논문 22-47-07-14 The Journal of Korean Institute of Communications and Information Sciences '22-07 Vol.47 No.07
https://doi.org/10.7840/kics.2022.47.7.1010

전술 엣지 클라우드에서의 컨테이너 이미지 및 상태 관리 기법

임 호 근◆, 김 영 한°

Container Image and State Management in the Tactical Edge Cloud Environment

Ho-keun Lim◆, Young-han Kim°

요 약

전술 엣지 클라우드 아키텍처는 컨테이너 기술을 활용하여 전술 환경에서 필요한 다양한 기능들을 경량 워크로드로 배포함으로써 제공한다. 전술 엣지 클라우드에서는 물리 자원의 최적화 및 장애 복구를 지원해야 하며 이는 워크로드를 다른 곳으로 이전 시키는 워크로드 마이그레이션으로써 해결이 가능하다. 워크로드 마이그레이션을 지원하기 위해서는 컨테이너 이미지와 상태 정보가 필요하다. 하지만 네트워크의 성능이 제약된 전술 엣지 클라우드 환경에서는 상위 노드로의 링크가 불안정하기 때문에 다운로드 속도에 영향을 준다. 워크로드를 배포하기 위한 컨테이너 이미지를 모든 노드에 미리 준비시키는 것으로 해결될 수 있지만 이는 자원의 낭비를 발생시킨다. 본 논문에서는 전술 엣지 클라우드에서 P2P 분산 시스템을 적용하여 마이그레이션 시 필요한 컨테이너 이미지 및 Stateful 워크로드의 상태를 이웃 노드에서 가져오는 기법을 제안하고 기능 구현을 통해 워크로드 마이그레이션 시의 지연시간 및 저장 용량 감소 효과를 검증했다.

Key Words : Tactical Edge Computing, Container Workload Migration, Resource Optimization, Container Image, P2P System

ABSTRACT

The tactical edge cloud architecture is provided by using container technology to deploy various functions required in the tactical environment as lightweight workloads. In the tactical cloud, physical resource optimization and disaster recovery must be supported, which can be solved by workload migration that moves workloads elsewhere. Container images and status information are required to support workload migration. However, in a tactical edge cloud environment where the performance of the network is limited, the link to the upper node is unstable, which affects the download speed. Although it can be solved by preparing container images for deploy workload to all nodes, this creates a waste of resources. In this paper, we propose a technique to download the container images and status information of Stateful workloads required for migration from neighboring nodes by applying P2P distributed systems in the aforementioned edge cloud, and verify the speed and storage capacity reduction effect during workload migration through feature implementation.

※ 본 연구는 방위사업청과 국방과학연구소가 지원하는 미래전투체계 네트워크기술 특화연구센터 사업의 일환으로 수행되었습니다.(UD190033ED)
◆ First Author : Soongsil University Department of Electronic Engineering, limhk@dcn.ssu..ac.kr, 학생회원
° Corresponding Author : Soongsil University Department of Electronic Engineering, younghak@ssu.ac.kr, 종신회원
논문번호 : 202205-085-0-SE, Received April 29, 2022; Revised May 14, 2022; Accepted May 14, 2022

Ⅰ. 서 론

엣지 컴퓨팅 기술은 사용자 가까운 곳에 클라우드 서버를 배치하여 가상화 서비스들을 제공함으로써 저지연 서비스 제공 및 트래픽 분산의 효과가 있다. 이러한 이유로, 모바일 네트워크를 비롯한 다양한 환경에서 엣지 컴퓨팅을 적용하기 위한 연구가 진행되고 있다. 특히, 제한적인 네트워크 자원을 가지고 있는 전술망 환경에서, 엣지 컴퓨팅은 전술 서비스에 대한 빠른 연결과 안정성을 제공할 수 있어, 이와 관련된 연구들이 제안되었다. 전술 엣지 클라우드에서는 제한된 자원을 극복하기 위해 데이터 처리에 소요되는 컴퓨팅, 스토리지 등의 자원을 최대한 가까운 곳에서 지원할 수 있어 변화는 미래 전투환경에서 데이터 처리를 통한 긴밀한 작전 수행이 가능할 뿐 아니라, 무선 링크 기반의 백본 네트워크 연결성이 취약한 상황에서도 단말과 연결될 수 있어 전술 수행 성능의 안정성을 제공할 수 있다는 장점이 있다[1-3].

하지만 엣지 클라우드 환경은 일반적인 상용망보다 제한적인 자원을 가지고 있다. 따라서 엣지 클라우드는 기존의 가상머신보다 경량화된 컨테이너 기술을 지원하는 형태로 인프라가 발전되었다. 엣지 클라우드에서는 물리 및 가상 자원의 최적화 또는 장애 복구 등에 이유로 클라우드 내 워크로드가 물리 서버에서 다른 서버로 이전하는 마이그레이션의 수행은 엣지 컴퓨팅 구조에서 분산된 클라우드의 자원 최적화와 이동 단말에 따른 접속 구간을 최적화하기 위해 필요하다[4,5]. 워크로드의 마이그레이션을 진행하기 위해서는 해당 워크로드를 배포하기 위한 기반 이미지가 필요하며 추가적으로 Stateful한 워크로드에 대한 마이그레이션을 진행하기 위해서는 현재 워크로드의 상태를 저장하는 것이 필요하다[6,7].

컨테이너 이미지는 일반적으로 인프라의 노드에서 컨테이너화된 애플리케이션을 시작하기 위해 컨테이너 레지스트리에서 제공되고 각 런타임 에이전트에 의해 다운로드 된다. 컨테이너 레지스트리는 클라우드에 배포되는 중앙 집중식 서비스인 경우가 많다. 즉 Docker Hub 또는 자체적인 레지스트리를 중앙 클라우드에서 제공하여 사용한다[8]. 단일 쿠버네티스 환경이 아닌 엣지 클라우드 환경에서도 각 클라우드를 관리하는 호스트 클라우드에서 컨테이너 레지스트리를 운영하는 방식을 취하고 있어 엣지 클라우드에서 이미지를 다운로드 할 경우 상대적으로 거리가 먼 호스트 클라우드에 접근해야 한다.

워크로드의 마이그레이션의 성능은 워크로드 실행을 위한 컨테이너 이미지와 워크로드의 상태 정보를 다운로드 받는 시간에 영향을 받는다. 하지만 네트워크 상황이 제약된 엣지 클라우드 환경에서 상위 링크에 접근하는데 어려움이 있으므로 전술 엣지 클라우드에서는 이와 같은 방법은 사용되기 어렵다. 전술 엣지 클라우드에서 사용되는 컨테이너 이미지를 미리 모든 노드에 다운로드할 수도 있지만 이러한 방법은 사용하지 않는 이미지를 미리 저장하여 자원의 낭비를 발생시킨다.

본 논문에서는 P2P 분산 시스템을 적용하여 전술 엣지 클라우드에서 컨테이너 워크로드 마이그레이션을 수행을 위해 컨테이너 이미지와 워크로드의 상태 정보를 전술 엣지 클라우드를 구성하는 노드에 분산 저장함으로써 자원을 효율적으로 사용하고, 이미지 및 상태 정보를 주변 노드에서 다운로드 함으로써 필요한 정보를 다운로드 하는 시간을 감소했다.

본 논문의 구조는 다음과 같다. 2장에서는 관련 연구로써 전술 엣지 클라우드 구조와 컨테이너 이미지 저장 방식에 대해 설명한다. 3장에서는 제안하는 전술 엣지 클라우드에서의 P2P 기반 컨테이너 이미지 및 상태 정보 관리 구조와 상세 절차를 설계한다. 4장에서는 제안 구조를 구현하여 제안 구조에 대한 성능 평가 및 결과를 분석하고, 5장에서 결론을 맺는다.

Ⅱ. 관련 연구 및 배경

2.1 전술 엣지 클라우드 구조 연구

엣지 컴퓨팅 기술의 핵심 아이디어는 SDN, NFV, 클라우드 컴퓨팅 기술 등을 활용하여 애플리케이션 기능, 네트워크 기능 및 이에 필요한 컴퓨팅, 스토리지 자원 등을 네트워크의 가장자리로 분산하는 것이다.[9] 이는 기존에 물리적으로 단말과 멀리 떨어진 클라우드 센터에서 제공하던 클라우드 컴퓨팅 기술에서 무선 자원의 희소성 및 트래픽 부하, 낮은 대역폭과 서비스 가용성 등의 문제[10]를 해결하기 위한 것으로, 다양한 형태의 엣지 컴퓨팅 구조가 제안되었다. MEC(Mobile Edge Computing)[11], Fog computing[12] 및 Cloudlet[13]과 같은 엣지 클라우드 기술은 세부적인 구조와 동작은 다르지만, 공통적으로 엑세스 네트워크 근처에서 다양 한 기능을 범용 물리 서버 위에 논리적으로 자원을 가상화하고, 애플리케이션 기능을 각 가상화된 서버 위에서 동작하도록 하는 공통점을 가지고 있다. 이러한 엣지 컴퓨팅 기술에서의 가장 주요한 활용 사례는 IoT 단말에 대한 관리 및 응용 프로그램으로써 헬스케어, 무선 센서 기반의 스마트 그리

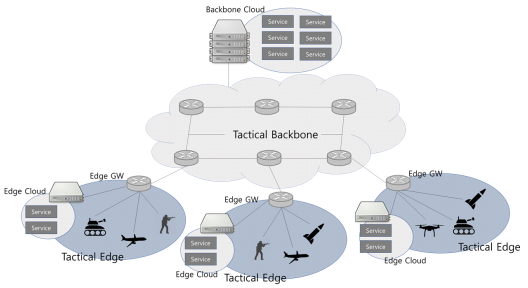

그림 1. 전술 엣지 클라우드 구조
Fig. 1. Tactical Edge Cloud Architecture

드, 스마트시티 등에 적용하기 위한 연구들도 지속적으로 제안되고 있고 위의 이점에 따라 전술망 환경에서도 사용되고 있다[14,15].

본 논문에서 고려하고 있는 전술 엣지 클라우드의 구조는 그림 1과 같다. 격자 형태의 전술 백본망의 백본 라우터에 하나 이상의 전술 엣지 클라우드들이 연결되며, 하나의 라우터에 하나 이상의 전술 엣지 클라우드가 연결될 수 있다. 상위 클라우드에 배치된 작전 수행을 위한 애플리케이션 및 망관리 기능들은 백본의 대규모 클라우드 센터에서 1차적으로 수행되며, 전술상의 필요에 따라 사용 중인 기능들이 배치된 전술 엣지 클라우드에 이전되어 동적으로 배치될 수 있다.

2.2 컨테이너 이미지 저장 기법 및 컨테이너 Live Migration 기법 연구

컨테이너 서비스를 배포하기 위해서는 컨테이너 이미지가 필요하며 컨테이너 이미지는 그림2와 같이 전술 엣지 클라우드를 구성하는 노드에 저장된다. 컨테이너 이미지는 매니페스트 및 레이어 Blob의 두 가지 주요 구성요소로 구성된다.

이미지 계층은 이미지 구축 과정에서 Dockerfile의 한 명령으로 생성된 읽기 전용 파일 시스템이다. 그 후, 이미지의 중간 상태를 설명하고 해당 단계에서 설치된 다른 소프트웨어 또는 종속성을 나타낸다. 압축된 레이어는 Blob이라고 하며 이미지로 구성된 모든 레이어를 포함하는 매니페스트에 나열된다. 도커 이미지의 계층 구조는 중복 제거도 가능하므로 이미지 간에 계층을 공유할 수 있다. 여러 컨테이너에서 사용되는 계층은 노드에서 한 번만 다운로드하면 된다. 컨테이너 시작 중에 도커 데몬은 먼저 매니페스트 파일을 로드한 다음 압축 해제된 계층에서 컨테이너를 만들고 쓰기 가능한 컨테이너 계층을 추가하고 네트워크를 구성하기 전에 로컬 노드에서 아직 사용할 수 없는 모든 압축 계층의 blob을 가져온다[16,17].

하나의 물리 서버에서 다른 물리 서버 또는 하나의 클러스터에서 다른 클러스터로 컨테이너를 이전하는 기술인 컨테이너 마이그레이션은 일반적으로 non-live 마이그레이션과 live 마이그레이션으로 구분된다. Live 마이그레이션 방식은 기존의 컨테이너을 종료하지 않고 특정 순간에서의 컨테이너 이미지 및 상태 정보를 이전할 서버로 복사한 뒤, 지속적으로 서비스를 통해 업데이트되는 상태 정보를 실시간으로 이전된 서버로 전달함으로써 상태 동기화를 수행하여 이전된 서버에서의 컨테이너와의 상태 동기화가 이루어지면 컨테이너을 종료하고 새로운 곳으로 서비스를 재개하는 방식이다. 이는 서비스 중단 시간을 최소화할 수 있다는 장점이 있다.

현재 컨테이너 Live 마이그레이션 지원을 위해서 CRIU(Checkpoint and Restore in Userspace) 기술이 사용되고 있으며 동작 방식은 그림 3과 같다. CRIU 기술은 동작 중인 컨테이너를 일시 정지한 뒤 현재 상태에 대한 정보를 저장한다. 저장한 파일을 동일 클라우드 내 다른 노드 또는 목적지 엣지 클라우드로 복사하여 새로운 컨테이너에서 복원한다. 상태 파일은 컨테이너 자체보다 용량이 가벼우며, 복원 후 일시 정지된 상태를 이어서 서비스 받을 수 있다. 다른 노드 또는 목적지 엣지 클라우드에는 기존 사용하던 워크로드의 컨테이너 이미지가 있어야 하며 존재하지 않는

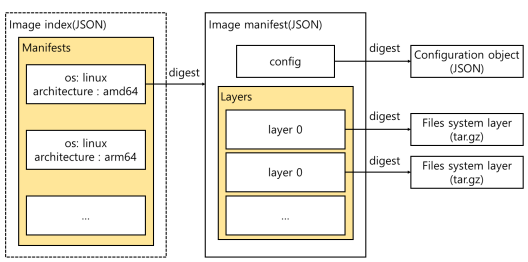

그림 2. 컨테이너 이미지 구조
Fig. 2. Container Image Architecture

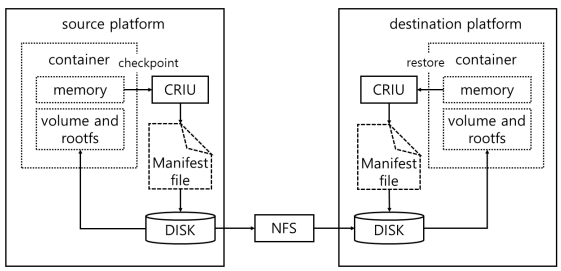

그림 3. CRIU를 이용한 Live Migration
Fig. 3. Live Migration Using CRIU

경우 해당 이미지에 대한 다운로드가 필요하다[18].

컨테이너 이미지의 다운로드는 컨테이너 레지스트리를 통해 제공될 수 있다. 하지만 단일 컨테이너 레지스트리는 다음과 같은 문제점을 가지고 있다. 단일 컨테이너 레지스트리는 네트워크 연결이 취약할 경우 바로 컨테이너 워크로드를 배포하지 못하며 특히 네트워크가 불안한 엣지 클라우드에 적용하기 어렵다. 추가적으로 컨테이너 레지스트리를 구성하는 노드는 다른 노드들에 비해 많은 자원을 가지고 있어야 하는데 전술 엣지 클라우드의 경우 모든 노드가 제약된 자원을 가지고 최적으로 운용되기 때문에 레지스트리를 구성하기 힘들다. 또한 현재의 컨테이너 오케스트레이션의 defacto인 쿠버네티스에서는 모든 노드에 같은 컨테이너 이미지가 중복적으로 저장된다. 따라서 자원이 부족한 전술 엣지 클라우드에서는 이러한 방식이 사용되기 어렵기 때문에 노드 자원에 제약이 있는 전술 엣지 클라우드에서 효과적인 관리 방안이 요구된다.

Ⅲ. 전술 엣지 클라우드 환경에서의 P2P 기반 컨테이너 이미지 및 상태 정보 관리 구조 및 절차

본 논문에서는 전술 엣지 클라우드 환경의 제한적인 네트워크 및 메모리 리소스를 고려하여 컨테이너 워크로드 배포 시 필요한 컨테이너 이미지 및 컨테이너 상태 정보 관리를 위한 P2P 기반 이미지 분산 시스템을 전술 엣지 클라우드에 구성했다. 제안된 시스템은 워크로드 이전 후 워크로드 구동에 필요한 컨테이너 이미지 및 상태 정보를 주변 노드에서 다운로드 함으로써 다운로드 속도를 증가시키고 중복적인 컨테이너 이미지를 미리 다운로드하지 않게 하여 메모리의 효율을 높였다.

그림 4는 본 논문에서 제안한 P2P 기반 컨테이너

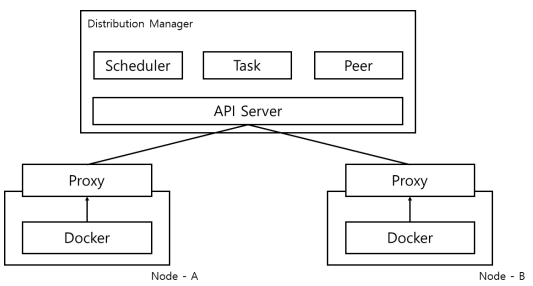

그림 4. 전술 엣지 클라우드에서의 P2P 분산 구조
Fig. 4. P2P Distribution Architecture in Tactical Edge Cloud

이미지 및 상태 정보 관리를 위한 구조이다. 서버 또는 가상머신에서 컨테이너 서비스를 제공하기 위한 컨테이너 런타임을 docker로 구성하였으며 그 위에 P2P 분산 시스템을 구성했다. 또한 현재 컨테이너 워크로드에 대한 상태 정보를 다운로드하고 새롭게 생성된 컨테이너에 상태 정보를 주입할 수 있는 CRIU 기술을 연동했다. P2P 분산 시스템은 아래의 두 컴포넌트로 구성된다.

- Proxy : Proxy는 P2P 네트워크에서 서로 데이터를 주고받을 수 있는 피어의 역할을 수행하고, 각 노드 간 컨테이너 이미지 및 상태 정보를 다운로드하는데 사용되는 클라이언트이다. 본 논문에서 설계된 Proxy는 두 개의 요청을 대신 처리한다. 첫 번째는 컨테이너 이미지 다운로드로써 컨테이너 런타임에서 이미지를 다운로드하는 명령어를 받을 시 Distribution Manager와 협업하여 다른 Proxy에서 이미지들을 다운로드할 수 있도록 peer를 구성하게 되고 각 노드들 간 데이터가 이동된다. 두 번째는 컨테이너 워크로드에 대한 상태 정보를 가져오는 것으로 컨테이너 이미지의 다운로드와 같은 방식으로 다른 노드에서 상태 정보에 대한 레이어들을 다운로드 한다.
- Distribution Manager : Distribution Manager는 전체 P2P 네트워크에서 파일 분산을 위한 역할을 한다. Scheduler는 각 노드 당 최적의 다운로드 경로를 찾는 역할을 수행하며 Peer는 P2P 네트워크를 통해 데이터를 다운로드할 수 있도록 도움을 준다. Task는 다운로드할 데이터 정보에 대한 관리를 수행한다. 또한 Distribution Manager는 다양한 컨테이너 워크로드 요청에 대한 레이어를 검사하며 기본 제공되고 있던 컨테이너 워크로드에 의해 이미 다운로드 된 레이어이면 이 부분을 제외하고 새로운 레이어만 적절한 노드에 저장시킨다.

컨테이너 워크로드의 이전 절차는 컨테이너 생성 및 상태 정보 복구로 나누어진다. 컨테이너 워크로드 생성시 컨테이너 런타임에서는 워크로드 생성에 필요한 이미지를 요청하게 되며 기존 상위 레지스트리가 아닌 매니저에게 이미지를 요청하게 된다. 컨테이너 이미지를 가져오는 절차는 그림 5와 같다. 컨테이너 워크로드 생성 시 docker pull 명령어를 실행하게 되고 Proxy가 이 명령을 가로챈다. 그다음 Proxy는 분산 매니저에 디스패치 요청을 보낸다. Distribution Manager는 요청을 수신한 후에 해당 다운로드 파일이 로컬로 캐시 되었는지 확인한다. 캐시 되지 않은

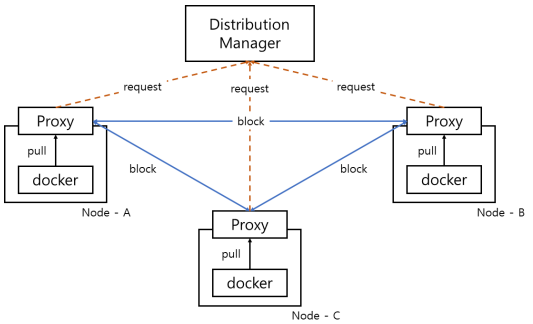

그림 5. 컨테이너 이미지 배포 절차
Fig. 5. Container Image Distribution Flow

경우 레지스트리에서 해당 파일을 다운로드하고 시드 블록 데이터를 생성한다. 이미 캐시 된 경우 블록 작업이 즉시 생성된다. 요청자는 해당 블록 작업을 분석하고 다른 피어 또는 분산 매니저에서 블록 데이터를 다운로드한다. 레이어의 모든 블록 다운로드가 완료되면 레이어가 다운로드 된 것이며 컨테이너가 동작된다.

워크로드를 배포하기 위한 기초 컨테이너가 생성되면 기존에 사용되던 컨테이너의 상태 정보를 생성된 컨테이너에 주입시켜야 한다. 기존에 사용되던 컨테이너가 배포된 노드에 CRIU를 통해 checkpoint 파일이 저장된다. 리눅스의 파일 시스템은 블록 파일 스토리지를 사용하기 때문에 checkpoint 파일 또한 복수 개의 block으로 쪼개지며 이 block들은 P2P 네트워크를 통해 공유된다. 컨테이너 워크로드 마이그레이션 시 필요한 상태 정보 파일을 주변 노드에서 다운로드하게 되며 다운로드 완료 후 CRIU를 통해 복구가 완료되며 마이그레이션 프로세스가 종료된다.

Ⅳ. 구현 및 실험 결과

본 논문에서는 제안한 컨테이너 이미지 및 상태 관리 기능을 구현한 하드웨어 스펙 및 소프트웨어 컴포넌트의 스펙은 표 1과 같다.

베어메탈 위에 오픈스택을 사용하여 가상머신을 생성하였고 해당 가상머신 위에 쿠버네티스를 설치하여 2개의 컨테이너 클러스터를 구성했다. 하나의 클러스터에 3.1장의 제안 구조와 같이 P2P 기반의 네트워크를 구성하여 P2P 기반 분산 관리 구조를 구성하고 다른 클러스터에서는 기본 쿠버네티스 클러스터를 구성하여 본 논문에서 제안한 기법이 기존 쿠버네티스 클러스터 대비 성능이 얼마나 개선되었는지 비교했다.

본 논문에서는 컨테이너 이미지 다운로드 속도 및 저장 용량만 측정을 했고, 제안한 기법의 성능을 측정

표 1. 테스트베드 하드웨어 및 소프트웨어 사양
Table 1. Specification of Testbed

Node	Classification	Specification
Controller/ Compute	CPU	Intel(R) Xeon(R) CPU E5-2697
	Memory	DDR4 2133MHz 16GB * 12
	OS	Ubuntu 18.04.4 LTS
	OpenStack	Victoria
Master/ Worker	Docker	20.10.7
	Kubernetes	v1.23.5

하기 전 몇 가지 가정했다. 컨테이너 이미지 저장의 경우 전술 엣지 클라우드에 모든 서비스들에 대한 컨테이너 이미지 데이터를 클러스터에 구성해 두어야 한다고 가정했다.

기존 쿠버네티스 방식에서 전술 엣지 클라우드에 배포되는 모든 서비스에 대한 이미지를 미리 저장시키고 있으려면 모든 노드에 이미지가 저장되거나 하나의 레지스트리 서버가 존재하는 경우이다. 하지만 전술 엣지 클라우드 구조에서 하나의 노드가 큰 리소스를 가지고 있기 어렵기 때문에 모든 이미지가 모든 노드에 저장되어야 한다. 하나의 컨테이너 이미지의 데이터 크기를 A라고 하고, 전술 엣지 클라우드를 구성하는 모든 노드들의 수 n이라고 했을 때 한 노드에 저장되는 이미지는 A의 크기를 갖고 모든 노드에 저장된다. 하지만 본 논문에서 제안한 시스템에서는 컨테이너를 구동하기 위한 이미지들을 분산 저장하여 준비하기 때문에 초기 전술 엣지 클라우드를 구성하는데 필요한 노드들의 수에 나누어 저장할 수 있으므로 한 노드에 저장되는 데이터는 A/n 만큼 저장되며 모든 노드들에 나누어져 저장된 크기가 A만큼 저장된

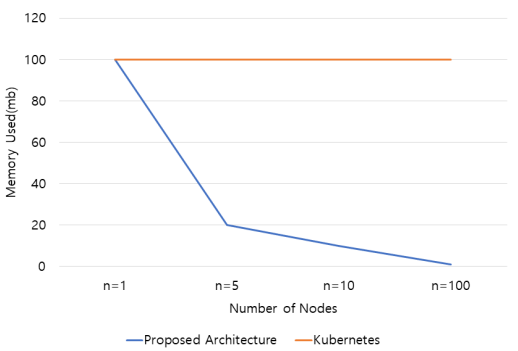

그림 6. 노드 당 컨테이어 이미지를 저장하는데 사용되는 메모리
Fig. 6. Used Memory for Store Container Image per Nodes

다. 단일 노드에 저장되는 데이터의 크기를 도표화하여 그림 6에서 표현하였으며 주황색 그래프는 기존 쿠버네티스 이미지 저장 방식을 사용하여 측정한 결과이고 파란색 그래프는 제안한 시스템을 사용하여 이미지를 저장한 방식이다. 노드의 수가 늘어날수록 초기 전술 엣지 클라우드를 구성할 때 한 노드에 필요한 이미지 저장 용량이 줄어드는 것을 볼 수 있다. 추가적으로 컨테이너를 구동할 시에는 각 노드에 컨테이너를 구동하기 위한 모든 block들이 다운로드 되지만 다른 노드에 있던 block들을 삭제하여 관리할 수 있기 때문에 기존 쿠버네티스의 이미지 저장 방식보다 리소스를 절약할 수 있다.

컨테이너 이미지 다운로드 속도를 측정하기 위해서 “time docker pull” 명령어를 사용했다. 컨테이너 이미지 시간 측정을 위해 한 개의 노드에 컨테이너 이미지를 미리 다운로드해두었으며 그 후 다른 노드에서 컨테이너 이미지를 다운로드하는 시간을 측정했다. 컨테이너 이미지 다운로드 완료까지 소요되는 시간을 확인하였으며 결과는 그림 7과 같다. 또한, 이미지 용량 별로 컨테이너 이미지를 다운로드하는 시간을 비교하기 위해서 여러 개의 용량에 대한 다운로드 속도를 측정했다. 주황색 그래프는 기존 쿠버네티스 이미지 저장 방식을 사용하여 측정한 결과이고 파란색 그래프는 제안한 시스템을 사용하여 이미지를 저장한 방식이다. 기존 쿠버네티스 방식은 다른 노드에 컨테이너 이미지 유무를 보지 않고 상위 계층에서 다운로드를 요청하기 때문에 다른 노드들과 차이가 없이 일정한 속도를 보인다. 하지만 본 논문에서 제안한 방식에서는 다른 노드에 이미지가 다운로드 되어 있으면 상위 계층에서 가져오는 방식이 아닌 다른 노드와의 연결을 통해 컨테이너 이미지를 다운로드하기 때문에 상위 계층 대비 네트워크 상태가 원활하므로 빠른 다운로드 속도를 나타내는 것을 볼 수 있다. 본 논문에서는 단일 클러스터 환경을 대상으로 컨테이너 이미지 다운로드 속도 및 저장 방식에 대해서만 실험을 진행하였지만 엣지 클라우드 환경에서 또한 중앙 집중형 레지스트리를 구성하여 운영되기 때문에 해당 레지스트리로 접근해야 하기 때문에 다운로드 시간이 주변 노드에서 이미지를 가져오는 방식인 본 논문에 비해 오래 걸릴 것으로 예상된다.

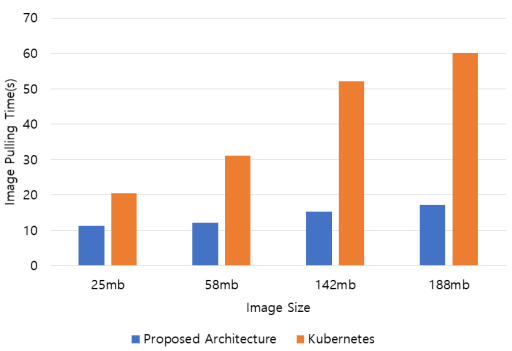

그림 7. 컨테이너 이미지 다운로드 시간
Fig. 7. Image Pulling Time

V. 결 론

본 논문에서는 전술 엣지 클라우드 환경의 제한적인 리소스를 고려하여 서비스 배포 시 필요한 컨테이너 이미지 및 Stateful 한 서비스의 마이그레이션 기능 제공을 위해 상태 정보를 효율적으로 관리하는 구조 및 절차를 제안했다. 제안하는 기법은 전술 엣지 클라우드를 구성하는 각 노드 간의 P2P 네트워크를 구성하여 서비스 실행에 필요한 이미지 정보 및 상태 정보를 모든 노드에 저장하지 않고 분산하여 저장함으로써 리소스를 효율적으로 사용하도록 했다. 추가적으로 컨테이너 생성에 필요한 이미지를 상위 계층에서 가져오는 것이 아니라 주변 노드에서 다운로드하여 이미지 다운로드 속도 또한 증가시켰다. 정량적인 성능 분석을 통해, 제안 기법에서의 컨테이너 이미지 및 상태 저장 방안은 기존 중앙 집중화된 컨테이너 레지스트리 및 네트워크 파일 시스템에 비해 다운로드 속도 측면 및 이미지 저장 측면에서 성능을 향상시켜 이미지 관리 및 상태 관리에 기여할 수 있음을 보였다.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” in *IEEE Internet of Things J.*, vol. 3, no. 5, pp. 637-646, Oct. 2016.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” in *IEEE Commun. Surv. & Tuts.*, vol. 19, no. 4, pp. 2322-2358, Fourthquarter 2017.

[3] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, “Edge cloud offloading algorithms: Issues, methods, and perspectives,” *ACM Comput. Surv.(ACUR)*, vol. 52, no. 1, Feb. 2019.

[4] ETSI GS MEC 003, *Multi-access Edge Computing (MEC)*; Framework and Reference Architecture, 2019.

[5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surv. Tuts.*, vol. 19, no. 3, pp. 1657-1681, May 2017.

[6] X.-T. Vu, M.-N. Tran, and Y.-H. Kim, "Service migration over edge computing infrastructure," in *Proc. Symp. KICS*, pp. 138-139, 2021.

[7] J. W. Lee and Y. H. Kim "A design for stateful migration of CNF in VNFM," in *Proc. Symp. KICS*, pp. 155-156, 2021.

[8] S. Das, M. Saraf, V. Jagadeesh, M. Amardeep, and H. L. Phalachandra, "Deduplication of docker image registry," *2021 IEEE MASCON*, pp. 1-8, 2021.

[9] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757-6779, Mar. 2017.

[10] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. and Mob. Comput.*, vol. 13, no. 18, pp. 1587-1611, Oct. 2011.

[11] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "*Mobile edge computing—A key technology towards 5G*," ETSI White Paper, vol. 11, no. 11, pp. 1-16, Sep. 2015.

[12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc First Edition of the MCC Wkshp. Mob. Cloud Comput.*, pp. 13-16, Aug. 2012.

[13] M. Satyanarayananm, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14-23, Dec. 2009.

[14] K. Sun and Y. H. Kim, "VM migration with IP mobility management for tactical edge cloud," *The J. Korean Inst. Next Generation Comput.*, vol. 15, no. 3, pp. 50-66, 2019.

[15] K. Sun and Y. H. Kim, "LISP-based service mobility management with improving survivability in the tactical edge cloud environment," *The J. Korean Inst. Next Generation Comput.*, vol. 17, no. 2, pp. 53-68, 2021.

[17] N. Zhao, et al., "Large-scale analysis of docker images and performance implications for container storage systems," in *IEEE Trans. Parall. and Distrib. Syst.*, vol. 32, no. 4, pp. 918-930, Apr. 2021.

[18] S. Oh and J. Kim, "Stateful container migration employing checkpoint-based restoration for orchestrated container clusters," *2018 ICTC*, pp. 25-30, 2018.

임 호 근 (Ho-keun Lim)

2018년 8월 : 숭실대학교 전자정보공학부 졸업
2021년 2월 : 숭실대학교 정보통신융합학과 석사
2021년 3월~현재 : 숭실대학교 정보통신공학과 박사과정

<관심분야> 모바일 네트워크, SDN/NFV, Edge Computing
[ORCID:0000-0002-1623-6873]

김 영 한 (Young-han Kim)

1986년 2월 : 한국과학기술원 전기 및 전자 공학과 석사
1990년 2월 : 한국과학기술원 전기 및 전자 공학과 박사
1994년~현재 : 숭실대학교 전자정보공학부 교수

<관심분야> 모바일 네트워크, 이동성 관리 기술, SDN/NFV, 센서 네트워크
[ORCID:0000-0002-1066-4818]