쿠버네티스를 활용한 MEC 에서의 서버리스 컴퓨팅 구현 구조

서장원, 김영한*
숭실대학교

sjw98052323@dcn.ssu.ac.kr, *younghak@ssu.ac.kr

Serverless computing implementation architecture in MEC using Kubernetes

Seo Jang Won, Kim Young Han*
Soongsil Univ.

요 약

쿠버네티스를 활용한 엣지 클라우드 관리 사례가 많아지면서, MEC 에서도 쿠버네티스 기반 저지연 서비스 제공의 필요성이 대두되고 있다. 또한, 서버리스 컴퓨팅은 비용 효율성과 이벤트에 따른 동적 서비스 제공 등의 강점으로 인해 클라우드 환경에 널리 도입되고 있다. 이에 따라 본 논문에서는 쿠버네티스를 활용한 MEC 에서의 서버리스 컴퓨팅 구현 구조를 제안하여 사용자에게 높은 서비스 품질과 사용자 경험을 제공할 수 있는 방법을 제시하고자 한다.

Ⅰ. 서 론

최근 클라우드 기술의 발전으로 엣지 클라우드가 주목받고 있다. 엣지 클라우드는 데이터 처리를 사용자나 디바이스에 가까운 지역에서 수행하여 서비스의 지연을 최소화하는 데 중점을 두고 있다. 유럽 통신 표준 협회(ETSI)는 5G 네트워크에서 엣지 컴퓨팅 구조를 표준화한 MEC(Multi-access Edge Computing)를 제시하였다. MEC 는 5G 네트워크를 통해 사용자에게 가깝게 배치되어 지연이 낮은 서비스를 제공할 수 있다는 장점이 있다. 하지만 MEC 은 VNF(Virtual Network Function)를 관리하는 시스템인 MANO(Management and Orchestration)를 이용하기 때문에 구조가 복잡하고 유연성과 확장성이 부족하다.

현대 클라우드 환경에서는 컨테이너 오케스트레이션 플랫폼인 쿠버네티스를 활용하여 효율적이고 유연한 서비스를 제공한다. 또한, 최근 클라우드 관련 오픈소스 프로젝트들은 쿠버네티스의 설계 개념을 기반으로 출시되고 있다. 따라서 MEC 에서도 쿠버네티스를 활용하여 서비스를 관리하는 필요성이 대두되고 있다.

또한, 서비스의 품질과 사용자 경험을 향상시키기 위해 이벤트에 따라 동적으로 기능을 제공해야 할 필요가 있다. 이에 따라 사용자의 리소스 사용의 효율성과 이벤트 기반의 동적인 서비스 제공으로 인한 유연성을 갖춘 서버리스 컴퓨팅을 활용하는 사례가 늘어나고 있으며 클라우드 관련 오픈소스 프로젝트 중에서 쿠버네티스 기반의 서버리스 서비스를 실행하기 위한 플랫폼들이 개발되고 있다. 그 중에서 Cloud Native Computing Foundation(CNCF) 산하에서 관리되는 오픈소스 서버리스 플랫폼인 Knative 가 대표적이다. Knative 는 쿠버네티스 환경에서 서버리스 애플리케이션의 배포 및 관리를 담당하고 이벤트 기반의 서버리스 애플리케이션의 호출을 가능하게 하는 오픈소스 플랫폼으로 유연하고 반응적인 서버리스 애플리케이션 운영을 가능하게 한다[1].

따라서, 본 논문에서는 쿠버네티스 환경에서 Knative 를 이용하여 MEC 에서의 이벤트 기반 서버리스 컴퓨팅 구현 구조를 제안하고, 이에 대한 추가 연구 방향을 제시하고자 한다.

Ⅱ. 본론

본론에서는 Knative 의 핵심 기능인 Knative Serving 과 Knative 의 Eventing 대해서 설명하고 이를 이용하여 MEC 에서의 이벤트 기반 서버리스 아키텍처를 제안한다.

Ⅱ-1. Knative Serving

Knative Serving 은 Kubernetes 를 기반으로 구축되어 애플리케이션 및 함수를 서버리스 컨테이너로 배포하고 관리하는 것을 지원한다[2]. Knative Serving 의 핵심 기능인 Scaling-to-zero 및 Scaling-from-zero 기능을 통해 요청이 없을 경우, 파드의 개수를 0 으로 조정하여 활동을 중단시켜 리소스를 효율적으로 관리한다. 특정 서버리스 파드로의 요청은 Istio Ingress Gateway 를 통해 전달되며, 파드가 동작 중이라면 직접 파드로 요청이 전달되지만 pod 가 비활성화된 경우 Activator 에서 우선적으로 요청을 받는다. 이 때, AutoScaler 는 pod 내의 Knative 컴포넌트인 Queue-proxy 의 상태를 모니터링하고 이에 따라 pod 의 개수를 동적으로 조절하여 pod 가 활성화되면 Activator 에 보관되어 있던 요청을 pod 로 전달한다.

Ⅱ-2. Knative Eventing

Knative Eventing 은 서버리스 애플리케이션에서 이벤트 기반 구조를 사용할 수 있게 해주는 기능을 제공한다[3]. 이를 통해 특정 이벤트가 발생했을 때 특정 함수를 실행하여 동적인 서비스 제공이 가능하다. Knative

Eventing 은 이벤트가 발생한 지점과 이벤트를 전달해야 할 지점을 명시해놓은 Source 컴포넌트를 통해 이벤트를 감지하고 전달하는데 CloudEvent 형식으로 이벤트를 생성하여 이벤트 소비자로 전달된다. CloudEvent 란 서비스, 플랫폼 및 시스템 전반에 걸쳐 상호 운용성을 제공하기 위해 일반적인 형식으로 이벤트 데이터를 설명하는 사양이다[4]. 이러한 표준을 통해 클라우드 사업자에 종속적이지 않은 이벤트를 처리할 수 있다. Source 로부터 생성된 이벤트는 두가지 방법으로 처리될 수 있다. 첫번째는 이벤트를 임시보관 하는 Channel 컴포넌트와 특정 채널을 구독하여 정해진 서비스르 이벤트를 전달하는 Subscription 컴포넌트의 조합으로 이벤트 처리 파이프라인을 구성할 수 있다. 두번째는 이벤트를 전달받아서 조건에 맞는 이벤트 소비자로 이벤트를 전달하는 중개자 역할인 Broker 컴포넌트와 조건에 맞는 이벤트가 들어왔을 때 특정 서버리스 함수를 실행시키도록 하는 Trigger 컴포넌트를 이용하여 이벤트 처리 파이프 라인을 구성할 수 있다.

Ⅱ-3. MEC 에서의 이벤트 기반 서버리스 구현

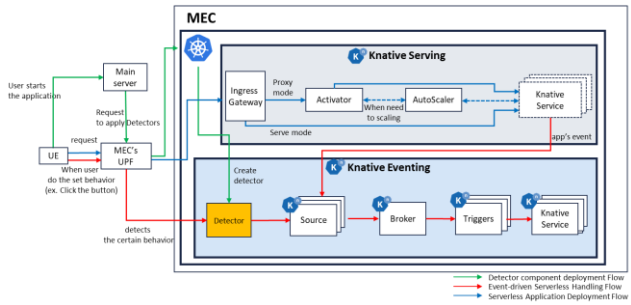

그림 1. Knative 를 이용한 MEC 에서의 서버리스 구조

그림 1 은 위에서 언급한 Knative 의 핵심 기능들을 이용하여 MEC 에서의 서버리스를 구현한 구조를 나타낸 그림이다. 클라우드 내 컴포넌트 간에는 CloudEvent 형식으로 이벤트를 주고받도록 설정되어 있지만 외부 단말 및 외부 애플리케이션으로부터의 이벤트는 CloudEvent 형식으로 이루어져 있지 않고 또한 외부의 이벤트를 Source 로 전달할 수 있는 방법이 없다.

따라서 본 논문에서는 그림 1 에 그려진 것처럼 외부 이벤트를 전달받아 Source 를 정의하여 생성하는 컴포넌트인 Detector 를 새롭게 추가한다. Detector 컴포넌트는 사용자 단말이 메인 응용 프로그램에 접속했을 때 생성되어 해당 사용자가 주요 응용프로그램을 이용하는 동안 발생되는 이벤트는 특정 방식를 통해 Detector 로 전달되도록 한다.

그림 2 는 Detector 를 추가하여 이벤트 기반 서버리스 서비스를 호출하는 과정을 나타낸 그림이다.

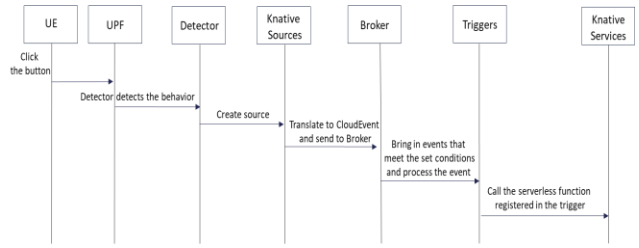

그림 2. 이벤트 기반 서버리스 애플리케이션 호출 과정

동작 과정을 설명하자면 UPF 를 통해 사용자나 외부 애플리케이션으로부터 특정 동작(예: 특정 버튼을 클릭)이 수행되면 Detector 는 해당 동작에 대한 이벤트를 감지하고 Source 를 정의하여 해당 이벤트를 CloudEvent 형식으로 생성하고 Knative 의 Broker 컴포넌트로 전달한 후 Knative 의 Trigger 컴포넌트는 설정된 조건에 따라 Broker 가 수신한 이벤트를 Knative Service 로 전달하여 이벤트가 처리된다. 이 과정을 통해 MEC 환경에서의 서버리스 아키텍처가 이벤트 중심으로 유연하게 동작할 수 있을 것이라 기대한다.

Ⅲ. 결론

본 논문에서는 저지연 서비스 제공 및 동적인 기능 제공 등의 장점을 가진 엣지 클라우드와 서버리스의 특징을 결합하여 높은 수준의 사용자 경험을 충족시키기 위해 서버리스 오픈소스 플랫폼인 Knative 의 핵심 기능들을 활용하여 MEC 환경에서의 서버리스 구현 구조를 제안하였고 이벤트 기반의 서버리스 동작 과정을 나타내었으며 Detector 라는 새로운 컴포넌트를 추가하여 클라우드 내 애플리케이션이 아닌 외부의 이벤트를 감지하는 기능을 더했다.

본 논문에서는 외부의 이벤트를 Detector 가 감지하기 위해 특정 프로토콜 및 메서드의 이용을 언급했지만 이에 대해서는 추후 5G 네트워크의 스터디와 Knative Eventing 의 깊은 탐구를 통해 구체화할 예정이며 이 후 Detector 를 구체적으로 설계하고 실험할 예정이다.

ACKNOWLEDGMENT

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터사업의 연구결과로 수행되었음" (IITP-2023-RS-2023-00258649)

참 고 문 헌

[1] What is Knative?, https://knative.dev/docs/concepts/

[2] Knative Serving, https://github.com/knative/serving/tree/main, 2023

[3] What is Knative Eventing?, https://github.com/knative/eventing?tab=readme-ov-file, 2023

[4] CloudEvent, https://github.com/cloudevents/spec, 2023