논문 2020-15-34

저지연 서비스를 위한 Multi-access Edge Computing 스케줄러

(Multi-access Edge Computing Scheduler for Low Latency Services)

김 태 현, 김 태 영, 진 성 근*
(Tae-Hyun Kim, Tae-Young Kim, Sunggeun Jin)

Abstract : We have developed a scheduler that additionally consider network performance by extending the Kubernetes developed to manage lots of containers in cloud computing nodes. The network delay adapt characteristics of the compute nodes were learned during server operation and the learned results were utilized to develop placement algorithm by considering the existing measurement units, CPU, memory, and volume together, and it was confirmed that the low delay network service was provided through placement algorithm.

Keywords : Cloud computing, Containers, Edge computing, Scheduling, Low latency, Network

Ⅰ. 서 론

최근 클라우드 컴퓨팅 시장이 빠른 속도로 성장하고 있다. 쉽게 접할 수 있는 클라우드 저장소를 넘어 사용 고성능 컴퓨팅 자원을 빌려주는 것에 이르렀다. 특히 MEC (Multi-access Edge Computing) 기술을 활용하여 모바일 사용자와 가까운 RAN (Radio Access Network) 내에 클라우드 컴퓨팅 환경을 제공함으로써 서비스 지연을 줄이는 등 다양한 측면에서 클라우드 컴퓨팅 서비스에 대한 성능 향상을 이루고 있다. 이는 지리적으로 멀리 있는 클라우드 센터에 컨테이너를 설치하지 않고 사용자와 최대한 가까운 곳에 컨테이너를 적절하게 배치함으로써 가능하다 [1]. 컨테이너의 적절한 배치는 서비스의 성능과 직접 연관되므로 컨테이너 배치를 위한 다양한 알고리즘들이 개발되었다. 쿠버네티스는 자원의 사용량을 측정하고 다양한 알고리즘을 조합하여 최적의 배치를 실행할 수 있도록 지원한다 [2, 3].

쿠버네티스는 현재 CPU와 메모리, 볼륨의 자원 사용량을 고려하여 Pod을 배치한다. Pod은 쿠버네티스에서 제공하는 컨테이너 배치단위이다. 5G 시대가 되고 모바일 환경이 발전함에 따라 네트워크 속도도 중요한 자원 중 하나로 고려되어야 한다 [4, 5]. 그러나 쿠버네티스는 Pod의 배치를 위해 네트워크 자원은 고려하지 않고 있다. 그러므로 우선순위에서 네트워크 자원 사용량 값을 추가적으로 고려하고, 스케줄링 정책을 재정립하여 저지연 서비스를 제공할 수 있는 Krane 스케줄러를 개발하였다.

Ⅱ. 배 경

MEC는 사용자 기기가 중앙 데이터센터와 직접 소통하는 방식이 아닌 엣지 컴퓨팅 기술을 통해 기기와 가까이 위치한 노드에 서버를 배치하여 지연시간을 최소화 하도록 지원하는 서버 운용 기술이다. 서버를 노드에 배치하는 방법에는 가상화 기술이 사용된다. 그러나 사용자 기기와 가깝다고 무조건적으로 한 노드에만 서버를 배치하게 된다면 부하가 하나의 노드에 집중되거나 컴퓨팅 자원이 빨리 소진 되는등 다양한 문제가 생길 수 있다. 그래서 쿠버네티스는 이런 문제를 해결하기 위하여 스

*Corresponding Author (sgjin@daegu.ac.kr)
Received: Oct. 5, 2020, Revised: Nov. 16, 2020, Accepted: Dec. 4, 2020
T.H. Kim: Daegu University (B.S. Student)
T.Y. Kim: Daegu University (B.S. Student)
S. Jin: Daegu University (Assoc. Prof.)
※ 이 연구는 2016학년도 대구대학교 학술연구비 지원으로 수행되었음.

 2020 Dec. 15(6) 299-305
ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2020.15.6.299

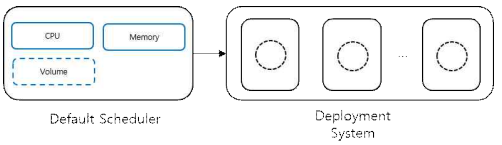

그림 1. 쿠버네티스 스케줄링을 위한 자원
Fig. 1 Resources for Kubernetes scheduler

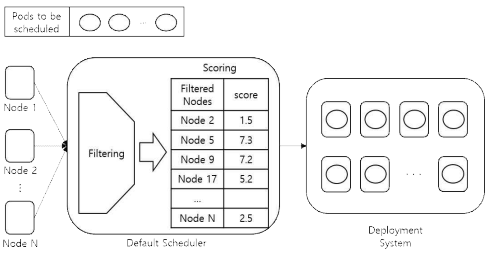

그림 2. 쿠버네티스 스케줄러 흐름도
Fig. 2 Procedure of Kubernetes scheduler

케줄러를 통해 안정적인 배치 정책을 설계하였다. 쿠버네티스 스케줄러는 그림 1과 같이 CPU와 메모리, 볼륨의 여유량 또는 상태를 고려하여 배치된다. 스케줄러의 전체적인 흐름도는 그림 2와 같다. 배치되어야 할 Pod들이 요청되면 스케줄러는 우선적으로 배치 불가능한 노드를 걸러낸다. 이를 필터링이라 한다. 이때 사용되는 알고리즘이 predicate이며, 목적은 스케줄링 프로세스에서 Pod의 모든 특정 요구사항을 충족시키는 노드만을 선별하기 위함이다. 필터링 과정을 처리하고 나면 선별된 노드들만 우선순위를 고려하면 되기 때문에 처리량 및 처리시간이 줄어들 수 있다.

Predicate 알고리즘들은 볼륨 및 디스크의 상태를 확인하거나 CPU와 메모리의 여유량이 Pod의 요구자원량을 충족하는지, Process ID (PID)의 남은 잔량은 여유로운지, master node와 worker node들 사이의 연결은 잘 되어있는지 등을 고려한다. 아울러, Pod의 배치를 위한 규격에서 nodeSelector필드에 레이블을 지정하면 레이블에 포함되지 않는 노드를 모두 제외할 수도 있다. 이와 반대로 tolerations 필드를 사용하여 지정한 레이블을 모두 제외할 수도 있다.

이처럼 predicate 알고리즘들은 Pod의 규격에서 지정된 레이블이나 CPU, 메모리, 볼륨 등을 주로 다루는 것을 알 수 있다. 그래서 쿠버네티스 스케줄러는 여유 자원이 모자라거나, 노드에 문제가 있는 경우를 제외하며 Pod을 배치하도록 되어있다. Predicates 알고리즘을 통해 사용 가능한 노드들을 일차적으로 선별하고 남은 노드 중 어디에 배치하는 것이 가장 좋을지를 결정해야 하는데 이를 해결하기 위해 필터링 되고 남은 노드들의 순위를 나눈다. 이를 스코어링이라 한다. 필터링은 배치 불가능한 노드들을 걸러내는 과정이라면, 스코어링은 가장 배치하기 좋은 노드를 선별하는 과정이라고 볼 수 있다. 이때 사용되는 알고리즘이 priority이다 [6].

$$S_{n,i} = \sum_{k=0}^{n} W_{i,k} \times f_{n,k}$$

- $S_{n,i}$ 서비스 프로파일에 대한 노드 n의 스케줄링 점수
- $W_{i,k}$ 서비스 프로파일에 대한 알고리즘 k의 weight factor
- $f_{n,k}$ 노드 n에 대한 알고리즘 k의 실행점수 (0~10)

Priority 알고리즘들은 노드의 우선순위 계산을 할 때 사용되며, 각 노드들의 자원의 균형을 위한 점수 산정 및 한 노드에 자원이 집중되지 않도록 하기 위한 알고리즘들로 구성되어 있다. 각 priority 알고리즘 점수는 (알고리즘 점수×weight)로 산정되며 배치 우선순위는 모든 가중 점수가 합산되어 가장 높은 순위부터 Pod을 배치한다.

Ⅲ. Krane 스케줄러

쿠버네티스 스케줄러는 네트워크 사용률을 전혀 고려하지 않고 스케줄링을 수행하므로 네트워크 대기 시간이 짧고 안정성이 요구되는 서비스를 충족하지 못할 수 있다. 예를 들어 기본 스케줄러에 따라 지연시간이 짧은 서비스를 배포하는 경우 네트워크 사용률에 관계없이 배포되므로 CPU와 메모리는 여유로우나 네트워크는 여유가 없는 노드에 배치할 수도 있다. 이와 같이 네트워크 요소를 고려하지 않고 Pod을 배치하게 된다면 Pod이 제공하는

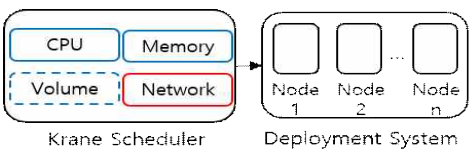

그림 3. Krane 스케줄러가 고려하는 자원
Fig. 3 Considering resources of Krane scheduler

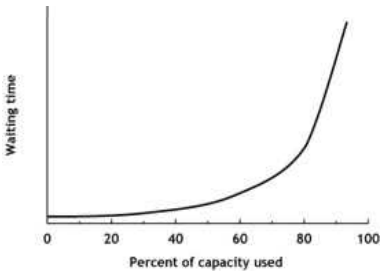

그림 4. Queueing system의 사용량에 따른 그래프
Fig. 4 Queuing system graph according to utilization

서비스를 사용하는 경우 네트워크 연결이 불안정하여 불편함을 느낄 수 있다. 이와 같은 문제점들을 해결하기 위해 그림 3과 같이 네트워크 자원을 추가적으로 고려하고 서비스 프로파일에 따른 그룹을 나누어 배치함으로써 네트워크 연결을 안정적으로 제공하는 스케줄러를 개발하여 문제를 해결하였다. Krane 스케줄러가 네트워크 요소를 고려하는 방법에 대한 내용은 다음과 같다.

1. 네트워크 자원 우선순위

위의 그림 1과 2를 보면 알 수 있듯이 기존 쿠버네티스의 스케줄러는 CPU, 메모리, 볼륨의 자원만을 고려하여 배치하고, 네트워크에 관련된 자원은 전혀 고려하지 않는다는 것을 알 수 있다. Krane 스케줄러는 네트워크 자원을 추가로 고려하여 노드를 결정하기 때문에 어플리케이션이 제공하는 지연시간 관리에 많은 도움을 줄 수 있다. Krane 스케줄러는 네트워크 자원을 추가로 고려하여 노드를 결정한다. 네트워크 시스템은 큐잉 시스템으로 볼 수 있는데 이는 일반적으로 사용량이 증가함에 따라 대기 시간이 함께 증가하는 특징이 있으며 그림 4와 같이 나타난다. 이에 따라 Krane 스케줄러에서의 네트워크 자원의 사용 원리는 다음과 같다. 각 노드는 네트워크 시스템의 특성을 반영하여 실측을 통해 네트워크 자원에 대해 특성 학습을 한다. 학습된 데이터를 토대로 네트워크 자원을 고려하여 필터링뿐만 아니라 자원사용량 및 지연시간에 따라 스코어링도 하여 Pod을 배치할 노드를 결정한다.

2. 서비스 프로파일 그룹화

참고문헌 [7]과 그림 5를 보면 알 수 있다시피 Pod의 특성이나 요구되는 서비스의 특성을 고려하지 않고, 단순 노드 상태에 따라 선별됨을 알수있

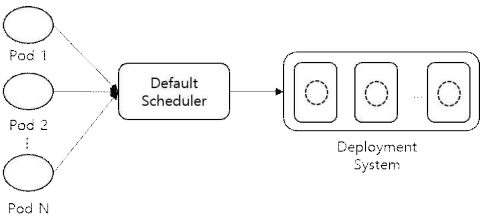

그림 5. 쿠버네티스의 스케줄링 정책
Fig. 5 Policy of Kubernetes scheduler

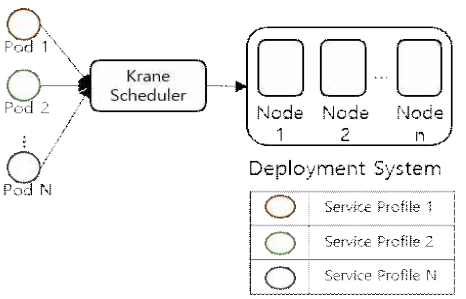

그림 6. Krane의 스케줄링 정책
Fig. 6 Policy of Krane scheduler

다. 하지만 프로그램을 운영 및 개발하는 입장에서는 배포할 프로그램에 따라 다양한 요구사항이 생길 수 있다. 이러한 문제점을 해결하기 위해 그림 6과 같이 Pod의 요구사항을 기입 가능하도록 서비스 프로파일을 Pod 배치규격에 추가하여 요구되는 서비스에 맞게 스케줄링되도록 하였다. Pod1부터 PodN까지의 서비스 프로파일이 각기 다른 서비스가 요청되면 서비스 프로파일마다 요구되는 지연시간이 다를 것이므로, 요구 지연시간을 그룹별로 나누어 각 그룹마다 지정된 지연시간을 넘지 않도록 네트워크 사용량을 고려하여 배치하도록 하였다.

저지연이 요구되는 서비스가 그룹화되지 않고 배치될 경우 서비스를 이용하는 사용자가 네트워크 연결을 안정적으로 제공받지 못할 수 있으므로 지연시간에 대해 불만을 가질 수 있다. 이를 위해 Krane 스케줄러는 네트워크의 특성을 학습하여 획득한 특성함수를 바탕으로 사용량을 조정함으로써 특정 서비스 프로파일을 만족시켜야 하는 최대 지연시간을 기준으로 그룹화하기 때문에 지연시간 및 네트워크 자원을 안정적으로 제공받을 수 있다. 네트워크를 안정적으로 제공하기 위해 서비스 프로파일의 표준 및 서비스 프로파일에 따른 노드 그룹을 나누는 방법은 다음과 같다.

표 1. 서비스 프로파일 요구사항 표준
Table 1. Service profile requirements standard

	CPU	Memory	Bandwidth	Latency
Cloud AR/VR	4 core	8GB	4k: 20~40Mbps 8k: 90~130Mbps 12k: 500~700Mbps	4k: <=50ms 8k: <= 20ms 12k: <= 10ms
Web Server	1 core	2GB	1.5Mbps	1~2 sec
Video Stream	1 core	4GB	FHD : 6 Mbps 4k : 35 Mbps	4~5 sec (w/ buffering)
Cloud Game	4 core	4GB	30Mbps	20~100ms
Smart Factory	6 Core	2GB	1~10Mbps	1ms

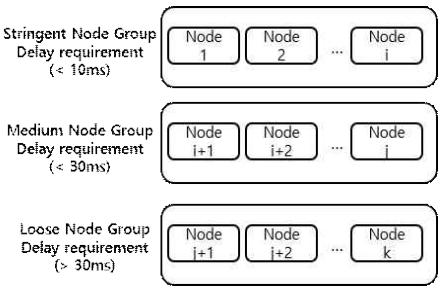

그림 7. Krane 스케줄러의 서비스 프로파일을 고려한 정책
Fig. 7 Policy of considering the service profile on Krane scheduler

서비스 프로파일은 크게 5가지로 분류하였다. 각 서비스 프로파일은 CPU와 메모리, 대역폭, 지연시간을 고려하여 표준화하였다. 표 1의 서비스 프로파일 요구사항 표준은 [8-10]을 참고하여 작성되었고, 위 표준을 참조하여 Pod을 배치할 노드 그룹을 지연시간에 따라 초저지연, 저지연, 저지연이 요구되지 않는 3가지 그룹으로 분류하였다. 노드 그룹을 나누는 것은 각 노드별로 요구되는 지연시간에 따라 안정적으로 네트워크 서비스를 제공하기 위함이다.

노드가 지연시간에 관한 요구를 만족하도록 하기 위해서 각 노드에서 발생하는 지연시간과 네트워크 사용량의 상관관계를 실측을 통해 수집하여 학습하도록 하였다. 그리고 수집된 지연 정보를 이용하여 지연에 대한 네트워크 사용량 임계값과의 관계를 찾아내고 이 관계를 바탕으로 노드 그룹들을 위한 네트워크 사용량 임계값을 설정하였다. 그림 7은 stringent, Medium, Loose 노드 그룹 세 가지로 나누어진 노드 그룹을 나타내고 있다. Stringent 노드 그룹은 지연시간이 초저지연의 서비스가 운영되어야 하는 그룹으로 Smart Factory,

$$f_H = 10 \times ((L_n - L(U_n + U_p)) / L_n)$$
$$f_L = 10 \times (1 - (L_n - L(U_n + U_p)) / L_n)$$

표 2. 지표 점수 계산 방법
Table 2. Method of calculating score

f_H	Algorithm score for the Stringent node group
f_L	Algorithm score for the Loose node group
L_n	Latency when network utilization on node n is a limit
U_n	Current network utilization on specific nodes n
U_p	Requiring network utilization of specific Pod p
$L(x)$	Latency function according to specific network utilization value x

Cloud AR/VR 서비스 프로파일이 포함된다. Medium 노드 그룹은 지연시간이 저지연인 서비스를 지원하는 그룹으로 Cloud Game 및 AR/VR이 포함된다. 그 외에 저지연이 요구되지 않는 서비스의 경우는 모두 Loose 노드 그룹으로 지정된다.

Stringent 노드 그룹은 초저지연을 유지해야 하므로 네트워크 자원 사용량이 낮은 노드를 우선적으로 선정하여 Pod을 배치하여야 한다. Stringent 노드 그룹의 알고리즘 실행점수는 표 2에 따라 노드 n의 네트워크 사용량 한계치의 지연시간 (Ln)과 현재 네트워크 사용량 (Un)을 더하고 새로 배치될 Pod의 요구 네트워크 사용량 (Up)을 뺀 뒤 Ln으로 나누어 네트워크 자원 여유량을 계산한다. Medium 및 loose 노드 그룹은 네트워크 자원의 사용량이

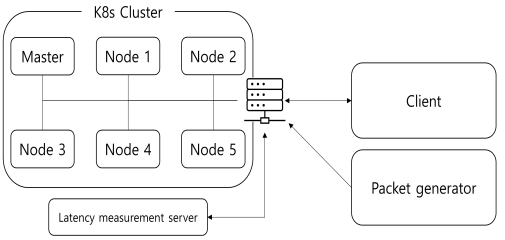

그림 8. 실험 환경 구성도
Fig. 8 Configuration of experiment environment

높은 노드에 우선적으로 배치하기 위해 Stringent 노드 그룹 알고리즘 실행점수와는 반대로 한 노드의 자원을 최대한 사용하도록 개발하였다.

Ⅳ. 성능 평가

본 실험의 구성도는 그림 8과 같이 구성되어 있다. 먼저 실험환경은 클러스터를 관리하는 master 노드 1대와 worker 노드 5대로 구성되어 있다. Worker 노드 1번과 2번은 노드 그룹 중 Stringent 노드 그룹에 해당하고 worker 노드 2, 3번은 Medium, 4, 5번은 Loose 노드 그룹에 해당한다. Latency measurement server는 클러스터에서 측정서버까지의 지연율을 측정하여 측정한 결과값을 각 노드의 node exporter에 전달하여 측정값을 시각화한다. 시나리오는 다음과 같다.

먼저 쿠버네티스의 기본 스케줄러만을 이용한다. 저지연을 만족해야하는 Stringent 노드 그룹은 자원 임계점을 대역폭 20%, CPU 사용량 40%로 하였으며, Loose 노드 그룹은 대역폭 70%, CPU 사용량 10%으로 하였다.

그림 9를 보면 쿠버네티스 스케줄러로 배치된 웹서버는 10ms로 서비스가 제공되고 있으며 지연시간이 자주 큰 폭으로 튀는 것을 확인할 수 있다. 그림 10 게임 서버는 3ms로 서비스가 제공되고 있으며 웹서버에 비해 좀 더 빈번하게 큰 폭으로 튀는 것을 확인할 수 있다. 쿠버네티스 스케줄러는 네트워크 요소를 고려하지 않으므로 CPU와 메모리, 볼륨 자원이 여유롭다면 어디든지 배치를 하기 때문에 특정 지연시간이 요구되는 서비스에 대해서는 안정적으로 제공한다고 보기 힘들다.

다음은 네트워크 요소를 추가적으로 고려하여 배치하는 Krane 스케줄러를 사용하여 지연시간을 측정하였다. 기본 스케줄러와는 다르게 Krane 스케줄러는 스코어링 단계에서 네트워크 사용량도 고려

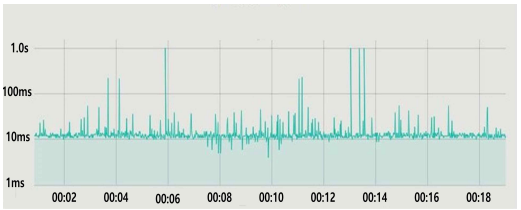

그림 9. 쿠버네티스 스케줄러로 배치된 웹서버 Pod의 지연시간
Fig. 9 Allocated web server Pod latency by Kubernetes scheduler

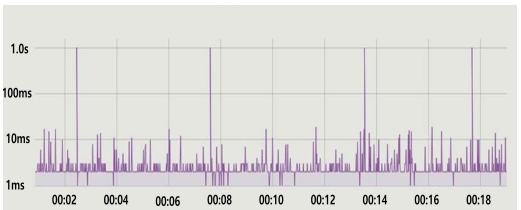

그림 10. 쿠버네티스 스케줄러로 배치된 게임 서버 Pod의 지연시간
Fig. 10 Allocated game server Pod latency by Kubernetes scheduler

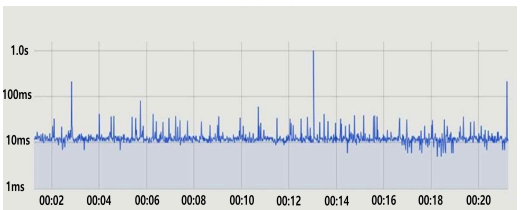

그림 11. Krane 스케줄러로 배치된 웹서버 Pod의 지연시간
Fig. 11 Allocated web server Pod latency by Krane scheduler

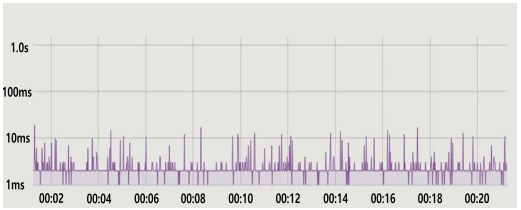

그림 12. Krane 스케줄러로 배치된 게임 서버 Pod의 지연시간
Fig. 12 Allocated game server Pod latency by Krane scheduler

표 3. 각 스케줄러의 평균 지연시간
Table 3. Average latency for each scheduler

Service / Scheduler	Web Server	Game Server
Default Scheduler	78.74ms	15.66ms
Krane Scheduler	34.77ms	5.32ms

하며 필터링 단계에서는 서비스 프로파일 별로 노드 그룹을 나누어 배치한다.

Krane 스케줄러로 배치된 웹서버는 10ms, 게임 서버는 3ms로 서비스가 제공되고 있으며 쿠버네티스 기본 스케줄러와는 다르게 네트워크 요소를 추가적으로 고려하여 배치하였으므로 보다 안정적으로 네트워크 서비스를 제공하는 것을 확인할 수 있다. 그림 11과 그림 12를 보면 쿠버네티스 스케줄러에 비해 튀는 빈도가 크게 줄어들었으며 네트워크 끊김 없이 서비스가 안정적으로 제공되는 것을 확인할 수 있다. 기본 스케줄러와 Krane 스케줄러의 성능 차이는 표 3을 통해 명확히 확인 가능하다.

V. 결 론

앞으로도 많은 노드를 관리하는 방법에 대한 연구는 계속될 것이고 스케줄러는 효율적 배치 및 관리가 가능하도록 하는 중요한 열쇠가 될 것이다. 현재 쿠버네티스의 스케줄러는 CPU와 메모리, 볼륨만을 고려하지만 Krane 스케줄러를 통해 네트워크를 추가 고려하고, 노드를 묶음으로 나누는 정책을 통해 안정적인 서비스를 제공함으로써 각 서비스 프로파일에 따른 요구 지연시간을 만족시키며 가상화된 컴퓨터를 배치할 수 있도록 하였다. 또한, 우리는 네트워크를 고려한 스케줄링 방법을 쿠버네티스를 확장하여 구현하였고 실험을 통하여 검증할 수 있었다. 성능평가를 통해 필터링 과정에 노드 그룹을 분류하고, 스코어링에 네트워크 자원을 추가 고려하여 배치된 서비스를 안정적으로 제공하는 것을 확인하였고 앞으로 이 기술을 토대로 GPU와 같은 여러 컴퓨팅 자원을 추가하여 서비스를 적절하게 배치하기 위한 판단 요소로 지정하거나, 노드의 특성을 고려하여 자동으로 서비스 프로파일에 맞게 배치하는 등 다양하게 발전 및 응용도 가능할 것으로 보인다.

References

[1] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," in IEEE Cloud Computing, Vol. 1, No. 3, pp. 81-84, 2014.

[2] M. Chiang, T. Zhang, "Fog and IoT: An Overview of Research Opportunities," in IEEE Internet of Things Journal, Vol. 3, No. 6, pp. 854-864, 2016.

[3] T.H. Kim, et al. "Virtualization and Kubernetes," OSIA Standards & Technology Review, Vol. 33, No. 2, pp. 4-10, 2020 (in Korean).

[4] R. Pérez de Prado, S. García-Galán, J.E. Muñoz-Expósito, A. Marchewka, N. Ruiz-Reyes, "Smart Containers Schedulers for Microservices Provision in Cloud-Fog-IoT Networks. Challenges and Opportunities," Sensors, Vol. 20, No. 6, pp. 1714, 2020.

[5] Y.C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young, “Mobile edge computing－A key technology towards 5G,” ETSI white paper, Vol. 11, No. 11, pp. 1-16, 2015.

[6] J. Santos, T. Wauters, B. Volckaert and F. De Turck, "Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing Applications," 2019 IEEE Conference on Network Softwarization (NetSoft), pp. 351-359, 2019.

[7] T.Y. Kim, J.R. Lee, T.H. Kim, I.G., Chun, J. Park, S. Jin, “Kubernetes Scheduler Framework Implementation with Realtime Resource Monitoring,” IEMEK J. Embed. Sys. Appl., Vol. 15, No. 3, pp. 129 - 137, 2020 (in Korean).

[8] D. Santoro, D. Zozin, D. Pizzolli, F. De Pellegrini, S. Cretti, "Foggy: A Platform for Workload Orchestration in a Fog Computing Environment," 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 231-234, 2017.

[9] S. Sarkar, S. Chatterjee, S. Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things," in IEEE Transactions on Cloud

Computing, Vol. 6, No. 1, pp. 46-59, 2018.
[10] M. Chima Ogbuachi, C. Gore, A. Reale, P. Suskovics, B. Kovács, "Context-Aware K8S Scheduler for Real Time Distributed 5G Edge Computing Applications," 2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1-6, 2019.

Tae-Hyun Kim (김 태 현)

He is currently in the course of B.S. degree at Daegu University, Daegu, Korea.

Email: tolriaz@daegu.ac.kr

Tae-Young Kim (김 태 영)

He is currently in the course of B.S. degree at Daegu University, Daegu, Korea.

Email: tkken12@gmail.com

Sunggeun Jin (진 성 근)

He received B.S. and M.S. degrees from Kyungpook National University, Daegu, Korea, in 1996 and 1998, respectively. Also, he received Ph. D. degree from Seoul National University, Seoul, Korea, in 2008.

In 1998, he joined the Electronics and Telecommunications Research Institute, Daejeon, Korea.

He is now an associate professor with Daegu University, Gyeongsan, Korea.

Email: sgjin@daegu.ac.kr