엣지/클라우드 컴퓨팅 환경에서의 서비스 캐싱 연구 동향

함동호, 곽정호
대구경북과학기술원

요 약

AR/VR, 고화질 스트리밍, 고FPS 게임 등 6G의 대표적 서비스들은 네트워킹과 컴퓨팅, 스토리지 자원을 함께 요구한다. 그러나 모바일 단말의 컴퓨팅 및 스토리지 자원 한계로 인해 원격 서버로부터 컴퓨팅 자원을 빌려 사용하는 코드 오프로딩 기술이 필수적이다. 원격 서버의 거리로 인한 전송 지연 문제를 해결하기 위해 MEC(Mobile Edge Computing) 기술이 각광받고 있으며, 이와 동시에, 클라우드 대신 최종 사용자와 가까운 엣지 서버에 필요한 서비스를 위치시키는 서비스 캐싱 기술 또한 주목을 받고 있다. 본고에서는 엣지/클라우드 컴퓨팅 환경에서의 서비스 캐싱의 필요성과 오프로딩 간의 관계를 알아보고, 관련 연구 동향에 대해 살펴본다.

Ⅰ. 서 론

모바일 기기 및 네트워크의 발전 그리고 클라우드 컴퓨팅의 등장으로 모바일 서비스를 목표로 한 여러 6G 서비스들이 등장하였다. 그러나 이러한 6G 서비스들은 대부분 높은 컴퓨팅 능력과 스토리지 자원, 그리고 낮은 지연을 필요로 하기 때문에, 여전히 모바일 기기 혹은 클라우드 컴퓨팅 만으로는 이러한 서비스를 완벽하게 즐기기 어려운 실정이다.

모바일 기기의 처리 지연 문제와 클라우드 컴퓨팅의 전송 지연 문제의 타협적인 해결책으로서, MEC(Mobile Edge Computing)기술이 제안되었다. MEC는 사용자에게 충분한 컴퓨팅 및 저장 자원을 물리적으로 더 가까이 배치함으로써, 처리 지연과 전송 지연을 모두 줄일 수 있다. 시장조사기관인 MarketandMarket 에 의하면 〈그림 2〉와 같이 엣지 컴퓨팅의 시장의 규모는 연평균 30.45%의 고성장을 통해 급증하고 있다. 이러한 성장세는 2024년에 전세계 기준 약 90억 달러에 이를 것으로 전망된다.

기존의 모바일 기기나 클라우드 컴퓨팅에 비해 많은 장점을 가진 엣지 컴퓨팅이지만 유지 및 보수에 있어서 단점은 바로 엣지 컴퓨팅을 사용하는데 드는 비용이다. 일반적으로 엣지 컴퓨팅 및 스토리지 자원은 클라우드 자원보다 비싼 것으로 알려져 있는데, 실제로 Amazon AWS에서 제공하는 엣지 자원은 클라우드의 자원에 비해 약 4배 정도의 가격이다[1]. 따라서, 모든 서비스를 MEC를 통해 제공하는 것은 비용 효율적이지 않으며, 서비

그림 1. 대표적인 6G 킬러 서비스들

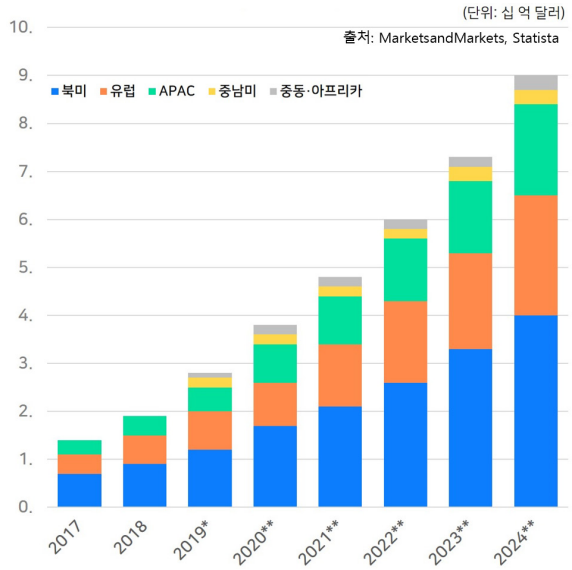


그림 2. 전 세계 지역별 엣지 컴퓨팅 시장 전망

표 1. 엣지 컴퓨팅과 클라우드 컴퓨팅 비교

	엣지 컴퓨팅	클라우드 컴퓨팅
접근성	네트워크 엣지 혹은 그 근처	중앙 데이터 센터
가격	상대적으로 비쌈	상대적으로 저렴함
처리지연	짧은 대기시간 및 실시간 처리	긴 대기시간
활용 어플리케이션	IoT, 모바일 AI 등 실시간 처리 어플리케이션	빅데이터, 머신러닝 등 대용량 컴퓨팅 처리 어플리케이션

스 제공업체는 엣지 및 클라우드 자원 활용을 최적화하여 엣지 컴퓨팅 사용과 서비스 품질 사이의 균형을 맞추어야 한다. 예를 들어, NVIDIA GeforceNow는 강력한 컴퓨팅 능력을 갖춘 클라우드 기반 게임 서비스를 제공한다. GeforceNow의 "Priority" 서비스는 1080p 해상도에서 60 fps를 보장하지만, 원격 게임 플레이를 하는 위치가 클라우드 서버로부터 많이 떨어져 있다면, 실제 성능을 보장할 수 없다. 다시 말해, 엣지 컴퓨팅 자원을 효과적으로 활용하기 위해서는 모바일 기기, 엣지, 그리고 클라우드 서버 간의 시간에 따라 변하는 네트워킹 및 컴퓨팅 환경을 고려한 서비스 캐싱과 오프로딩에 대한 적응적인 결정이 필요하다. 이를 위해서는 원격 서버들의 특성을 파악하는 것이 매우 중요하다. 〈표 1〉은 엣지 컴퓨팅과 클라우드 컴퓨팅의 장단점을 정리한 표이다. 일반적으로 엣지 서버는 클라우드 서버에 비해 사용자와 가까이 위치하기 때문에 네트워크 지연 측면에서 이득이 있다. 그러나 이는 곧 사용자와 가까이 위치한 여러 엣지 서버를 구축하는 것과 같은 의미이고, 자연스럽게 유지 및 보수 비용이 클라우드 서버보다 상대적으로 비싼 편에 속한다. 그럼에도 불구하고 IoT 기기나, 모바일 AI기반 서비스 등 실시간 처리 어플리케이션에서는 엣지 서버의 활용도가 굉장히 높다. 반대로, 클라우드 서버는 사용자와의 거리가 멀어 엣지 서버에 비해 높은 네트워크 지연을 가지지만 더 강력한 컴퓨팅 능력을 지니고 있다. 따라서 빅데이터 분석 혹은 머신러닝 등 대용량 컴퓨팅 처리 어플리케이션에 적합하다. 물론 두 원격 서버 모두 모바일 기기의 컴퓨팅 성능을 월등히 상회하며, 활용 방안이 해당 어플리케이션들에만 국한되지 않는다.

본 고에서는 엣지/클라우드 컴퓨팅 환경에서의 서비스 캐싱 및 오프로딩 기술에 대해 살펴보고 더 나아가 실제로 연구되고 있는 최신 서비스 캐싱 연구동향에 대해 알아본다.

Ⅱ. 서비스 캐싱과 코드 오프로딩

서비스 캐싱 기술은 서비스와 관련된 라이브러리와 데이터베이스를 엣지 서버에 저장하는 기술이다[2]. 엣지 서버의 컴퓨팅 자원과 저장 자원을 모두 활용한다는 측면에서 서비스 캐싱은 모바일 단말에서 프로세싱을 요구하는 워크로드를 엣지 서버에서 대신 실행하게 하는 코드 오프로딩과 자원활용에서의 유사성이 있다.

1. 서비스 캐싱

먼저 〈그림 3〉은 서비스 캐싱 기술의 예시를 나타낸 그림이다. 일반적으로, 자원이 넉넉한 클라우드는 많은 서비스들을 저장할 수 있지만, 자원이 제한되어 있는 엣지 서버에는 사용자의 요구사항을 제일 많이 반영할 수 있는 서비스들로 구성하는 것이 가장 비용 효율적이다. 따라서, 엣지 서버에서는 사용자들의 네트워크 상황, 서비스의 인기도, 엣지 서버의 컴퓨팅 사용률 등 다양한 환경 변수를 고려하여야만 가장 최적의 서비스 캐싱 결정을 내릴 수 있다.

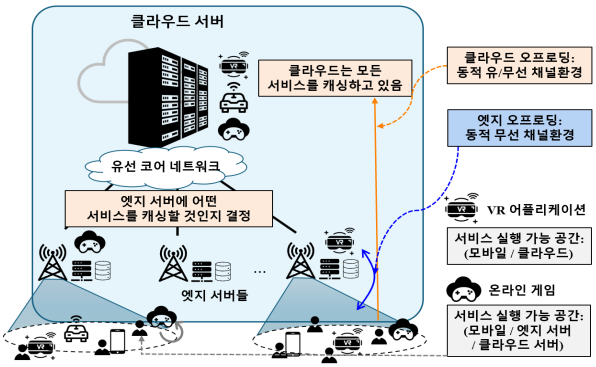


그림 3. 서비스 캐싱의 예시

만약 서비스가 엣지 서버에 캐싱되지 않았다면 사용자는 해당 서비스를 모바일 혹은 클라우드에서 처리할 수 있게 된다. 반대로, 서비스가 엣지 서버에 캐싱되어 있다면 사용자에게는 세가지 경우의 수(모바일, 엣지, 클라우드)가 존재한다. 이렇듯 서비스 캐싱이 적절하게 이루어진다면, 사용자들에게 더 많은 선택지를 제공할 수 있고 사용자들이 자신의 환경에 알맞은 최선의 서비스 실행 방식을 선택할 수 있기 때문에, 이는 바로 사용자들의 서비스 품질 향상으로 이어진다.

2. 코드 오프로딩

코드 오프로딩 기술은 모바일 기기에서 발생하는 서비스 작업을 모바일, 엣지 서버 혹은 클라우드 서버 중 어디에서 처리할 지 결정하는 기술이다. 〈그림 4〉는 오프로딩 기술을 설명하는 간단한 예시이다. 사용자는 모바일 기기에서 작업하기 어려운 서비

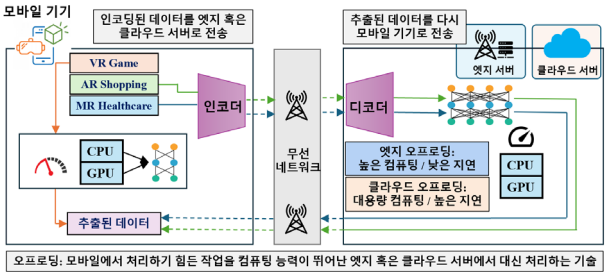


그림 4. 코드 오프로딩의 예시

스들을 원격 서버에 요청하여 대신 처리할 수 있는데, 이를 코드 오프로딩이라 부른다.

사용자는 서비스의 특성에 따라 엣지 서버 혹은 클라우드 서버에 오프로딩을 요청할 수 있고, 만약 네트워크가 혼잡한 상태 등으로 인해 코드 오프로딩을 하기에 적합하지 않은 상황이라면 모바일 기기 내에서 직접 서비스를 실행할 수도 있다. 만약, 사용하려는 서비스가 단순 체스 게임 같이 간단한 서비스라면 모바일 기기의 자원을 활용하여 게임을 플레이 하는 것이 효율적일 것이다. 반대로, 사용하려는 서비스가 고성능을 요구하는 게임이거나 AI 기반 사물인식 서비스 등 높은 컴퓨팅 능력을 요구하는 서비스라면, 네트워크 지연을 감내하더라도 원격 서버에서 서비스를 처리하는 것이 효과적일 것이다.

정리하자면 서비스 캐싱 결정은 사용자들의 서비스 품질을 높이기 위한 더 나은 선택지를 가질 수 있게 해주며, 이는 사용자들의 오프로딩 선택에 직접적으로 영향을 받는다. 그러나 제한된 자원의 엣지 서버에서는 모든 서비스를 캐싱할 수 없기 때문에 사용자들의 네트워크 상태, 서비스 인기 및 코드 오프로딩 결정과 같은 요소들을 고려하여 공동으로 고려되어야 한다.

3. 컴퓨팅 성능과 네트워크의 영향

앞서 언급한 내용에 따르면, 서비스 캐싱은 처리하는 서버의 컴퓨팅 성능과 네트워크의 영향에 따라 결정되는 것을 알 수 있다. 우리는 실제로 컴퓨팅 성능은 좋지만 멀리 떨어져 네트워크 상황이 좋지 않은 클라우드 서버에 비해 엣지 서버가 가질 수 있는 효용성을 확인하기 위해 간단한 실험을 진행하였다.

〈그림 5〉는 고성능 자원을 요구하는 게임 중 하나인 원신 모바일 게임을 원격 서버에서 처리하여 플레이할 때와, 모바일 기기의 CPU/GPU에서 처리하여 플레이할 때의 성능을 나타낸 그래프이다. 왼쪽 그래프에서 모바일 단말에서 직접 서비스를 처리하는 것보다 원격 서버에서 처리하는 것이 성능과 사용 자원 측면에서 뛰어남을 확인할 수 있다. 오른쪽 그래프는 호스트 서버가 대구에, 모바일 단말은 각각 대구와 서울에 있을 때의 원격플레이 성능 비교 그래프이다. 이때, 서울에서 HD 해상도로 원신을 원격플레이하는 것이 대구에서 4K 해상도로 원격플레이하는 것보다 더 높은 (처리+전송) 지연이 발생하는 것을 확인할 수 있다. 즉, 컴퓨팅 성능과 네트워크 환경은 서비스 성능에 직접적인 영향을 준다는 것을 알 수 있다.

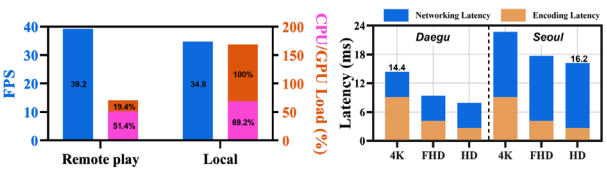


그림 5. 원격플레이의 위치에 따른 컴퓨팅 성능과 네트워크의 영향

Ⅲ. 서비스 캐싱 연구 동향

본 장에서는 서비스 캐싱과 코드 오프로딩을 동시에 다룬 연구 동향에 대하여 알아본다. 서비스 캐싱과 오프로딩에 대한 기존 연구들은 정적 및 동적 전략으로 분류될 수 있다. 서비스 캐싱 결정이 시간대를 고려하지 않고 이루어진다면(즉, 시간이 지남에 따라 엣지 서버에 저장되는 서비스가 변화하지 않는다면), 정적 서비스 캐싱으로 정의된다. 반면에, 서비스 캐싱 결정이 다양한 환경에 적응하며 시간이 지남에 따라 변하는 경우, 동적 서비스 캐싱으로 정의된다. 비슷하게, 사용자가 모바일 기기내에서 직접 처리하는 옵션 없이 항상 작업을 엣지 혹은 클라우드 원격 서버로 오프로딩한다면, 정적 오프로딩으로 부른다. 반면에, 사용자가 다양한 환경에 적응하여 오프로딩과 모바일 직접 처리 사이의 오프로딩 정책을 선택할 수 있다면, 동적 코드 오프로딩으로 정의한다.

〈그림 6〉은 서비스 캐싱과 오프로딩을 동시에 적용하였을 때, 각각의 기술이 동작하는 네 가지 조합에 대한 개념도를 나타낸 그림이다.

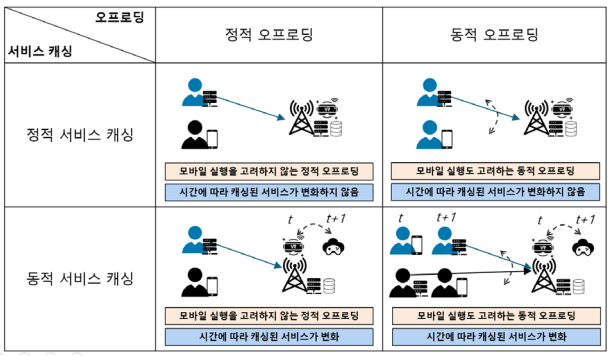


그림 6. 서비스 캐싱과 오프로딩 연구의 개념도

1. 서비스 캐싱과 오프로딩 동시 최적화 연구

서비스 캐싱과 오프로딩의 동시 최적화 연구들의 목적은 일반적으로 사용자들로 하여금 가장 최선의 서비스 처리 방식을 구하기 위함이다. 다시 말해, 사용자가 원하는 서비스가 엣지 서버에 캐싱되어 있는지, 사용자가 모바일 기기로 서비스를 직접 처리할 수 있는지 등 여러 환경에 대해서 서비스 캐싱과 오프로딩 결정을 최적화 하여 사용자들의 서비스 품질 향상을 이끌어낼 수 있도록 하는 것이 연구의 주된 목적이다.

가. 정적 서비스 캐싱/정적 오프로딩:

이 경우는 〈그림 6〉의 첫 번째 경우에 해당한다. 그림에서도 알 수 있듯이, 이러한 상황을 다룬 연구들은 모바일 실행을 고려하지 않고 항상 엣지 서버로 모바일 서비스 작업을 오프로딩하는 정적 오프로딩 환경을 고려한다. 마찬가지로 서비스 캐싱 결정도 시간에 따라 변화하지 않는 정적 서비스 캐싱 전략을 유지한다. 정적 서비스 캐싱/정적 오프로딩 전략을 제안한 연구들의 예시는 다음과 같다. Ko et al.은 동종/이종 서비스 환경에서 가장 인기있는 서비스를 엣지 서버에 저장하는 정적 서비스 캐싱 전략을 연구하였다[2]. 또한, Bi et al.은 하나의 서비스가 상호작용하는 여러 작업들로 이루어져 있을때, 이를 고려하여 서비스를 저장하는 정적 서비스 캐싱 알고리즘을 개발하였다[3]. 그러나 이러한 연구들은 사용자들이 동시에 엣지 서버를 사용하는 사용을 고려하지 않거나, 모바일 기기 내의 처리 가능성을 염두에 두지 않았다. 추가적으로 이러한 연구들은 서비스 캐싱과 오프로딩 결정을 정적으로 하기 때문에, 시간에 따라 다양하게 변화하는 환경에 대처하기 어렵다는 한계점을 가지고 있다.

나. 정적 서비스 캐싱/동적 오프로딩:

이 경우는 〈그림 6〉의 두 번째 경우에 해당한다. 첫 번째 경우처럼 서비스 캐싱의 경우 시간에 따라 서비스 캐싱의 결정이 변화하지는 않지만, 서비스를 모바일에서 실행하는 경우도 고려하기 때문에 조금 더 다양한 환경에 대응할 수 있다. 정적 서비스 캐싱/동적 오프로딩을 제안한 연구들의 예시는 다음과 같다. Dai et al.은 엣지 서버와 연동하여 작업을 처리할 수 있는 AIoT (Artificial Intelligence of Things)기기의 비용(지연 등)을 최소화 하기위한 인기도 기반의 정적 서비스 캐싱 전략을 제안하였다[4]. 추가적으로, Yan et al.은 서비스의 저장 비용이 서비스별로 다른 환경에서, 엣지 서버가 기지국에 장착되어 있는 경우에 기지국의 비용적 이득을 최대화하기 위한 서비스 캐싱 전략을 연구하였다[5]. 그러나, 이러한 연구들은 동적으로 모바일 기기와 엣지 서버 중 성능이 좋은 처리 방식을 선택할지라도, 서비스 캐싱 결정이 시간에 따라 변화하지 않기 때문에 실시간으로 변화하는 사용자들의 오프로딩 요구를 온전히 반영하기 어렵다는 한계점들이 있다.

다. 동적 서비스 캐싱/정적 오프로딩:

이 경우는 〈그림 6〉의 세 번째 경우에 해당한다. 두 번째 경우와 반대로, 서비스 캐싱 결정은 시간에 따라 변화할 수 있지만 서비스가 모바일에서 처리되는 경우는 고려하지 않는다. 정적 오프로딩만을 허용하는 환경에서 동적 서비스 캐싱을 다룬 연구들은 다음과 같다. Xu et al.은 사용자들의 서비스 수요에 따라 서비스 캐싱을 동적으로 관리하는 정적 코드 오프로딩 정책에 기반한 다중 셀 MEC 시스템을 제안하였다[6]. 또한, Ouyang et al.은 정적 코드 오프로딩 정책 하에서 MEC 시스템에서 이동성을 고려한 동적 서비스 캐싱 정책을 연구하였다[7].

최근에는 서비스 캐싱과 정적 코드 오프로딩의 결합 최적화 문제를 해결하기 위한 유망한 접근법으로 강화학습이 주목을 받고 있다. 예를 들어, Sun et al.은 계층적 심층 강화 학습(DRL) 프레임워크를 제안하여 DRL 기반 동적 서비스 캐싱과 정적 코드 오프로딩 프레임워크를 효과적으로 해결할 수 있도록 한 연구 방법을 제안하였다[8]. 그러나 이러한 연구들은 모바일 기기내 처리의 잠재력을 충분히 탐구하지 않았으며, 따라서 동적 서비스 캐싱 정책 하에서 최적화된 동적 오프로딩 정책이 제안되지 않았다.

라. 동적 서비스 캐싱/동적 오프로딩:

이 경우는 〈그림 6〉의 마지막 경우에 해당한다. 서비스 캐싱 결정은 다양한 사용자의 요구에 맞춰서 바뀔 수 있고, 서비스의 모바일 실행 또한 고려한다. 실제로 사용자들의 환경 및 요구사항은 실시간으로 달라지기 때문에 서비스 캐싱 결정이 이에 맞춰서 동적으로 이루어져야 한다. 동적 오프로딩을 허용하는 환경에서 동적 서비스 캐싱 전략을 구사한 연구는 다음과 같다. Ham et al.은 3계층(모바일-엣지-클라우드) 네트워크에서 모바일 기기의 비용(에너지, 원격 서버 이용 비용)을 최소화하는 동적 서비스 캐싱 및 자원할당 문제를 푸는 알고리즘을 제안하였다[9].

2. 경제적 관점에서 서비스 캐싱과 자원관리

앞서 서론에서 언급한 것처럼 서비스 캐싱은 비용적인 측면에서 예민하게 반응하기 때문에, 경제적 관점에서 서비스 캐싱과 자원관리를 다룬 여러 연구들이 존재한다. 이 연구들의 목적은 일반적으로 서비스 제공자들이 사용자들에게 엣지 컴퓨팅을 제공하고자 할 때, 가격이 비싼 엣지 서버를 어느 가격에 얼마나 구비해야 하는지 등을 다룬다.

가. 2계층(사용자-엣지): 경쟁적 모델

먼저, 사용자와 엣지 두 가지의 계층이 존재하는 상황에서의 연구들이 존재한다. 이 연구들은 일반적으로 모바일 기기에서

서비스를 엣지 오프로딩으로 실행할 때, 엣지 서버에 계산 작업을 오프로딩하여 지불할 수 있는 상황을 고려한다. 다시 말해, 모바일 기기내의 컴퓨팅 자원이 부족할 때, 비용을 지불하게 된다면 더 나은 서비스의 성능을 얻을 수 있다는 것이다. 이때, 엣지 서버는 제한된 저장 공간을 갖기 때문에 모든 서비스를 저장할 수 없다. 따라서, 사용자들로부터 엣지 컴퓨팅 이용 비용을 최대로 얻을 수 서비스를 선택하여 저장해야 한다. 엣지 서버에 어떤 서비스를 캐싱 할지에 대한 전략은 자연스럽게 사용자들의 오프로딩 결정에 의해 영향을 받게 된다. Tütüncüolu et al.은 이러한 상황에서 기지국에 연결된 엣지 서버가 서비스의 엣지 컴퓨팅 이용 가격을 공표하고, 서비스의 엣지 컴퓨팅 이용 가격에 따라 사용자가 엣지 서버로 오프로딩 할지 혹은 처리 지연을 감수하고 비용을 줄일 수 있는 모바일 내 처리를 선택할 것인지를 결정하는 사용자-엣지 서버 경쟁적 알고리즘을 제안하였다[10].

그러나, 실제로는 대부분의 서비스 제공업체가 서비스가 배치된 엣지 시설을 운영하지 않기 때문에 2계층 모델보다는 엣지 시설 자체를 공급해줄 수 있는 세력이 존재하는 3계층 모델이 더 현실적이다. 아직까지 3계층 모델에서 서비스 캐싱을 다룬 연구는 부족한 상황이나, 오프로딩을 고려한 자원관리 연구들은 활발하게 진행되었다. 3계층 모델에서는 엣지 운영자와 서비스 제공업체 간 이해관계의 충돌이 발생할 수 있고, 혹은 3계층 모든 계층들이 서로 협력하여 시스템 전체의 효율을 올리기 위해서 노력하기도 한다.

나. 3계층(사용자-엣지-클라우드): 순환적 모델

먼저 3계층 시스템에서 순환적 모델은 3계층의 모든 계층들이 협력을 기반으로 시스템 전체의 효율을 올리는 모델이다. 대부분의 서비스들은 엣지 노드의 컴퓨팅 능력 및 네트워크 자원을 포함한 다양한 자원에서 사용자 데이터뿐만 아니라 서비스 제공업체(클라우드)의 지원도 필요로 한다. Ma et al.은 엣지 노드와 서비스 제공업체의 이익은 각각 독립적으로 고려되는 상황에서 서비스 제공업체와 엣지 운영자 간의 협상을 통해 사용자의 만족도를 최상으로 높일 수 있도록 하는 알고리즘을 제안하였다[11]. 이 연구에서 사용자는 서비스 제공자에게 서비스를 요청하고, 서비스 제공자는 서비스를 엣지에 배치하며 엣지에서는 사용자들을 서비스하는 순환적 모델을 기반으로 문제를 형성하였고, 서비스 제공업체와 엣지 운영자 간의 협상을 통해 이를 해결하였다.

다. 3계층(사용자-엣지-클라우드): 경쟁적 모델

다음으로 3계층이 서로 경쟁적인 성향을 띠고 있는 경쟁적 모델에 대한 연구이다. 이러한 상황에서 사용자는 저렴한 가격을 통해 높은 품질의 서비스를 이용하고 싶을 것이고, 엣지 서버의 서비스 사업자는 사용자로 부터는 많은 돈을, 클라우드 서버의 클라우드 사업자로 부터는 싼 가격에 서비스 환경을 이용하고 싶어한다. 마지막으로, 클라우드 사업자는 자신이 어느 정도의 자원을 가져야 하며, 어느 정도의 가격으로 제공해야 자신의 이익이 가장 최대화되는지 고민한다. 이러한 철학을 바탕으로 Kim et al.은 엣지 서버가 사용자와 클라우드 사이에서 중간다리의 역할을 할 때, 엣지-사용자 사이의 오프로딩 비용을 결정하고 엣지-클라우드 사이의 엣지 서버 배치 비용 및 엣지 서버 설치 대수를 결정하는 문제를 푸는 알고리즘을 제안하였다[12].

Ⅳ. 결 론

본 고에서는 엣지/클라우드 환경에서 서비스 캐싱 기술이 가지는 의미와 연구 동향에 대해서 살펴보았다. 서비스 캐싱 기술의 경우 클라우드 자원보다 비싼 엣지 서버의 한정된 자원을 최대한 효율적으로 사용하기 위해 필수적인 기술이며, 자연스럽게 오프로딩 기술과도 엮여있음을 확인하였다. 서비스 캐싱의 최신 연구 동향들에 대해서도 알아보았으며, 서비스 캐싱 연구는 서비스 캐싱 전략과 오프로딩 전략의 유동성을 기반으로 나누어 분석할 수 있음을 확인하였다. 추가적으로, 비용적인 측면이 항상 고려되어야 하는 엣지 서버의 특성상 경제적 관점에서 서비스 캐싱을 다룬 연구들이 많이 진행되었고, 계층의 크기에 따라 다양한 방식으로 연구를 진행할 수 있음을 파악하였다.

Acknowledgement

이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2022-0-01053, 다중 통신기술 네트워크 로드밸런싱 기술개발). 또한, 본 연구는 과학기술정보통신부에서 지원하는 DGIST 기관고유사업에 의해 수행되었습니다 (23-HRHR-01).

참 고 문 헌

[1] Amazon, “Amazon AWS,”[Online]. Available: https://aws.amazon.com/.

[2] S.-W. Ko, S. J. Kim, H. Jung, and S. W. Choi, “Computation offloading and service caching

for mobile edge computing under personalized service preference," IEEE Transactions on Wireless Communications, vol. 21, no. 8, pp. 6568-6583, Aug. 2022.

[3] S. Bi, L. Huang, and Y.-J. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," IEEE Transactions on Wireless Communications, vol. 19, no. 7, pp. 4947-4963, Jul. 2020.

[4] X. Dai, Z. Xiao, H. Jiang, M. Alazab, J. C. S. Lui, G. Min, S. Dustdar, and J. Liu, "Task offloading for cloud-assisted fog computing with dynamic service caching in enterprise management systems,"IEEE Transactions on Industrial Informatics, vol. 19, no. 1, pp. 662-672, Jan. 2023.

[5] J. Yan, S. Bi, L. Duan, and Y.-J. A. Zhang, "Pricing-driven service caching and task offloading in mobile edge computing," IEEE Transactions on Wireless Communications, vol. 20, no. 7, pp. 4495-4512, Jul. 2021.

[6] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in Proc. of IEEE INFOCOM, Honolulu, USA, Apr. 2018, pp. 207-215.

[7] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," IEEE Journal on Selected Areas in Communications, vol. 36, no. 10, pp. 2333-2345, Oct. 2018.

[8] C. Sun, X. Li, C. Wang, Q. He, X. Wang, and V. C. M. Leung, "Hierarchical deep reinforcement learning for joint service caching and computation offloading in mobile edge-cloud computing," IEEE Transactions on Services Computing, early access, Jan. 2024.

[9] D. Ham, Y. Kim, and J. Kwak, "Dynamic interplay between service caching and code offloading in mobile-edge-cloud networks," in Proc. of IEEE SECON, Madrid, Spain, Sep. 2023, pp. 339-347.

[10] F. Tütüncüolu, and G Dán, "Joint Resource Management and Pricing for Task Offloading in Serverless Edge Computing," IEEE Transactions on Mobile Computing, early access, Nov. 2023.

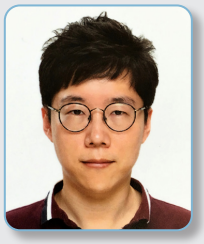
[11] S. Ma, S. Guo, K. Wang, W. Jia, and M. Guo, "A cyclic game for service-oriented resource allocation in edge computing," IEEE Trans. Services Computing, vol. 13, no. 4, pp. 723-734, Jul. 2020.

[12] D. Kim, H. Lee, H. Song, N. Choi, and Y. Yi, ''Economics of fog computing: Interplay among infrastructure and service providers, users, and edge resource owners,'' IEEE Transactions on Mobile Computing, vol. 19, no. 11, pp. 2609-2622, Nov. 2020.

약 력

함동호

2020년 대구경북과학기술원 기초학부 융복합 공학사
2020년~현재 대구경북과학기술원 전기전자컴퓨터공학과 석박통합과정
관심분야: 클라우드/엣지 컴퓨팅, 네트워크 자원 최적화, 5G+/6G 무선 통신 등



곽정호

2001년~2008년 아주대학교 전자공학과 학사
2009년~2011년 한국과학기술원 전기 및 전자공학부 석사
2011년~2015년 한국과학기술원 전기 및 전자공학부 박사
2015년~2017년 캐나다 국립과학연구소 박사후 연구원
2017년~2018년 아일랜드 Trinity College Dublin Marie Curie Fellow
2019년 대구대학교 전자전기공학부 조교수
2020년~현재 대구경북과학기술원 전기전자컴퓨터공학과 부교수
관심분야: 클라우드/엣지 컴퓨팅, 5G 셀룰러 시스템, 모바일 인공지능, 저궤도 위성 시스템 등