Airflow 를 활용한 쿠버네티스 환경에서의 머신러닝 워크플로우 구조 분석

김승현, 김영한*
*숭실대학교

kim11776@dcn.ssu.ac.kr, *younghak@ssu.ac.kr

Analysis of Machine Learning Workflow Structure using Airflow in Kubernetes Environment

Kim Seung Hyun, Kim Young Han*
*Soongsil Univ.

요 약

컨테이너 기반 클라우드 환경에서 일정 주기로 머신러닝 모델을 fine-tuning 시키기 위해서는, 워커 노드에서 새롭게 생성된 다양한 형태의 데이터들을 가공하고 정제한 뒤 모델에 학습시키는 반복적인 작업이 필요하다. 따라서 본 논문에서는 Apache Airflow 에서 제공하는 KubernetesExecutor 와 KubernetesPodOperator 를 활용해 쿠버네티스 환경에서 주기적으로 반복 실행 가능한 머신러닝 워크플로우 구조를 제안하고 분석한다.

Ⅰ. 서 론

머신러닝 워크플로우는 인공지능 모델 훈련 및 검증 과정에 해당하는 전체 작업 순서를 자동화함으로써 관리자에게 편의성을 제공한다. 그러나 ETL(Extract, Transform, Load)[1] 및 학습 모델과 하드웨어 스펙 사이의 종속성으로 인해 새로운 환경에서의 워크플로우 재사용은 불가능하며 인공지능 모델의 정확도를 향상시키기 위해 주기적으로 실행되는 fine-tuning 같은 반복 작업에서의 한계 또한 존재한다. 그러나 컨테이너 기반의 쿠버네티스 환경에서 Apache Airflow 머신러닝 워크플로우는 각 작업들이 DAG(Directed Acyclic Graph)로 구성되며 노드에서 개별 작업으로 실행되어 하드웨어 종속성 문제를 해결할 수 있으며 자원을 효율적으로 관리할 수 있다. 또한 원하는 작업에 맞게 관리자가 정의한 Operator 를 사용할 수 있으며 Airflow 스케줄러를 활용한 반복 작업도 가능하다. 또한 관리자는 파이썬 코드로 머신러닝 워크플로우를 작성함으로써 동적 변경이 쉽도록 설계할 수 있고 Web Server 를 통해 지속적으로 작업 상태를 모니터링 할 수 있다. 따라서 본 논문에서는 Apache Airflow 에서 쿠버네티스 환경을 위해 제공하는 Operator 를 활용해 주기적으로 fine-tuning 이 가능한 머신러닝 워크플로우 구조를 제안하고 분석하고자 한다.

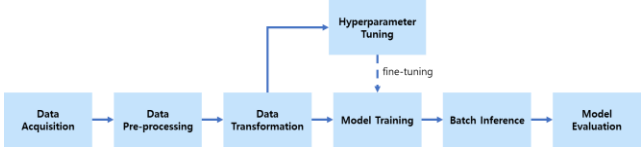

그림 1. 머신러닝 워크플로우

Ⅱ. 관련연구

Airflow 는 Apache 에서 개발한 워크플로우 스케줄링, 모니터링 오픈소스 플랫폼이며 개발자가 코드로 작성한 데이터의 흐름을 뜻하는 DAG 와 작업의 단위를 뜻하는 Task 로 구성되어 있다. Apache Airflow 의 동작 원리는 다음 그림과 같다.

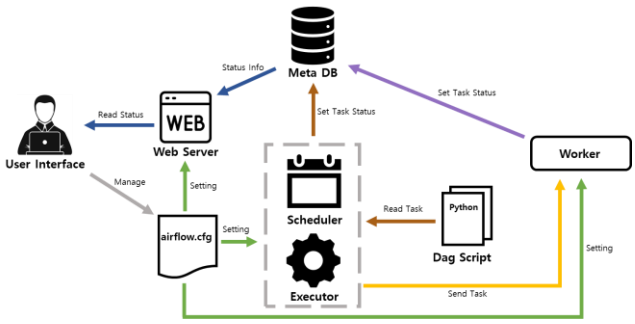

그림 2. Apache Airflow 동작 원리

관리자는 airflow.cfg 파일을 활용해 웹서버를 사용할 포트 정보, 워커 노드가 참조할 DAG 스크립트 위치 정보 및 스케줄러 Dag 스크립트 파일을 어떤 주기로 반복해서 확인할지 설정한다. 정해진 주기로 스케줄러는 사용자가 파이썬 코드로 작성한 DAG 스크립트를 읽어서 각 Task 들의 워크플로우 정보를 메타 데이터 베이스에 저장하며, 사용자가 airflow.cfg 파일에 미리 정의한 Executor 를 실행해준다. Executor 는 현재 시점에 실행되어야하는 Task 를 찾고 작업의 상태 정보를 업데이트 한 후 워커 노드에서 해당 작업을 실행한다. 워커 노드는

Executor 의 종류에 따라서 동작 방식이 나뉘며 CLI(Command Line Interface) 명령어로 실행된다. 관리자는 UI(User Interface)를 통해서 각 Task 의 실행 상태, 성공 여부, Dag 구성도를 모니터링 할 수 있다.

Airflow 에서 일반적으로 사용하는 LocalExecutor[3] 또는 CeleryExecutoer[3]를 쿠버네티스 환경에서 사용하면 구성이 간단하고 템플릿화가 쉽다는 장점이 있지만 유지보수 비용이 크며 라이브러리 의존성에 따라서 무거워질 수 있다. 따라서 Airflow 에서는 쿠버네티스 환경을 위해서 자원을 효율적으로 사용하며 원하는 도커 컨테이너를 골라 pod 로 실행함으로써 라이브러리 의존성을 낮춰 유지보수 비용을 절감할 수 있게 해주는 KubernetesExecutor 와 KubernetesPodOperator 를 제공한다.

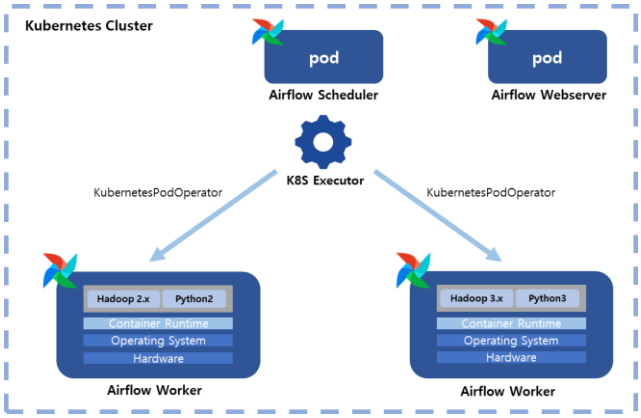

그림 3. KubernetesExecutor & KubernetesPodOperator

Ⅲ. 본 론

클라우드 환경은 노드에서 생성되는 다양하고 방대한 데이터들을 수집하고 머신러닝 모델을 간편하게 워커 노드에 배포할 수 있기 때문에, 인공지능 서비스를 적용하기 위한 최적의 환경을 제공한다. 클라우드 환경에서 인공지능 서비스를 구축하기 위해서는 다음과 같은 일련의 반복 작업이 요구된다. 각 노드에서 실시간으로 다양한 데이터를 서비스 목적에 맞게 추출한 후, 가공하고 정제하는 과정을 거친 후 학습 데이터로 변환시킨다. 변환된 학습 데이터로 머신러닝 모델을 훈련시킨 후 적절한 테스트 배치 데이터로 머신러닝 모델 테스트를 진행한다. 추가로 머신러닝 모델의 정확도를 높이기 위한 fine-tuning 을 하기 위해 노드에서 새롭게 생성된 데이터로 앞선 일련의 과정들을 주기적으로 반복한다. 본 논문에서 제안하는 쿠버네티스 환경에서 Airflow 를 활용한 머신러닝 워크플로우 구조는 다음 그림과 같다.

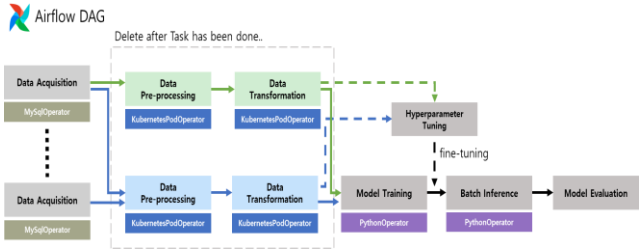

그림 4. 머신러닝 워크플로우 (Airflow)

Airflow 스케줄러는 설계된 DAG 의 순서대로 Task 를 실행한다. MySqlOperator[3]를 활용해 노드에서 데이터를 수집한다. 데이터의 형식 또는 목적에 따라 KubernetesPodOperator 로 개별 Task 에서 요구되는 라이브러리가 설치된 새로운 pod 를 배포한다. 하이퍼파라미터 튜닝 Task 는 워크플로우 첫 실행에서 success 로 설정되어 작업이 실행되지 않는다. 따라서 수집된 학습 데이터로 PythonOperator 를 사용해 머신러닝 모델을 훈련시키는 Task 를 실행한다. 훈련이 완료된 머신러닝 모델은 적절한 배치값을 활용해 테스트를 진행한 후 인공지능 서비스를 제공하는 클라우드 환경 내의 모든 워커 노드에 배포된다. KubernetesPodOperator 로 사용되는 pod 들은 Task 가 끝난 후 자동 삭제됨으로 쿠버네티스 자원을 효율적으로 관리할 수 있도록 해준다. Airflow 스케줄러는 주기적으로 Airflow DAG 를 확인한 후 같은 작업을 반복 실행한다. 그러나 두번째로 실행되는 워크플로우에서는 Airflow 스케줄러의 머신러닝 모델 학습에 해당하는 Task 를 success 로 수정한 후 진행하게 된다. 새로 생성된 데이터에 맞게 KubernetesPodOperator 에 의해서 pod 가 새롭게 생성되고 하이퍼파라미터 튜닝 Task 를 실행함으로 머신러닝 모델을 fine-tuning 할 수 있다. fine-tuning 된 머신러닝 모델의 테스트를 마친 후 서비스 되고 있는 워커 노드의 모델들을 새롭게 교체한다.

Ⅳ. 결 론

본 논문에서는, Airflow 에서 제공한 KubernetesExecutor 와 KubernetesPodOperator 를 활용한 머신러닝 워크플로우 구조를 제안하고 분석했다. 작업이 끝난 Task 들의 자원을 점유하고 있는 기존 방법과는 다르게 동적으로 Task 를 생성한 후 사용한 자원을 반납함으로써 효율적인 자원 관리가 가능하다. 또한 쿠버네티스에서 제공하는 자동화 배치 작업인 CronJob[2]과는 달리 각각의 작업을 동적으로 간편하게 변경이 가능하며 웹 UI 를 통해 DAG 를 관리하고 모니터링할 수 있다. 추후 연구에서는 워크플로우 개별 Task 에 대한 구체적인 기능 및 구조를 설계하고 구현하고자 한다.

ACKNOWLEDGMENT

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의

대학 ICT 연구센터육성지원사업의 연구결과로 수행되었음"

(IITP-2021-2017-0-01633)

참 고 문 헌

[1] Panos Vassiladis, Alkis Simitsis, Sprios Skiadopoulos. "Conceptual Modeling for ETL processes", DOLAP'02: Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, Nov 2002

[2] Kubernetes, https://kubernetes.io/

[3] Apache Airflow, https://airflow.apache.org/