KIPS Trans. Softw. and Data Eng.
Vol.12, No.9 pp.399~406
ISSN: 2287-5905 (Print), ISSN: 2734-0503 (Online)

https://doi.org/10.3745/KTSDE.2023.12.9.399

Deep Learning-Based Dynamic Scheduling with Multi-Agents Supporting Scalability in Edge Computing Environments

JongBeom Lim†

ABSTRACT

Cloud computing has been evolved to support edge computing architecture that combines fog management layer with edge servers. The main reason why it is received much attention is low communication latency for real-time IoT applications. At the same time, various cloud task scheduling techniques based on artificial intelligence have been proposed. Artificial intelligence-based cloud task scheduling techniques show better performance in comparison to existing methods, but it has relatively high scheduling time. In this paper, we propose a deep learning-based dynamic scheduling with multi-agents supporting scalability in edge computing environments. The proposed method shows low scheduling time than previous artificial intelligence-based scheduling techniques. To show the effectiveness of the proposed method, we compare the performance between previous and proposed methods in a scalable experimental environment. The results show that our method supports real-time IoT applications with low scheduling time, and shows better performance in terms of the number of completed cloud tasks in a scalable experimental environment.

Keywords : Dynamic Scheduling, Deep Learning, Cloud Computing, Edge Cloud, Multi-Agent

멀티 에이전트 에지 컴퓨팅 환경에서 확장성을 지원하는 딥러닝 기반 동적 스케줄링

임 종 범†

요 약

클라우드 컴퓨팅은 에지 서버가 동작하는 포그(fog) 레이어가 결합된 에지(edge) 컴퓨팅 아키텍처로 진화하고 있다. 에지 컴퓨팅 아키텍처가 관심을 받는 이유는 짧은 통신 지연으로 실시간 IoT 응용을 지원할 수 있기 때문이다. 이와 동시에 인공지능 기술을 도입한 많은 클라우드 작업 스케줄링 기법들이 제안되었다. 인공지능 기반의 클라우드 작업 스케줄링 기법은 기존 기법보다 더 좋은 성능을 보이지만 스케줄링 시간이 다소 소요된다는 단점이 있다. 이 논문에서는 에지 컴퓨팅 환경에서 분산 딥러닝 학습 기반의 동적 스케줄링 기법을 제안한다. 제안하는 기법은 기존 기법보다 스케줄링 시간이 짧은 장점이 있다. 또한 멀티 에이전트를 통한 분산 딥러닝 학습의 효과성을 보이기 위해 확장적인 실험 환경에서 제안 기법과 기존 인공지능 기법의 성능일 비교 평가하였다. 성능 실험 결과 기존 인공지능 기반 클라우드 작업 스케줄링 기법보다 짧은 스케줄링 시간을 보여 IoT 실시간 응용에 적합함을 보였으며, 확장적인 실험에서도 제안 기법이 완료된 작업의 수에 대하여 우수한 성능을 보임을 증명하였다.

키워드 : 동적 스케줄링, 딥러닝, 클라우드 컴퓨팅, 에지 클라우드, 멀티 에이전트

1. 서 론

최근 인공지능 응용의 활용성이 높아지면서 인공지능 응용을 클라우드 컴퓨팅 환경에서 처리하는 사례가 증가하고 있다[1-2]. 이러한 응용의 예는 전문가 시스템, 자연어 처리, 데이터 마이닝, 컴퓨터 비전, 지능 로봇 등을 들 수 있다[3]. 또한 5G 통신 환경 및 고속 이더넷 환경이 대중화되면서 통신 지연이 줄어듦에 따라 실시간 데이터 처리 응용에 에지(edge) 컴퓨팅 기반 클라우드 컴퓨팅 환경이 각광을 받고 있다[4]. 에지는 네트워크 에지에 있는 컴퓨팅 위치와 해당 물리적 위치에 있는 하드웨어와 소프트웨어를 말한다. 일반적인 클라우드 컴퓨팅 환경에서는 에지 서버가 아닌 클라우드 내의 가상머신 또는 컨테이너 내에서 워크로드가 실행되는 환경임에 반해, 에지 컴퓨팅 환경에서는 에지 장치에서 워크로드가 실행된다[5,6].

에지 컴퓨팅의 스케줄러의 역할은 요청된 서비스에 해당하는 컨테이너를 어느 호스트에 할당할지에 대한 물음의 답을 제공하는 것이다[7,8]. 즉, 스케줄러의 역할은 기본적으로 연

※ 이 논문은 2022학년도 평택대학교 학술연구비의 지원에 의하여 연구되었음.
† 종신회원 : 평택대학교 ICT융합학부 스마트콘텐츠전공 조교수
Manuscript Received : March 6, 2023
First Revision : May 31, 2023
Accepted : June 5, 2023
* Corresponding Author : JongBeom Lim(jblim@ptu.ac.kr)

산 노드들이 여러 개 있을 때 서비스를 할당하기에 가장 알맞은 연산 노드를 찾는 것이다. 에지 컴퓨팅은 기존 중앙 집중형 클라우드 기반의 서비스를 제공하는 솔루션을 에지 플랫폼에서 활용할 수 있도록 에지와 에지 간에 협업하고 연동하는 인터페이스를 제공하여 에지 기반 솔루션의 신속한 전환을 가능하게 하여 새로운 서비스를 제공하는 연구가 필요하다. 기존 에지 기반 클라우드 컴퓨팅 연구에서는 일반적인 데이터 및 연산 처리를 위한 방법들이 논의되었으며, 딥러닝을 비롯한 인공지능 응용에 적합한 에지 기반 클라우드 컴퓨팅 아키텍처 및 분산 스케줄링 기법에 대한 연구는 아직 미비한 수준이다.

이 연구에서는 에지 클라우드 환경에서 확장성을 지원하는 멀티 에이전트 기반 동적 스케줄링 기법에 대한 연구를 진행한다. 에지 컴퓨팅을 위한 스케줄러를 개발하는 것이 연구의 범위이며, 사용자로부터 요청된 서비스들을 하나의 에지 클러스터에 국한되지 않고 클라우드 및 지리적으로 분산된 에지 클러스터들을 통합하여 인공지능 응용에 대하여 멀티 에이전트가 리소스에 할당 및 배치하는 연구를 수행한다. 여기에 멀티 에이전트 기반의 분산 하이퍼파라미터 최적화 방법을 적용하여 스케줄링에 적합한 값을 도출하고 이를 에지 클라우드에 적용한다.

제안 연구를 통해 얻을 수 있는 장점은 다음과 같다.

- 단일 클러스터의 스케줄러의 역할을 넘어서 다중 클러스터를 고려한 스케줄러로 기능을 확대하여 적용할 수 있다.
- 제안하는 에지 클라우드 스케줄러는 멀티 에이전트 기반 순차 처리를 위한 스케줄러 형식과 서로 경쟁하는 스케줄러 형식이 복합된 구조를 통해 다중 클러스터를 지원할 수 있다.
- 응답 속도 민감형 서비스나 클라우드-에지-단말 간 협업 기반 확장성을 지원하는 지능형 서비스를 효과적으로 지원할 수 있다.

이 논문의 구성은 다음과 같다. 2장에서는 에지 컴퓨팅 환경에서 분산 딥러닝 학습에 대한 연구 배경 및 동기를 설명하고, 3장에서는 멀티 에이전트 기반 스케줄링 기법의 알고리즘을 상세히 소개한다. 4장에서 인공지능 기반 다른 기법들과의 성능 비교를 보이고, 마지막으로 5장에서는 논문의 결론을 맺는다.

2. 연구의 배경 및 동기

온프레미스(on-premise) 컴퓨팅 대비 클라우드 컴퓨팅의 경제성 장점을 활용하고자 실시간 데이터 저장 및 처리를 위해 IoT 응용을 배포하기 위한 에지 컴퓨팅이 사용되고 있다[9]. 클라우드-포크 컴퓨팅에서의 IoT 응용 배포를 위해서는 중앙집중형 클라우드 서버가 아닌 사용자 및 클라이언트에 가까운 에지 서버에서 데이터 저장 및 처리를 지시할 수 있다[10-11]. 이로써 통신 지연을 낮추고 실시간 응용에 적합한 응답을 해줄 수 있다.

Fig. 1은 에지 컴퓨팅 아키텍처를 보여준다. 최상위단에는 중앙집중형 클라우드 서버가 존재하고 최하위단에는 IoT 장

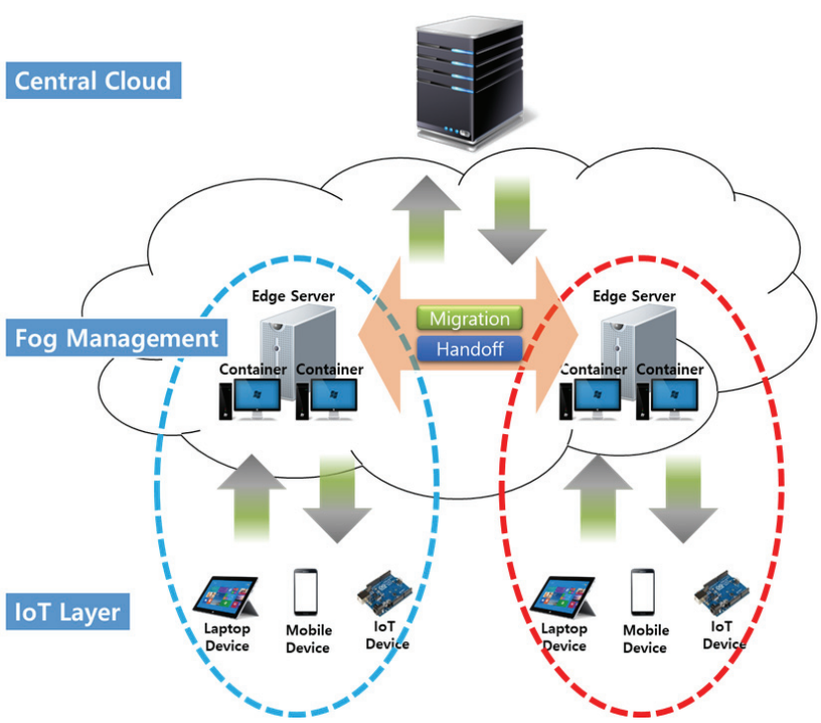


Fig. 1. Edge Computing Architecture

치들이 존재한다. 그 사이에 포그(fog) 관리 레이어가 존재하는데 여기에 에지 서버들이 존재하여 중앙집중형 클라우드 서버와 IoT 장치들을 중재하는 역할을 한다. 모든 IoT 장치들이 중앙집중형 클라우드 서버와 통신하는 것도 가능하지만 물리적으로 멀리 떨어져 있고, 병목현상이 발생하는 구조이기 때문에 통신 지연이 발생하는 단점이 있다. 포그 레이어에 존재하는 에지 서버들이 IoT 장치와 중앙집중형 클라우드 서버를 중재함에 따라 IoT 장치들은 적은 통신 지연으로 실시간 응용을 처리할 수 있다.

이러한 에지 컴퓨팅 환경에서 클라우드 작업을 포그 장치 및 에지 서버에 할당하고 자원을 관리하는 스케줄링 문제는 아직 완벽히 해소되지 않았으며 현재까지 다양한 연구가 활발히 진행 중이다. 그 중에서 DRL(deep reinforcement learning) [12] 및 DQL(deep q-learning)[13]을 사용하는 클라우드 작업 스케줄링 방법은 기존의 에지 환경에서의 스케줄링 문제를 해결함에 있어 전처리 및 복잡도 측면에서 한 층 도움을 주고 있다.

하지만 DRL 및 DQL 기반 에지 작업 스케줄링 기법의 단점은 다소 높은 스케줄링 시간을 들 수 있다. Fig. 2는 DRL, DQL과 이 논문에서 제안하는 기법인 Ours1, Ours2의 스케줄링 시간을 보여준다. 그림에서 색상 배경은 신뢰구간을 나타낸다. Ours1과 Ours2는 뉴럴 네트워크를 이용한 기법으로 이에 대한 자세한 내용은 4장에서 다룬다. 제안하는 분산 딥러닝 학습의 확장성을 측정하기 위해 분산 딥러닝 학습을 제외한 스케줄링 기법과 분산 딥러닝 학습을 포함한 스케줄링 기법을 비교하였다. 전자를 Ours1로 표기하였으며, 후자를 Ours2로 표기하였다. 그림에서 볼 수 있듯이 DRL과 DQL의 스케줄링 시간은 Ours1과 Ours2에 비해 높게 나타났다. 이에 비해 Ours1과 Ours2의 스케줄링 시간은 0.15초 이내로 짧다. 스케줄링 시간이 짧은 것과 더불어 DRL과 DQL은 스케줄링 시간의 편차가 크기 때문에 실시간성이 요구되는 IoT 응용 및 클라우드 작업을 수행하기에는 무리가 있고 SLA(service level agreement) 및 SLO(service level objective) 위반이 높게 나타날 우려가 있다[14].

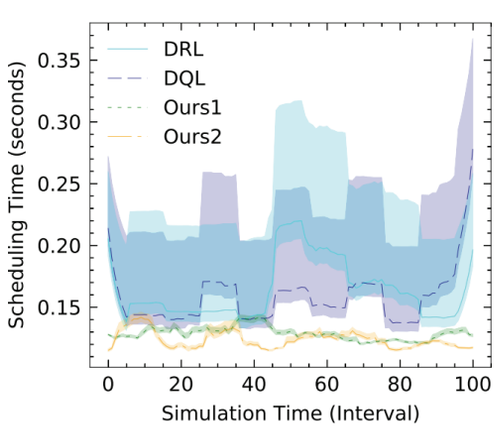


Fig. 2. Mini Benchmark for Scheduling Time

최근에는 IoT 장치의 폼팩터 기술이 발전함에 따라 소형 장치에서도 여러 가지 응용을 처리할 수 있는 모듈이 내장되고 있다. 이에 따라 인공지능 및 딥러닝 응용도 에지 서버 및 IoT 장치에서 처리할 수 있게 되었다[15]. 따라서 이 논문에서는 중앙집중형 클라우드에서 수행했었던 딥러닝 기반 에지 작업 스케줄링의 연산을 에지 서버에서 분산처리하고 학습의 결과를 집성하는 스케줄링 구조 및 기법을 제안한다. 각각의 에지 서버에는 분산 딥러닝 학습을 위한 소프트웨어인 에이전트를 배포하고, 배포된 멀티 에이전트가 에지 작업 스케줄링의 학습 일부를 진행하고 자가적인 방법으로 자원 및 작업을 할당한다.

3. 제안하는 분산 딥러닝 기반 클라우드 작업 스케줄링 알고리즘

이 장에서는 멀티 에이전트 에지 클라우드 환경에서 확장성을 지원하는 딥러닝 기반 동적 스케줄링에 대한 알고리즘을 소개한다. Fig. 3은 제안하는 분산 딥러닝 학습 구조를 보여준다. 그림 상에서 Agent의 역할은 제안 연구에서 에지 서버가 담당한다. 그림과 같은 분산 딥러닝 학습을 통해 스케줄링과 관련된 하이퍼파라미터를 학습하여 클라우드 작업에 대한 처리를 최적화한다. Fig. 3에서 파라미터 서버는 전체 딥러닝 학습에 대한 정보를 가지고 있으며 이는 중앙집중형 클라우드 서버에서 관리한다.

중앙집중형 클라우드 서버는 에지 컴퓨팅 환경에 대한 정보를 가지고 있으므로, 자신이 관리하는 에지 서버들에 대하여 모니터링하고 이를 기반으로 각각의 에지 서버가 처리해야 하는 학습 분량을 정하여 해당 에지 서버들에게 학습을 지시한다. 이는 그림 상에서 Agent 1, Agent 2, ..., Agent N으로 표시되어 있다. 에지 서버에 할당된 Agent 스레드를 모두 수행하였다면 해당 에지 서버에 할당된 분산 딥러닝 학습이 완료되었으므로 학습된 결과를 파라미터 서버에게 돌려준다. 파라미터 서버는 취합된 학습 결과를 통합하여 스케줄링에 사용한다.

Table 1과 Table 2는 이에 대한 상세한 알고리즘을 보여준다. Table 1에서 파리미터 서버에 대한 분산 딥러닝 학습 알고리즘은 액티브 스레드와 패시브 스레드에 대한 알고리즘으로 나뉘어진다. 액티브 스레드에서는 학습해야 할 딥러닝 구조를 에지 서버에게 쪼개어 나누어주고, 모든 에지 서버에서 학습이 완료되었을 때 글로벌 스케줄링 정보를 취합한다. 패시브 스레드에서는 에지 서버들이 보내오는 학습 파편들을 수신하고 수신된 학습 정보를 취합한다. 모든 에지 서버로부터 학습 파편을 수신하였다면 *Edges_learned* 변수의 값을 *true*로 설정하여, 액티브 스레드에서 글로벌 스케줄링 정보를 취합하도록 한다.

파라미터 서버에서 에지 서버에게 나누어주는 분산 학습 파편을 나누는 기준은 크게 두 가지로 나눌 수 있다. 하나는 모

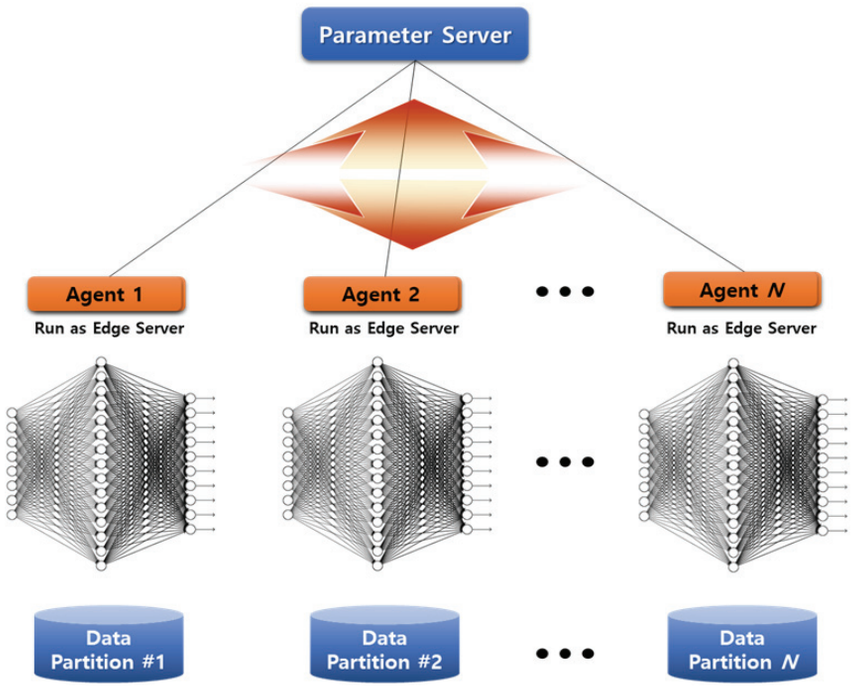


Fig. 3. Proposed Distributed Deep Learning Structure

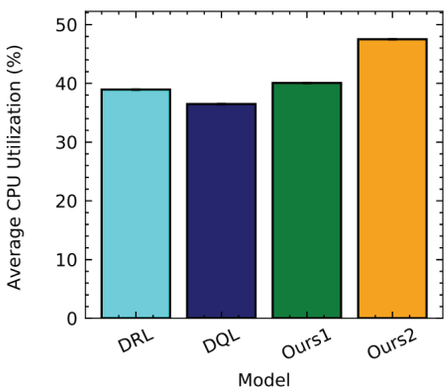
Table 1. Algorithm for the Parameter Server

```
• Input
  - Dataset, Cloud_tasks, Sys_info
• Output
  - Model_global, Sched_global
• Initialization
  - Edges ← Sys_info.Get_edge();
  - Train ← mix_datasets(Dataset, Cloud_tasks);
  - Edges_learned ← false;
• Pseudocode for active thread
  Splits ← split_data (Train, Edges.Length());
  for all Edge_i ∈ Edges do
      Split_i ← get_split(Splits);
      distribute(Edge_i, Split_i);
  end for
  if Edges_learned == true then
      Sched_global ← schedule(Model_global);
  end if
• Pseudocode for passive thread
  for all Edge_i ∈ Edges do
      Trained_i ← wait_for(Edge_i);
      Model_global ← Model_global ∪ Trained_i;
  end for
  Edges_learned ← true;
```

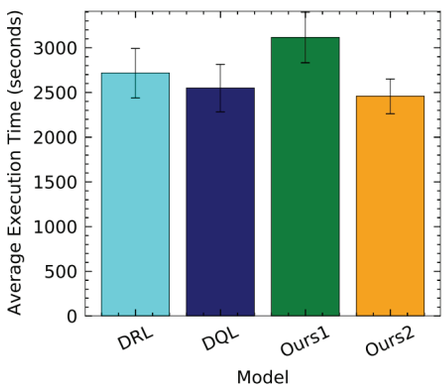
Table. 2. Algorithm for Edge Servers

```
• Input
  - Split_i
• Output
  - Model_partial
• Initialization
  - Params ← Sys_info.Get_params(edge);
• Pseudocode
  Model_partial ← perform_train(Split_i, Params);
  send(Model_partial, Server);
```

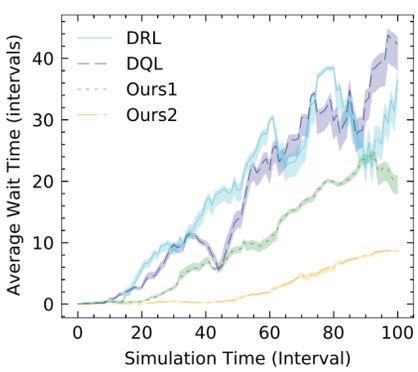
델 기반 파티셔닝 방법이고, 다른 하나는 샘플 기반 파티셔닝 방법이다[16]. 모델 기반 파티셔닝 기법은 분할된 하나의 파티션이 CPU, GPU, TPU(tensor processing unit), NPU(neural processing unit)에서 학습이 가능하도록 해야 하는 추가적인 요구사항이 있다. 따라서 모델 기반 방법은 파티셔닝 시 주의가 요구된다. 반면, 샘플 기반 파티셔닝 방법은 랜덤 또는 셔플링 방식으로 데이터를 뒤섞은 다음 단위 작업으로 나누어 에지 서버에게 나누어 준다. 샘플 기반 파티셔닝 방법은 특징 차원 기반으로 데이터가 나누어지고 셔플링을 통하여 더 좋은 일반화 능력을 가질 수 있다. 따라서 이 연구에서는 샘플 기반 파티셔닝 방법을 사용하여 분산 딥러닝 학습을 이루어지도록 하였다.



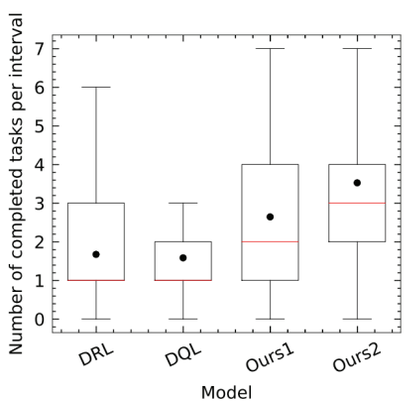
(a) Average CPU utilization



(b) Average execution time



(c) Average wait time



(d) Number of completed tasks

Fig. 4. Performance Results for CPU Utilization, Execution Time, Wait Time, and Number of Completed Tasks

4. 실험 평가

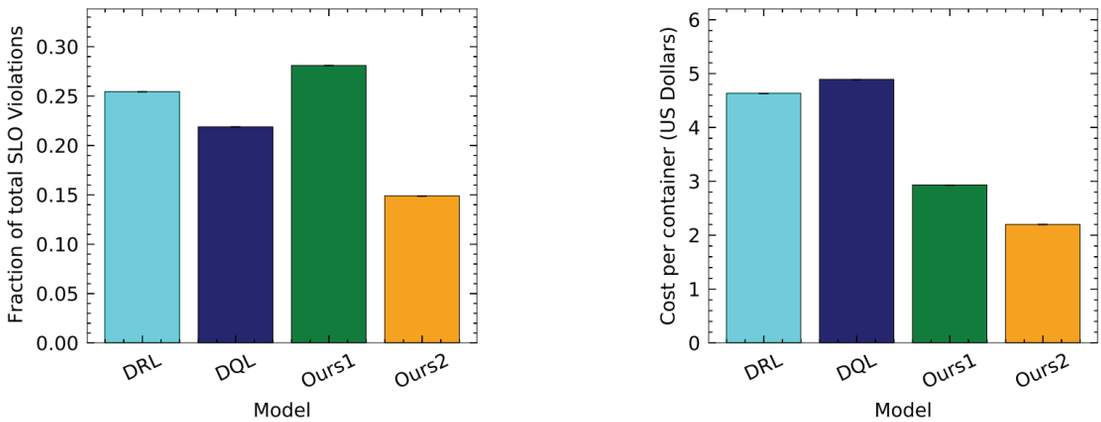
이 장에서는 멀티 에이전트 에지 클라우드 환경에서 확장성을 지원하는 딥러닝 기반 동적 스케줄링에 대한 실험 환경 및 실험 결과를 비교한다. 제안하는 분산 딥러닝 학습의 확장성을 측정하기 위해 분산 딥러닝 학습을 제외한 스케줄링 기법과 분산 딥러닝 학습을 포함한 스케줄링 기법을 비교하였다. 전자를 Ours1로 표기하였으며, 후자를 Ours2로 표기하였다. 또한, 기존 인공지능 기반 클라우드 작업 스케줄링과의 비교를 위해 DRL과 DQL 기법을 구현하여 Ours1과 Ours2와의 성능을 비교하였다.

에지 작업에 대한 데이터셋은 BitBrain[17]과 DeFog[18] 데이터셋을 사용하였으며, 에지 서버의 사양은 Azure B2 모델과 B4 모델을 사용하였다. 두 모델 모두 인텔 E5-2673 사양이며, 램의 용량만 각각 4GB와 16GB로 차이가 있다. 에지 서버의 수는 30개로 설정하여 확장적인 실험을 지원하도록 설정하였다.

뉴럴 네트워크 구조는 피드-포워드 구조이며, 두 개의 히든 레이어를 가진다. 히든 레이어의 크기는 각각 128과 64로 설정하였다. 활성화 함수는 시그모이드 함수를 사용하였으며 비어파인 근사에는 softplus와 tanhshrink 함수를 사용하였다. 뉴럴 네트워크의 입력은 작업의 크기, 에지 서버의 수, 작업의 수, 특징 벡터의 수에 따라 상이하게 구성된다. 이러한 뉴럴 네트워크의 구조는 [19]와 유사한 구조를 가지며, 입력에 대한 역전파 학습이라고 볼 수 있다.

Fig. 4(a)는 DRL, DQL, Ours1, Ours2의 평균 CPU 사용률을 보여준다. CPU 사용률은 클라우드 작업을 수행하는 것뿐만 아니라 스케줄링 및 분산 딥러닝 학습을 모두 포함하기 때문에 Ours2가 Ours1 보다 더 높은 사용률이 나온 것을 볼 수 있다. 분산 딥러닝 학습에 대한 CPU 사용률을 제외한다면 CPU 사용률이 높을 수도록 클라우드 작업을 더 빨리 수행한다고 판단할 수 있다. Ours2가 다른 세 개의 기법보다 약 4% 더 높은 CPU 사용률을 보였다.

Fig. 4(b)는 DRL, DQL, Ours1, Ours2의 평균 작업 수행 시간을 보여준다. 작업 수행 시간이 높다는 것은 클라우드 작업에 대한 스케줄링의 효율이 떨어진다는 것을 의미하며, 작업 수행 시간이 낮다는 것은 같은 클라우드 작업에 대하여 스케줄링이 잘 이루어져 클라우드 작업을 빠른 시간 내에 끝낸다는 것을 의



(a) Fraction of total SLO violations.

(b) Cost per container.

Fig. 5. Performance Results for SLO Violations and Cost

미한다. Ours1이 다른 세 개의 기법보다 작업 수행 시간이 높은 이유는 뉴럴 네트워크의 학습이 없이 클라우드 작업 할당 및 이주 기법만을 사용하였을 때 시너지 효과가 떨어진 것으로 보인다. 반면 Ours2는 클라우드 작업에 대한 스케줄링과 더불어 멀티 에이전트를 통한 뉴럴 네트워크 학습이 이루어져서 다른 세 개의 기법보다 더 짧은 작업 수행 시간이 나타났다.

Fig. 4(c)는 DRL, DQL, Ours1, Ours2의 스케줄링 간격 당 평균 대기 시간을 보여준다. 그림에서 색상 배경은 신뢰구간을 나타낸다. 대기 시간은 클라우드 작업이 에지 서버에서 시작한 시간에서 클라우드 작업이 할당된 시간을 뺀 값을 의미한다. 따라서 대기 시간은 짧을수록 좋으며, 대기 시간이 짧다는 것은 클라우드 작업에 대한 스케줄링이 잘 이루어졌다는 것을 의미한다. 그림에서 알 수 있듯이 DRL과 DQL은 Ours1과 Ours2 보다 더 높은 평균 대기 시간을 보였으며, 편차도 더 큰 것을 알 수 있다. 반면 Ours2는 다른 세 개의 기법보다 가장 짧은 평균 대기 시간을 보였다.

Fig. 4(d)는 DRL, DQL, Ours1, Ours2의 수행 완료된 클라우드 작업의 수를 보여준다. 앞선 실험 결과를 기반으로 예상한 것과 같이 Ours2가 나머지 세 개의 기법보다 가장 많은 수의 클라우드 작업을 완료한 것을 볼 수 있다. Ours1도 DRL과 DQL보다는 더 많은 수의 클라우드 작업을 완료하였지만 뉴럴 네트워크에 대한 학습이 이루어지지 않아 Ours2 보다는 적은 수의 클라우드 작업을 완료한 것으로 볼 수 있다. 이러한 결과를 토대로 에지 컴퓨팅 환경에서의 분산 딥러닝 학습에 대한 제안 기법이 효과적으로 잘 적용됐다고 볼 수 있다.

Fig. 5(a)는 DRL, DQL, Ours1, Ours2의 SLO 위반 정도를 보여준다. SLO 위반은 워크로드에 대한 이상적인 스케줄링 대비 초과된 응답 시간을 기준으로 측정하였다. 여기서 응답 시간은 클라우드 작업이 제출되고 완료된 시간까지의 시간 차이를 의미한다. SLO 위반은 Ours1이 가장 높게 나타났으며 Ours2가 가장 낮게 나타났다. Ours2는 15% 미만의 결과를 보이며 Ours1은 약 28%의 결과를 보였다.

Fig 5(b)는 DRL, DQL, Ours1, Ours2의 컨테이너당 비용($)을 보여준다. 에지 서버가 Intel E5-2673 v3를 사용였을 때 Microsoft Azure 데이터센터의 비용을 기준으로 컨테이너당 비용을 측정하였다. Ours2의 컨테이너당 비용은 약 $2.2 정도로 네 개의 기법 중 가장 낮은 비용의 결과를 보였으며, DQL이 약 $4.9 정도로 가장 높은 결과를 보였다. 각 기법에 따라 전체 컨테이너의 수가 다르므로 컨테이너당 비용과 전체 컨테이너의 수를 곱하더라도 Ours2가 DQL 대비 약 50% 적은 비용이 든 것을 알 수 있다. DQL의 스케줄링 간격 당 평균 컨테이너의 수는 75개 이며, Ours2의 스케줄링 간격 당 평균 컨테이너의 수는 85개로 나타났다.

성능 실험 결과에 대하여 결과의 원인을 생각해보면 다음과 같이 정리할 수 있다. 실험 대상군의 인공지능 기반 스케줄링 방식에 비하여, 제안 연구는 뉴럴 네트워크 기반 스케줄링 방식으로 동일한 작업 종류에 대한 학습이 진행되어 새로운 작업이 요청되었을 때 더 적은 스케줄링 시간을 보여주며, 에이전트가 분산 딥러닝 학습을 통하여 학습에 대한 병목현상 및 오버헤드를 감소시킬 수 있는 것을 알 수 있다. 학습의 양과 에지 서버의 수에 따라 분산 딥러닝의 효과의 양상이 다르게 나올 수 있으나, 실험 환경에서의 설정(에지 서버의 수: 30)과 실험 결과를 토대로 성능적인 이득을 확인하여 실제 배포 환경에서의 적용을 검토할 수 있을 것으로 기대한다.

5. 결 론

이 논문에서는 에지 컴퓨팅 환경에서 분산 딥러닝 학습 기반의 동적 스케줄링 기법을 제안하였다. 제안하는 기법은 기존 딥러닝 기반 클라우드 작업 스케줄링 기법 대비 분산 딥러닝 학습을 진행하기 때문에 중앙집중형 클라우드 서버의 병목

현상을 줄일 수 있으며 에지 서버에서 로컬 학습이 가능하여 호스트(에지 서버)의 수가 증가하더라도 성능 저하가 많이 발생하지 않고 확장성을 지원한다. 에지 서버에 존재하는 멀티-에이전트는 로컬 학습 데이터의 결과만을 중앙집중형 클라우드 서버에 전송하고, 중앙 집중형 클라우드 서버는 취합된 모델을 통하여 스케줄링을 수행하기 때문에 스케줄링 시간을 줄일 수 있어 IoT 실시간 응용에 적합하다. 성능 실험 결과 기존 인공지능 기반의 클라우드 작업 스케줄링 기법보다 우수한 성능이 보임을 증명하였다.

References

[1] L. Sun, X. Jiang, H. Ren, and Y. Guo, "Edge-Cloud computing and artificial intelligence in internet of medical things: Architecture, technology and application," *IEEE Access*, Vol.8, pp.101079-101092, 2020.

[2] U. F. Mustapha, A.-W. Alhassan, D.-N. Jiang, and G.-L. Li, "Sustainable aquaculture development: A review on the roles of cloud computing, internet of things and artificial intelligence (CIA)," *Reviews in Aquaculture*, Vol.13, No.4, pp.2076-2091, 2021.

[3] Y. Pan and L. Zhang, "Roles of artificial intelligence in construction engineering and management: A critical review and future trends," *Automation in Construction*, Vol.122, pp.103517, 2021.

[4] G. Ananthanarayanan et al., "Real-time video analytics: The killer app for edge computing," *Computer*, Vol.50, No.10, pp.58-67, 2017.

[5] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, Vol.8, pp.85714-85728, 2020.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, Vol.3, No.5, pp.637-646, 2016.

[7] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Communications Surveys & Tutorials*, Vol.23, No.4, pp.2131-2165, 2021.

[8] X. Li, J. Wan, H. N. Dai, M. Imran, M. Xia, and A. Celesti, "A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing," *IEEE Transactions on Industrial Informatics*, Vol.15, No.7, pp.4225-4234, 2019.

[9] S. Kunal, A. Saha, and R. Amin, "An overview of cloud-fog computing: Architectures, applications with security challenges," *Security and Privacy*, Vol.2, No.4, pp.e72, 2019.

[10] D. Kimovski, R. Mathá, J. Hammer, N. Mehran, H. Hellwagner, and R. Prodan, "Cloud, fog, or edge: Where to compute?," *IEEE Internet Computing*, Vol.25, No.4, pp.30-36, 2021.

[11] V. Prokhorenko and M. A. Babar, "Architectural resilience in cloud, fog and edge systems: A survey," *IEEE Access*, Vol.8, pp.28078-28095, 2020.

[12] G. Rjoub, J. Bentahar, O. Abdel Wahab, and A. Saleh Bataineh, "Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems," *Concurrency and Computation: Practice and Experience*, Vol.33, No.23, pp.e5919, 2021.

[13] Y. Ran, H. Hu, X. Zhou, and Y. Wen, "DeepEE: Joint optimization of job scheduling and cooling control for data center energy efficiency using deep reinforcement learning," *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp.645-655, 2019.

[14] A. Alqahtani, Y. Li, P. Patel, E. Solaiman, and R. Ranjan, "End-to-End service level agreement specification for IoT applications," *in 2018 International Conference on High Performance Computing & Simulation (HPCS)*, 16-20 July 2018, pp.926-935, 2018.

[15] Q. Liang, P. Shenoy, and D. Irwin, "AI on the Edge: Characterizing AI-based IoT applications using specialized edge architectures," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*, 27-30 Oct. 2020, pp. 145-156, 2020.

[16] X. Xie et al., "A transferable approach for partitioning machine learning models on multi-chip-modules," *Proceedings of Machine Learning and Systems*, Vol.4, pp. 370-381, 2022.

[17] S. Shen, V. V. Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 4-7 May 2015, pp.465-474, 2015.

[18] J. McChesney, N. Wang, A. Tanwer, E. De Lara, and B. Varghese, "Defog: Fog computing benchmarks," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pp.47-58, 2019.

[19] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks," *IEEE Transactions on Mobile Computing*, Vol.21, No.3, pp.940-954, 2022.

임 종 범

https://orcid.org/0000-0001-8954-2903
e-mail : jblim@ptu.ac.kr
2009년 백석대학교 정보통신학부(학사)
2011년 고려대학교 컴퓨터교육과(석사)
2014년 고려대학교 컴퓨터교육과(박사)
2015년 ~ 2017년 동국대학교 IT융합교육센터 초빙교수
2017년 ~ 2021년 한국공학대학교 게임공학부 조교수
2021년 ~ 현 재 평택대학교 ICT융합학부 스마트콘텐츠전공 조교수
관심분야 : Distributed & Cloud Computing, Resource Management, Artificial Intelligence Application