엣지 기반 미디어 서비스 구성을 위한 AI모델 정보 관리구조의 제안

*염정철 **금승우
한국전자기술연구원 정보미디어연구센터
*jcyeom@keti.re.kr **swkum@keti.re.kr

Proposed of AI-Model Information Management Structure for Media Service Construction based on Edge

*Yeom, Jeongcheol **Kum, Seungwoo
Korea Electronics Technology Institute
Information and Media Research Center

요약

최근 미디어, 금융 등 다양한 분야의 기업들이 AI를 활용해 제공하는 서비스가 늘어남에 따라 학습된 모델을 엣지 자원에 배포하여 기능을 제공하는 서비스형태 또한 늘어나고 있다. AI-Application이 동작하기 위해서는 AI-Model 파일뿐 아니라 동작을 위한 설정 파일들이 필요하여 AI-Application이 사용 중인 AI-Model의 정보를 수집, 관리하는 것은 중요한 이슈라고 할 수 있다. 하지만 단일 서비스서버에서 동작하는 형태가 아닌 각 자원이 산재되어 다양한 형태로 서비스를 제공하는 엣지컴퓨팅의 구조적인 특성상 AI-Application의 기존 서비스구조, 기능을 수정하지 않고 정보를 수집하는 과정은 다양한 문제에 부딪치게 된다. 이에 따라 본 논문에서는 기존 서비스구조를 변경하지 않고 독립적으로 AI-Application에서 사용중인 AI-Model의 정보를 파악하고, 사용자 요청에 대응할 수 있는 관리구조를 제안한다.

1. 서론

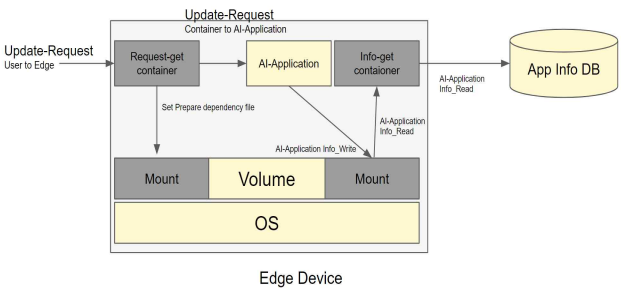
2016년 다보스포럼에서 4차산업혁명에 대한 언급이 시작되고, 핵심기술들이 기업의 서비스에 접목되어 일반 소비자들에게까지 그 영향이 미치기 시작했다. 그중 인공지능(AI)기술은 기존에 활용되던 단순한 데이터 예측의 범주를 넘어, 미디어•통신•금융 등 각 서비스 분야에서 예측된 정보를 접목한 서비스를 제공하기 시작했다[1][2]. 인공지능 서비스를 위한 딥러닝기술은 두 과정으로 나눌 수 있는데, 학습과정은 매우 많은 컴퓨팅 자원를 요구하지만 추론단계에서는 비교적 적은 자원을 요구하기 때문에, 서비스 업체들은 서버 혹은 클라우드 등에서 학습을 진행한 모델을 서비스 서버 혹은 엣지에 배포해서 사용하는 서비스형태를 사용하고 있다[3].

AI-Application이 동작하기 위해서는 AI Model에 대한 Weight파일, Config파일 등 다양한 종속성을 필요로 하는데 각 모델별로 필요한 파일들이 달라, AI Model의 변경, 갱신 등의 관리를 위해서는 각 디바이스에서 활용중인 AI-Model을 파악하는 것이 필수적이다. 단일 서비스 서버에서 동작하는 AI-Application의 정보를 파악하기는 쉽지만, 각 엣지자원에서 AI-Application이 동작하는 엣지 컴퓨팅은 기능을 제공하는 자원들이 다양한 형태로 산재되어 있어 AI-Application의 서비스구조 변경 없이 정보를 획득하는 과정에서 몇몇 문제에 직면하게 된다.

본 논문에서는 이러한 문제를 해결하기 위해 기존 서비스 구조의 변경 없이 엣지에서 동작중인 AI Application의 AI-Model의 정보를 파악하고, 모델의 변경, 업데이트등 사용자의 요청에 대응할 수 있는 형태의 관리 기법을 제안한다.

2. 제안 연구

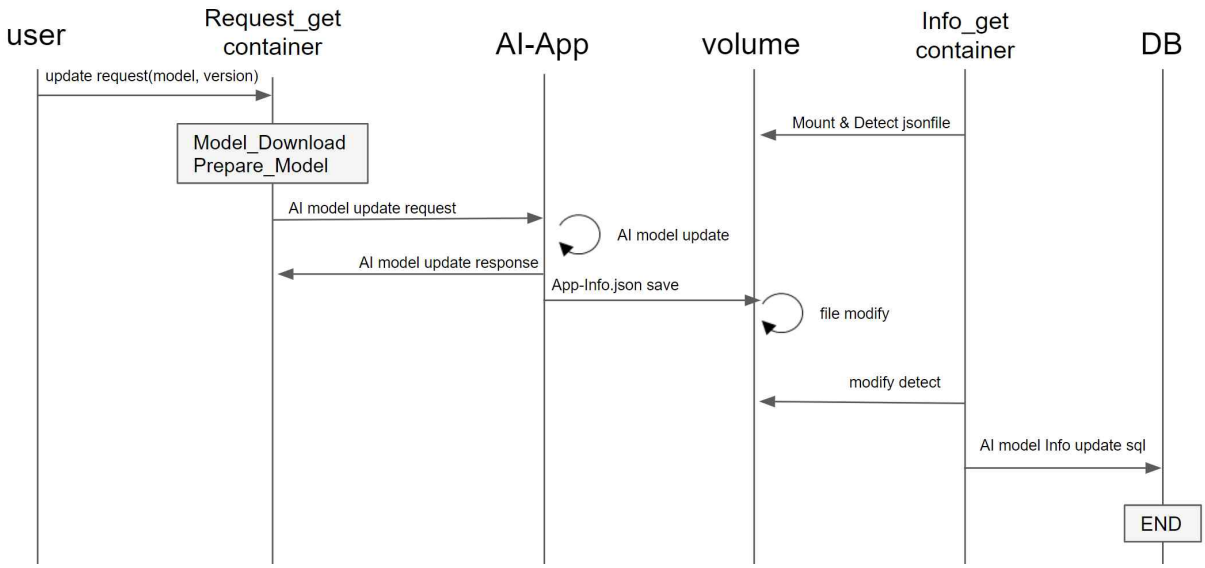
본 논문에서 제시하는 관리기법의 구성은 다음의 그림 1과 같다.



[그림 1] 제안하는 관리기법의 구성

제안하는 구성은 'Request-get Container'와 'Info-get Container'라는 두 개의 컨테이너로 이루어져 있으며 각 컨테이너는 로컬볼륨에 마운트 해야한다. 'Request-get Container'는 업데이트와 같은 사용자의 요청을 수신한 뒤 해당 모델의 동작을 위해 모델의 다운로드, 설정파일의 위치 설정의 기능을 가지는 컨테이너고, 'Info-get Container'는 동작이 완료된 Application의 변경정보를 추적하여 Database에 기록하는 컨테이너이다.

아래의 그림 2은 업데이트 요청시 AI-Application과 컨테이너간의 동작절차를 보여준다. 그림에서 'Request-get Container'는 업데이트 요청을 수신한 이후 모델을 다운로드하고, 종속성 파일들의 위치를 설정



[그림 2] 제안구조의 동작절차(업데이트 요청)

한다. 그 이후 AI-Application에게 업데이트를 요청하면 업데이트 완료 후 'update response'를 수신하는 것으로 프로세스가 종료된다. 'Info-get Container'는 초기 동작시 마운트 한 이후 파일의 변화를 감지한다. AI-Application이 업데이트한 내용을 기반으로 파일을 수정하면 해당내용을 감지하고, DB에 변경된 내용을 업데이트한다.

3. 구현

3.1. Sample AI-Application

논문에서 제안된 구조와 절차를 사용하기 위해서 Sample AI-Application은 외부와 통신할 수 있는 Interface를 가지고 있고 어플리케이션의 실행 혹은 변경시 해당 내용을 jsonfile형태로 'Info-get Container'가 마운트한 경로에 저장할 수 있도록 구현되어있다.

3.2. 실험 환경

구현을 위해 사용된 운영체제는 Linux-Ubuntu, 동작을 위한 프로그램의 구현에는 Python, 컨테이너 매니지먼트 툴로는 Docker를 사용하였으며 각 버전은 다음과 같다. 마지막으로 컨테이너 오케스트레이션 플랫폼인 쿠버네티스를 사용하여 클러스터를 구성하고 해당 컨테이너들을 배포하여 동작기능을 검증하였다.

- Ubuntu 20.04
- Docker: 20.10.14
- Python: 3.8.10
- Postgresql: 13.3
- Docker: 20.10.14

3.3. 실험 과정 및 결과

구현된 구조의 실험을 위해 다음의 그림 3과 같은 request를 동작시켜 'Request-get Container'에게 업데이트를 요청하였다.

```
import requests

# Create a dictionary to be sent.
json_data = {
    "ai_model": {
        "url": "file://./test.zip",
        "model_name": "nametest",
        "model_version": "vtest"
        }
    }

# Send the data.
response = requests.post(url='http://127.0.0.1:5050/update', json=json_data)
print("Server responded with %s" % response.status_code)
```

[그림 3] 테스트 업데이트 요청

위의 그림과 같은 업데이트 요청을 통해 그림 4와 같이 ai-app.json이 생성된 것을 확인할 수 있고, 그림 5을 통해 DB에 요청한 정보가 입력된 것을 확인할 수 있다.

```
root@798fe29c332d:/app-info# ls
ai-app.json
root@798fe29c332d:/app-info# cat ai-app.json
{
    "ai_model": {
        "model_name": "nametest",
        "model_version": "vtest",
        "url": "file://./test.zip"
    }
```

[그림 4] 생성된 ai-app.json파일

```
aidb=> select * from edgetable;
    NODE     | CLUSTER |  INPUT   |  OUTPUT   | MODEL | VERSION
-------------+---------+----------+-----------+-------+---------
 vibes-master | default | testinput | testoutput | nametest | vtest
```

[그림 5] DB 입력 확인을 통한 동작 검증

4. 결론

본 논문은 엣지에서 동작하는 AI-Application의 AI-Model의 정보를 파악하고, 사용자의 요청할 수 있는 관리구조에 대해 제안하였다. 제시된 아키텍처는 실험실 환경에서 멀티 엣지-싱글 어플리케이션에 대응 및 동작하도록 구현되어 있으며, 제안 내용과 동일하게 AI-App의 실행, 업데이트에 따라 변경되는 내용이 데이터베이스에 기록되는 것을 확인하였다. 제시된 제안구조는 두 개의 컨테이너를 활용하여 기존 플랫폼에 영향을 미치지 않고 독립적으로 운용이 가능하여, 기존에 동작하는 다양한 시스템에 적용할 수 있다는 장점이 있다.

향후 멀티 어플리케이션에 대한 대응, AI-App과의 통신방법, AI-Model 다운로드 구조 등의 기능을 확장을 통해 다양한 환경의 엣지 자원에서 동작하는 유연하게 적용시킬 수 있도록 진행할 예정이다.

감사의글

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2021-0-00619, (세부 4) 커넥티드 자율주행서비스 엣지 AI 요소기술개발)

Reference

[1] 곽우정 외 1인, “도서관의 인공지능(AI) 서비스 현황 및 서비스 제공 방안에 관한 연구,” Journal of Korean Library and Information Science Society vol.52 no.1, 2021. 03

[2] 남현우, “비대면 IoT 피부진단-치료 AI 서비스 접근방향 연구,” 한국디자인리서치학회지 vol.5 no.3, 2020. 03

[3] 김귀훈 외 2인, “IoT와 AI를 위한 에지 컴퓨팅 표준화 및 기술 동향,” 한국통신학회지 vol.34 no.12, 2017. 12

[4] https://github.com/ultralytics/yolov5

[5] 김동명 외 1인, “인공지능을 위한 Edge Computing 기술 동향,” 한국통신학회지 vol.37 no.12, 2020. 12