Fog/Edge 컴퓨팅 환경에서 효율적 오프로딩 기술

김규범* · 백승현 · 장민석 · 이연식

군산대학교

Efficient Offloading Technology in Fog/Edge Computing Environments

Gyu-Beom Kim* · Seung-Hyun Baek · Yon-Sik Lee · Min-Seok Jang

Kunsan National University

E-mail : slaow@naver.com / bluems@kunsan.ac.kr / yslee{msjang}@kunsan.ac.kr

요 약

최근 네트워크 에지에 배치된 컴퓨팅 리소스를 효율적으로 조정 관리함으로써 많은 수의 장치들 및 새로운 IoT 수요를 충족시키기 위한 Fog/Edge Computing(FEC)에 대한 다양한 연구가 진행되고 있다. 본 논문은 FEC 환경에서 실행 지연시간 최소화를 유도하기 위한 오프로딩 대상 결정 및 오프로딩 방법의 효율성 제고를 위한 주요 고려사항과 적용 방법들을 제시한다. 이는 향후 관련 이해 관계자들에게 필요한 FEC 프레임워크 구축에 효과적으로 적용될 수 있다.

ABSTRACT

Recently, various studies have been conducted on Fog/Edge Computing (FEC) to meet the demand for large numbers of devices and new IoT by efficiently coordinating and managing computing resources deployed on network edges. This paper presents key issues and its solutions for determining offloading targets and improving the efficiency of offloading methods to induce minimizing execution latency in FEC environments. TThe proposals in this paper can be effectively applied to building the FEC framework required by relevant stakeholders in the future.

키워드

Fog/Edge 컴퓨팅, 계산 오프로딩, 실행 지연시간, 이동에이전트

Ⅰ. 서 론

애플리케이션을 중앙 집중식 클라우드로 오프로딩하는 모바일 클라우드 컴퓨팅 방법은 IP기반의 네트워크 토폴로지 측면에서 네트워크 부하와 시간 지연을 초래한다[1,2,3]. 사용자 중심의 차세대 컴퓨팅 서비스를 위하여 사용자 기기로 부터의 정보 분석 필요성과 사용자 수의 증가에 따라 더 높은 성능과 더 많은 트래픽 처리를 위한 투자 요구가 급등하는 추세이며, 애플리케이션 서비스의 지연시간 문제 해결을 위하여 클라우드 서비스를 FEC 패러다임의 모바일 네트워크의 가장자리로 이동시켜 사용자 장비와의 근접성을 확보해야 함이 필수적이다[3,4]. 오프로딩은 컴퓨팅 자원 측면에서 낮은 지연시간 및 근접 지원기능과 모바일 리소스 제한을 확장하는 흥미로운 기술이며[3,4], 클라우드 컴퓨팅의 한계 극복을 위한 향후 타임라인에 실용적인 기술이며[5,6], FEC 패러다임은 오프로딩 기술을 위한 명확한 시너지 효과로써 시간 제약 애플리케이션에 필연적 적용이 예상된다[7].

본 논문은 이와 같은 환경에서 실행 지연시간 최소화를 유도하기 위한 오프로딩 대상 결정 및 오프로딩 방법의 효율성 제고를 위한 주요 고려사항과 이들의 해결 방법들을 제시한다.

Ⅱ. FEC 환경에서 효율적 오프로딩을 위한 주요 문제들과 해결 방법 제안

2.1 오프로딩 가능 부분 추출 및 결정

오프로딩 가능성, 대상 및 방법 등의 결정에 직접적인 영향을 주는 주요 요인은 처리 대상 애플리케이션의 모델 및 유형이며[8,9], 다음과 같은 기준으로 애플리케이션을 분류하여 적용이 가능하다.

1) 오프로딩 대상에 따른 분류: 다중 목적 오프

* speaker

로딩을 위한 코드의 분할 및 병렬화를 지원하는 애플리케이션 모델을 분류하고[9,10], 이를 기반으로 오프로딩 가능성을 적용한다.

2) 처리할 데이터양에 따른 분류: 데이터의 양을 미리 추정할 수 있는 여부로 구분하여 적용하며, 능동규칙 기반 에이전트 기술과 대량 코드의 오프로딩을 위한 분산객체 및 에이전트 경량화 기술 적용한다[6,17].

3) 오프로딩 대상의 상호 의존성 관련: 개별 성분 간의 관계를 종속성 그래프나 함수 호출 그래프를 사용하여 오프로딩 가능 부분을 적용한다[9].

위와 같은 오프로딩 결정 및 구현 프로세스를 위하여, 사용자 기기, FEC시스템 및 서버 관리자를 통하여 코드 분석(오프로딩 항목 결정), 시스템 분석(매개변수 모니터링) 및 결정 엔진(오프로딩 여부 결정) 등의 기능과 역할의 구현이 요구된다.

2.2 전체 및 부분 오프로딩

오프로딩 대상인 코드의 크기는 통신 및 실행 지연시간뿐만 오프로드 되는 코드를 탑재하여 이주시키는 이동에이전트의 이주 성능과 직접적인 연관이 있으므로[16,17], 전체 및 부분 오프로딩의 가능성 분석 기반의 오프로딩 방법 적용이 요구된다[7]. 애플리케이션을 로컬 또는 FEC 환경에서 처리할지의 오프로딩 결정 자체는 사용자 기기의 오프로딩 정책 모듈을 통해 결정하도록 한다.

1) 전체 오프로딩 모델

애플리케이션의 모델 및 크기와 이동에이전트에의 탑재 가능성을 우선 적용하고, 오프로딩을 위한 코드 기반, 스레드 기반, 프로세스 기반 등의 각각의 방법에 대한 분석을 기반으로 최적의 전체 오프로딩 수행 방법을 제안한다.

2) 부분 오프로딩 모델

부분 오프로딩은 애플리케이션의 구성 부분들 간의 상호 의존성 및 종속성에 대한 처리 방법이 필요하며, 기존 연구들에 의해 높은 시간복잡도의 비선형 문제로 공식화된 부분 오프로딩 결정은 애플리케이션의 분할 여부와 대상의 수, 사용자 기기 수에 영향을 받으므로[7,12,13], 본 논문에서는 조합 최적화 방법 기반의 적응 알고리즘[12]과 다중 사용자 기기 시나리오를 위하여 기기 간의 공정성[13], 지연시간 제약에 따른 우선순위[14] 및 임계값 적용 알고리즘을 통한 시간복잡도 감소를 유도할 수 있는 정책 기반의 모델 구현을 제안한다.

2.3 오프로딩 효율성을 위한 이동에이전트 적용

오프로딩 효율성과 관련한 기존 연구들의 주요 미흡한 점은 사용자 기기만의 지연 시간 고려, 정적 시나리오 기반의 오프로딩 및 다중 컴퓨팅 노드 대상의 오프로딩 미적용 등이다. 본 논문에서는 이들의 해결방법을 다음과 같이 제안한다.

1) 이동에이전트 미들웨어 플랫폼 기반 이동에이전트

센서네트워크 환경에서 사용자 기기들의 제어 및 Fog/Edge 환경에의 통합을 보장하는 미들웨어 플랫폼 구축과 이를 기반으로 오프로딩을 위한 이동에이전트의 모듈 탑재 및 이주 모델의 구현이 필요하다[15,17]. 이동에이전트 이주는 라디오 스택의 추상화 기술을 확장 적용하고, 탑재 모듈의 크기와 성격에 따른 D2D(Device-to-Device) 기술의 구현이 요구된다.

이동에이전트 이주 모듈은 별도의 이주 리스트를 적용하여 이주대상을 결정하며, 이주 시 최적 이주경로 탐색 및 조정 알고리즘을 적용하고, 탑재되는 모듈의 크기로 인한 이주 지연시간 문제 해결을 위하여 분산객체 개념[17]과 푸시기능을 포함하는 에이전트 경량화 기술[18] 적용을 제안한다.

2) 최적 이주경로 탐색 및 조정 알고리즘

오프로딩을 위한 이동에이전트의 이주시간 감소 및 이주 신뢰성 보장을 위하여, 최적 이주경로 탐색 알고리즘과 네트워크 장애 시 이주경로를 조정하는 알고리즘이 필수적으로 요구된다[15].

최적 이주경로 탐색 알고리즘은 FEC 환경에서 트래픽 분석 기반의 지연시간 최소화를 위하여, 패킷 송/수신시간과 패킷 분실 수 기반의 라우팅 테이블 재구성 방법을 적용하며[18], 경로 조정 알고리즘은 장애 노드 판단을 위하여 이주 대기시간 임계값을 적용하고 에이전트 객체의 분실방지를 위한 객체 복제 방법 적용을 제안한다.

개발 시 이동에이전트를 스레드 형태로 동작시키고, 이주, 실행, 반환 등의 필수적 기능들을 메소드 형태로 구현하며, 경로결정 지연시간과 이주 대상노드 수 사이의 trade-off 분석을 통하여 총 지연시간의 최소화를 유도하도록 구현한다.

3) 다중 컴퓨팅 노드 대상 오프로딩 적용

이동에이전트 기술을 적용하여 해결하고자 하는 다중 컴퓨팅 노드로의 오프로딩은 이동에이전트의 작업 처리시간이 성능 및 효율성 향상 문제와 직결되므로[9,10,15], 이동에이전트 구성 및 이주 방법 적용이 매우 중요하다[17,18]. 다중 컴퓨팅 노드로의 오프로딩은 이동에이전트의 병렬 이주모델과 네임 스페이스의 메타 정보를 이용한 능동적 라우팅 스케줄링 방법 적용을 제안한다.

Ⅲ. 결 론

본 논문은 FEC 환경에서의 오프로딩에 관한 기존 연구들에서 수행이 미흡하거나 적용되지 않은 애플리케이션 서비스의 지연시간을 사용자 기기뿐만 아니라 FEC 환경 측면에서도 적용하는 것, 동적 시나리오 대상의 오프로딩 결정, 다중 컴퓨팅 노드 대상 오프로딩 방법, 이동에이전트 기반 기술 적용 등에 대한 연구 제안으로써, FEC 환경에서 이동에이전트 기반의 효율적 오프로딩 기술의 실

질적인 모델을 구축하기 위하여 필요한 주요사항에 대한 고찰 및 방법에 대한 제안이다.

특히, 자율성과 이동성을 지원하는 이동에이전트 기술은 복잡한 문제를 세분화하여 해결하는 수단으로 다양한 융합공학에서 많은 관심을 받고 있으며, 이를 적용한 다중 목적 오프로딩 기술은 기존의 일부 다중에이전트 적용 기술과 비교하여 개념적 차별성을 가진 독창적 연구 제안이다. 고도화된 FEC 기술은 자원 활용성을 높이며, 사용자 중심의 설계를 도입하면 FEC 자원을 사용자 수준에서 제어 및 거래가 가능하므로 보다 빠른 IoT 시대의 도래를 가능하게 하며, 현재 클라우드 서비스의 구조적 문제 해결을 위한 창의적 기술로의 발전 가능성이 명확하다.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1B07051045).

References

[1] R. Buyya, S. Srirama (ED), Fog and Edge Computing: Principles and Paradigms, and Applications, 1st ed, John Wiley & Sons Inc., 2019.

[2] X. Sun, N. Ansari, "EdgeIoT: Mobile Edge Computing for the Internet of Things," *IEEE Communications Magazine*, 54(12), pp. 22-29, 2016.

[3] Mach, P., Becvar, Z., "Mobile edge computing: A Survey on Architecture and Computation Offloading," *IEEE Communication Survey Tutorial*, 19, pp. 1628-1656, 2017.

[4] Y. Liu, C. Xu, Y. Zhan, Z. Liu, H. Zhang, "Incentive Mechanism for Computation Offloading using Edge Computing," *Computer Networks*, 129(2), pp. 339-409, 2017.

[5] Ji. Cheng, Y. Shi, B. Bai, and W. Chen, "Computation Offloading in Cloud-RAN Based Mobile Cloud Computing System," *IEEE International Conference on Communications*, pp. 1-6, 2016.

[6] Josilo, S., Dan, G., "Selfish Decentralized Computation Offloading for Mobile Cloud Computing in Dense Wireless Networks," *IEEE Trans. Mobile Computing*, 18, pp. 207-220, 2018.

[7] C. You, K. Huang, H. Chae, B. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," *IEEE Trans. on Wireless Communications*, 16(3), pp. 1397-1411, 2017.

[8] Y. Mao, et. al., "Power-Delay Tradeoff in Multi-User Mobile-Edge Computing Systems," *IEEE Global Communications Conference*, pp. 1-6, 2016.

[9] Chen, M,. Liang, B., Dong, M., "Multi-user Multi-task Offloading and Resource Allocation in Mobile Cloud Systems," *IEEE Trans. Wireless Communication*, 17, pp. 6790-6805, 2018.

[10] X. Chen, L. Jiao, W. Li, X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, 24(5), pp. 2795-2808, 2016.

[11] C. You and K. Huang, "Multiuser Resource Allocation for Mobile Edge Computation Offloading," *IEEE Global Communication Conference*, pp. 1-6, 2016.

[12] Alameddine, H., et. al., "Dynamic Task Offloading and Scheduling for Low-latency IoT Services in Multi-access Edge Computing," *IEEE Journal Sel. Areas Communication*, 37, pp. 668-682, 2019.

[13] Al-khafajiy, M., et. al., "Improving Fog Computing Performance via Fog-2-Fog Collaboration," *Future Generation Computing System*, 100, pp. 266-280, 2019.

[14] J. Dolezal, Z. Becvar, T. Zeman, "Performance Evaluation of Computation Offloading from Mobile Device to the Edge of Mobile Network," *IEEE CSCN*, pp. 1-7, 2016.

[15] P. Ardakani, "A Mobile Agent Routing Protocol for Data Aggregation in Wireless Sensor Networks," *International Journal of Wireless Information Networks*, 24(1), pp. 27-41, 2017.

[16] D. Parygin, et. al., "Multi-Agent Approach to Distributed Processing Big Sensor Data based on Fog Computing Model for the Monitoring of the Urban Infrastructure Systems," *Proc. of International Conf. on SMART*, pp. 305–310, 2016.

[17] Y. Lee,"Distributed Control Framework based on Mobile Agent Middleware," *Journal of The KSCI*, 25(12), pp. 195-202, 2020.

[18] Y. Lee, "Lightweight and Migration Optimization Algorithms for Reliability Assurance of Migration of the Mobile Agent," *Journal of The KSCI*, 25(5), pp. 91-98, 2020.