논문 2021-16-26

HeteroAccel: 엣지 컴퓨팅 환경에서의 다양한 영상 추론을 위한 쿠버네티스 기반의 이종 연산·가속기 자원 관리 시스템

(Kubernetes-based Heterogeneous Computational and Accelerator Resource Management System for Various Image Inferences in Edge Computing Environments)

전 재 호, 김 용 연, 강 성 주*
(Jaeho Jeon, Yongyeon Kim, Sungjoo Kang)

Abstract : Edge Computing enables image-based inference in close proximity to end users and real-world objects. However, since edge servers have limited computational and accelerator resources, efficient resource management is essential. In this paper, we present HeteroAccel system that performs optimal scheduling in Kubernetes platform based on available node and accelerator information for various inference requests. Our experiments showed 25.3% improvement in overall inference performance over the default scheduling scheme in edge computing environment in which four types of inference services are requested.

Keywords : Heterogeneous Accelerator, Image-based Inference, Edge Server, Kubernetes, Scheduling

Ⅰ. 서 론

초저지연 서비스의 기반이 되는 5G 이동통신과 엣지 컴퓨팅 기술의 발전으로 실세계의 사물과 근접한 곳에서 다양한 인공지능 서비스가 가능해지고 있다. 하지만 복잡한 연산 처리와 데이터 저장을 위한 대규모 자원이 필요한 인공지능 서비스의 특성상 이를 효과적으로 운영하는 방법이 필요하며, 최근에는 학습은 데이터센터의 클라우드에서, 그리고 추론은 실세계 현장의 엣지 서버를 연계해서 인공지능 서비스를 수행하는 방법들이 시도되고 있다 [1].

인공지능 서비스를 이루는 주요 기능 중 영상 기반의 추론은 실세계에서 일어나는 다양한 상황에 대한 영상을 획득하고, 이를 기존에 학습된 데이터와 비교해 확률적으로 근접한 결과를 도출하는 기술로서 엣지 컴퓨팅 기술이 점차 고도화됨에 따라 예전의 산업현장에서는 할 수 없었던 다양한 현장 상황 추론이 시도되고 있다 [2]. 예를 들어 공항, 항만, 철도 등 주요 시설에서의 위험물 자동 판별, 열악한 현장 작업자의 보행 동작을 분석한 낙상사고 검출, 방문객에 대한 얼굴 인식이나 번호판 인식 등이 대표적이다.

하지만 실세계와 근접한 엣지 컴퓨팅 환경은 클라우드 컴퓨팅 환경에 비해 연산자원이 제한적이다. 실제로 엣지 컴퓨팅 환경에 배치되는 엣지 서버 클러스터의 규격은 단일 서버나 4U 수준의 클러스터 시스템이며 [3, 4], 특히 빠른 추론 기능에 필수적인 GPU, FPGA 등 가속기 자원은 여러 응용에 의해 공유되기 어려우므로 더욱더 제한적이다 [5]. 따라서 다수의 영상 획득 장치 (ex. CCTV)들로부터 유입되는 대규모의 영상 기반 추론 요청을 제한된 연산·가속기 자원으로 처리하기 위해서는 엣지 서버 클러스터의 컴퓨팅 자원에 대한 효율적 관리 방법이 필요하며, 최근 들어 이와 관련한 연구들이 수행되고 있다. Zeng은 엣지 컴퓨팅 환경에서 CPU와 GPU를 활용한 인공지능 서비스 시 에너지 효율을 고려한 자원 관리 방법을 제안하였다. 하지만 엣지 컴퓨팅 환경에서 추론 성능에 대한 고려는 부족하였다 [6]. Cho는 엣지 컴퓨팅 환경에서 비디오 감시 시스템을 위한 GPU와 FPGA 자원 스케줄링 방법을 제안하였다. 본 연구에서는 CNN 계층별 실행 시간 예측 결과를 스케줄링에 활용하였으나, 실제 노드의 자원 상태가 스케줄링에 반영되지는 않았다 [7]. Liu는 얼굴인식을 위한 추론 서비스를 에너지 효율 관점에서 CPU/GPU/FPGA에 효율적으로 스케줄링하여 전력 소비와 실행 시간을 비교하였다. 본 연구를 통해 얼굴 인식 성능을 대폭 개선하였으나, 엣지 컴퓨팅 환경에서 요구하는 다양한 추론 기능을 범용적으로 제공하기에는 부족하다 [8].

본 논문에서는 실세계 현장의 다양한 추론 요청에 대응하여 엣지 서버 클러스터 내 이종 연산·가속기의 효율적 관리를 통해 전체적인 추론 성능을 향상하는 HeteroAccel 시스템을 제안한다. HeteroAccel은 먼저 임의의 현장 내 영상 획득 장치들의 다양한 추론 (얼굴 인식, 동작 인식, 문자 인식, 객체 인식 등) 요청을 처리하기 위한 추론 엔진 (응용)들과 해당 현장에 배치된 엣지 서버가 보유한 이종 연산·가

*Corresponding Author (sjkang@etri.re.kr)
Received: Aug. 31, 2021, Revised: Oct. 13, 2021, Accepted: Oct. 18, 2021.
J. Jeon: ETRI (Senior Researcher)
Y. Kim: ETRI (Senior Researcher)
S. Kang: ETRI (Principal Researcher/Project Leader)
※ 이 논문은 2021년도 정부 (과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2020-0-00844, 엣지 서버 시스템 자원 관리 및 제어를 위한 경량 시스템 소프트웨어 기술 개발).

ⓒ IEMEK J. Embed. Sys. Appl. 2021 Oct. 16(5) 201-207
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2021.16.5.201

속기의 조합 (pair)만큼의 추론 응용을 컨테이너 형태로 가상화하여 저장한다. 이어서 HeteroAccel은 실제 영상 획득 장치에서 추론 요청 시 현재 엣지 서버 클러스터 상의 이종 연산·가속기 자원 중 가장 적합한 자원을 선택하고, 선택된 종류의 연산·가속기 자원을 활용해 추론 기능을 수행하는 응용을 컨테이너 저장소로부터 추출해 클러스터 내의 노드에 활성화한다. 이를 통해 임의의 추론 요청을 처리하기 위한 추론 응용이 다른 응용에 의해 선점된 가속기 자원을 장시간 대기하거나, 동일한 추론이 가능한 다른 연산·가속기가 가용한 상태임에도 불구하고 추론 기능이 개시되지 않는 문제를 해결하여 엣지 컴퓨팅 환경 내의 전체적인 추론 성능을 향상하였다. 본 연구에서는 다양한 추론 응용 및 가속기의 조합을 위해 인텔 오픈비노 (OpenVINO) 툴킷을 사용하였고 [9], 추론 응용의 가상화 및 이에 대한 배포, 확장 및 관리를 위한 오케스트레이션 플랫폼인 쿠버네티스 (Kubernetes)를 사용하였다 [10].

본 논문의 내용은 다음과 같이 구성되어 있다. 2장에서는 본 연구를 위한 배경 설명으로서 쿠버네티스 스케줄러와 추론 프레임워크를 설명한다. 3장에서는 HeteroAccel 시스템의 구조와 동작 방식을 제안한다. 4장에서는 제시한 방법을 통한 실험 사례와 결과를 비교하고 5장에서 결론을 맺는다.

Ⅱ. 배경 기술

1. 쿠버네티스

쿠버네티스는 컨테이너화된 응용의 배포, 확장 및 클러스터 관리를 자동화하기 위한 컨테이너 오케스트레이션 시스템이다. 쿠버네티스 클러스터는 관리 역할을 수행하는 마스터 노드와 실제 응용이 실행되는 워커 노드로 구성된다. 마스터 노드는 스케줄링, 노드 프로비저닝 및 프로비저닝 해제와 같은 클러스터에 대한 전역 이벤트를 처리하고, 마스터 노드에 의해 제어되는 워커 노드는 실행 중인 애플리케이션과 쿠버네티스 런타임 환경을 유지 관리하는 가상머신의 인스턴스 또는 물리적 시스템이 될 수 있다 [11]. 쿠버네티스는 현재 구글과 리눅스 파운데이션에서 공식적으로 지원하고 있어 오픈소스 프로젝트 수준에서 개발중인 타 솔루션과 비교해 폭넓게 활용되고 있으며, 특히 국내외 주요 클라우드 (구글의 GCP, 아마존의 AWS, 마이크로소프트의 Azure, 네이버 클라우드 등)에서 공통적으로 지원하고 있다.

쿠버네티스 클러스터 내에서는 컨테이너의 집합으로 이루어진 파드 (Pod)를 응용 최소 단위로 관리하며, 쿠버네티스 스케줄러는 파드의 실행 요청 시 해당 파드를 실행할 수 있는 최적의 노드 선택과 배포를 담당한다. 최적 노드의 선택은 요청된 파드를 실행하기 위한 최소 조건을 만족시키는지를 판단하는 필터링 단계와 필터링을 통해 결정된 노드 중 파드의 실행 적합성 관점에서 가장 높은 점수를 가진 노드를 선택하는 스코어링 단계를 통해 이루어진다. 또한 쿠버네티스는 스케줄링 시 참조할 지표 추가, 필터링 기준, 스케줄링 정책 등의 확장을 위한 스케줄러 확장 개발 방법론을 제공하고 있어 본 연구에서 활용하였다.

표 1. 쿠버네티스 스케줄러의 대표적인 필터링 기준
Table 1. Filtering Elements in Kubernetes Scheduler

Parameter	Description
PodFitsHostPorts	Checks if a Node has free ports for the requestsed Pod ports
PodFitsHost	Checks if a Pod specifies a specific Node by its hostname.
PodFitsResources	Checks if the Node has free resources to meet the requirement of the Pod.
MatchNode Selector	Checks if a Pod's Node Selector matches the Node's label(s).
NoVolumeZone Conflict	Evaluate if the Volumes that a Pod requests are available on the Node, given the failure zone restrictions for that storage.

표 2. 쿠버네티스 스케줄러의 대표적인 스코어링 정책
Table 2. Scoring Policy in Kubernetes Scheduler

Priority	Description
SelectorSpread Priority	Spreads Pods across hosts, considering Pods that belong to the same Service, StatefulSet or ReplicaSet.
InterPodAffinity Priority	Implements preferred inter pod affininity and antiaffinity.
LeastRequested Priority	Favors nodes with fewer requested resources.
BalancedResource Allocation	Favors nodes with balanced resource usage.
NodeAffinity Priority	Prioritizes nodes according to node affinity scheduling preferences indicated in PreferredDuringSchedulingIgnoredDuringExecution.

1.1 쿠버네티스 스케줄러의 필터링 단계

파드에는 오브젝트 설정 파일 (object configuration file)이 존재하며, 필터링 단계에서 쿠버네티스 스케줄러는 해당 파일내에 정의된 요청 자원 정보를 통해 파드를 스케줄링할 수 있는 노드 집합을 찾는다. 이 과정을 클러스터 내 모든 후보 노드에 적용해서 임의의 노드가 파드의 요청 자원 요건을 충족할 수 있으면 '가용 노드', 그렇지 않으면 '배제 노드'로 구분된다. 표 1은 필터링 단계에서 적용될 수 있는 대표적인 기준이다.

1.2 쿠버네티스 스케줄러의 스코어링 단계

필터링 단계를 통해 가용 노드 집합이 결정되면 스케줄러는 스코어링 단계를 통해 목록에 남아있는 노드 중 가장 높은 점수를 가진 노드를 선택한다. 스케줄러는 파드의 특성에 따라 다양한 스케줄링 정책을 사용할 수 있으며, 표 2는 대표적인 스케줄링 정책을 보여준다.

이때 노드 선택 기준이 되는 최종 점수는 아래의 공식 (1)에 따라 결정된다.

$$finalScoreNode = \sum_{i}^{N} weight(i) * priorityFunc(i). \quad (1)$$

최종 점수를 도출하는 방법은 모든 가용 노드(N)를 대상으로 개별 노드에 대해 응용 개발자에 의해 설정된 가중치(weight)와 선택한 스코어링 정책 고유의 우선순위 함수(priorityFunc) 간의 곱의 총합이다. 이후 스케줄러는 바인딩이라는 프로세스로 쿠버네티스 API 서버에 해당 결정을 전달하고, 파드는 선택된 노드에 실행된다.

2. 오픈비노 기반의 추론 프레임워크

오픈비노는 딥러닝 응용을 신속한 개발을 지원하는 오픈소스 툴킷이다. 오픈비노는 CNN을 기반으로 하여 인텔 하드웨어 가속기에 대한 딥러닝과 Xeon CPU, Iris GPU, Arria FPGA 및 Movidius NPU 등의 연산·가속기에서의 추론 기능 실행을 가능하게 한다. 오픈비노 모델 서버 (Model Server)는 오픈비노 툴킷을 통해 최적화된 모델을 이용해 추론 기능을 서비스 형태로 제공할 수 있도록 하며, 이는 외부 영상 획득 장치에 대한 추론 서버로 동작이 가능하다. 또한 추론 응용의 경우 별도의 학습 모델이 필요한데, 쿠버네티스에서 동작하기 위한 추론 파드를 구성하는 경우 파드 내에 추론 응용 컨테이너에 학습 모델을 통합하거나 외부의 학습 모델 저장소를 연동할 수 있다.

3. 이종 연산·가속기 자원 모니터링

쿠버네티스 클러스터 내의 자원은 정확하고 효율적인 스케줄링을 위해 지속적인 상태 모니터링이 되어야 하며, 이종 연산·가속기 자원에 대한 모니터링은 쿠버네티스 내부 또는 외부 컴포넌트를 통해 복합적으로 이루어질 수 있다.

3.1 코어 메트릭스 파이프라인

쿠버네티스의 컴포넌트들로부터 데이터를 수집하는 시스템으로서 메트릭 서버 (Metrics Server)를 통해 시스템 메트릭, 즉 노드와 파드의 메모리, CPU 사용량과 같은 기본적인 데이터를 수집할 수 있다. 하지만 대부분의 가속기 정보를 수집할 수 없고, 데이터가 메모리에 저장되기 때문에 영속적이지도 않다.

3.2 모니터링 메트릭스 파이프라인

쿠버네티스 외부 컴포넌트를 이용해 모니터링하는 시스템으로서 프로메테우스 (Prometheus)가 대표적이다 [12]. 모니터링 메트릭스 파이프라인은 다양한 서드 파티 컴포넌트인 익스포터 (exporter)를 활용해 다양한 가속기 정보에 대한 접근이 가능하다.

Ⅲ. HeteroAccel

1. HeteroAccel 시스템 개요

앞 장에서 살펴본 대로 쿠버네티스 스케줄러는 요청된 파드 실행을 위한 다양한 조건을 고려해서 최적의 노드에 파드를 배포하지만, 다음과 같은 제약사항이 존재한다. 먼저 쿠버네티스의 기본 스케줄러는 노드의 자원 중 CPU, 메모리, 스토리지 세 가지만을 고려해서 스케줄링 정책을 수행하고 이에 따라 파드를 노드에 배포한다. 따라서 상기 세 가지 외의 자원, 특히 이종 가속기와 같이 추론과 밀접한 관련이 있는 자원의 정보나 상태가 노드 선택에 반영될 수 없다. 이 경우 외부 단말에서 쿠버네티스 클러스터로 추론 서비스를 요청할 때 기본 쿠버네티스에서는 다음과 같은 이유로 배포를 보류 (pending) 또는 거절 (reject)시킨다.

- 쿠버네티스에서 해당 가속기를 인식하지 못해서 추론

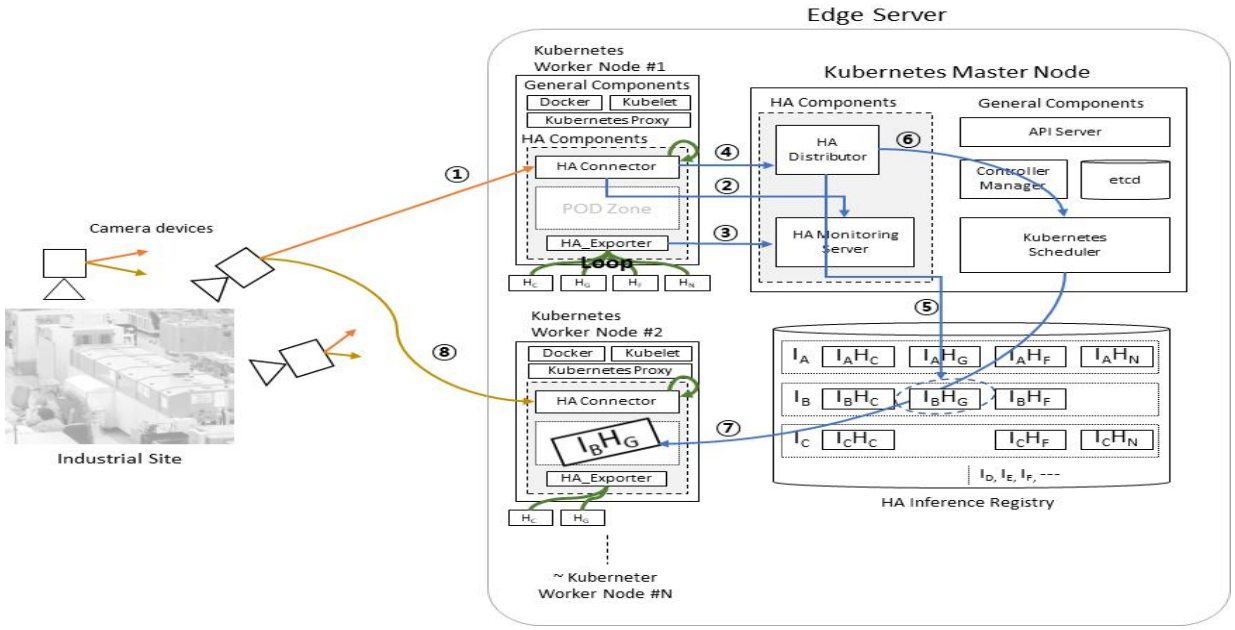

그림 1. HeteroAccel의 구조 및 동작 방식
Fig. 1. Architecture and Workflow of HeteroAccel

파드의 배포 요청이 무한 보류 상태가 되는 경우

- 쿠버네티스에서 해당 가속기를 인식하였지만, 이미 다른 추론 파드에 할당이 되어서, 요청된 파드의 배포가 보류 상태가 되는 경우

또한 다음과 같은 이유로 서버 자원이 비효율적으로 사용될 수 있다.

- 요청된 추론과 이를 처리할 가속기 (ex. GPU) 외에 실제로 다른 연산·가속기 (ex. CPU/FPGA)가 가용한 상태임에도 불구하고 유휴한 가속기를 사용할 수 있는 파드가 준비되지 않아 추론이 진행되지 않는 경우

HeteroAccel 시스템은 실세계 현장의 다양한 추론 요청에 대응하여 엣지 서버 클러스터 내 이종 연산·가속기의 효율적 관리를 통해 위와 같은 문제를 해결하여 전체적인 추론 성능을 향상하는 것을 목표로 한다. HeteroAccel 시스템은 쿠버네티스 상에서 동작하며, 오픈비노 툴킷으로 작성된 이종 연산·가속기 활용 추론 파드를 요구 자원의 상태에 따라 최적 노드에 배치하기 위해 확장된 쿠버네티스 스케줄러 기반의 시스템이다.

구체적으로 쿠버네티스 기본 스케줄러의 경우 서비스 개발자가 파드 정의 파일에 명세하는 요구자원 정보에 기반해 스케줄링이 실행되는 기존 방식으로는 정확한 스케줄링이 어려워, 실시간 메트릭을 획득할 수 있는 메커니즘을 추가하였다.

2. HeteroAccel 구조

HeteroAccel은 다수의 영상 획득 장치로부터 영상 신호를 수신하는 엣지 컴퓨팅 환경에서 동작한다. 그림 1은 HeteroAccel 시스템의 구조를 나타내며, 각 구성 요소에 대한 설명은 다음과 같다.

- HA Connector: 쿠버네티스 워커 노드에 파드 형태로 배포가 된다. HA Connector의 역할은 일정한 시간 간격마다 또는 외부의 영상 획득 장치에서 추론 요청이 들어왔을 때 자신이 배치된 워커 노드의 가속기 정보를 모니터링한다. 이어서 HA Monitoring Server에서 수집 중인 워커 노드들의 상태 정보를 바탕으로 해당 워커 노드에 어떠한 연산·가속기 (CPU, GPU, FPGA, NPU)가 유휴한 상태인지를 판단한다. 파악된 상태 정보는 최종적으로 HA Distributor에게 전달된다.

- HA Exporter: HA Exporter가 배포된 노드의 연산·가속기 정보를 수집한다. HA Exporter가 저장하는 정보는 해당 노드의 연산·가속기 종류, 가속기 ID, 가속기 상태이다.

· 가속기의 종류: CPU, GPU, FPGA, NPU 등
· 가속기 ID: 해당 가속기의 고유 ID
· 가속기 상태: allocated, unallocated (가속기를 다른 파드에서 사용 중이라면 allocated, 그렇지 않다면 unallocated)

각 워커 노드에서 수집한 연산·가속기의 정보는 HA Monitoring Server에 취합된다.

- HA Monitoring Server: 각 워커 노드에 배포된 HA Exporter가 수집하는 정보를 스크랩 (scrape)하는 프로메테우스 기반의 서버이다. 이를 통해 워커 노드별 <유휴 가속기 종류, 가속기 ID> 정보를 수집한다.

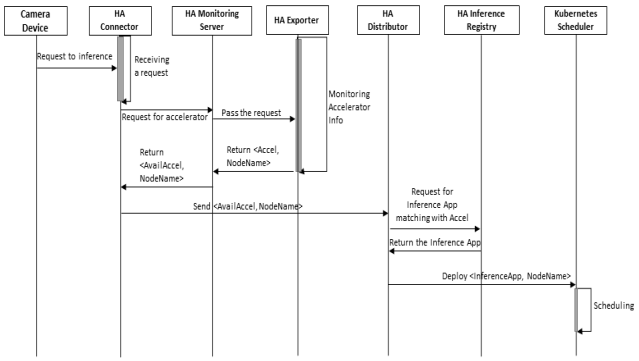

그림 2. HeteroAccel의 동작 순서도
Fig. 2. Sequence Diagram of HeteroAccel

- HA Distributor: 클러스터 내 모든 워커 노드의 HA Connector로부터 가속기 정보를 전달받고, HA Monitoring Server로부터 유휴 노드 정보를 전달받아 <유휴 가속기 종류, 가속기 ID, 노드 이름> 정보를 갱신한다. 이어서 사용자가 요청한 <추론 종류>와 매칭되는 추론 컨테이너를 HA Inference Registry로부터 확인한다. 끝으로 선택된 추론 응용 파드를 선택된 노드에 배포하도록 쿠버네티스 스케줄러에게 스케줄링을 요청한다.

- HA Inference Registry: HA Container Registry는 엣지 서버 내 연산·가속기 (CPU, FPGA, GPU, NPU)로 구동되는 추론 응용 전용 컨테이너와 정의 파일 (YAML 형식)의 저장소이다. 각각의 추론 응용 컨테이너들은 하나의 특정 가속기 정보가 정의 파일에 명시되어 있고 HA Distributor에 의해서 스케줄러에게 요청되어 실행될 수 있다. 그림 1의 HA Container Registry 내의 추론 응용 컨테이너의 표현은 다음과 같은 의미를 가진다. (A, B, C, D는 얼굴 인식, 객체 인식 등 각기 다른 추론 응용의 종류)

· I_AH_C= Inference A using CPU
· I_BH_G = Inference B using GPU
· I_CH_F = Inference C using FPGA
· I_DH_N = Inference D using NPU

- Kubernetes Scheduler: HA Distributor로부터 전달받은 정보를 바탕으로 사용자가 요청한 추론을 수행할 수 있는 추론 파드들의 집합을 스케줄링한다. 이후 동작은 2장에서 언급한 기본 쿠버네티스 스케줄러 동작을 수행한다. 다만, 다수의 가속기가 유휴한 상태이면 스케줄러의 필터링과 스코어링 정책을 통해서 우선순위 점수가 가장 높은 하나의 노드에서 추론 응용이 배포되고 실행된다.

3. HeteroAccel 동작 원리

실제 산업현장에 엣지 서버와 HeteroAccel이 배치되면, 다수의 영상 획득 장치와 엣지 서버와의 연결이 설정된다. HeteroAccel은 주기적으로 각 노드 내의 이종 연산·가속기의 상태를 모니터링하며 추론 요청을 대기한다. 이후 실제 영상

획득 장치에서 추론 요청이 발생하면 현재 엣지 서버 클러스터 상의 이종 연산·가속기 자원의 상태 모니터링을 통해 유휴성 측면에서 가장 적합한 노드 자원을 선택한다. 이후 동작 원리는 그림 2의 HeteroAccel 컴포넌트 간의 동작 순서도와 같으며, 이는 그림 1 내의 실행 흐름과도 일치한다.

① 영상 획득 장치 (Camera Device)에서 임의의 추론 서비스를 요청하고, HA Connector가 이를 수신한다.
② HA Connector는 HA Monitoring Server에 추론 요청을 전달한다.
③ HA Monitoring Server는 전체 HA Exporter로부터 현재 보유 중인 연산·가속기 정보와 가용 정보를 획득한다.
④ HA Connector는 파악된 유휴 연산·가속기 정보와 노드 정보를 HA Distributor에 전달한다.
⑤ HA Distributor는 영상 획득 장치가 요청한 추론 종류와 매칭되는 컨테이너 중에 현재 유휴 노드와 가속기 정보를 고려해서 선택한다.
⑥ 선택된 추론 컨테이너 정보를 쿠버네티스 스케줄러에 전달한다.
⑦ 쿠버네티스 스케줄러는 선택된 워커 노드에 추론 파드를 실행한다.
⑧ 실행된 추론 서버의 주소가 영상 획득 장치에 전달되고, 추론 서비스가 시작된다.

Ⅳ. 실 험

1. 실험환경

실험환경은 표 3과 같이 3개의 노드 (노드 G, 노드 F, 노드 N)로 하나의 엣지 컴퓨팅 클러스터를 구성하였으며 각각의 노드는 최소한 1개 이상의 가속기를 포함한다. 영상 획득 장치는 태블릿 PC를 사용하였다.

영상 획득 장치에는 4가지의 서로 다른 추론 서비스를 요청하도록 화면과 추론 결과 가시화 기능을 분할하였다. 실험에 사용된 추론 종류는 얼굴 인식, 동작 인식, 문자 인식, 객체 (손) 인식이며 오픈비노에서 기본적으로 제공하는 추론 응용을 쿠버네티스 파드로 가상화하여 사용하였다. 그림 3은 영상 획득 장치에서 HeteroAccel에 4가지 추론을 요청했을 때 추론 결과가 각각 표시되는 것을 보여준다.

HeteroAccel에 4개의 영상 획득 장치가 서로 다른 4가지의 추론 서비스 요청을 반복하였을 때 연산·가속기를 효율적으로 활용하여 전체 추론 성능이 향상되는지 확인하기 위한 목적으로 다음과 같이 비교 실험을 구성하였다.

첫 번째 실험군 (실험 A)은 그림 4와 같이 사용자가 파드 정의 파일에 요구 가속기 정보를 GPU로 명시한 후 4개의 영상 획득 장치가 4가지 서로 다른 추론 요청을 하는 것이다. 이 경우 기본 쿠버네티스 스케줄러에 의해 GPU가 탑재된 노드에 4가지 추론 종류에 대한 GPU 관련 파드 (IAHG, IBHG, ICHG, IDHG)가 배포된다. 실험군 A를 통해 GPU 외에 유휴 연산·가속기가 있음에도 추론이 진행되지 않는 상황을 유도할 수 있다.

표 3. 클러스터 구성과 연산·가속기 종류
Table 3. Cluster and Type of Accelerators

System	Processor	Accelerator
Node G	CPU: 11th Gen Intel(R) Core(RM) i7 @ 2.80GHz	GPU : Intel Iris Xe
Node F	CPU: Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz	FPGA: Intel Arria 10 GX
Node N	CPU: Intel(R) Xeon(R) CPU X5550 @ 2.67GHz	NPU : Intel Neural Compute Stick 2
Device	Samsung Galaxy Note 10 with Exynos 9 / Android 11	-

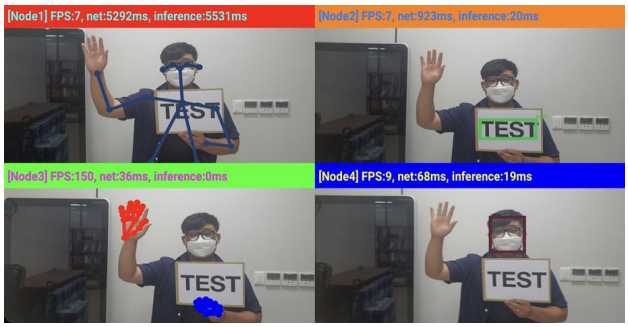

그림 3. HeteroAccel로의 다양한 추론 요청과 결과 가시화 (동작 인식, 문자 인식, 객체 (손)인식, 얼굴 인식)
Fig. 3. Various Inference Requests and Result of HeteroAccel

```
apiVersion: batch/v1
kind: Job
metadata:
  namespace: openvino
  name: openvino-gpu
  labels:
    jobgroup: intelgpu-demo
spec:
  parallelism: 1
  template:
    metadata:
      labels:
        jobgroup: intelgpu-demo
    spec:
      restartPolicy: Never
      containers:
        - name: intelgpu-demo-job-1
          image: openvino/cpu_gpu_npu:21.04
          imagePullPolicy: IfNotPresent
          command: [bash, "-c"]
          args: ["cd /opt/intel/openvino/deployment_tools/demo &&
                 ./demo_squeezenet_download_convert_run.sh -d GPU
                 &&sleep 1000"]

          resources:
            limits:
              gpu.intel.com/i915: 1
          volumeMounts:
          - mountPath: /dev
            name: dev
          securityContext:
            privileged: true
```

그림 4. 파드 정의 파일에 명시한 요구 가속기 정보
Fig. 4. Pod Information of Accelerator

두 번째 실험군 (실험 B)은 사용자가 가속기 정보를 따로 설정하지 않고 4개의 영상 획득 장치가 HA Connector에 각각 추론 요청을 하는 것이다. 이 경우 HeteroAccel은 유휴 노드와 연산·가속기 정보를 고려해 추론 파드를 배포한다.

반복적인 실험을 통해 아래와 같은 결과를 도출하였다. 실험 A에서는 4가지 추론 종류에 대한 GPU 추론 파드들이 노드 G에 배포 요청되었다. 기본 스케줄러의 우선 순위 정책에 따라 4가지 추론 파드 중 1개의 파드가 노드 G에 배포가 되면, 다른 파드는 그림 5와 같이 배포가 보류된다. 배포된 파드의 추론이 처리되고 가속기가 해제되면, 스케줄러

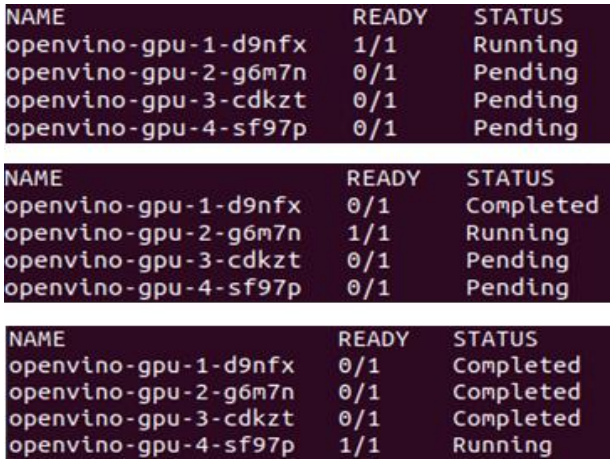

그림 5. 가속기 경쟁에 따른 파드의 배포 보류와 재개
Fig. 5. Deployment Progress of Accelerator Competition

표 4. 실험 결과
Table 4. Experiment Result

Test	# of Inference Completion	FPGA	CPU	NPU	GPU
A	260 times / 19,672 ms	-	-	-	260
	Time interval between requests	24.3 ms			
B	326 times / 19,542 ms	1	37	134	154
	Time interval between requests	21.8 ms			

우선순위 큐에 대기 중인 다음 추론 파드가 노드 G에 배포된다.

HeteroAccel 시스템을 통해 수행한 실험 B의 경우 첫 번째 요청에 대해서는 실험 A과 동일하게 유휴한 가속기가 있는 노드 G에 GPU 관련 추론 응용이 배포되었다. 이후 요청에 대해서 노드 G의 GPU 가속기가 해제되지 않았을 경우 노드 F나 노드 N의 FPGA나 NPU, 또는 CPU에서 동작할 수 있는 대체 파드가 스케줄러에 요청되어 배포되었다. 이렇게 사용자가 다수의 추론 서비스를 요청하더라도 HeteroAccel 시스템은 노드의 정보를 지속적으로 모니터링하다가 가속기가 해제되는 순간 스케줄링큐에 등록된 추론 파드들 중 유휴가속기를 사용할 수 있는 파드를 우선적으로 배포한다.

표 4는 각각 20초가량 진행된 실험 A, B에 대한 결과를 보여준다. 실험군 A에서는 약 24.3 ms, 실험군 B에서는 약 21.8 ms 간격으로 추론 요청이 전달된 것으로 확인되었다. 실험 A는 4개의 영상 획득 장치의 추론 요청 260번을 처리하였고, 실험 B는 326번 처리하여, 25.3%의 추가적인 추론을 처리하였다. 이를 통해 실험군 B에서는 GPU 외에 유휴 연산·가속기가 있음에도 추론이 진행되지 않는 상황 시 다른 가속기 활용된 것을 확인할 수 있다. 또한 실험 B의 추론 요청 326번에 대해 연산·가속기 별로 처리한 횟수는 FPGA 1회, CPU 37회, NPU 134회, GPU 154회였다. 결론적으로 기존 쿠버네티스 스케줄러를 통해 파드 배포를 하는 경우 가속기 경쟁에 따른 파드 배포 보류가 발생하는 반면, HeteroAccel의 경우 대체 가속기 활용을 통해 파드 배포를 효율적으로 처리해 전체 추론 성능이 향상되었음을 확인하였다. 다만 실험에 사용된 Intel Arria 10 FPGA가 추론에 특화된 장치가 아니었고, FPGA가 설치된 동일 노드의 CPU 부하, 다른 노드의 GPU의 유휴도 등에 따른 스케줄링 스코어링 단계에서의 점수 하락으로 인해 FPGA가 1회만 선택된 점은 HeteroAccel의 실제 활용을 위해서는 보다 많은 실험과 스케줄링 알고리즘의 최적화가 필요함을 보여준다.

V. 결 론

애플리케이션 개발과 배포의 단순화, 그리고 복잡한 리소스 관리 등을 해결해주는 쿠버네티스는 클라우드와 엣지 컴퓨팅 환경에서 점점 중요성이 높아지고 있다. 하지만 엣지 컴퓨팅 환경에 배치되는 엣지 서버 클러스터는 연산자원이 제한적이며, 추론 등 인공지능 서비스를 위해서는 효율적인 이종 연산·가속기 자원 관리가 필요하다. 파드 정의 파일에 응용 실행을 위해 요구되는 자원을 기입하고, 이를 바탕으로 쿠버네티스 스케줄러에 의해 파드가 배포되는 현재의 방식에서 서비스 운영자는 다양한 유휴 가속기를 사용하기 위해 파드 정의 파일을 수정하는 작업을 반복하고 배치를 요청하는 작업도 반복해야 하는 번거로움이 있다. 본 논문의 HeteroAccel 시스템은 다양한 추론 요청에 대해 유휴 노드에서 비할당된 가속기를 검색해서 원하는 추론 서비스를 대기 없이 수행할 수 있게 해준다. 이는 관련 서비스 개발 및 운영 효율을 높이는데 기여할 수 있고, 결과적으로 쿠버네티스 기반의 엣지 컴퓨팅 환경에서의 전체 추론 성능을 향상시킬 수 있다. 하지만 실세계 산업 현장에서 다수의 영상 획득 장치로부터 요청되는 다양한 추론 서비스를 제한적인 컴퓨팅 자원으로 제공하기 위해서는 보다 정밀한 자원 관리가 필요하다.

향후 연구에서는 추론 서비스에 특화된 연산·가속기 자원 관련 메트릭과 다양한 추론 종류와 연산·가속기 특성 간의 관계 등을 분석하고 이를 반영한 특화된 스케줄러에 관한 연구가 이루어져야 할 것이다. 또한 추론에 보다 최적화된 FPGA 활용, 스케줄링 정책 등 다양한 실험 조건의 구성을 통해 평균 지연 시간이나 꼬리 지연 시간 (tail latency), 연산·가속기 별 배치 특성 등의 결과를 도출하고, 이에 대한 분석을 통해 엣지 환경에서 효율적 다중 추론 시 제안하는 기법의 유효성을 보이는 연구로 확장될 것이다.

References

[1] J. Ren, D. Zhang, S. He, Y. Zhang, T. Li, “A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms: Transparent Computing, Mobile Edge Computing, Fog Computing, and Cloudlet,” ACM Computing Surveys, Vol. 52, No. 6, pp.1-36, 2020.

[2] X. Wang, Y .Han, V.C.M. Leung, D. Niyato, X. Yan, X. Chen, “Convergence of Edge Computing and deep Learning: A Comprehensive Survey,” IEEE Communications Surveys & Tutorials, Vol. 22, No. 2, pp. 869-904, 2020.
[3] Karbon 700, https://www.onlogic.com/computers/rugged/karbon700/
[4] MK400-70, https://www.onlogic.com/mk400-70/
[5] S. Kim, Y. Kim, “A design of GPU container co-execution framework measuring interference among applications,” KNOM Review, Vol. 23, No. 1, pp. 43-50, 2020.
[6] Q. Zeng, Y. Du, K. Huang, K.K. Leung, “Energy-Efficient Resource Management for Federated Edge Learning with CPU-GPU Heterogeneous Computing,” arXiv:2007.07122v2 [cs.IT], 2020.
[7] G. Cho, “Hybrid Resource Scheduling Scheme for Video Surveillance in GPU-FPGA Accelerated Edge System,” Master’s thesis, KAIST, https://koasas.kaist.ac.kr/handle/10203/284802
[8] X. Liu, J. Yang, C. Zou, Q. Chen, “Collaborative Edge Computing With FPGA-Based CNN Accelerators for Energy-Efficient and Time-Aware Face Tracking System,” IEEE Transactions on Computational Social Systems (Early Access), doi: 10.1109/TCSS.2021.3059318
[9] OpenVINO, https://docs.openvinotoolkit.org/
[10] Kubernetes, https://kubernetes.io/
[11] Z. Zhong, and R. Buyya, “ A Cost-Efficient Container Orchestration Strategy in Kubernetes-Based Cloud Computing Infrastructures with Heterogeneous Resources,” ACM Trans. on Internet Technology, Vol. 20, Issue 2, pp. 1-24, 2021
[12] Prometheus, https://prometheus.io/

Jaeho Jeon (전 재 호)

2008 SW Engineering from Auburn University (B.S.E)
2010 SW Engineering from Auburn University (M.S.E)
2010~Electronics and Telecommunications Research Institute (ETRI), a senior researcher
Fields of interests: Edge Computing, CPS, AI Platform
Email: jeonjaeho11@etri.re.kr

Yongyeon Kim (김 용 연)

2008 Computer Engineering from Chungnam National University (B.S.)
2010 Computer Engineering from Chungnam National University (M.S.)
2010~Electronics and Telecommunications Research Institute (Senior Researcher) researcher.
Fields of interests: Embedded systems and operations, System, communication middleware, artificial intelligence SW platform
Email: yyeon@etri.re.kr

Sungjoo Kang (강 성 주)

2003 Electronic Engineering from Hanyang University (B.E.)
2005 Electronic Engineering from Hanyang University (M.E.)
2011 Software Engineering from Korea Advanced Institute of Science and Technology (KAIST) and Carnegie Mellon University, respectively, (M.S.E)
2018 Computer Engineering from Chungnam National University (Ph.D.)
Field of Interests: Cyber-Physical Systems, Edge Computing, Multimodal Interactions in Metaverse, Autonomous Systems, Digital Twin, Blockchain and Cryptocurrency
Email: sjkang@etri.re.kr