JDCS
디지털콘텐츠학회논문지
Journal of Digital Contents Society
Vol. 20, No. 12, pp. 2525-2533, Dec. 2019

가상화 플랫폼을 통한 CNN기반 모니터링 애플리케이션의 안정적인 응답 속도 보장 방안 연구

오 승 민 · ASHIQUZZAMAN AKM · 이 동 수 · 김 진 술* · 박 재 형
전남대학교 전자컴퓨터공학부

A Study on Stable Response of the CNN-Based Monitoring Applications Through Virtualized Platform

Seungmin Oh · ASHIQUZZAMAN AKM · Dongsu Lee · Jinsul Kim* · JaeHyung Park
School of Electronics and Engineering, Chonnam National University, Gwangju, Korea

[요 약]

최근 가상화 기술이 적용된 가상화 플랫폼(Virtualized Platform)을 도입하게 되면서 단일 하드웨어 리소스의 파티셔닝을 통한 리소스 활용률 상승과 마이그레이션을 통한 확장성의 이점을 통해 서비스의 안정적인 응답 속도를 기대할 수 있게 되었다. 기존 단일 하드웨어 서버기반 모니터링 애플리케이션 서비스에서는 필요 이상의 리소스를 사용하거나 사용자의 요청에 비해 리소스가 부족하여 응답 속도가 저하되는 문제가 발생하였다. 본 논문에서는 이를 해결하기 위해 오픈 소스 가상화 플랫폼인 OpenBaton, OpenStack과 Docker를 통해 이미지에 대해 화재 및 연기 예측이 가능한 ResNet50 기반의 CNN 모델이 적용된 모니터링 애플리케이션을 구현하였다. 이를 통해 본 논문에서는 기존 단일 하드웨어 서버와 제안된 시스템의 응답 속도 비교를 통해 안정적인 응답 속도 보장 방안에 대해 연구하였다.

[Abstract]

With the recent introduction of a virtualized platform with virtualization technology, the benefits of increased resource utilization through partitioning of a single hardware resource and scalability through migration provide a reliable response rate for services. Traditional single hardware server-based monitoring application services have had problems with using more resources than needed or lack of resources compared to the user's request, resulting in slower response times. To address this, this paper implemented a monitoring application with a CNN model based on ResNet50 that enables fire and smoke prediction for images through open source virtualization platforms, OpenBaton, OpenStack and Docker. In this paper, we studied how to ensure a stable response rate through comparing the response speed of the existing single hardware server with the proposed system.

색인어 : 가상화 플랫폼, 클라우드 컴퓨팅, OpenStack, Docker, IaaS, CNN, 모니터링
Key word : Virtualized Platform, Cloud Computing, OpenStack, Docker, IaaS, CNN, Monitoring

http://dx.doi.org/10.9728/dcs.2019.20.12.2525

Received 23 October 2019; **Revised** 20 November 2019
Accepted 15 December 2019

***Corresponding Author; Jinsul Kim**
Tel: ※ 개인정보 표시제한
E-mail: osm5252kr@gmail.com

Ⅰ. 서 론

과거에 대부분의 기업에서는 단일 하드웨어 서버에 단일 벤더 서비스만을 운영하고 있었기 때문에 다른 벤더의 하드웨어 서버에서는 레거시 애플리케이션을 실행할 수 없었다. 이 때문에 다양한 벤더가 제공하는 저렴한 상용 서버, 운영 체제, 애플리케이션은 IT 환경을 업데이트하면서 점점 자원 활용률이 낮은 단일 하드웨어 서버에 종속되었으나 가상화 기술을 도입을 통해 하드웨어 서버의 파티셔닝이 가능하였고 여러 운영체제 및 적절한 리소스(CPU, Memory 등)에서 레거시 애플리케이션을 실행할 수 있게 되었다. 기업은 서버를 다양한 환경과 효율적 리소스 사용할 수 있어 냉각 및 유지관리 비용이 절약되었고 클라우드 컴퓨팅의 기반을 다지는 데 도움이 되었다. 최근에는 시스템을 가상화 환경에서 관리하기 위한 전문적인 소프트웨어인 가상화 플랫폼을 활용되고 있다. 여기서 가상화 플랫폼이란 클라우드 컴퓨팅이 확산되면서 관심을 끌고 있는 기술 중 하나로 컴퓨터의 리소스를 파티셔닝을 하여 구분되어 사용할 수 있게 함으로써 최종 사용자들이 여러 리소스와 상호작용하는 방식으로 이용할 수 있는 기술이다[1],[2].

가상화 기술 중 서버 가상화는 하나의 컴퓨터에서 여러 대의 운영 시스템을 구동할 수 있으며 하이퍼바이저(Hypervisor)에 의해 운영/관리된다. 서버 가상화의 작동 원리로는 하이퍼바이저 소프트웨어가 필요로 하는 가상 머신(Virtual Machine)에 따라 하드웨어의 물리 리소스를 분리하여 제공한다. 하이퍼바이저는 노트북, 데스크톱 등의 운영체제에 배포하거나 서버 등의 하드웨어에 직접 설치할 수 있다. 이 때 하이퍼바이저는 필요에 따라 물리 리소스를 분할하여 가상 머신에서 사용할 수 있도록 파티셔닝 된 후 사용자가 가상 머신과 상호 작용하며 가상 머신 내에서 컴퓨팅을 실행한다. 최근에는 클라우드 컴퓨팅 환경을 구축하는 데에 있어서 시스템 환경에 대한 의존성을 없애고 경량화를 통한 속도 및 이식성 향상을 추구하는 컨테이너(Container) 기반의 오픈소스 가상화 플랫폼 기술인 Docker가 등장하였다. Docker는 서버의 운영체제, 애플리케이션들의 버전과 같은 환경을 컨테이너에 담아 사용자가 필요할 때 사용할 수 있다는 장점이 있고 두 개 이상의 컨테이너는 서로 별도의 공간을 가짐으로써 다른 컨테이너와 상호 간의 영향을 받지 않는다[3]-[5].

우리는 이러한 가상화 플랫폼의 효율적인 리소스 분배와 할당을 통해 안정적인 서비스가 필요한 모니터링 시스템을 구현하여 기존 단일 하드웨어에서 제공하는 것보다 제안된 방안을 사용하는 것이 안정적인 응답 속도 제공할 수 있다는 것을 기대하였다. 따라서, 본 논문에서는 가상화 플랫폼 기술(OpenBaton, OpenStack, Docker)과 ResNet50기반의 CNN 모델이 적용된 화재 및 연기 감지 모니터링 시스템을 구축하였고 이를 통해 컴퓨팅 리소스(CPU, Memory 등)를 효율적으로 사용하는 가상화 플랫폼의 이점과 CNN기반 화재 및 연기 감지 모니터링 시스템을 제공하는 기존 단일 하드웨어 서버와 비교하여 안정적인 응답 속도를 보장하는 방안에 대해 연구하였다. 본 논문의 나머지 부분은 다음과 같이 정리되어 있다. 2장은 이론적 배경 및 관련 연구에 관해 기술하였고 3장은 가상화 플랫폼과 모니터링 시스템을 이용한 연구 제안에 관해 설명하였다. 4장은 실험 및 결과를 설명하였으며 이어서 5장의 결론에 관해 설명하였다[6],[7].

Ⅱ. 이론적 배경 및 관련 연구

2-1 가상화(Virtualization)

가상화는 물리적인 컴퓨터의 리소스를 논리적인 객체로 추상화시키는 것을 일컫는 용어이다. 컴퓨터의 하드웨어 리소스를 가상화를 통해 분리하여 여러 개의 가상 머신으로 나누어 사용할 수 있게 한다. 가상화를 사용하게 되면 컴퓨터의 리소스를 보다 효율적으로 사용할 수 있고 하드웨어 장비를 하나로 묶거나 하나의 하드웨어 장비를 여러 개의 머신인 것처럼 동작시키는 것도 가능하다. 가상화가 되는 대상은 주로 프로세서(CPU), 메모리(Memory), 스토리지(Storage), 네트워크(Network)가 되는데 이러한 대상들로 구성된 서버나 장치들을 가상화함으로써 높은 수준의 리소스 관리과 분산 처리를 가능하게 된다. 또한, 한 대의 머신을 여러 대의 머신처럼 사용할 수 있으므로 필요한 서버의 수도 줄어들 뿐만 아니라 보유하고 있는 서버를 최대한 활용할 수 있다. 이처럼 가상화는 클라우드 컴퓨팅을 가능하게 하는 주요 핵심 기술 중 하나로 클라우드 공급업체의 경우 가상화를 사용하면 하나의 서버에서 여러 고객에게 가상 환경의 클라우드 서비스를 제공할 수 있으므로 많은 기업들이 가상화 및 클라우드 컴퓨팅을 사용하여 기업 내 시스템의 효율성을 높이고 있는 추세이다.

2-2 하이퍼바이저(Hypervisor)와 컨테이너(Container)

가상 머신의 속성은 사용자의 기본 하드웨어와 관계없이 완전한 기능을 갖춘 시스템과 같은 독립적인 시스템을 제공한다. 가상 머신 기술은 언제든지 작업 프로세스를 인스턴스화하고 종료할 수 있을 정도의 세부적인 제어가 가능하며 이러한 방식으로 리소스 프로비저닝(Resource Provisioning)을 유연하게 수행할 수 있도록 한다. 그러나 가상 머신은 추상화된 물리적 하드웨어로 애플리케이션 및 서비스를 호스팅하기 위한 전체 게스트 운영체제(OS; Operation System) 이미지와 추가 바이너리 및 라이브러리가 필요하고 가상 머신은 해당 서비스가 필요하지 않으면 전체 운영체제의 부팅으로 인해 느린 실행시간과 많은 양의 리소스를 비효율적으로 사용하게 된다는 문제점이 있다. 반면 컨테이너는 가상 머신보다 작은 크기로 인스턴스를 생성하기 때문에 밀리 초 단위로 인스턴스화가 쉽고 마이그레이션이 빠르며 CPU, 메모리, 디스크 및 네트워킹에서 기본 성능으로 빠르게 배포할 수 있다는 장점이 있다.

2-3 도커(Docker)

Docker는 응용 프로그램들을 컨테이너 안에 배치시켜 애플리케이션을 구축, 테스트 및 배포할 수 있어 가상 머신의 대안으로 사용되는 오픈소스 가상화 플랫폼이다. Docker는 소프트웨어를 컨테이너라는 표준화된 유닛으로 패키징하며 컨테이너에는 라이브러리, 시스템 도구, 코드, 런타임 등 소프트웨어를 실행하는 데 필요한 모든 것이 포함되어 있다. Docker를 사용하면 코드를 더 빨리 전달하고 애플리케이션 운영을 표준화할 수 있으며 마이그레이션을 원활하게 하여 리소스 활용률을 높여 기존 방식보다 애플리케이션 운영 비용을 절감할 수 있다. 가상 머신과의 명확한 차이로는 가상 머신은 서버 하드웨어를 가상화하지만 컨테이너 개념을 사용하는 Docker는 서버의 운영 체제를 가상화한다는 특징이 있다[8].

2-4 오픈스택(OpenStack)

클라우드 컴퓨팅의 부상과 함께 IaaS(Infrastructure as a Service) 형태의 서비스로 활발하게 운영되고 있는 OpenStack은 현재 CEF digital의 BDTI(Big Data Test Infrastructure), TM Forum 등에서 채택하여 사용하고 있는 클라우드를 구축할 수 있는 소프트웨어 플랫폼이다. OpenStack은 표준 하드웨어에서 운용할 수 있는 모듈형 클라우드 인프라를 제공하여 단일 위치에서 필요한 모든 툴을 필요한 시기에 배포할 수 있다. OpenStack의 아키텍처는 다양한 코드 이름을 가지고 있는 모듈 방식의 오픈소스 프로젝트로 이루어져 있다. 컴퓨팅, 네트워킹, 스토리지, ID, 이미지를 처리하는 6가지의 핵심 서비스인 Nova, Neutron, Swift, Cinder, Keystone, Glance와 그 외에도 Horizon, Heat, Mistral 등의 서비스 등이 있다.

OpenStack은 그림 1과 같이 구성되어 있으며 앞서 설명한 핵심 서비스들에 의해 다양한 구성으로 리소스를 나누어 사용할 수 있다. 논문[9]에서 수행된 연구는 OpenStack을 기반으로 한 클라우드 서비스에서 위성영상정보 분석처리 서비스를 설계 및 구축하였다. 그러나 Docker나 OpenBaton을 통한 OpenStack 관리를 통한 성능 향상 방안에 대한 내용은 언급되지 않아 본 논문에서 제안된 시스템과의 차이가 있다.

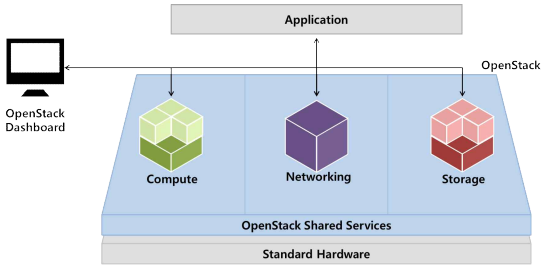


그림 1. 오픈스택 구조
Fig. 1. OpenStack Architecture

2-5 오픈바톤(OpenBaton)

OpenBaton은 특정 클라우드 환경에 네트워크 기능을 추가하여 가상 네트워크 인프라의 개발을 보장하는 기술이다[10]. 기본 인프라, 소프트웨어 아키텍처, 네트워킹, 관리 및 오케스트레이션을 병합하여 전체 인프라의 성능 향상과 보안에 중점을 둔다. OpenBaton의 프레임워크는 VNFM(Virtual Network Functions Manager) 어댑터를 기반으로 VNF(Virtual Network Functions)의 라이프사이클 관리한다. OpenBaton은 분산된 이벤트의 관리와 스케일링 관리를 위한 자동 스케일링을 통합하여 운영한다. OpenBaton에는 네트워크 하드웨어 관리 시스템인 Zabbix를 사용하여 모니터링 정보를 수집하여 자동 런타임 관리를 하며 오케스트레이션의 로직 내에서 추가 및 삭제를 할 수 있는 플러그인도 제공한다. OpenBaton은 최초의 Point of Presence로 그림 2와 같이 OpenStack을 컨트롤 할 수 있다.

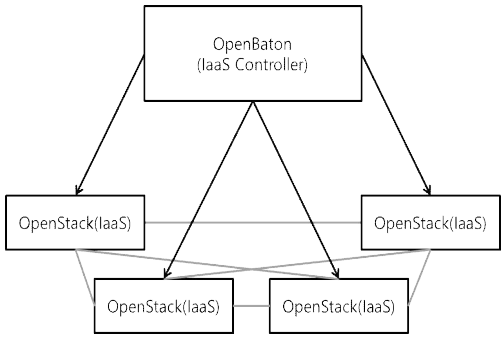


그림 2. 오픈바톤 구조
Fig. 2. OpenBaton Architecture

2-6 컨볼루션 뉴럴 네트워크(Convolution Neural Network, CNN)

CNN은 Convolutional Neural Network의 약자로서, 학습 가능한 가중치(Weight)와 바이어스(Bias)로 구성되어 있다. CNN은 일반 신경망과 달리 입력 데이터로 이미지를 주로 사용하기 때문에 이미지가 갖는 특성들을 인코딩할 수

있다. CNN 모델은 이미지, 비디오, 텍스트 또는 음향을 분류하는 딥러닝에 가장 많이 사용되며 이미지에서 객체, 얼굴, 장면을 인식하기 위한 패턴을 찾는 데 유용하다. 그림 3은 CNN의 가장 기본적인 모델 구조를 보여주고 있으며 입력 데이터로부터 필터 역할을 해주는 컨볼루션 연산을 통해 특징맵을 만들고 특징맵에 서브샘플링과정을 거쳐 로컬 특징을 얻어낸다. 얻어진 로컬 특징에 컨볼루션과 서브샘플링을 통해 전체적인 특징을 얻게 되고 이러한 특징들을 가지고 Fully Connected Network의 입력으로 연결되어 최종적으로 출력을 통해 이미지 등을 분류하게 된다. 논문[11]에서 수행된 연구는 CCTV 카메라, 영상화재수신기 등에서 수집된 영상을 기반으로 자동화재감지 알고리즘을 통해 화재 영상 분석 프로그램의 성능 향상에 대한 논문으로 가우시안 혼합모델 해석, 화재 후보영역 해석, 입력된 이미지로부터 연기 및 화염의 특징과 움직임 특징 추출 등을 통한 화재 및 연기를 감지하는 모델의 성능을 향상시켰고 이러한 특징들을 추출하여 화재를 감지할 경우 더 좋은 성능을 기대할 수 있음을 알 수 있었지만 딥러닝과 가상화 플랫폼을 결합한 기술에 관한 내용은 언급되지 않았다. 논문[12]에서 수행된 연구는 CNN을 이용하여 기존의 컬러 모델을 이용한 화재 감지 방법에서의 오경보 발생의 가능성이나 사람이 직접 정의한 화염의 특성을 이용해서 화염 발생 여부를 감지할 경우에 발생했었던 문제를 해결하였다. 하지만 OpenBaton, OpenStack, Docker 등의 가상화 플랫폼과의 결합이나 제시되어 있는 CNN은 본 논문에서 제시한 ResNet50기반 CNN모델보다 낮은 예측 정확도를 가지는 것을 알 수 있었다.

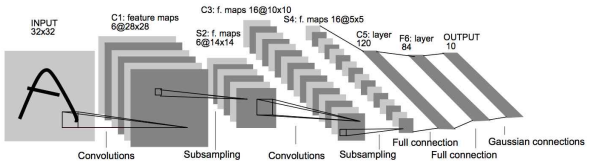


그림 3. 기본적인 CNN 구조
Fig. 3. Basic CNN Architecture

Ⅲ. 가상화 플랫폼과 모니터링 애플리케이션을 이용한 안정적인 응답 속도 보장 방안 연구

3-1 프레임워크 아키텍처 개요

본 논문에서 제안한 가상화 플랫폼을 CNN기반 화재 및 연기 감지 모니터링 시스템은 몇 가지 구조로 구성되어 있다. 그림 4는 본 논문에서 제안하는 가상화 플랫폼 프레임워크 아키텍처이다. 하드웨어 위에 가상 머신을 생성 및 관리 할 수 있는 OpenStack을 구성하였고 OpenBaton으로 OpenStack을 제어할 수 있게 하였다. OpenStack은 가상 머신 인스턴스를 생성하며 각 가상 머신에는 Docker를 이용하여 애플리케이션을 구동할 수 있는 환경을 빠르게 구성할 수 있도록 하였다. 각 가상 머신의 인스턴스는 웹서버(WebServer)의 역할과 클라이언트(Client)의 역할을 수행할 수 있도록 구성하였다. 앞서 제안된 가상화 플랫폼 모니터링 시스템에 대하여 전체적으로 상세히 설명하였다.

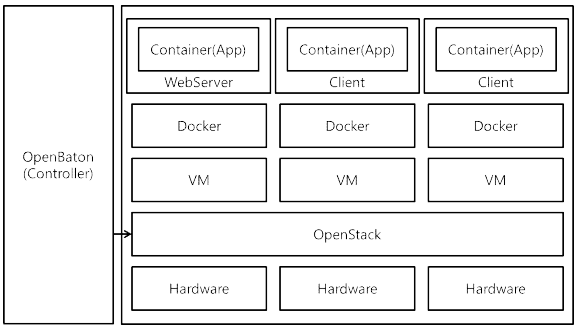


그림 4. 제안한 프레임워크 아키텍처 구조 개요
Fig. 4. Proposed Framework Architecture Overview

3-2 가상화 플랫폼과 모니터링 애플리케이션 시스템

전술한 프레임워크 구조를 통해 본 논문에서 제안하는 시스템과 응답 속도 보장하기 위한 전체적인 흐름은 그림 5와 그림 6의 순서도로 나타낼 수 있다. 그림 5는 클라이언트 역할을 하는 가상 머신이 웹서버의 웹사이트에 이미지를 업로드하여 해당 이미지의 화재 및 연기를 감지하는 작업을 요청한다. 업로드된 이미지는 224*224*3 이미지로 전처리되어 ResNet50기반 CNN 모델을 통해 해당 이미지가 화재 및 연기 상황인지를 예측하고 해당 이미지가 화재나 연기 상황으로 예측되는 확률이 95% 이상이라고 판단될 시 웹서버를 통해 웹사이트에 해당 이미지를 전송하여 사용자는 이를 확인할 수 있는 모니터링 애플리케이션 시스템을 구현하였다.

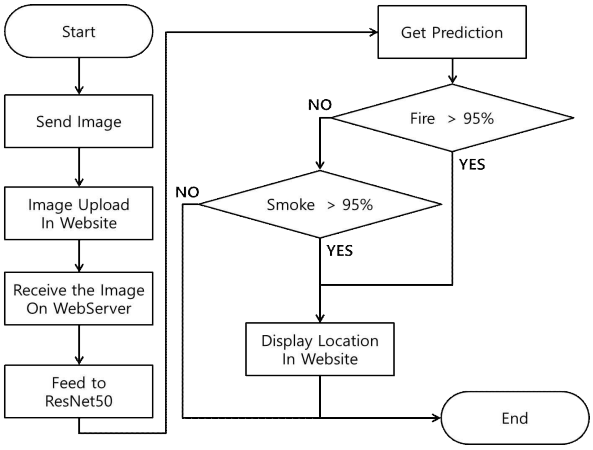


그림 5. 제안된 화재/연기 감지 흐름도
Fig. 5. Proposed Fire/Smoke Detection Flow

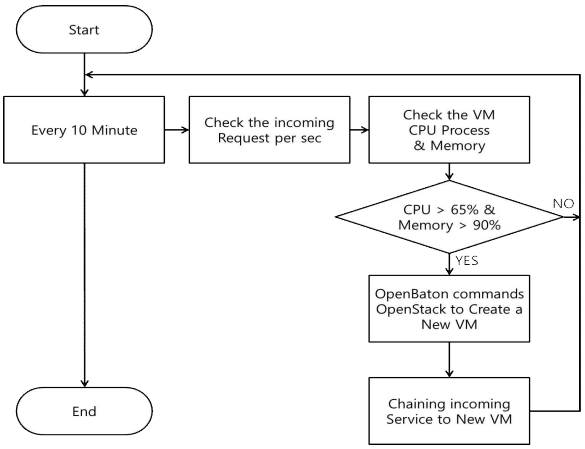


그림 6. 리소스 활용률에 따른 오픈바톤의 오픈스택 가상 머신 제어
Fig. 6. OpenBaton Controls the VMs of OpenStack According to Resource Utilization

3-3 응답 속도 보장을 위한 자원 할당 시스템 연구

그림 6은 클라이언트가 웹사이트에 화재 및 연기 감지를 위한 이미지를 업로드할 때 이를 처리하기 위한 웹서버 가상 머신은 CPU, Memory 등의 컴퓨팅 자원을 사용하여 요청을 처리한다. 이때 OpenBaton은 해당 OpenStack의 가상 머신의 리소스 중 Memory 사용률이 90% 이상이고 CPU 이용률이 65% 이상일 때 OpenBaton은 OpenStack에게 새로운 리소스를 할당하며 새로운 가상 머신을 생성을 명령한다. 가상 머신이 새로 생성되면 이를 웹서버로 운영할 수 있도록 새로운 가상 머신을 연동하여 새로운 웹서버 역할을 수행할 수 있도록 하였다. 이를 통해 그림 7과 같이 가상 머신의 리소스 사용량를 확인하여 OpenBaton을 통한 OpenStack 컨트롤로 가상 머신의 리소스 활용률을 향상시킬 수 있고 이를 통하여 기존 단일 하드웨어 서버를 운영하는 것보다 더 안정적인 응답 속도를 보장할 수 있었다[13].

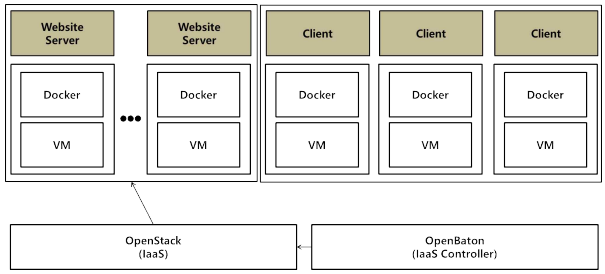


그림 7. 오픈바톤과 오픈스택에 의한 웹서버의 증가
Fig. 7. Increase of WebServer in VMs by OpenStack and OpenBaton

Ⅳ. 실험 및 결과

4-1 화재 및 연기 감지를 위한 데이터 세트

화재 및 연기 감지 모델을 위한 주요 데이터 세트는 화재 비디오, 연기 발생 비디오, 일반 비디오 3종류로 이루어져 있으며 화재 비디오는 13개, 연기 발생 비디오는 12개, 일반 비디오는 12개로 이루어져 있다. 각 비디오의 크기는 224 * 224 * 3의 크기를 가지고 있고 각 비디오의 프레임 이미지를 데이터 학습에 사용하였다. 각 비디오에 대해 화재 및 연기 데이터로 분류하고 5종류의 Augment를 통한 총 37,000개의 이미지 데이터 세트로 구성되어 있으며 화재 이미지, 비화재 이미지, 연기 이미지, 비연기 이미지 총 4개로 각 7,600개, 14,900개, 7,628개, 7,145개의 데이터로 이루어지게 되며 이를 학습하여 기존 ResNet50 신경망에 화재 및 연기 감지 학습을 함으로써 새로운 모델 제시하였다.

4-2 CNN기반 신경망과 감지 예측 정확도

화재 및 연기 감지를 위한 CNN 모델은 Inception v3, VGG16, ResNet50 3가지 모델로 구현하여 전술한 데이터를 통해 학습하였다. Inception v3, VGG는 ResNet50보다 떨어지는 성능을 보여 본 논문에서는 ResNet50 구조 모델을 사용하여 학습하였다. 그림 8은 본 논문에서 사용한 CNN 모델인 ResNet50의 구조이다[14]. ResNet은 기존의 CNN모델에서의 컨볼루션 레이어과 활성화 함수를 통해 진행하는 과정에서 레이어의 입력을 레이어의 출력에 바로 연결 시키는 Skip Connection을 사용하여 성능을 향상 시킨 모델이다. 학습된 모델에 224 * 224 * 3 사이즈로 전처리한 이미지 입력을 통해 최종적으로 화재감지, 연기감지, 감지없음에 대한 3종류의 값을 출력함으로써 화재 및 연기 상황에 대하여 감지 및 예측하는 모델을 구현하였다.

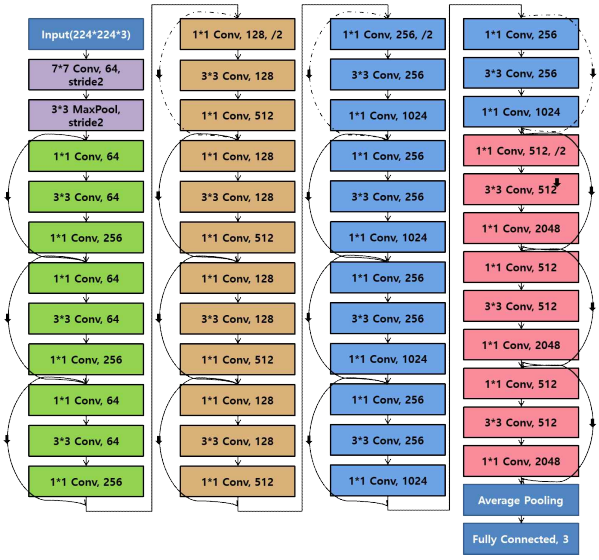


그림 8. Resnet50기반 CNN 구조
Fig. 8. Resnet50 based CNN Architecture

학습한 ResNet50 모델 신경망의 배치 크기는 20, 각 Epoch은 5가지의 Augment를 적용하여 학습에 사용하였고 총 100 epoch으로 학습시켰다. CNN의 Loss 그래프는 그림 9과 같이 도출하였다. x축은 Epoch, y축은 loss를 나타내었으며 20 Epoch에서 0.2 이하의 손실을 보여주었다. 그래프 중 전반적으로 아래쪽에 위치하는 붉은색 그래프는 Testing 데이터 세트로부터의 Loss, 위쪽에 위치하는 푸른색 그래프는 Training 데이터 세트의 Loss를 나타내었고 Testing 데이터 세트보다 적은 손실을 보여주었다.

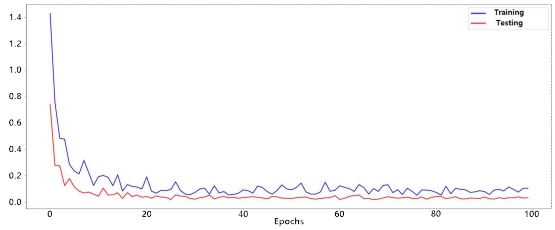


그림 9. 학습 중 표현된 CNN 손실 그래프
Fig. 9. CNN Loss Graph during Training

또한, 학습 모델에 대한 정확도 그래프는 그림 10으로 도출하였다.

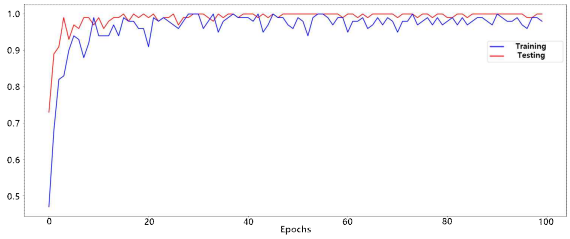


그림 10. 학습 중 표현된 CNN 정확도
Fig. 10. CNN Accuracy graph during training

x축은 Epoch, y축은 Accuracy를 나타내었으며 20 Epoch에서 97% 정도의 정확도를 보여주었다. 그래프 중 전반적으로 위쪽에 위치하는 붉은색 그래프는 Testing 데이터 세트로부터의 Accuracy, 아래쪽에 위치하는 푸른색 그래프는 Training 데이터 세트의 Accuracy를 나타내었고 Testing 데이터 세트보다 높은 정확도를 보여주었다. 따라서, 신경망의 정확도는 데이터 세트와 제안된 ResNet 모델에 따라 입력받은 이미지에 대한 화재 및 연기 예측에 대하여 98%의 정확도를 가지고 있음을 확인 할 수 있었다.

4-2 모니터링 시스템을 위한 가상화 플랫폼의 중요성

기존의 모니터링 시스템들은 주로 하나의 하드웨어나 서버를 통해 운영되어 왔으나 간혹 사용자들의 요청 과부화 등의 이유로 갑작스런 데이터의 양이 폭주하게 되면 급박한 상황에 해당하는 요청에 대해 처리를 하지 못하는 경우 중대한 정보를 놓치거나 서버가 다운되는 문제가 발생하기도 하였다. 또한, 필요 이상의 서버 리소스를 사용하여 리소스를 낭비하거나 요청에 비해 리소스가 부족하여 응답 속도가 저하되는 상황이 발생할 수도 있었다[15]. 모니터링 시스템은 지속적이고 안정적인 운영이 필수적이기 때문에 본 논문에서 제안한 가상화 플랫폼(OpenBaton, OpenStack, Docker)을 이용하면 이러한 단점을 해결할 수 있는 적절한 방안이 된다. 따라서 본 논문에서의 실험을 위해 클라이언트 역할을 하는 가상 머신이 지속적으로 웹서버에게 이미지를 전송하게 하고 웹서버는 클라이언트가 전송한 이미지에 대하여 화재 및 연기를 예측하는 모델을 지속적으로 수행하였으며 웹서버는 이러한 작업을 수행하며 리소스가 부족하게 되면 OpenBaton은 이를 판단하여 OpenStack의 가상 머신 추가 생성하고 생성된 가상 머신을 웹서버로 사용할 수 있도록 하여 사용자의 요청을 처리할 수 있도록 하였다. 이러한 가상화 플랫폼을 통한 모니터링 시스템은 기존 단일 하드웨어 서버를 이용하는 방식보다 빠르고 더 안정적인 응답 속도의 보장할 수 있게 되었다. 그림 11은 지속되는 사용자의 요청에 따른 기존 단일 하드웨어 서버와 가상화 플랫폼 기반 서버의 응답 속도 비교를 그래프로 나타내었다. x축은 실제시간, y축은 숫자 단위로서 제일 아래 그래프부터 가상화 플랫폼을 통한 애플리케이션의 응답 속도(붉은색, 다이아몬드), 사용자 로드(푸른색, 사

각형), 분당 요청 수(노란색, 삼각형), 일반적인 서버의 응답 속도(초록색, 원형) 순으로 나타내었고 사용자 로드와 분당 요청수의 증가와 시간의 경과에 따라 기존의 단일 하드웨어 서버에서는 응답 속도가 증가되는 것에 비해 가상화 플랫폼을 통한 화재 및 연기 감지 모니터링 시스템으로 제공하게 되면 5ms 이하의 응답 속도를 보장하여 운영할 수 있음을 알 수 있었다[16],[17].

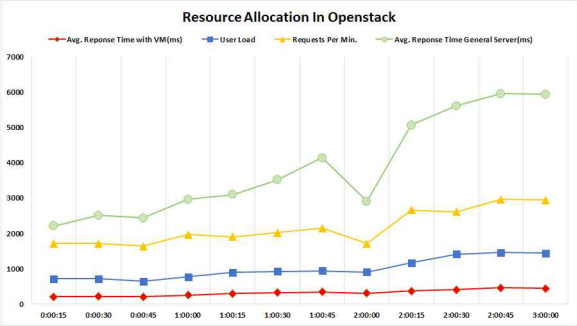


그림 11. 오픈스택의 리소스 할당을 통한 안정적인 응답 시간 보장

Fig. 11. Provide Reliable Response Time Through OpenStack Resource Allocation

Ⅴ. 결론

본 논문에서는 화재 및 연기 감지 모니터링 애플리케이션의 가상화 플랫폼 적용을 통한 안정적인 응답 속도 보장 방안과 프레임워크 구조에 대해 제안하였다. 이를 위해 가상화 플랫폼을 통한 화재 및 연기 감지 예측 모델인 ResNet50 기반 CNN 모델은 5종류의 Augment를 통한 37,000개의 화재 및 연기 이미지 데이터 세트를 이용하여 학습하였고 98% 예측 정확도를 확인 할 수 있었다. 가상화 플랫폼 기반의 화재 및 연기 감지 모니터링 애플리케이션은 기존 단일 하드웨어 서버를 운영했을 때보다 리소스 활용률 향상과 마이그레이션의 확장성의 이점으로 빠르고 안정적인 응답 속도를 보장하는 결과를 도출하였다. 향후에는 효율적인 리소스 활용도가 요구되는 다른 애플리케이션 및 서비스 영역과 다양한 오픈소스 플랫폼(ONAP, OpenNFV, OpenMANO 등)과 결합하여 더 안정적인 성능을 도출하는 연구를 기대할 수 있다.

감사의 글

본 연구 논문은 과학기술정보통신부 및 정보통신기획평가원의 출연금 등으로 수행하고 있는 한국전자통신연구원의 위탁연구과제(초연결 공통 네트워킹 서비스 연구인프라 구축 및 2019-미래통신·전파-1)와 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음" (IITP-2019-2016-0-00314)의 연구결과입니다.

참고문헌

[1] N. Fernando, S.W. Loke, W. Rahayu, “Mobile cloud computing: A survey,” *Future generation computer systems,* Vol. 29, No. 1, pp. 84-106, 2013.

[2] H. T. Dinh, C. Lee, D. Niyato, P. Wang, “A survey of mobile cloud computing: architecture, applications, and approaches,” *Wireless communications and mobile computing*, Vol. 13, No. pp18 (2013): 1587-1611.

[3] M. Satyanarayanan, Fundamental challenges in mobile computing, Defense Technical Information Center, Philadelphia, USA, Technical Report CMU-CS-96-111, 1996

[4] M. Satyanarayana, V. Bahl, R. Caceres, N. Davies, “The case for VM-based cloudlets in mobile computing,” *IEEE Pervasive Computing*, Vol. 8, No. 4, pp. 14–23, 2009.

[5] C. A. C. Medina, O. J. S. Parra, F. F. Rodiriguez, “A brief survey of new generation networks,” *Electronic vision*, Vol. 7, No. 1, pp. 194-205, 2014

[6] M. Fazio, A. Celesti, F. G. Marquez, A. Glikson, M. Villari, “Exploiting the FIWARE cloud platform to develop a remote patient monitoring system,” *IEEE Symposium on Computers and Communication (ISCC)*, Larnaca, Cyprus, pp. 264-270, 2015.

[7] A. Spanias, J. H. McClellan, A. Docef, “Teach-Ware: DSP education resources,” *IEEE Signal Processing Magazine*, Vol.23, No.4, pp.136-137., 2006

[8] P. Smet, B. Dhoedt, P. Simoens, “Docker Layer Placement for On-Demand Provisioning of Services on Edge Clouds,” *IEEE Transactions on Network and Service Management*, Vol. 15, No. 3, pp. 1161-1174, 2018

[9] S.G. Kang, K.W. Lee, “Testing Implementation of Remote Sensing Image Analysis Processing Service on OpenStack of OpenSource Cloud Platform,” *Journal of the Korean Association of Geographic Information Studies*, Vol. 16, No.4, pp.141-152, 2013

[10] OpenBaton. [Internet]. Available: https://openbaton.github.io/documentation/.

[11] J. H. Ku, “A Study on Characteristics Analysis for Detection Performance of a Video-based Automated Fire Detection System,” *Journal of the Korean Society of Hazard Mitigation,* Vol.13, No. 5, pp. 239-245, 2013

[12] Y. J. Kim, E. G. Kim, “Image based Fire Detection using Convolutional Neural Network,” *Journal of the Korean Institute of Information and Communication Engineering*, Vol.

20, No. 9, pp. 1649-1656, Sep 2016.
[13] A. Muhammad, W. C. Song, “Framework for Resource Utilization-Based Dynamic Migration of VMs in OpenStack Clouds,” *The Journal of the KICS*, Vol. 42, No. 8, pp. 1562-1572, 2017
[14] K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition,” In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, pp. 770-778. 2016.
[15] G. Cattaneo, F. Giust, C. Meani, D. Munaretto, P. Paglierani, “Deploying cpu-intensive applications on mec in nfv systems: The immersive video use case,” *Computers*, Vol. 7, No. 4, pp 55-75, 2018
[16] R. L. D. Santos, Deploying and managing network services over programmable virtual networks, Ph.D. Federal University of South Portugal, Lisbon, 2018
[17] S. Vural, R. Minerva, G. A. Carella, A. M. Medhat, L. Tomasini, S. Pizzimenti, U. Stravato, “Performance Measurements of Network Service Deployment on a Federated and Orchestrated Virtualisation Platform for 5G Experimentation,” in *Proceeding of In 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Verona, Italy, 2018

오승민(Seungmin Oh)

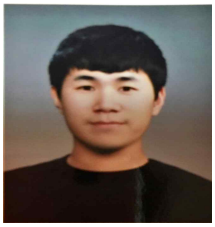
2019년 : 나사렛대학교 디지털콘텐츠학과 학사
2019년~현재 : 전남대학교 전자컴퓨터공학과 석사과정

※관심분야 : 모바일 클라우드 컴퓨팅, 에너지 절약 시스템, 사물인터넷, 인공지능 및 강화학습

ASHIQUZZAMAN AKM

2017년 : Computer Science and Engineering from University of Asia Pacific, Dhaka Bangladesh 학사
2018년~현재 : 전남대학교 전자컴퓨터공학과 석사과정

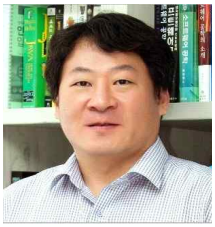
※관심분야 : 지능형 네트워킹 시스템, 인간-컴퓨터 상호작용, 컴퓨터 비전, 딥러닝 및 머신러닝



이동수(Dongsu Lee)

2018년 : 광주대학교 정보보안학과 학사
2018년~현재 : 전남대학교 전자컴퓨터공학과 석사과정

※관심분야 : 모바일 클라우드 컴퓨팅, 스마트팩토리, 사물인터넷



김진슬(Jinsul Kim)

2001년 : University of Utah, Salt Lake City, Utah, USA 학사
2005년 : 한국과학기술원 (KAIST) 석사
2008년 : 한국과학기술원 (KAIST) 박사

2005년～2008년: 한국전자통신연구원(ETRI) IPTV 인프라 기술, 융·복합 방송/통신 분야 연구원
2009년～2012년: 나사렛대학교 멀티미디어학과 교수
2012년～현재: 전남대학교 전자컴퓨터공학과 교수
※관심분야 : QoS/QoE 예측/분석/관리, 모바일 미디어 처리, 통신, 클라우드 컴퓨팅 디지털 미디어 아트 및 네트워크 지능기술

박재형(JaeHyung Park)

1991년 : 연세대학교 전산과학과 학사
1993년 : 한국과학기술원 (KAIST) 석사
1997년 : 한국과학기술원 (KAIST) 박사

1997년~1998년 : 한국과학기술원 (KAIST) 인공지능 연구센터 연구원
1998년~2002년 : 한국전자통신연구원 (ETRI) 네트워크기술연구소 선임연구원
2002년~현재 : 전남대학교 전자컴퓨터공학부 교수
※관심분야 : 인터넷 라우팅, 멀티캐스트 라우팅, 네트워크 보안, 무선 메쉬 네트워크, 무선 메쉬/애드혹 네트워크, 인공지능