분산 엣지 클라우드 환경에서 이벤트 기반 Function as a Service 연구

김도현, 송수현, 송지원, 윤창섭, 김영한*

*숭실대학교

{kdh951102, suhyun709, jwsjws5346, ckdtjq6037}@soongsil.ac.kr, *younghak@ssu.ac.kr

A Study on the Event-Driven Function as a Service in Distributed Edge Cloud Systems

Kim Do Hyun, Song Su Hyun, Song Ji Won, Yoon Chang Seop, Kim Young Han*

*Soongsil Univ.

요 약

본 논문은 지역별 서버로 구성된 쿠버네티스 클러스터에서 이벤트 기반으로 동작하는 서버리스 모델의 비용 절감 및 인프라 효율에 관해 연구한다. 제안된 모델은 평상시에 동작하지 않는 상태에서 특정 이벤트가 발생했을 때 부하가 큰 딥러닝 모델을 이용해 사고 상황(교통사고, 화재, 신변위험)을 인식하고, 필요한 곳에 적절한 서비스를 제공하도록 FaaS 형태로 서비스를 구축한다. 간단한 움직임 감지부터 점차 복잡한 세부 상황을 인식하도록 함으로써 기존의 지능형 CCTV와 비교하여 FaaS 형태로 인식 기능 서비스를 구현했을 때 이식성과 유연성을 보장하는지. 하드웨어 자원을 얼마나 효율적으로 사용할 수 있는지 분석한다.

Ⅰ. 서 론

기존의 전통적인 서버 구성은 필요한 자원만큼 서버를 빌려 애플리케이션을 배포하는 형태였다면, FaaS(Function-as-a-Service)[1]는 서비스를 함수 형태로 정의하고 서비스가 필요할 때만 호출하는 방식이다. 이러한 방식은 특정 이벤트가 발생했을 때 함수가 실행되므로 자원 효율성과 비용적 측면에서 합리적이다. 본 연구에서는 이러한 FaaS의 특성을 활용하여 특정 상황이 발생했을 때만 파드(pod)가 적절한 엣지 클라우드에 할당되어 서비스를 제공할 수 있도록 한다. 기존에는 서비스를 제공하는 파드가 반드시 한 개 이상 작동해야 했던 것과는 다르게 scale-to-zero를 구현하여 이벤트 기반으로 파드가 생성되어 동작하는 서버리스 인프라 환경을 제안한다.

Ⅱ. KEDA

컨테이너화 기술은 서비스에 필요한 라이브러리, 환경을 패키지화하여 제공하므로 어느 환경에서나 컨테이너 이미지를 받아와 사용할 수 있다는 이식성 측면에서 매우 큰 장점을 준다. 본 연구에서는 컨테이너 오케스트레이션 툴인 쿠버네티스(Kubernetes)를 통해 서비스 배포를 자동화한 인프라를 구축한다. KEDA(Kubernetes-Event-Driven-AutoScaling)[2]를 이용하여 파드가 존재하지 않는 상태에서 이벤트 기반으로 파드가 생성되는 scale-to-zero를 구현해 자원의 활용성을 극대화하고자 한다.

KEDA는 그림 1과 같은 구조로 이벤트 발생 메시지가 전송되었을 때 ScaledObject로 정의된 서비스를 배포하는 방식으로 동작한다. ScaledObject의 트리거로는 CPU 부하량, Prometheus[3] 쿼리 결과 등 다양한 scaler를 사용해서 정의할 수 있고, pollingInterval, cooldownPeriod 필드를 통해 오브젝트를 생성한 후 해당 오브젝트가 완전히 소멸하기까지의 시간을 지정할 수 있다. 서비스의 확장이 필요할 때, 노드에 1개 이상의 파드가 존재하면 기존의 쿠버네티스 HPA(Horizontal Pod AutoScaler)를 이용하여 scale-out하고 그렇지 않으면 KEDA를 통하여 파드를 생성해 트래픽을 원활하게 처리한다.

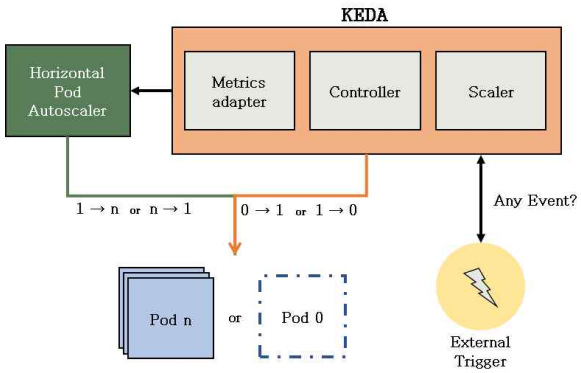


그림1. KEDA 구성도

Ⅲ. 본 론

본 논문의 클러스터 및 시스템은 표 1과 같이 구성하였다.

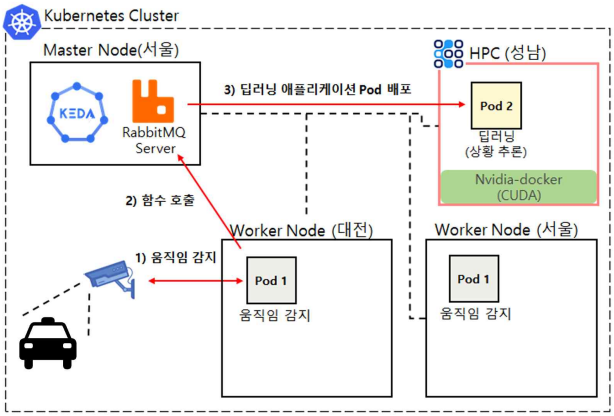
표 1. 시스템 환경 및 클러스터 구성

요소	내용
OS	Ubuntu 18.04
Kubernetes	v1.21.3
KEDA	v2.3
Container Runtime	Docker v20.10.7

본 연구에서는 KEDA의 외부 이벤트 트리거로써 RabbitMQ[4]의 브로커 큐에 메시지가 쌓이는 정도를 scaler로 사용한다. 특정 서비스가 필요하다고 판정되면 RabbitMQ 서버에 메시지가 전송되고, 미리 정의한 ScaledObject에 따라 scale-to-zero를 수행한다. 정의한 메시지 수를 초과하여 큐에 쌓이게 되면 HPA는 해당 파드를 scale-out하는 동시에 워커노드에 부하 균등하게 배포하여 로드밸런싱을 제공한다.

본 연구의 시나리오는 그림 2와 같다. 쿠버네티스 클러스터로 구성된 워커 노드에서 물체의 움직임을 감지하는 경량화 애플리케이션을 컨테이너 형태로 동작시킨다. 해당 경량화 애플리케이션은 CPU만으로 동작한다. 움직임이 감지되면 마스터 노드의 RabbitMQ 브로커 큐에 메시지를 전송하고, KEDA가 이를 감지하여 HPC(High-Performance-Computer)로 교통사고, 화재 등 세부 상황을 인식하는 딥러닝 애플리케이션을 배포한다. 해당 딥러닝 애플리케이션은 GPU를 사용한다. 이처럼 특정 상황(교통사고, 화재 등)을 인식하는 부하가 큰 서비스가 FaaS형태로 동작하면, 많은 하드웨어 자원을 요구하는 세부 상황 감지 딥러닝 모델은 항상 동작할 필요 없이 움직임이 감지된 이후 배포되므로 자원 효율을 극대화할 수 있다.

그림 2. 클러스터 구성도



움직임을 감지하는 모델은 이미지 처리를 지원하는 OpenCV를 활용하여 전 장면과의 비교를 통해 움직임을 감지하는 간단한 알고리즘을 이용하고, 세부 상황을 인식하는 딥러닝 모델은 Darknet 프레임워크의 YOLOv4 모델을 사용한다. 해당 모델들은 도커(docker)를 이용해 컨테이너 형태로 빌드하여 배포하였기 때문에 쉽게 수정 및 배포가 가능하며 클라우드 환경에서의 확장성과 이식성을 보장한다.

Ⅳ. 실험결과

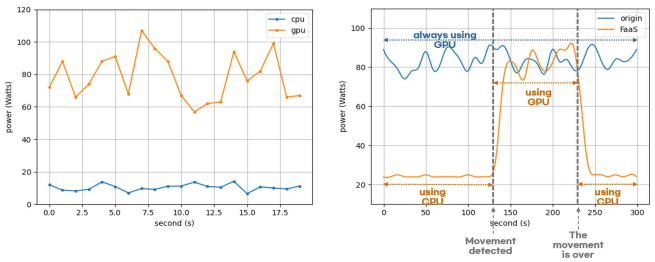


그림 3. FaaS 인프라 적용 전후 소비전력량 측정

FaaS 인프라 유무에 대한 소비전력량 비교는 그림 3과 같다. 그림 3의 왼쪽 그래프는 FaaS 인프라에서 움직임을 감지하는 파드가 동작할 때 소모되는 CPU 전력과 세부 상황을 인식하는 딥러닝 모델 파드가 동작할 때 사용되는 GPU 전력을 측정하여 나타낸 것이다. 이를 통해 딥러닝 모델이 동작할 때 65W만큼 많은 소비전력이 소모됨을 확인할 수 있다.

그림 3의 오른쪽 그래프는 세부 상황 인식 딥러닝 모델을 HPC에서 항상 동작시킬 경우(origin)와 FaaS 인프라로써 서비스를 제공할 경우(FaaS)를 비교하여 각각의 자원 소모량을 나타낸 것이다. 기존 인프라 구조에서는 HPC에서 딥러닝 모델이 항시 동작하면서 GPU 전력을 계속 소모한다. 반면, FaaS 형태로 서비스를 제공하면 움직임 이벤트가 발생할 때만 HPC에서 세부 상황 감지 딥러닝 파드가 동작하고, 해당 시간에만 GPU 전력을 소모한다. 세부 상황 감지가 끝난 후에는, 자동으로 딥러닝 애플리케이션은 종료되고 다시 움직임 감지 파드가 동작하여 소량의 CPU 전력을 소모하는 형태로 돌아간다. 부하가 큰 애플리케이션이 상시 돌아가지 않아도 된다는 이점은 이러한 프로세서의 전력 소모 외에도 메모리 측면에서 이득을 가져온다.

표 2. 딥러닝 애플리케이션과 움직임 감지 애플리케이션의 메모리 사용량 비교

	세부 상황 판단 딥러닝 애플리케이션 (NVIDIA Tesla V100)	움직임 감지 애플리케이션 (CPU + RAM)
시작 전 메모리 사용량	239 MiB/ 16160 MiB	5846 MiB/ 167954 MiB
시작 후 메모리 사용량	1237 MiB/ 16160 MiB	5986 MiB/ 167954 MiB
평균 사용량	998 MiB	140 MiB

표 2와 같이 해당 시나리오에서 동작시키면 움직임이 없을 때, 858MiB 만큼의 메모리를 아낄 수 있다. 실제 서비스에 적용했을 때, 움직임이 많이 없는 시간대나 서비스 이용량이 많지 않은 환경이라면 비용 절감 효과는 더욱 극대화된다.

Ⅴ. 결 론

본 논문에서는 서버리스 인프라 구축을 통해 필요한 상황에만 답러닝 등의 서비스가 FaaS로써 수행되므로 자원 절약 및 비용 절감의 효과를 증명하였다. 실험결과 통해 불필요하게 작동하던 딥러닝의 수행을 없애 최대 65W의 전력을 절약하는 결과를 도출하였고, 이를 통해 사용자는 간간이 사용하는 서비스에 대한 인프라의 구축 부담을 덜고 기존의 클라우드 사업자를 통해 인프라를 필요한 만큼만 선택하여 사용할 수 있다. 또한, FaaS의 확정성 및 이식성은 새로운 기술의 빠른 적용 및 효율적인 서비스 배포를 기대할 수 있다. 더 나아가 IoE 시대로 도래하면서 처리해야 할 정보가 방대해짐에 따라 엣지 디바이스에서의 수많은 트래픽 요청을 FaaS를 통해 처리한다면 훨씬 더 효율적인 운영이 가능하다.

ACKNOWLEDGMENT

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT 연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2021-2017-0-01633)

참 고 문 헌

[1] Paul Castro, Vatche Ishakian, Vinod Muthusamy, Aleksander Slominski, "Serverless Programming (Function as a Service)", 2017 IEEE 37th International Conference on Distributed Computing Systems(ICDCS).

[2] Cloud Native Computing Foundation, "Keda" (https://keda.sh/).

[3] Cloud Native Computing Foundation, "Prometheus" (https://prometheus.io/).

[4] VMware Tanzu, "RabbitMQ Message Broker" (https://www.rabbitmq.com/).