사용된다.

키워드: 부분 오프로딩(partial offloading), DAG 구조(DAG topology), 분기구조 분할(partition of branch structure)

I. Introduction

FEC는 리소스가 제한된 모바일 장치에서 에지 서버로 부분적인 계산 오프로딩을 수행하여 에너지 절약과 연산 처리 속도를 높일 수 있도록 한다. 계산 오프로딩은 프로그램 분석 및 프로파일링을 통해 모듈별 처리의 분할 및 처리 위치 결정을 요구한다.

본 논문은 모바일 장치에서 요구되는 서비스의 구현 모듈을 에지 서버에 부분 오프로딩하기 위하여, DAG 구조인 서비스 모델 토폴로지에 최소-컷 문제를 적용한 분기구조 분할 방법을 제안한다.

II. Preliminaries

1. Topology of the application service model

일반적으로 서비스 구현 모듈은 정점 vi (i = 1~n)로 구성되는 DAG 구조로 모델링된다. 실행단계에서 모바일 장치는 DAG에 따라 서비스 구현 모듈들을 분할하고, 오프로딩 정책에 따라 에지 서버에 부분 오프로딩을 수행한 후 전체 작업을 완료한다. m개의 구현 모듈을 각각 파티션 대상으로 간주하여 파티션 집합을 DAG = {DAG_1, DAG_2, · · ··, DAG_m}으로 정의한다. 각 DAG_i는 모바일 장치와 에지 서버와의 협력으로 실행되며, 로컬에서 처리되는 DAG^l_i와 에지 서버에서 처리되는 DAG^e_i로 구분한다. 그림 1_a)는 $t^l_1 > (t^e_1 + tt_1 + tt_2)$일 경우 DAG_1 모듈을 에지 서버로 오프로딩하여 처리하는 경우이며, 1_b)는 모듈 간의 종속성 분석을 기반으로 DAG_2 모듈을 에지 서버로 오프로딩하는 경우를 나타낸다.

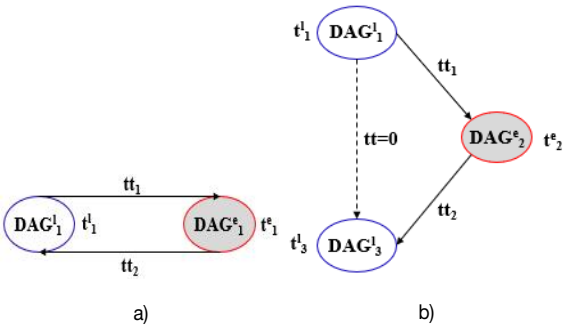


Fig. 1. Characteristics of DAG_i according to processing position

2. Computation offloading

지연시간 최소화는 최적의 계산 오프로딩 방법 결정을 통해 실현될 수 있다. 작업의 총 처리 지연시간은 로컬 및 에지 서버 처리시간과 전송시간으로 구성되며, 식 (1)과 같다. 이는 해당 기지국 내에서의

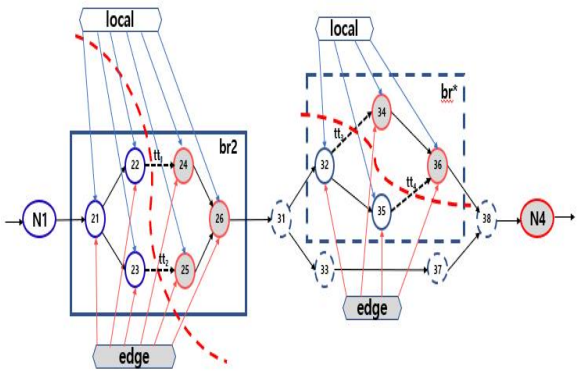
체류시간보다 짧고, 클라우드 서버 이용 시간보다 짧아야 함을 보장해야 한다.

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

식 (1)에서 T_l은 모든 DAG_i^l의 로컬 처리시간이며, T_e는 모든 DAG_i^e의 에지 서버 처리시간이다. T_{tt}는 오프로딩으로 인한 출력 전송시간의 합이다.

III. Branch Structure Partitioning

서비스 구현 모듈에 대한 부분 오프로딩 결정을 위하여, DAG 토폴로지의 각 모듈의 분기구조들을 분석하여 체인 토폴로지로 변환하고 이를 DAG로 재구성해야 한다. 먼저 구현 코드의 모듈 중 둘 이상의 분기를 가지는 DAG의 모든 다중 분기구조의 집합 (Bran = {br1, br2, · · ,bri, · ·})을 생성한다. 분기구조 bri는 분기의 정점들의 집합이며, 각 정점들은 부분 오프로딩 대상 모듈이 된다. 본 논문에서는 부분 오프로딩을 위하여, 최소-컷 문제를 이용하여 분기구조를 두 개의 부분 그래프로 분리(그림 2의 점선 표시)한다. 분리된 좌측과 우측 부분은 각각 로컬과 에지 서버에서 처리되며, 점선에 의해 절단된 간선들의 가중치 합은 총 전송시간이다.



a) b)

Fig. 2. Partitioning of Branch Structure Using Min-Cut Problem

그림 2_a) br2의 좌측은 로컬에서 처리되며 처리시간은 t^l_2 ($t^l_2=t^l_{21}+t^l_{22}+t^l_{23}$) 이고, 우측은 에지 서버에서 처리되며 처리시간은 t^e_2 ($t^e_2=t^e_{24}+t^e_{25}+t^e_{26}$) 이다. 최소-컷에 의해 잘려지는 두 개의 간선은 로컬에서 에지 서버로 출력을 전달하는 시간이며, 총 전달시간은 (tt_1+tt_2)이다. 따라서 br2의 총 처리시간은 ($t^l_2+t^e_2+tt_1+tt_2$)이며, DAG의 N2로 변환된다. 그림 2_b)와 같이 내포된 branch br*를 포함하는 br3와 같은 경우에는, 먼저 br*를 br2에 적용한 같은 방법으로 분리하여 노드들의 처리 위치를 결정한다. br*를 포함하는 br3는 br*를 하나의 노드로 간주하여 같은 방법으로 분리한다. 이때, br3의 로컬 처리시간에 br*의 에지 처리시간이 포함되지만 실행경로 추출을 DAG로 재구성되면 하나의 노드 처리시간으로 계산되므로 총 처리시간에는 영향을 주지 않는다. 그림 3은 N1은 로컬에서 N4는 에지에서 처리됨을 가정하에 br2와 br3가 각각 DAG 노드인 N2와 N3로 변환된 결과를 나타낸다. N1과 N4의 처리 위치에 따라 N2와 N3의 처리 지연시간이 변동될 수 있다.

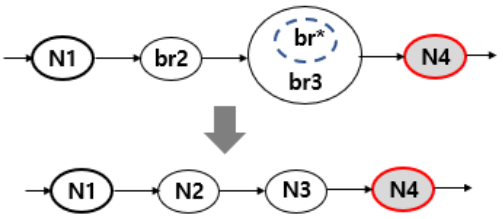


Fig. 3. Result of converting branch structure to DAG

N1~N4로 이루어진 DAG 구조는 노드별 최단 경로 알고리즘을 적용하여 최적의 부분 오프로딩 실행경로 추출이 가능하다.

IV. Conclusions

본 논문은 FEC 환경에서 부분 오프로딩을 위한 기반연구로써, 서비스 구현 모듈을 여러 개의 독립적 파티션으로 분할할 때 발생하는 분기구조 분할 방법을 제안하였다. 최소-컷 문제를 적용한 제안 방법은 다양한 분기구조들의 오프로딩 여부 결정과 최적 실행경로 추출에 유용하게 사용될 수 있다. 실제 부분 오프로딩을 통한 최소 처리시간은 이전 노드의 처리 위치와 관련이 있으므로 모든 노드의 최적 오프로딩 결정 방법을 찾는 것이 중요하다.

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