NET 챌린지 캠프 시즌7 챌린지리그 최종결과보고서

과제제목	엣지 클라우드 환경에서 초고신뢰/초저지연 컴퓨팅(URLLC) 서비스를 위한 Network-Aware Placement System
팀 명	MJU IDPL

본 보고서를 NET 챌린지 캠프 시즌7 챌린지리그(학생팀)의 최종결과보고서로 제출합니다.

2020년 11월 13일

지 도 교 수 : 안희철 (명지대학교 컴퓨터공학과)

참여 구성원 : 엄상현 (명지대학교 컴퓨터공학과 학사과정)

노지환 (명지대학교 컴퓨터공학과 학사과정)

과학기술정보통신부장관 귀하

< 안 내 사 항 >

1. 본 연구보고서는 한국정보화진흥원의 출연금으로 수행한 NET 챌린지 캠프 시즌7 챌린지리그(학생팀)의 연구망 활용 연구과제 결과입니다.
2. 본 연구보고서의 내용을 발표할 때에는 반드시 한국정보화진흥원 NET 챌린지 캠프 시즌7 챌린지리그(학생팀)의 연구망 활용 연구과제 결과임을 밝혀야 합니다.

요 약 문

1. 제 목

엣지 클라우드 환경에서 초고신뢰/초저지연 컴퓨팅(URLLC) 서비스를 위한 Network-Aware Placement System

2. 아이디어 개발의 목적 및 필요성

5G 상용화와 IoT에 의한 통신량 증가를 극복하기 위해 엣지 컴퓨팅기술이 떠올랐지만, 차세대 네트워크에 맞는 스케줄링 정책의 미흡할 뿐만 아니라 엣지 컴퓨팅을 효율적으로 활용하기 위한 작업할당이 이루어지지 않았다. 때문에 본 과제에서는 컨테이너 오케스트레이션 시스템인 쿠버네티스가 Edge cloud에 환경에서 URLLC (Ultra-Reliable Low Latency Communication)서비스를 배치하기에 적합한 스케줄러를 개발하는 것을 목표로 한다.

3. 아이디어 개발의 내용 및 범위

본 과제에서는 쿠버네티스의 기본 컴포넌트인 스케줄러를 통하여 네트워크 환경의 두 가지 요소인 RTT(Round-Trip Time)와 대역폭(Bandwidth)을 고려하여 URLLC서비스에 적합한 배치를 하도록 하였고, 이에 따라 네트워크 정보를 모니터링하는 응용프로그램과 성능검증을 위한 API 기반 서비스를 개발하였다.

요 약 문

4. 개 발 결 과

본 과제에서는 클러스터 서버 기반 가상 엣지 클라우드, KOREN SDI 기반 엣지 클라우드와 이에 따른 네트워크 정보를 모니터링 할 수 있는 API 기반의 툴 그리고 최적의 네트워크 엣지 노드를 선정할 수 있는 스케줄러를 개발하였다. 개발한 결과물 중, NAS와 HRDF 스케줄러를 통하여 기본적인 스케줄링 방식보다 평균적으로 30% 이상의 유의미한 지연율 감소 효과를 보았다.

5. 기 대 효 과

본 과제에서 제안한 기술을 통해 VR/AR 그리고 자율주행 부분에서의 필요 latency를 충족시킬 것으로 기대할 수 있고, 엣지 클라우드를 위한 솔루션이나 시스템 프레임워크의 부족을 해결하여 기술의 접근성을 쉽게 하여 5G와 스마트 시티(Smart City) 그리고 기업들의 배포환경에서도 유용하게 사용할 수 있도록 만들어 줄 뿐만 아니라 URLLC 서비스의 성능개선을 기대할 수 있다. Future Work로는, Channel estimation 데이터를 통하여 머신러닝을 도입해 NAS의 노드선정 과정에서 개선된 선택이 가능하게 하고, Application Aware를 통한 스케줄링 과정에서의 시간을 줄이는 방안을 통하여 기술에 대한 발전 가능성 또한 기대된다.

목 차

그림 목차

표 목차

I 서 론

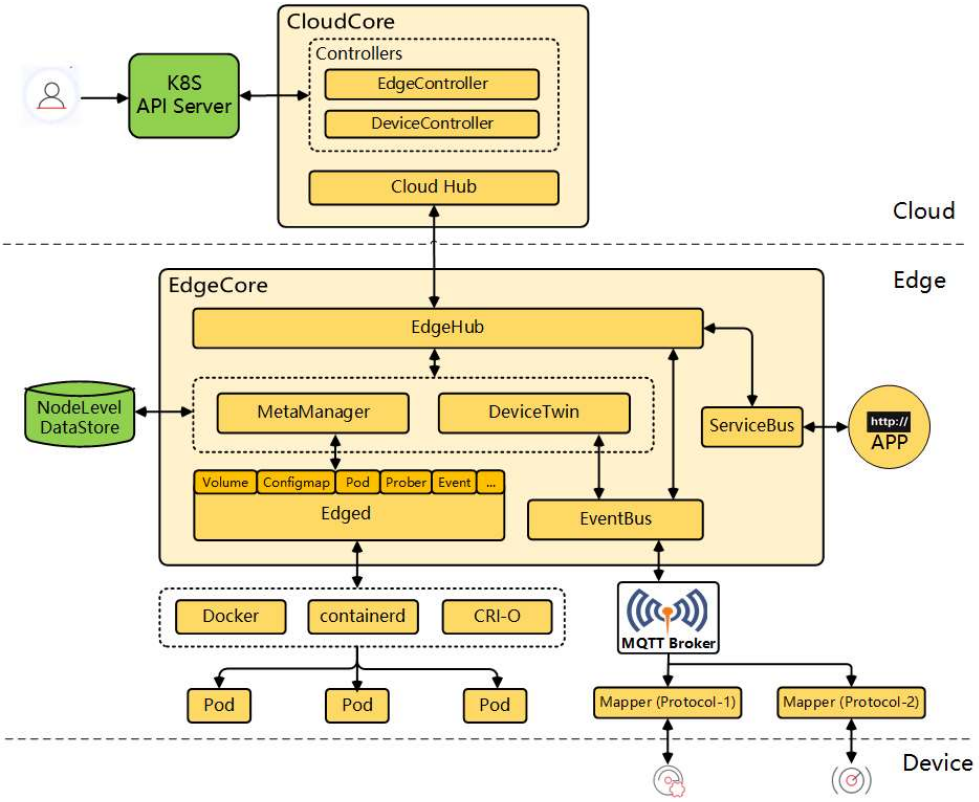
1. 개발 목적 및 필요성

5G의 상용화와 IoT에 의한 통신량 증가를 극복하기 위한 엣지 컴퓨팅[1]이라는 새로운 컴퓨팅 패러다임이 등장했다. 대부분의 엣지 클라우드는 많은 범위의 응용프로그램을 다루지 않는 특성이 있다. 때문에 엣지 컴퓨팅이 필요하지 않은 작업에 대해서는 일반적인 클라우드에서 처리를 요청하는 것이 엣지 컴퓨팅의 가치를 저해하지 않는 방법이다. 때문에 엣지 클라우드에서는 단말에서 요청한 작업을 효율적으로 처리하기 위한 작업 할당은 중요한 문제 중 하나이다. 하지만 아직 대부분의 클라우드 플랫폼들이 엣지 컴퓨팅을 위한 기능을 개발단계에 있다. 마이크로서비스 기반 애플리케이션을 효율적으로 관리하기 위해서 개발된 오케스트레이션 툴 중 사실상의 표준 (*De facto*)인 쿠버네티스[2]는 서비스 스케줄링을 위한 많은 정책과 Topology-Awareness를 고려한 특성들을 가지고 있다. 하지만 이러한 정책들은 대부분 호스트의 자원상황(CPU, RAM)을 고려한다. 따라서 쿠버네티스는 디바이스와 컴퓨팅 머신 사이의 네트워크의 속도가 중요한 URLLC [3](Ultra-Reliable Low Latency Communication)을 기반으로 하는 IoT 서비스를 다루기에는 한계가 있다. [4]

최근 쿠버네티스를 기반으로 하는 엣지 클라우드 플랫폼인 쿠브엣지(KubeEdge)가 CNCF (Cloud Native Computing Foundation)의 인큐베이팅 프로젝트로 채택되면서[5] 엣지 클라우드 환경에서 컨테이너 가상화 기반으로 서비스를 개발 및 배포하는 것이 차세대 엣지 클라우드의 동향이다. 이러한 서비스 환경에서 네트워크를 고려한 서비스 배치는 필수적이다. 따라서 본 과제는 쿠버네티스 플랫폼을 기반으로 하는 엣지 클라우드 환경에서 쿠버네티스 환경을 바탕으로 적합한 자원 할당 결정 및 서비스 배치시스템을 구현하고 KOREN 네트워크를 통해 그 성능을 평가하는 것을 목적으로 한다.

2. 특징(동일・유사 아이디어에 대한 차별성, 독창성, 혁신성)

본 과제와 유사한 기술을 목표로 하는 시스템으로는 현재 쿠버네티스의 확장프로그램으로 개발 진행 중인 KubeEdge[6]가 있다. KubeEdge는 엣지 클라우드 환경을 대상으로 컨테이너 오케스트레이션과 에지-노드를 관리할 수 있는 기능을 가진다. 주요한 시스템 구조는 그림 5과 같다.



[그림 1 KubeEdge의 시스템 구조]

그림 1를 통하여 알 수 있듯이 KubeEdge는 쿠버네티스 API를 통해 사용자와 상호작용하는 것을 알 수 있다. 이는 기존에 쿠버네티스가 가지고 있는 시스템의 특성과 정책을 사용한다고 할 수 있다. 하지만 앞에서 언급한 바와 같이 쿠버네티스의 기본 스케줄러는 노드의 자원상황을 먼저 고려하기 때문에 적합하지 않은 스케줄링 방식을 가지고 있다.

실제로 KubeEdge Roadmap[7]에서도 엣지 클라우드에서의 지능적 스케줄링을 미래의 과제로 지정하고 있다. 본 과제는 엣지 클라우드 시스템의 성능을 더욱 개선할 수 있을 것으로 생각된다. 엣지 클라우드에 적합한

작업 디스패칭 (dispatching)과 스케줄링은 많은 연구가 이루어져 왔다. 하지만 실제 상용 네트워크 환경에서 적용할 수 있는 플랫폼 기반 시스템은 아직 개발되지 않았다. 본 과제는 상용 환경에서 운용 가능한 시스템을 개발하는 것을 목표로 한다.

Ⅱ 아이디어 개발 추진 내용과 범위

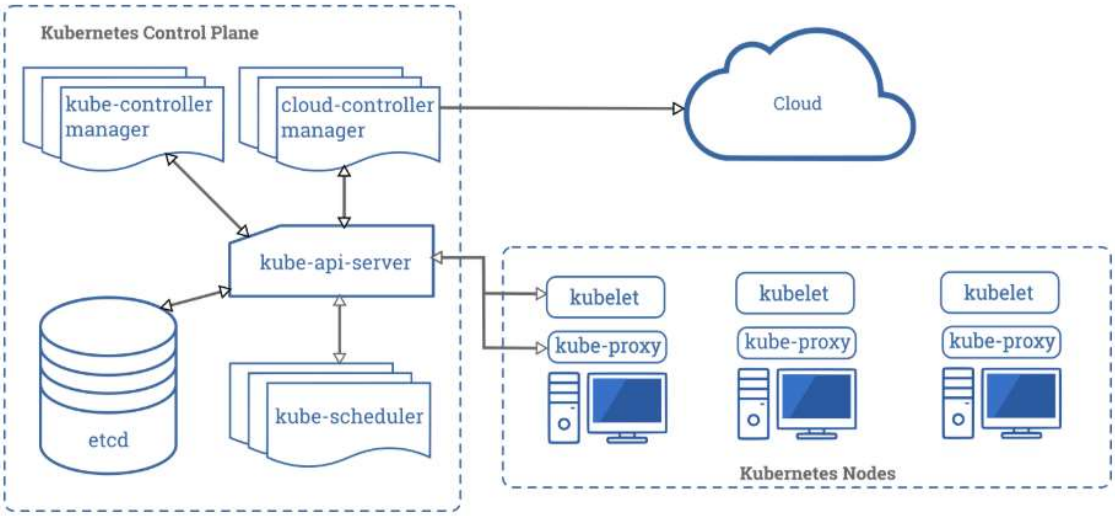
1. 개발 목표 및 추진 내용

가. 개발 목표

본 과제는 쿠버네티스를 구성하고 있는 기본 컴포넌트 중에서 스케줄러를 개선하여 Network-Aware Service Placement를 위한 Network-Aware Scheduler (NAS)를 개발함과 동시에 개선된 스케줄링 정책인 HRDF (Highest Residual Density First)를 구현하는 것을 목표로 한다.

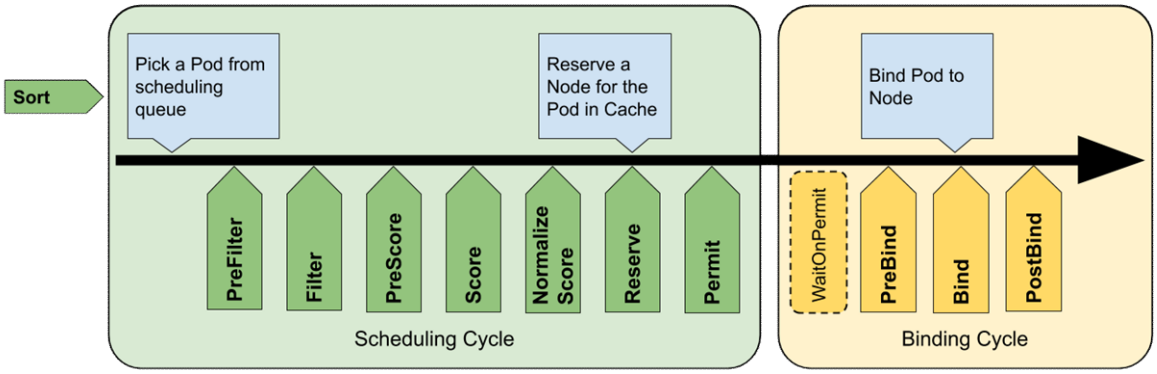
나. 세부 추진 내용 및 범위

본 과제의 개발은 쿠버네티스의 시스템을 기반으로 한다.
기본적인 쿠버네티스 시스템 구조[8]는 그림 2과 같다.



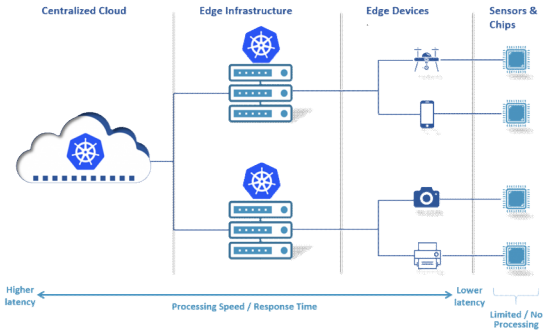
[그림 2 쿠버네티스 시스템 구조]

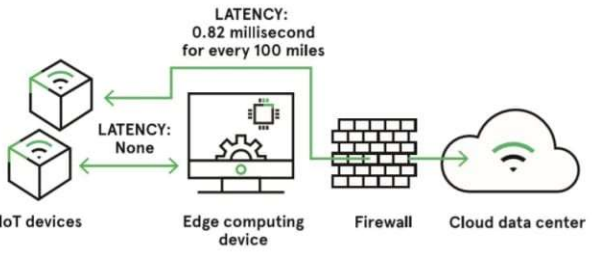
쿠버네티스 시스템을 구성하고 있는 기본 컴포넌트 중에서 스케줄러(Kube-scheduler)[9]는 크게 필터링 단계와 스코어링 단계를 거쳐 작업에 가장 적합한 노드를 찾게 된다. 쿠버네티스는 스케줄러의 사용자화를 위해서 각 스케줄링의 단계마다 기능을 확장할 수 있는 포인트를 제공하는 이는 그림 3과 같다.



[그림 3 Service Scheduling Context]

쿠버네티스 시스템을 기반으로 개발한 Network-Aware Service Placement System은 IoT 단말와 엣지 클라우드 사이의 RTT (Round-Trip Time)과 대역폭(Bandwidth)을 고려하여 URLLC 서비스에 적합한 배치를 가능하게 한다. 이를 통해 URLLC 서비스에 대해서 지연시간이 거의 존재하지 않는 최적의 성능을 낼 수 있을 것으로 예상한다. 예상하는 네트워크 구조와 그에 대한 성능은 그림 4, 5과 같다.





[그림 4 예상 결과물 네트워크 구조] [그림 5 예상 결과물 시스템 성능]

다. 참여구성원의 역할

성 명	역 할
엄상현	시스템 개발 총괄 및 실험
노지환	시스템 개발 보조 및 실험 보조

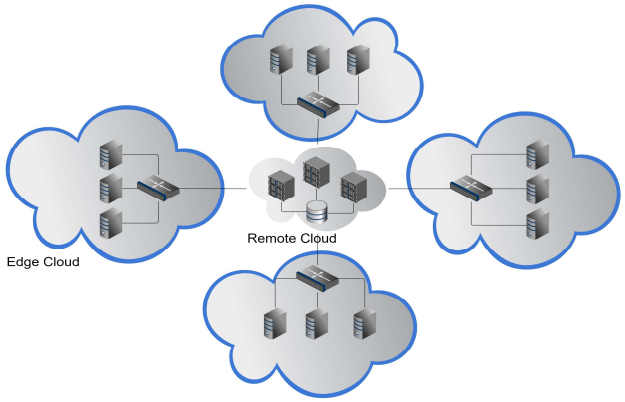
[표 1 참여구성원의 역할]

엄상현 팀장은 NAS(Network Aware Scheduler)의 아이디어 제공, HRDF(Highest Residual Density First)의 개발 방향 및 시스템 개발을 총괄하였다. 이러한 스케줄러의 알고리즘을 구현하고, 스케줄러에 필요한 API의 방향성을 제시하였으며, 본 과제의 총체적인 기획을 담당했다. 또한, UNIST 시스템 소프트웨어 연구실에서 Whale이라는 클러스터를 구축하여, KOREN망에서 쓰일 스케줄러의 연구와 성능 측정 그리고 성능 검증 실험을 설계하였다.

노지환 팀원은 검증을 위해 필요한 Whale 클러스터에 Docker, Kubernetes를 이용하여 API가 돌아갈 수 있는 환경을 구성하였고, 클러스터에 API를 띄워 주기적으로 네트워크 상태에 대하여 수집 및 기록하였다. 총체적인 실험과 검증을 바탕으로, 발표할 자료와 수치들을 수집하고, 검증하였으며 HRDF(Highest Residual Density First)의 기본적인 개념을 정리하였다. 또한, 스케줄러들의 성능검증을 위한 웹서버를 만들어 시험검증을 진행하였다.

2. 단계별 세부 구현 계획

아이디어 구현 및 추진 계획은 하드웨어 부분과 소프트웨어 부분으로 나눌 수 있다. 첫 번째로 하드웨어 부분에서 소프트웨어 개발을 위한 클러스터 기반 가상 엣지 클라우드를 구축하는 것을 목표로 한다. 클러스터에서 Traffic Control 툴을 이용하여 RTT와 대역폭을 제어한다. 이를 이용하여 각각의 노드의 물리적 거리를 개념적으로 구현한다. 이를 통해 엣지 클라우드와 같은 환경을 구성하여 실험을 진행한다. 구체적인 실험환경의 구조는 그림 6과 같다.



[그림 6 가상 엣지 클라우드 환경 네트워크 구조]

시스템 성능을 검증하는 단계에서 사용하는 실제 엣지 클라우드의 구조는 그림 7과 같다. 실험을 위한 테스트베드는 KOREN SDI[10]를 이용하여 3개의 엣지 노드와 9개의 가상 엣지 노드를 이용하여 구성한다. 가상 엣지 노드는 '코로나19'로 인하여 물리적인 노드 배치가 힘들어지면서 고안해낸 방식이다. 물리적인 거리를 통하여 발생하는 rtt와 대역폭 등의 네트워크 환경을 논리적인 방식을 통하여 실제로 물리적 노드와 유사한 환경을 만들어 구성하였다. URLLC 서비스에서 time-critical한 응용을 수행하기 위해서 가장 중요하게 고려되어야 할 요소를 정확하게 시뮬레이션 할 수 있었다. 각 엣지 노드는 네트워크뿐만 아니라 자원상황을 시뮬레이션 하기 위해서 구글에서 제공하고 있는 클러스터 워크로드의 일부를 수행하는 환경을 구성한다.

[그림 7 시스템 성능검증을 위한 엣지 클라우드]

소프트웨어 부분에서 구현할 프로그램은 크게 세 가지로 구분할 수 있다.

먼저 서비스 배치를 직접 결정하는 Network-Aware Scheduler (NAS) 는 쿠버네티스 스케줄러를 개선하여 개발을 진행할 예정이다. 제안한 스케줄러는 그림 3에서 나타나는 스케줄링 단계 중 필터링(filtering) 단계를 확장한다. 필터링 단계는 조건에 부합하는 엣지 클라우드를 선정하는 과정이다. 구현 시스템은 필터링 단계에서 네트워크 환경의 두 가지 요소를 고려하게 된다. 첫 번째 요소는 RTT (Round-Trip Time)이다. 단말과 엣지 클라우드 사이의 RTT는 time-critical 시나리오에서 low-latency 서비스가 가장 중요하게 고려해야 하는 요소이다. RTT가 가장 작은 엣지 클라우드는 URLLC 서비스를 위한 최적의 노드라고 할 수 있다.[11]

두 번째 요소인 대역폭은 최적의 노드 후보가 데이터 전송에 적합한지 판단하는 기준이다. URLLC 서비스를 위해 사용자가 요청한 대역폭 이상을 지원할 수 있는지 판단한다. 최종적으로 데이터 전송에 가장 적합한 노드를 선정할 수 있다. 세부적인 알고리즘은 그림8 과 같다.

Algorithm 1 Network Aware Scheduler (NAS)

```
1: Input: Remaining Node for Filtering Process
2: Output : Node for the service placement
3: NAS_handler(http.request){
   receivedNodes = decode(http.Request);
4: receivedPod = decodePod(http.Request);
5: node = selectNode(receivedNodes, receivedPod);
6: return node
7: }
3: // Return the best candidate Node (recursive)
4: function selectNode(Nodes, Service){
5:   targetLocation = getLocation(Service);
6:   minBandwidth = getBandwidth(Service);
7:   min = math.MaxFloat64;
8:   copy = receivedNodes;
9:   // find min RTT
10:  for for node in range Nodes {
11:      rtt = getRTT(node, targetLocation);
12:      min = math.Min(min,rtt)
13:     }
14:// find best Node based on RTT and minBandwidth
15:  for node in range Nodes{
16:    if min == getRTT(node, targetLocation){
17:      if minBandwidth <= getAvgBandwidth(node){
18:          return node;
         }
19:     else
20:          copy = removeNode(copy,node)
21:   }
22:    if copy == null
23:          return null, Error("No suitable nodes found");
24:     else
25:        return selectNode(copy, Service);
26: }
```

[그림 8 Network Aware Scheduler 알고리즘]

본 과제에서는 최종적으로 스케줄러가 네트워크뿐만 아니라 응용프로그램 간의 자원 경쟁(Resource Contention)을 극복하기 위한 스케줄링 정책인 HRDF(Highest Residual Density First)[12]를 제안 시스템에 추가함으로써 더욱 완성도 높은 시스템을 개발한다. HRDF는 업로드/다운로드 지연시간 (Upload/Download Delay), 처리시간 (Processing Time), 가중치를 고려하여 가장 적합한 노드를 찾는 방법이다. 이러한 스케줄링 가능한 이유는 엣지 클라우드는 처리시간을 예상할 수 있는 특정한 워크로드를 수행하기 위한 시스템이기 때문이다. 결과적으로 처리시간이 서비스 속도의 가장 큰 원인일 경우 통신시간(Communication Time)과 처리시간(Processing Time)을 트레이드 오프(Trade off)하여 전체 단말 간 작업 완료시간(End to End Job Completion Time)을 단축할 수 있다. 자원 사용량에 대한 정보는 쿠버네티스 API를 사용하여 얻을 수 있다.

HRDF는 다음 식을 이용하여 최적의 노드를 찾을 수 있다. $Q_{k'j}$ 구함으로써 가장 적합한 노드를 찾게 된다.

$$\text{server } k = \text{argmin}_{k'} Q^{\text{selfish}}_{k'j}(t)$$

$$Q^{\text{selfish}}_{kj}(t) = \frac{1}{1+\varepsilon}\left\{ w_{kj} \sum_{(j',t')\in \mathcal{A}^1_{kj}(t)} p_{j'}(t') + w_{kj}(p_{kj} + \Delta^{\uparrow}_{kj} + \Delta^{\downarrow}_{kj}) \right\}.$$

w_{kj}는 각 서버가 가진 연산능력에 대한 가중치이며 중괄호 안의 첫 번째 항은 Job의 대기시간이다. 작업 j가 스케줄링 되기 전에 이미 서버에 스케줄링 되어있는 작업의 수행시간을 의미한다. 덧셈의 두 번째 항은 스케줄링 될 작업의 j의 작업시간이다. 이 두 값의 합에 w_{kj}를 곱한 이후에 $\frac{1}{1+\varepsilon}$을 취함으로써 작업 j 에 대해서 각 서버가 가지고 있는 처리능력을 계산할 수 있다.p_{jk}는 j 의 처리시간, $\triangle_{k}^{\uparrow}{}_{j}$는 업로드 지연시간, $\triangle_{k}^{\downarrow}{}_{j}$는 다운로드 지연시간을 의미한다. $Q_{k'j}$의 값이 가장 작은 엣지 노드가 작업을 가장 빨리 끝낼 수 있는 노드이기 때문에 계산 집중적인 작업에 대해서 응답시간을 줄일 수 있다.

계산 집중적 응용에 대한 정보는 프로파일링된 정보를 이용하고 각 노드의 자원 사용현황과 스케줄링 정보는 쿠버네티스 API를 이용하여 구현할 수 있다.

기본적인 스케줄링에 대한 설명은 다음과 같다. 작업을 스케줄링하기 위해서 먼저 2가지 작업으로 분류한다. 작업은 Nearest와 Selfish로 분류하게 된다. Nearest는 논리적으로 가장 가까운 위치에 있는 엣지 클라우드에 배치되어야 하는 작업이다. 이는 NAS를 이용하여 최적의 노드를 찾을 수 있다. 두 번째로 분류되는 작업으로 Selfish가 있다. Selfish는 실제 작업 워크로드에 가까운 작업이라고 볼 수 있다. 이러한 작업은 자기 자신의 실행이나 성능만 신경 쓴다. Selfish 작업을 다루기 위해서 HRDF 스케줄링 정책을 사용하게 된다.

또 다른 구현 소프트웨어로는 스케줄러에 전송해야 할 네트워크 정보를 모니터링하는 응용프로그램이 있으며 파이썬(python)을 이용하여 구현할 예정이다. psutil, pyping 모듈을 이용하여 네트워크 정보를 수집한 후 Flask 프레임워크[13]를 이용하여 API 통신을 통해 정보를 전송하는 기능을 구현한다.

마지막 구현 소프트웨어는 성능검증을 위한 응용프로그램이다.
결과물에 대한 성능 측정은 기존 쿠버네티스 시스템과 제안한 시스템의 실행시간과 URLLC 서비스를 비교하였다. 응용프로그램은 표2와 같다.

서비스 이름	설 명
Birch	단일 디바이스를 서비스하기 위한 RESTful API 서비스
Robust	다중 디바이스를 서비스하기 위한 RESTful API 서비스
CIFAR-10	CNN을 이용한 이미지 추론 서비스

[표 2 시스템 성능 검증을 위한 서비스]

실험을 위한 표 2의 서비스는 빠른 배포와 실제 개발환경을 고려하려 컨테이너 기반 마이크로서비스로 작성한다. 초연결 KOREN망에 구축된 테스트베드에서 각 서비스의 평균 RTT (AverageRTT)와 서비스 요청부터 배치까지 걸리는 시간 (Start-up time) 및 서비스 응답시간(Respose Time)을 측정함으로써 성능을 검증한다. 또한, 이미지를 추론하는 서비스에 대한 성능을 측정한다. ImageNet[14] 데이터 셋을 학습시킨 CNN(Convolutional Neural Network)를 이용한다. 딥러닝 추론 단계의 서비스를 엣지 디바이스에 가깝게 배치하여 기존의 환경과의 비교를 통해서 성능을 측정 및 검증한다. 추론을 위한 CNN은 ResNet[15]을 이용한다.

검증 순서는 실제로 각 기능의 정상적인 작동 여부를 확인하는 시험검증 단계와 기존 시스템과 비교/분석하는 실증시험순으로 시스템 성능을 검증한다. 시험검증 단계에서는 가상환경, 실증시험 단계에서는 실제 엣지 클라우드에서 실험을 진행하였다.

Ⅲ 아이디어 개발 진행 내용 및 결과

1. 개발 수행 현황 및 최종 결과물

가. 하드웨어 부분

수행 현황	최종 결과물
- 가상 엣지 네트워크 환경 구축	각각 RTT, 대역폭을 다르게 조정한 네트워크 Testbed
- 쿠버네티스 클러스터 구축	10node 클러스터
- 엣지 클라우드 구축 (실험용)	KOREN SDI VM 기반 Cluster
- Docker 설치 스크립트	Docker.sh
- Kubernetes 설치 스크립트	Kubernetes.sh

[표 3 최종 결과물 (하드웨어 부분)]

나. 소프트웨어 부분

수행 현황	최종 결과물
- 네트워크 모니터링 프로그램 (server)	network_api_server.py
- 네트워크 모니터링 프로그램 (client)	network_api_client.py bandwidth_client.py
- 네트워크 모니터링 프로그램 (slave)	network_api_slave.py
- 스케줄러 프로그램	network_aware_scheduler.py
- HRDF 스케줄링 모듈	hrdf.py
- 스케줄러 dokerfile 및 컨테이너 파일	Dockerfile
- 쿠버네티스 배포 파일	deployment.yaml
- Resnet 모델 파일	resnet_tf.py
- CIFAR-10 서비스 prototype	tf_server.py
- tensorflow_server 모듈	resnet.py
- bandwidth 측정 모듈	bandwidth.py
- RTT 측정 모듈	ping.py

[표 4 최종 결과물 (소프트웨어 부분)]

2. KOREN 연동 및 활용 방안

KOREN 연동 계획		
- KOREN 광역망 (WAN)을 이용한 실제 엣지 클라우드 (서울-대전PoP) 구축 완료 - 실험을 위한 네트워크 환경 구성 - 시스템 테스트 및 검증서비스를 위한 서비스 배포 - 시스템 최종 성능검증 및 평가		
테스트베드		**테스트 내용**
이용 날짜	**이용 장소**	
8/16(월)	UNIST 시스템 소프트웨어 연구실	- KOREN HPC 이용 클러스터 구축 시도
8/17(화)	UNIST 시스템 소프트웨어 연구실	- KOREN HPC 이용 클러스터 구축 시도
8/18(수)	UNIST 시스템 소프트웨어 연구실	- KOREN SDI VM 생성 (서울)
8/18(수)	UNIST 시스템 소프트웨어 연구실	- KOREN SDI VM 생성 (대전PoP)
8/18(수)	UNIST 시스템 소프트웨어 연구실	- SDI VM 연결 및 클러스터 구축
8/31(월)	명지대 지능형 데이터처리 연구실	- 스케줄러 시스템 결합
9/7(월)	명지대 지능형 데이터처리 연구실	- Binch 서비스 배포 및 실험
9/14(월)	명지대 지능형 데이터처리 연구실	- Robust 서비스 배포 및 실험
9/21(월)	명지대 지능형 데이터처리 연구실	- 네트워크 환경 구성
9/21(월)	명지대 지능형 데이터처리 연구실	- CIFAR-10 서비스 배포 및 실험
9/28(월)	명지대 지능형 데이터처리 연구실	- 시스템 수정 및 보완
10/12~16(월~금)	명지대 지능형 데이터처리 연구실	- 최종 성능검증/평가
10/19~23(월~금)	명지대 지능형 데이터처리 연구실	- 최종 성능검증/평가 및 결론 도출

[표 5 KOREN 연동 및 활용 방안]

3. 멘토 의견에 따른 개선사항

가. 멘토 의견에 따른 개선사항

날 짜	멘토링 의견	보완 및 수정사항	멘토
7/22	- 프로젝트 수행 인원 부족	- 인원 1명 추가	홍충선 교수님
7/22	- KOREN 망 연결	- 서울-대전 간 VM 연결	안희철 교수님
7/22	- 분산 딥러닝 응용의 어려움	- 응용 잠정 보류 및 응용 대체	권택원 교수님
7/22	- 스케줄링 정책의 고도화	- 관련 논문 검색	권택원 교수님
8/13	- 스케줄링 머신러닝 적용	- 스케줄링 오버헤드 발생	홍충선 교수님
8/13	- 성능검증을 위한 가시적, 구체적인 방안 필요	- 시나리오 작성	권택원 교수님
8/13	- 시뮬레이션 환경에 대한 구체화 및 실제 환경과의 검증비교가 필요	- 실제 환경에서 실험 및 구현	안희철 교수님
8/26	- 실험 환경 개선 필요	- 실험환경 확장	안희철 교수님
8/26	- 머신러닝 기능의 필요여부	- 기능 개선	안희철 교수님
9/25	- 기존 기술과의 차별성 보완	- 비교를 통한 차별성 검증	홍충선 교수님
9/25	- URLLC 컴퓨팅 서비스에 대한 시나리오	- 필요 산업 기준 비교 검증	송왕철 교수님
9/25	- 검증시나리오 보충	- 실증시험 방식 추가	안희철 교수님

9/25	- 새로운 아이디어 필요성과 차별성에 대한 보충	- 스케줄러 추가	권택원 교수님
10/26	- 네트워크의 안정성이 보장되지않을 때에 대한 대비책	- 스케줄링 방식 개선	권택원 교수님
10/26	- 비교를 통한 목표 성능 제시	- 기본 스케줄러 성능 비교 및 산업 기준 비교	권택원 교수님
10/26	- 채널에 대한 지연값의 머신러닝을 통한 예측	- 데이터 부족으로 인한 잠정 보류	송왕철 교수님
10/26	- 스케줄러에서 Application aware 기능 추가	- Future Work로의 전환	안희철 교수님

[표 6 멘토링 의견 및 개선사항]

4. 결과물의 통합 및 시험/검증 결과

본 과제의 개발 최종 결과물의 성능검증은 결과물의 통합, 시험검증, 실증시험, 향후 연구 계획 순으로 이루어졌다.

가. 결과물의 통합

개발 결과물은 각 기능별로 마이크로서비스 형태로 개발되었다.

결과물은 개발자에 의해서 TDD (Test-Driven Development, 테스트 주도 개발) 되었으며, 단위테스트를 완료 후에 시스템에 이식하는 과정으로 시스템 통합을 진행하였다. 시스템 통합이 완료된 이후에 시스템의 정상 작동 여부를 판단하는 통합 테스트를 진행하였다. 통합 테스트는 발생 가능한 시나리오를 기반으로 검증을 진행하였다. 통합 테스트단계에서는 시스템이 각 단계에서 정상적인 작동 여부를 확인하였다.

나. 시험검증

통합된 결과물의 기능들이 정상적으로 상호작용하고 스케줄러의 의도한대로 스케줄링이 가능하고 서비스가 실행되는지 검증하기 위해 하였다. 시스템의 정상적인 작동에 대해서 각 단계별로 가상 클러스터 상에서 검증하였다. 각 단계에 대한 체크리스트는 표7,8,9와 같다. 체크리스트를 통해서 시스템의 각 기능이 정상적으로 동작함을 알 수 있었다.

순번	체크리스트
1	**쿠버네티스 시스템이 정상적으로 동작하는가? (실험 환경의 적합성)**
1-1	KOREN 네트워크를 포함하는 엣지클라우드를 사용하는가?
1-2	실험에 적합한 각각의 다른 네트워크 환경을 가진 엣지 노드로 구성된 엣지 클라우드 인가?

[표 7 실험 환경의 적합성에 관한 체크리스트]

순번	체크리스트
2	**단말에서는 URLLC 서비스를 사용할 수 있는가? (end device)**
2-1	단말에서 서비스를 요청할 때, 네트워크 속도 (rtt)를 측정할 수 있는가?
2-2	단말에서 서비스를 요청할 때, 네트워크 대역폭을 (bdw)를 측정 할 수 있는가?
2-2-1	단말에서 대역폭 측정 시 소켓 서버통신이 가능한가?
2-3	단말에서 legacy 시스템보다 더 빠른 connection 가능한가?
2-4	성능 향상이 차세대 네트워크를 기반으로 한 URLLC 서비스에 적합한가?

[표 8 end device에 관한 체크리스트]

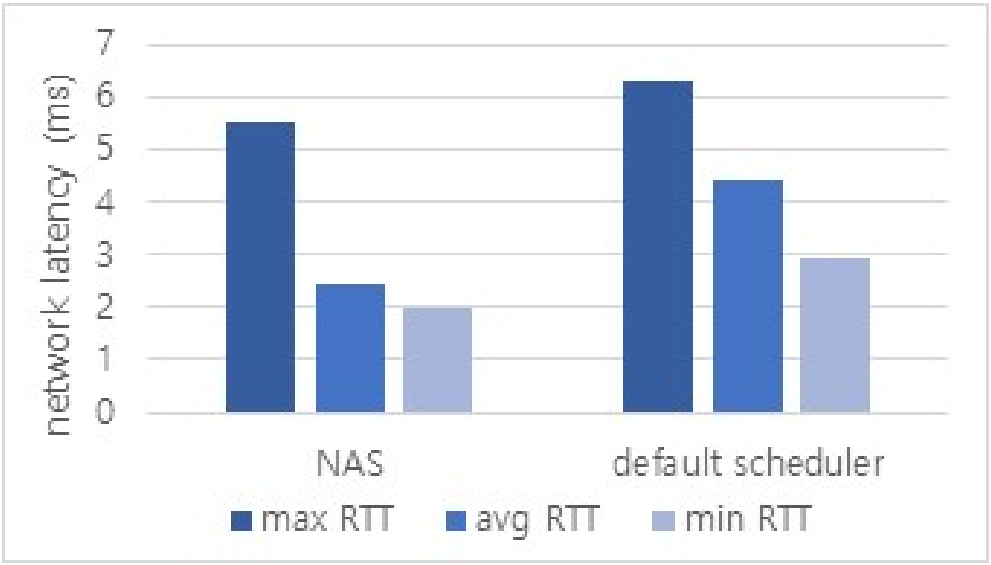
순번	체크리스트
3	**단말 위치에 적합한 엣지 노드에 컨테이너(서비스)가 실행되는가?**
3-1	서비스가 네트워크 상황에 맞게 배치되는가? (NAS)
3-1-1	단말로부터 얻은 네트워크 환경을 스케줄러에서 사용할 수 있는가?
3-1-2	RTT를 고려한 후보군 노드를 선정할 수 있는가?
3-1-3	3-1-1에서 선정한 candidate중 bdw를 통한 최적의 노드를 선정할 수 있는가?
3-2	서비스가 네트워크 이외의 상황을 고려해 적합하게 배치될 수 있는가?(HRDF)
3-2-1	warm-container 를 고려한 스케줄링이 가능한가? (start-up time)
3-2-2	자원 상황에 대한 유동적인 스케줄링이 가능한가?

[표 9 Scheduler에 관한 체크리스트]

다. 실증시험

실증 시험은 KOREN망 기반 엣지 클라우드에서 진행되었다. 실험환경은 12개의 컴퓨팅 노드로 구성된 엣지 클라우드를 이용하여 구성하였다. 시스템 사양은 Intel Core Processor(Broadwell, IBRS, 4core), 8GB 메모리로 구성된 테스트 베드 3개의 서버노드와 Intel(R) Xeon(R) CPU E5-26500 @ 2.00GHz, 64G로 구성된 서버노드 9개로 구성되어있는 테스트 베드를 사용하고 마스터 노드 1개와 워커 노드 11개로 구성된 클러스터를 구성하였다. 엣지 클라우드는 서울, 용인, 대전에 있으며 논리적으로 위치를 조정한 9개의 가상 노드를 이용하여 클러스터를 구성하였다. 실험에 사용한 소프트웨어 버전은 Ubuntu 18.04 LTS, Docker 19.03.6, Kubernetes v1.19.2 이다.

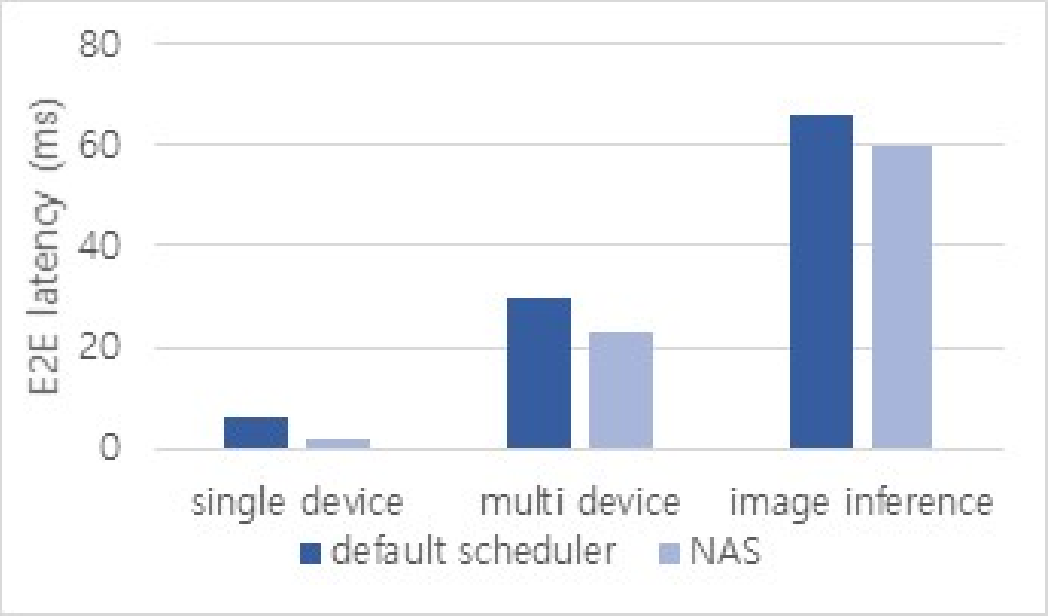
본 프로젝트에서 제안한 첫 번째 스케줄러 NAS에 대한 성능검증은 단말에서 서비스를 요청한 이후에 엣지 노드와 단말 간의 통신 지연시간을 비교 및 분석하였다. 성능검증을 위해서 응용프로그램을 쿠버네티스에 제출한 이후에 단말과 서비스 간의 통신 속도를 측정하였다. 실험을 위한 응용프로그램은 웹 기반의 단일/다중 디바이스를 지원하는 웹 기반 애플리케이션과 딥러닝을 이용한 이미지 추론 어플리케이션으로 구성되어있다. 그림 9은 NAS와 쿠버네티스의 기본 스케줄러를 통해서 배포한 응용 프로그램의 통신 지연시간을 비교한 것이다.



[그림 9 스케줄러 별 평균 통신 지연시간 비교표]

쿠버네티스의 기본 스케줄러는 네트워크 토폴로지에 대한 최적화를 제공하지만, 단말과의 네트워크 연결 상태에 대해서는 고려하지 못한다. 또한, 호스트 자원상황을 우선으로 고려하기 때문에 단말과의 거리가 먼 엣지 노드에서 서비스가 실행되는 경우가 빈번하게 발생하였다. 예를 들어 대전에 있는 단말에서 서비스를 요청했을 때 스케줄러가 자원상황이 적합한 노드에 서비스를 스케줄링하였다. 이는 물리적으로 거리를 늘이기 때문에 실시간 서비스에 대해서 통신 지연을 증가시키는 원인이 되었다.

본 프로젝트에서 제안한 NAS의 경우 단말과 엣지 간의 네트워크 상황을 가장 우선으로 고려한다. 그 때문에 NAS는 쿠버네티스의 기본 스케줄러보다 네트워크 지연시간의 최댓값, 평균값, 최솟값이 각각 12.69%, 53%, 30% 감소하였음을 그림 9을 통해서 알 수 있다.

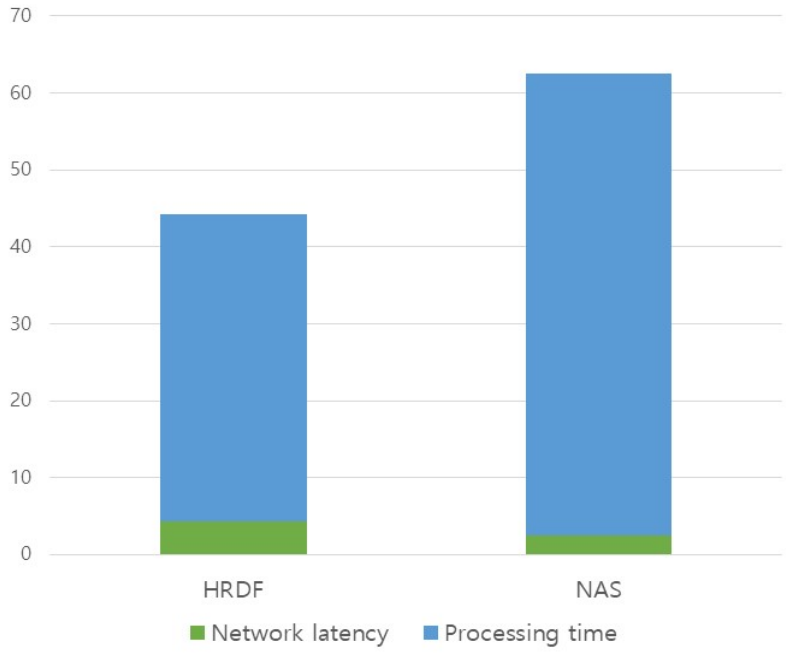


[그림 10 스케줄러 별 평균 통신 지연시간 비교표]

그림 10는 검증 서비스별 E2E (End to End) 지연시간을 나타낸 표이다. 그림 2에서 알 수 있듯이 기존 스케줄러를 대체한 NAS를 사용하여 E2E 지연율이 개선되었음을 알 수 있다. 하지만 단일/다중 디바이스를 지원하는 웹 애플리케이션과는 다르게 딥러닝 기반 이미지 추론 서비스는 네트워크 통신 지연을 개선하였음에도 불구하고 서비스의 E2E 지연율이 6%밖에 개선되지 않았다. 딥러닝처럼 계산 집중적인 응용프로그램은 전체 서비스

지연율에서 네트워크 통신 지연율이 차지하는 비율이 낮다는 사실을 알 수 있었다. 결론적으로 첫 번째 비교표를 통해 스케줄러의 성능 차이를 비교해볼 수 있다. 반면에, 두 번째 비교표는 특정 어플리케이션을 수행하는 환경에서의 성능 차이를 확인할 수 있었다.

NAS와 쿠버네티스의 기본 스케줄러를 비교 실험한 결과, 평균 30%의 유의미한 네트워크 지연시간 감축 효과를 볼 수 있다. 이러한 결과는 향후 엣지 컴퓨팅을 활용한 여러 산업의 상용화를 앞당길 수 있을 것으로 예상한다. 하지만 딥러닝과 같은 계산 집중적인 응용프로그램에 대해서 네트워크뿐만 아니라 컴퓨팅 자원까지 고려하는 스케줄링 정책에 대한 필요성을 인지하게 되었다. 따라서 본 프로젝트는 계산 집중적인 응용프로그램의 성능을 향상하기 위한 HRDF (Highest Residual Density First)를 개발하고 그 성능을 측정하였다. HRDF는 기본적으로 엣지 클라우드가 특정한 워크로드를 수행하는 시스템이라는 것에서 착안한 스케줄링 방식이다. 특정한 워크로드는 작업의 특성을 모두 분석되어 스케줄링에 도움을 줄 수 있다. 따라서 HRDF는 응용프로그램의 업/다운로드 지연시간, 처리시간 및 우선순위가 높은 다른 작업에 대한 대기시간을 이용해서 작업의 E2E 지연시간의 시간을 감소시킨다.



[그림 11 NAS와 HRDF 이미지 추론 응용프로그램 응답시간 비교표]

그림 11은 본 프로젝트에서 제안한 두 번째 스케줄러인 HRDF에 대한 성능 검증은 계산 집중적 응용프로그램인 이미지 추론 프로그램의 응답시간을 측정한 것이다. 실험은 앞서 서술한 컴퓨팅 테스트베드와 동일하게 사용되었으며 구글에서 학술연구용으로 제공하는 Google Cluster Workload trace[16]를 이용하여 실제 워크로드를 수행하는 환경을 시뮬레이션 하였다. 엣지 컴퓨팅 환경에서 100개의 작업을 무작위로 실행하면서 실험용 응용프로그램을 스케줄러를 통해서 실행하였다.

NAS는 비선점식 스케줄링 방식으로 동작하는 것과 반대로 HRDF는 선점식 스케줄링의 이점을 활용하는 방식을 가지고 있기 때문에 더 나은 성늘을 보여주었다. 그림 11을 통해서 알 수 있듯이 HRDF는 이미지 추론과 같은 계산 집중적 응용에 대하서 NAS 보다 30% 전체 지연시간이 개선되었음을 알 수 있었다. 네트워크 지연을 NAS와 HRDF를 비교하였을 때 HRDF가 50%정도 더 느렸지만, 처리시간은 34%의 감소하였다.

계산 집중적 응용프로그램이 전체 응답시간에서 네트워크 지연시간이 차지하는 비율을 보았을 때 처리시간에 비해 현저히 적기 때문에 네트워크 지연시간을 늘리고 응용프로그램의 처리시간을 줄이는 것이 효율적인 방식임을 알 수 있었다.

라. 향후 연구 계획

위와 같은 실증실험을 거치며, 스케줄링 방식을 바꾸는 것이 전체 작업의 처리시간과 응답시간을 감소시킬 수 있다는 것을 알 수 있었다. 또한 Network-aware 스케줄링과 HRDF 뿐만 아니라 Application-aware한 스케줄링 방식[17]을 통해서 주어진 작업이 어떠한 종류의 작업인지 분류하고 파악한 이후에 종류에 맞는 스케줄링 방식을 사용한다면 향후 엣지 클라우드의 서비스 효율과 자원 사용률을 더 높일 수 있을 것으로 발전 가능성에 대해서 생각하였다.

NAS와 같은 경우 본 프로젝트에서 개발한 방식은 네트워크 단절이나 보안적인 위협에 대해서 취약하다. 때문에 네트워크의 안정성이 보장되지 않을 때의 대처방안을 시스템에 추가함으로써 시스템의 안정성을 높일 수 있을 것으로 예상하였다. 이러한 기술은 채널에 대한 지연을 측정 및 평가할 수 있는 머신러닝 기반의 시스템을 도입하여 기존의 NAS의 정적인 스케줄링 방식에 비해 더 지능적이고 유동적인 스케줄링 가능할 것을 기대한다. 본 과제에서 수행한 NAS와의 네트워크 성능 비교를 통해서 시스템을 발전시킬 향후 계획을 수행할 계획이다.

또한 본 과제에서 수행한 HRDF에서 대해서도 향후 연구 계획을 가질 수 있었다. HRDF 스케줄링 방식이 엣지 클라우드가 특정한 도메인을 수행한다는 가정하에 고안한 스케줄링 방식이다. 이러한 스케줄링 방식은 기존에 없던 새로운 응용이 시스템에서 실행되면 자원 관리 상황에서 굉장히 취약하다는 단점을 보이고 있는데 이를 극복하기 위해서 실행되고 있는 응용프로그램이 어떤 자원을 중점적으로 사용하는지에 대한 프로파일링 할 수 있는 기능을 추가하는 것을 계획하고 있다. 프로파일링을 통해서 기존에 없던 사용자가 사용하기 이전에 시스템 단계에서 더 빨리 응용프로그램에 대한 정보를 스케줄링 단계에서 이용하여 자원 활용률에 대한 성능 향상이 있을 것으로 기대하고 있다.

Ⅳ 결 론

1. 개발 최종 결과물

엣지 클라우드 환경에서 초고신뢰/초저지연 컴퓨팅(URLLC) 서비스를 위한 Network-Aware Placement System의 최종 결과물은 다음과 같다.

- 클러스터 서버 기반 가상 엣지 클라우드, KOREN SDI 기반 엣지 클라우드

최종 결과물	설명
UNIST Whale Cluster Testbed	실험용 쿠버네티스 클러스터 (가상 클라우드)
Edge Network Testbed	네트워크 환경 실험용 클러스터
KOREN SDI VM cluster	서울-대전PoP로 구성된 실제 엣지 클라우드
Docker.sh	docker.io 설치용 스크립트
Kubernetes.sh	kubectl, kubeadm, kubelet 설치용 스크립트

[표 10 최종 결과물 (클러스터 부분)]

- 엣지 클라우드의 네트워크 정보를 모니터링 할 수 있는 API 기반의 툴

최종 결과물	
network_api_server.py	flask, flask_restful 모듈을 이용한 마스터 API 서버 - '/api/rtt/<node_ip> : 해당 ip 주소의 rtt를 json 형식으로 반환함
network_api_client.py bandwidth_client.py	flask, flask_restful 모듈을 이용한 워커 API 서버 - '/api/rtt/' : 엣지 디바이스간에 rtt, 대역폭 측정 후 이 정보는 마스터 서버 API로 전송
bandwidth.py	소켓 서버에 Synthetic 데이터를 송신함
bandwidth_server.py	Synthetic 데이터 (10Mb)를 수신함으로써 bandwidth를 측정하는 소켓 서버
ping.py	ping 모듈을 이용한 RTT 측정

[표 11 최종 결과물 (네트워크 모니터링 API 부분)]

- 최적의 네트워크 환경의 엣지 노드를 선정할 수 있는 스케줄러

최종 산출물	설명
network_aware_scheduler.py	네트워크 모니터링 API를 통해서 최적의 네트워크를 가지는 노드를 선정하는 스케줄러 구체적인 메커니즘은 'Algorithm 1' 참고
Dockerfile	NAS (Network Awareness Scheduler)를 컨테이너로 빌드하기 위한 설정 파일
deployment.yaml	스케줄러를 쿠버네티스에 배포하기 위한 설정 파일
hrdf.py	어플리케이션의 정보(처리시간, 업/다운로드 시간)을 이용한 계산 집중적 응용처리를 위한 HRDF 모듈
hrdf_scheduler.py	Job을 selfish와 nearest로 나누고 NAS와 HRDF의 적용을 분류한 이후 hrdf모듈을 이용해서 계산 집중적 응용프로그램에 대해서 각 엣지클라우드의 연산 및 업/다운로드 가중치를 계산 및 적용하는 스케줄러

[표 12 최종 결과물 (스케줄러 부분)]

- 시스템 검증을 위한 응용 프로그램

최종 산출물	설명
single_device.py	단일 디바이스를 지원하는 웹기반 응용프로그램
multi_device,py	다중 디바이스를 지원하는 웹기반 응용프로그램
alexnet_cifar10_tf.py	Deep Neural Network를 이용한 cifar-10 이미지 Dataset 학습 및 추론
tensorflow_serve.py	Convolutional Neural Network, Resnet을 이용한 ImageNet 학습 및 추론
resnet.py	tensorflow프레임 워크를 이용한 resnet-50 구현 프로그램

[표 13 최종 결과물 (검증서비스 부분)]

2. 기술 구현 에 따른 기대 효과

본 과제는 무선링크에서도 유선링크 수준의 실시간 통신과 제어를 위한 디바이스들을 효과적으로 처리할 수 있는 서비스 배치시스템이 필수적이라는 것에 기인하여 시작하였다. 초고신뢰/초저지연 중에서도 초저지연에 초점을 맞춰 프로젝트를 진행하였고, 그 결과 NAS(Network Aware System)라는 네트워크 속도를 고려하여 노드를 찾는 스케줄러와 Processing Time을 현저하게 줄여주는 노드를 찾아 처리하는 스케줄러인 HRDF(Highest Residual Density First)라는 스케줄러를 구현하였다.

본 과제를 수행하여 얻은 결과물을 토대로 기술적, 경제적, 사회적인 3가지 측면의 기대효과를 서술하고자 한다.

(1) 기술적 측면의 효과

기술적 측면의 효과에서 첫 번째로 기대할 수 있는 점은 적합한 자원할당을 통한 시간 단축’ 이다. AR/VR, 커넥티드 자동차(Connected Car), 스마트 시티(Smart City) 등 10ms 이하의 지연(Latency)이 필요한 실시간 애플리케이션이 증가하고 있는데, 이러한 응용프로그램들은 지터 (Jitter, 신호 불안정)에 민감한 특징을 가지고 있기 때문에 NAS를 통해 단말 혹은 이용자 최근접에 인프라를 배치하여 지연을 최소화할 수 있는 인프라를 제공함으로써 제안 시스템이 활용될 수 있다는 장점이 있다.

두 번째로는 ‘엣지컴퓨팅을 활용할 수 있는 인프라 구축’이다. 현재 상황은 엣지 컴퓨팅을 적용하여도, 이에 맞는 작업할당이 원하는 수준까지는 이루어지지 않아, 엣지 컴퓨팅의 충분한 효과를 기대하기 어려운 상황이다. 현재 개발한 NAS, HRDF 스케줄러를 통해서 엣지 클라우드 환경에서 쿠버네티스를 바탕으로 적합한 자원할당 및 서비스 배치시스템에 대한 구성이 가능할 것으로 예상되고, 이에 따라 타 프로젝트나 서비스를 개발할 때, 조금이나마 효율적이고 빠른 인프라가 구축될 것으로 기대된다.

세 번째로는 ‘엣지 클라우드의 서비스 배포 간소화’이다. 본 과제는 엣지클라우드에 필수적인 마이크로서비스 기반 애플리케이션을 효율적으로 관리하는 쿠버네티스 환경을 기반으로 이루어져 있다. 이를 통해, 기관이나 기업에서 엣지 클라우드를 이용한 서비스 개발 및 배포가 편리해질 거라 예상한다. 또한, Devops 관점에서도 간소화된 서비스 배포를 통해 빠른 피드백이 가능하고, 효율적으로 환경을 구성할 수 있을 거라 예상하고, 이에 따라 개발 속도가 향상될 것으로 기대한다.

(2) 경제적 측면의 효과

경제적 측면에서의 기대할만한 점으로는 첫 번째로 ‘On Demand형 네트워크 서비스 제공’이 가능하게 된다는 점이다. 현 네트워크 시스템은 URLLC의 빨라진 네트워크 속도와 엣지 컴퓨팅의 장점을 통한 대역폭 제한 해제(Mitigating bandwidth limits)로 대용량의 정보 전송이 가능하다. 이러한 특징은 사용자의 요구가 있을 때, 언제 어디서나 사용자 중심에서 사용자의 요구를 해결하는 On Demand형 네트워크 서비스를 좀 더 효율적인 방식으로 제공이 가능할 것이다.

두 번째로는 앞으로 더욱 많은 분야에 사용될 차세대 네트워크인 ‘5G의 경제적 가치’이다. 한국무역보험공사의 분석[18]에 따르면, 5G로 인한 경제적 수익이 글로벌 매출은 2026년경 US $1.3조로 예상하고, 국내경제의 파급효과는 2030년경 47.7조 원으로 전망하고 있다. 또한, 여러 가지 분야에서 이점이 있겠지만, 현재 예상 가능한 분야는 COVID-19(코로나바이러스)으로 인한 언택트 시대가 도래하면서 폭발적인 데이터 트래픽으로 인한 트래픽 수준 증가가 가장 큰 5G의 사용처가 되지 않을까 예상해본다. 그리고 제조업에서도 변화가 있을 것으로 본다. 처음 고객의 주문부터 사용자의 재고관리 요청까지 실시간으로 모든 과정을 볼 수 있어 접근성이 뛰어나기 때문에, 생산성의 효율 역시 증대할 것으로 본다.

(3) 사회적 측면의 효과

현재 URLLC 기술은 발전 가능성이 넘치고, 미래지향적인 기술로써 상용화가 이루어진다면, 사회적인 기대효과가 큰 기술이라고 생각한다. 본 과제를 통해서 URLLC 기술이 상용화가 이루어진다면 어떤 영향을 미칠지에 대한 관점에서 사회적 측면의 효과를 서술하겠다.

첫 번째로는 URLLC 기술이 진보함에 따라 발전하는 'VR/AR'의 효과를 들 수 있다. 가장 큰 효과로는 사용자의 안전이 보장된다는 것이다. 군사 분야와 항공 우주 같은 사용자가 목숨에 위협을 받는 상황에서 정교하고, 빠르게 작동하는 VR/AR은 위험성은 0에 가깝게 만들어주고, 그 상황에서의 경험과 지식은 사용자가 쌓을 수 있으므로 안전성이 보장된다고 할 수 있다. 또한, 재난이나 의료 분야와 같은 사람의 생명이 오가는 분야에서도 실제 상황이나 전문 인력이 빠르게 투입되어야 하는 위급상황에서도 이러한 기술을 통해 미리 교육하고 훈련할 수 있어서 실제 경험이 없는 구조대원이나 의사도 충분히 구체적이고 긴박한 상황에 대응할 수 있다.

두 번째로는 '초저지연을 통한 자율주행차량의 완전 자율주행화'이다. 현재 차량들의 자율주행 단계는 2단계 정도로 속도 조절과 장애물 회피 등 두 가지 정도의 자동주행 기능을 수행하는 정도이다. 여기서 5단계인 운전자가 필요 없는 상태가 되는 완전 자율주행화 단계로 넘어가기 위해서는 차량 센서나 주행 상태 데이터 그리고 주변 차량 및 도로상황 등을 관제센터로 전송하여 인프라의 보조를 받아야 한다. 그러므로 여기서 초저지연과 초고신뢰의 속성이 가장 필요하다고 보여진다. 이를 통해 완전한 자율주행이 가능해지면, 사람들의 차내에서의 여유시간이 늘어남에 따라서 차량운행의 메커니즘이 달라질 것으로 예상한다.

V 참 고 문 헌

[1]. X. Chen, L. Jiao, W. Li and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," in IEEE/ACM Transactions on Networking, vol. 24, no. 5, pp. 2795-2808, October 2016, doi: 10.1109/TNET.2015.2487344.

[2]. kubernetes homepages
https://kubernetes.io/ko/docs/concepts/overview/what-is-kubernetes/

[3]. P. Popovski, K. F. Trillingsgaard, O. Simeone and G. Durisi, "5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View," in IEEE Access, vol. 6, pp. 55765-55779, 2018, doi: 10.1109/ACCESS.2018.2872781.

[4]. 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), Paris, France, pp. 351-359, 2019

[5]. https://www.cncf.io/blog/2020/09/16/toc-approves-kubeedge-as-incubating-project/

[6]. kubeEdge 홈페이지, https://kubeedge.io/ko/

[7]. kubeEdge 로드맵, https://kubeedge.io/ko/docs/roadmap/

[8]. https://kubernetes.io/ko/docs/concepts/overview/components/

[9]. kubernetes sceduler,
https://kubernetes.io/ko/docs/concepts/scheduling-eviction/kube-scheduler/

[10]. KOREN(지능형 연구개발망) 이용안내서, 한국정보화진흥원(NIA)

[11]. R. Islambouli, Z. Sweidan and S. Sharafeddine, "Dynamic Multipath Resource Management for Ultra Reliable Low Latency Services," 2019 IEEE Symposium on Computers and Communications (ISCC), Barcelona, Spain, 2019, pp. 987-992, doi: 10.1109/ISCC47284.2019.8969731.

[12]. H. Tan, Z. Han, X. Li and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," IEEE INFOCOM 2017 – IEEE Conference on Computer Communications, Atlanta, GA, 2017, pp. 1-9, doi: 10.1109/INFOCOM.2017.8057116.

[13]. Flask frame work, https://flask-docs-kr.readthedocs.io/ko/latest/

[14]. Image net, http://www.image-net.org/

[15]. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[16]. J. Wilkes. Google cluster-usage traces v3. Technical report at https: //github.com/google/cluster-data, Google, Mountain View, CA, USA, Nov. 2019.

[17]. Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: resource-efficient and QoS-aware cluster management. In Proceedings of the 19th international conference on Architectural support for programming languages and operating systems (ASPLOS '14). Association for Computing Machinery, New York, NY, USA, 127–144. DOI:https://doi.org/10.1145/2541940.2541941

[18]. [산업동향보고서] 5G 및 이동통신산업 동향 분석http://www.alio.go.kr/informationResearchView.do?seq=2494863

회의록

o 일　시 : 2020년 07월 22일, 15:00 ~ 17:00
o 장　소 : 한국정보화진흥원 서울사무소
o 참석자 : 엄상현, 안희철, 홍충선, 송왕철, 권택원 등 3그룹

1. 회의록

멘토링 및 조치사항

- **프로젝트 수행 인원 부족 : 9월부터 개발인원 1명 추가 예정**
- **KOREN망 연결 : 서울-부산 망 연결, 부경대와 협의중**
- **분산 딥러닝 응용의 어려움 : 잠정 보류 및 다른 응용으로 대체**
- **스케줄링 정책의 고도화 : 특정한 서비스에 대해서 활용될 시스템 상황을 고려, 관련 논문 검색중**

1. 회의 사진

o 일　시 : 2020년 08월 13일, 10:00 ~ 11:00
o 장　소 : 온라인 (BreakOut 이용)
o 참석자 : 엄상현, 안희철, 홍충선, 송왕철, 권택원 등 3그룹

1. 회의록

금일 멘토 그룹과 지도교수님과의 2차 멘토링을 가졌다.
진행 상황과 지난 1차 멘토링 결과에 대한 조치결과를 보고했다.
지도 교수님들께서 프로젝트 진행에 대해서 보완해야 할 점에 대해서 알려주셨다.

홍충선 교수님 : 스케줄링에 머신러닝을 적용할 수 있도록 생각
권택원 교수님 : 성능검증을 위한 가시적이고 구체적인 것이 필요
안희철 교수님 : 시뮬레이션 환경에 대한 구체화와 실제 환경과의 검증비교가 필요

1. 회의 사진

o 일　시 : 2020년 08월 26일, 10:00 ~ 11:00
o 장　소 : 온라인 (BreakOut 이용)
o 참석자 : 엄상현, 안희철, 홍충선, 송왕철, 권택원 등 3그룹

1. 회의록

금일 멘토 그룹과 지도교수님과의 3차 멘토링을 가졌다.

진행 상황과 지난 2차 멘토링 결과에 대한 조치결과를 보고했다.
지도 교수님들께서 프로젝트 진행에 대해서 보완해야 할 점에 대해서 알려주셨다.

홍충선 교수님 : 실험환경의 개선이 더 필요
권택원 교수님 : 성능검증을 위한 가시적이고 구체적인 것이 필요
안희철 교수님 : 스케줄러의 머신러닝 기능이 의미가 있는지

2. 회의 사진

o 일　시 : 2020년 09월 01일, 13:00 ~ 18:00
o 장　소 : 명지대 부근 카페 (coffe320)
o 참석자 : 엄상현, 노지환

1. 회의록

기본적인 프로젝트 계획 수립 및 조정
중간발표에 대한 계획 수립
개발 내용 인수인계 및 역할 분담
중간보고 관련 사항 정의

1. 회의 사진

o 일　 시 : 2020년 09월 16일, 13:00 ~ 18:00 *o 장　 소 : 명지대 부근 카페 (브라운 리퀴드)* *o 참석자 : 엄상현, 노지환*
1. 회의록
스케줄러 관련 개발 토의 **시나리오 구축 및 분석 방안 토의** **스케줄러 개선 방안 토의** **개발 진행 상황 보고**
1. 회의 사진


o 일　시 : 2020년 09월 25일, 10:00 ~ 11:00
o 장　소 : 온라인 (BreakOut 이용)
o 참석자 : 엄상현, 노지환, 안희철, 홍충선, 송왕철, 권택원 등 3그룹

1. 회의록

차별성에 대한 보완점
URLLC 컴퓨팅 서비스에 대한 시나리오 요구
구현에 대한 시나리오가 만들어졌지만 검증에 대한 시나리오가 보충해야함
새로운 아이디어의 필요성과 차별성에 대한 보충

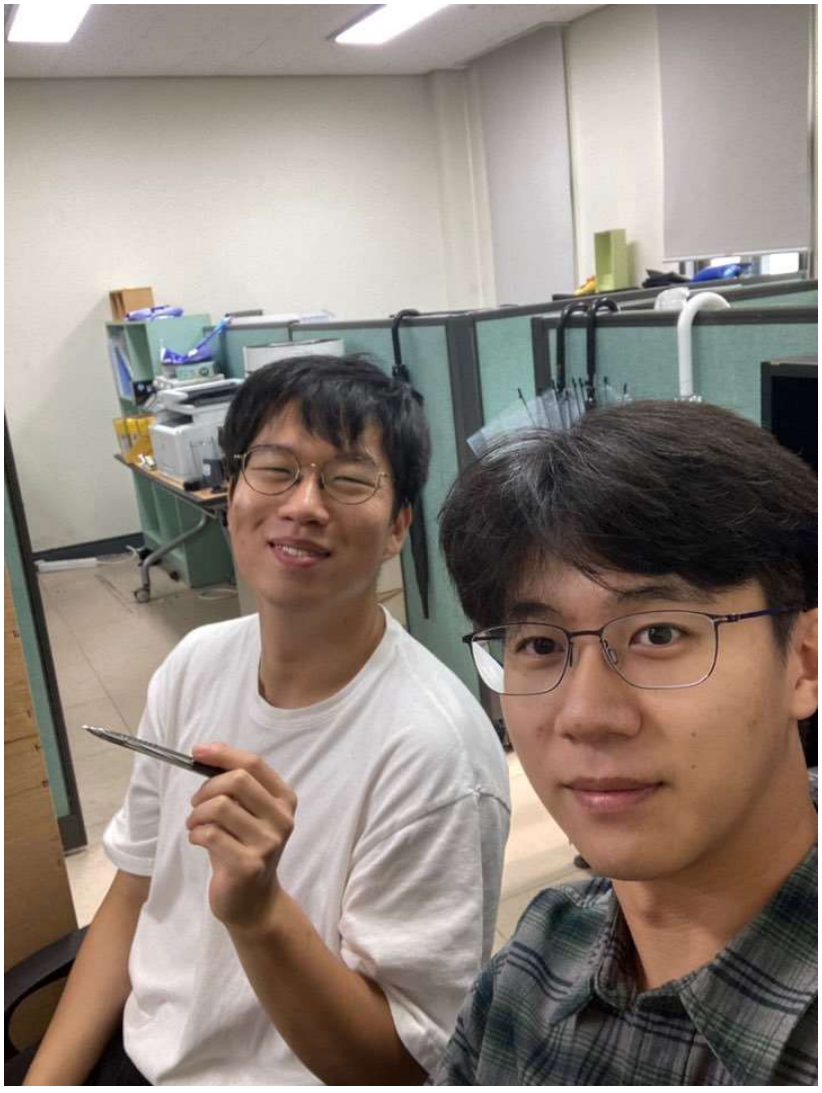
1. 회의 사진

o 일　시 : 2020년 09월 29일, 13:00 ~ 17:00
o 장　소 : 명지대학교 지능형 데이터처리연구실
o 참석자 : 엄상현, 노지환

1. 회의록

실험 시나리오 구성 및 클러스터 구축 방안토의
스케줄러 빌드 및 작동 방식 토의
개발방향에 대한 토의
IoT and Edge computing with Kubernetes white paper review

1. 회의 사진



o 일　시 : 2020년 10월 05일, 10:00 ~ 19:00
o 장　소 : 명지대학교 지능형 데이터처리연구실
o 참석자 : 엄상현, 노지환

1. 회의록

실험 시나리오 구성 및 클러스터 구축
스케줄러 성능 측정 및 성능 검증 실험
개발방향에 대한 토의
Extend Cloud to Edge with KubeEdge review

1. 회의 사진

o 일　시 : 2020년 10월 23일, 10:00 ~ 12:00
o 장　소 : 온라인 (BreakOut 이용)
o 참석자 : 엄상현, 노지환, 안희철, 홍충선, 송왕철, 권택원 등 3그룹

1. 회의록

현재까지의 진행상황 발표
네트워크의 안정성이 보장되지 않을 때 대처의 필요성
비교목표 성능을 제시하여 작성
채널에 대한 지연값을 estimation에 대한 머신러닝을 도입 가능성
NAS의 네트워크 성능 비교방식을 머신러닝을 통한 값들로 해보는 것에 대한 방향성

1. 회의 사진

o 일　시 : 2020년 11월 11일, 12:00 ~ 17:00
o 장　소 : 역북동 249 커피
o 참석자 : 엄상현, 노지환

1. 회의록

진행 상황 보고
발표 자료 작성 및 실증실험에 대한 결과 보고 양식
최종보고서 수정 사항 반영
과제 종료 이후 향후 연구 계획에 대한 토의

1. 회의 사진