Regular paper | Journal of the KIECS. pp. 1039-1046, vol. 17, no. 6, Dec. 31. 2022, t. 116, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2022.17.6.1039

SDN 환경에서 서버 상태 기반 가중치 부하분산 기법

이경한* · 권태욱**

Server State-Based Weighted Load Balancing Techniques in SDN Environments

Kyoung-Han Lee* · Tea-Wook Kwon**

요 약

코로나-19 판데믹 이후 언택트 문화의 확산과 다양한 유형의 데이터를 생성하는 4차 산업 혁명으로 이전과는 비교되지 않을 정도로 많은 데이터가 생성되었다. 이는 보다 높은 데이터 처리율을 요구하게 되었고, 벤더와 하드웨어를 중심으로 하는 기존 네트워크 체계의 한계를 조금씩 드러나게 하였다. 최근 이런 한계점을 극복할 수 있는 사용자와 소프트웨어 중심의 SDN이 주목받고 있다. 또한, SDN을 기반으로 한 부하분산 기법은 방대하고 다양한 데이터를 생성하고 처리하는 데이터 센터의 서버 클러스터의 부하분산 영역에 효율을 높여 줄 것으로 보인다. 본 논문은 기존 SDN 부하분산 연구들과 달리 모니터링 기법을 통한 주기적인 확인 아닌 이벤트 발생에 따라 컨트롤러가 서버의 상태를 확인하고, 부하율에 따른 가중치를 부여하여 사용자의 요청을 할당하는 부하분산 기법을 제안하고 있다. 소기 실험결과 제안기법이 대조기법과 비교하여 3%가량 균등한 부하분산 효과를 보여 소기의 성과를 보였기에 규모가 크고 패킷의 흐름이 많은 데이터 센터의 서버 클러스터에서의 좀 더 효과적일 것으로 기대된다.

ABSTRACT

After the COVID-19 pandemic, the spread of the untact culture and the Fourth Industrial Revolution, which generates various types of data, generated so much data that it was not compared to before. This led to higher data throughput, revealing little by little the limitations of the existing network system centered on vendors and hardware. Recently, SDN technology centered on users and software that can overcome these limitations is attracting attention. In addition, SDN-based load balancing techniques are expected to increase efficiency in the load balancing area of the server cluster in the data center, which generates and processes vast and diverse data. Unlike existing SDN load distribution studies, this paper proposes a load distribution technique in which a controller checks the state of a server according to the occurrence of an event rather than periodic confirmation through a monitoring technique and allocates a user's request by weighting it according to a load ratio. As a result of the desired experiment, the proposed technique showed a better equal load balancing effect than the comparison technique, so it is expected to be more effective in a server cluster in a large and packet-flowing data center.

키워드

SDN, Load Balancing, Server Cluster, Weighting
소프트웨어 정의 네트워크, 부하 분산, 서버 클러스터, 가중치

* 국방대학교 석사과정(luckykh07@naver.com)
** 교신저자 : 국방대학교 컴퓨터공학과 교수
· 접 수 일 : 2022. 09. 30
· 수정완료일 : 2022. 11. 05
· 게재확정일 : 2022. 12. 17
· Received : Sep. 30, 2022, Revised : Nov. 05, 2022, Accepted : Dec. 17, 2022
· Corresponding Author : Tea-Wook Kwon
Dept. Computer Science, Korea National Defense University
Email : luckykh07@naver.com

Ⅰ. 서 론

글로벌 시장 조사기관(IDC)에서는 전 세계 데이터 총량이 18년도 33ZB에서 25년에는 175ZB로 늘어날 것으로 전망했다. 올해 상반기까지 코로나-19 여파로 Untact(언-택트) 문화의 활성화로 인터넷쇼핑, OTT 콘텐츠, 재택근무 등 온라인 활동이 생활의 중심이 되었고[1], 이와는 상반되는 어감을 가진 다양한 스마트 기술의 연결을 더한 Ontact(온 택트) 문화 또한 확산되어 앞으로 데이터의 비중과 역할을 더욱 커져갈 것으로 예측된다. 이와 더불어 산업계 곳곳에서도 데이터와 네트워크 기반의 인공지능(AI) 챗봇이 등장하고 있어 데이터 증가량이 기존 예상보다 훨씬 빠른 속도로 급증할 것으로 판단된다. 이런 데이터량의 증가는 온라인 트래픽을 대량으로 증가시키면서 트래픽 폭증으로 이어지고 있다[2].

데이터의 폭증과, 코로나 19로 인한 디지털전환이 가속화되면서 데이터산업의 폭발적인 성장이 예상됨에 따라 보안 및 데이터 관리에 적합하고 대규모 데이터 센터의 필요성이 강조되고 있다. 데이터 센터는 ICT 산업의 근간이자 주요 부가가치를 창출하는 산업인 사물인터넷, 자율주행, 빅데이터, 클라우드 등의 구현을 위해서 필수적인 인프라이자, 국가정보화 전략, 데이터 주권 확보를 위한 중요한 기반시설이다.

이러한 이유로 네트워크의 영역은 폭증하는 데이터 처리와 데이터 센터에 적합한 네트워크 구성을 위해 지속적으로 확장이 요구되어 왔으나, 기존 체계는 벤더별 제공되는 하드웨어 중심의 일괄된 정책 적용으로 벤더가 다른 기종간의 융통성 있는 확장 제한과 상황에 따른 사용자의 요구사항에 실시간으로 대응이 제한되는 등의 한계점을 들어내게 되었고, 이런 배경으로 소프트웨어 중심으로 유연한 대응이 가능한 소프트웨어 정의 네트워크(SDN, Software-Defined Networking)가 주목받고 있다[3].

SDN의 유연성과 확장성의 이점으로 많은 분야에서 적용연구가 이루어지고 있다. 특히 다양한 유형의 데이터를 생성하는 4차 산업 혁명의 데이터 센터의 대규모 서버 클러스터의 부하분산에 적합할 것으로 판단되며, 기존 네트워크 체계보다 높은 효율을 달성할 것으로 기대된다[4].

본 논문에서는 SDN환경에서 연구되었던 불필요한 트래픽이 발생할 가능성이 있는 지속적인 모니터링 기법과 임계치에 도달한 서버를 포워딩 목록에서 제외하는 기법들과 달리 이벤트 발생시에만 컨트롤러가 서버 상태를 확인하고, 서버 상태에 따라 가중치를 부여하여, 모든 서버의 가용 능력을 고려하여 요청을 처리하는 기법을 제안한다.

Ⅱ. 관련연구

2.1 SDN(: Software-Defined Networking)

SDN[5]은 네트워크 장비에서 하드웨어와 소프트웨어 기능을 분리하는 것이 핵심이다. 하드웨어 영역은 Data Plane으로써 데이터를 전송하는 역할을 한다. 소프트웨어 영역은 Control Plane과 Application Plane으로 나뉘고, Control Plane은 네트워크에 대한 정책이 적용되는 영역으로 볼 수 있으며, Application Plane은 정책 적용을 위한 사용자 지원 영역이다. Control Plane은 SDN 컨트롤러로 구현되며, 논리적으로 중앙 집중화하여 프로그래밍할 수 있도록 고안되었고, 이를 통해 네트워크의 제어나 흐름, 혼잡 제어 등에 유연성을 제공한다.

기존 네트워크 장비는 다른 어떤 자원보다 더 하드웨어 종속성이 강해 장비 업체마다 다른 주문형 반도체를 사용하는 등 폐쇄적인 설계로 제품을 사용할 때 통합관리가 힘들었다. 하지만 SDN은 네트워크 장비의 역할에서 데이터전송 처리만 남기고 그 외 모든 Configuration, 액세스제어목록(ACL), 배포 등의 기능을 컨트롤러에게 맡겨 네트워크 인프라를 쉽고 빠르게 운영하여 유연하고 효율적인 네트워크가 관리가 가능하다. 이런 SDN의 특징으로 데이터 센터의 서버 부하분산에 적용는 연구가 활발히 이루어지고 있다[6].

데이터 센터는 조직의 방대한 정보저장을 위한 서버, 네트워크 회선 등을 제공하여 데이터를 안정적으로 통합·관리하는 인프라 시설이다. 가상화 기술 발전과 클라우드 등의 서비스 제공과 함께 빠르게 발달중이다[7]. 데이터센터는 웹을 기반으로 하는 다양한 서비스 제공을 위해 부하분산 목적의 서버 클러스터를 구축하고 있으며, 데이터 사용량이 늘어날수록 서버는 고집적화되고 있다. 고용량의 서버가 집적된 데이터센터에서 부하분산은 매우 중요한 부분이며,

SDN을 기반으로 SW를 이용해서 서버 클러스터를 제어하고 활용하는 것은 데이터센터 효율을 높일 수 있일수 있을 것으로 기대된다[8].

2.2 서버상태 기반의 Load Balancing 연구 동향

Zhong은 Control Plane과 Data Plane이 분리되어 있는 SDN 특성을 활용하여 Control Plane으로 서버의 응답시간을 측정하고 측정된 응답시간을 사용자 요청 처리 과정에서 서버를 선택하는데 반영함으로써 부하를 분산하였다[9].

J. Kim의 연구에서는 데이터센터로 유입되는 패킷 형태와 특정 이슈에 영향을 받아 편중된 데이터를 요구할 때, 미들박스를 활용 데이터의 유형을 분류하고, 그중 폭증하는 데이터를 서버 응답시간을 고려하여 우선 처리함으로써 부하를 분산하는 개념을 제시하였다[10].

J.Y LEE의 연구에는 서버의 부하측정을 위해 발생하는 비효율적인 트래픽 교환의 최소화를 목표로 하며, 이를 위해 컨트롤러와 서버의 주기적 통신이 아닌 서버의 특정 임계치 도달 시에만 서버에 의해 경고 패킷을 컨트롤러로 전송하고, 임계치에 도달된 서버를 포워딩에서 제외하여, 정상 서버를 대상으로 사용자의 요청을 처리하게 함으로써 임계치에 도달된 서버의 부하율을 관리하는 부하분산 방식을 제안하였다[11].

P. Deepalakshmi의 연구에서는 컨트롤러를 통해 부하분산에 최적화된 부하분산 요인으로 서버의 CPU, Memory, 연결 가능 세션 수를 매개변수로 선택하였으며, 매개변수별 중요도에 따라 가중치를 부여하여 서버의 부하를 판단하였다. 컨트롤러가 서버의 부하를 정기적으로 모니터링하고 서버의 부하에 따라 3가지 계층으로 분류하여 부하 분산하는 기법을 제안하였다[12].

그 외 연구들은 서버의 부하 판단을 위해 CPU idle, Free Memory를 변수로 선정하여 매개변수별 중요도에 따라 가중치를 부여하여 서버의 부하분산을 수행하는 기법[13]과 SDN 컨트롤러 내 모듈을 배치하여 서버 응답시간을 수집하고, 스위치 포트 트래픽 정보를 수집하여 서버를 선택하는 일련의 절차를 통해 부하분산을 수행하는 방안[14]이 있다.

기존 연구들에서는 컨트롤러가 지속적으로 서버의 상태를 확인하기 위해 패킷 교환하기 때문에 필요 이상의 트래픽이 생성되어 대역폭을 낭비할 가능성이 있다. J.Y LEE[12]의 기법은 지속적 확인이 아닌 일정 임계치에 도달했던 서버의 보고에 의해 해당 서버를 포워딩 목록에서 제외하는 기법으로 임계치에 도달한 서버의 가용능력을 사용하지 않고 낭비한다는 개선점이 있다. 본 논문의 제안기법은 이벤트 발생 시 서버들의 가용능력을 고려하여 가중치 부여함으로써 기존 기법들의 개선점을 해결하고자 한다.

III. 서버상태 기반 가중치 부하분산 기법

3.1 제안하는 기법의 동작방법

제안기법은 기존 연구의 서버와 컨트롤러 간의 불필요한 트래픽을 최소화하고, 서버들의 가용한 능력을 효율적으로 사용하여, 균등한 부하율 달성을 목표로 한다. 이를 위해 컨트롤러와 서버간의 지속적인 통신이 아닌 이벤트(서버 부하가 설정 임곗값(Threshold) 이상 올라가거나, 임곗값을 초과했던 부하가 임곗값 이하로 내려갈 경우) 발생 시 컨트롤러는 서버로부터 이벤트 보고를 받고, 전 서버의 상태를 확인하며, 서버의 가용능력에 따른 가중치를 부여하여 모든 서버의 가용 능력을 활용하는 기법을 제안한다. 그림 1과 그림 2는 각각 제안기법의 시스템 설계 및 기본적인 동작절차와 알고리즘이다.

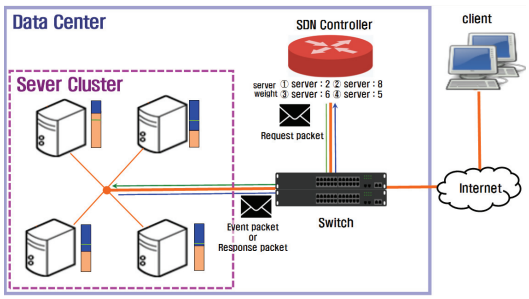

그림 1. 시스템 설계 및 동작절차
Fig. 1 System design and operation procedures

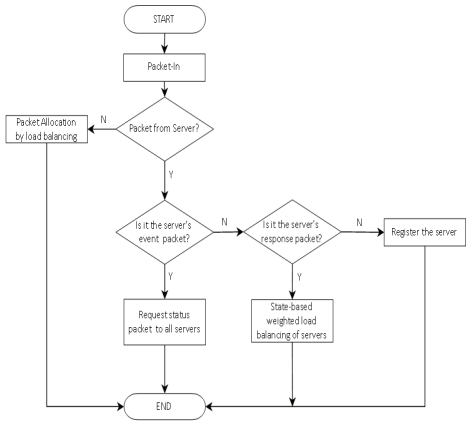

그림 2. 제안하는 기법의 알고리즘 순서도
Fig. 2 Algorithm flowchart of proposed method

시스템 시작 컨트롤러는 서버로부터 기본정보 패킷을 받아 서버 목록을 작성한다. 이후 최초 라운드 로빈 방식으로 부하분산을 실시하며, 동작 중 특정 서버의 부하가 설정 임곗값을 초과한 서버는 이벤트 패킷을 컨트롤러로 전송하다. 이벤트 발생 패킷을 받은 컨트롤러는 가중치 할당을 위해 전 서버를 대상으로 부하 상태확인을 위한 요청 패킷(Request Packet)을 전송하게 되며, 요청 패킷을 받은 서버는 자신의 CPU와 메모리 사용률을 응답 패킷(Response Packet)을 통해 컨트롤러로 전송한다. 응답 패킷을 받은 컨트롤러는 서버의 상태에 따른 가중치를 계산하여 부여하고, 사용자의 요청을 각 가중치에 따라 서버에 할당한다. 이후 이벤트가 발생했던 서버의 부하율이 설정했던 임곗값 이하로 내려가거나, 혹은 다른 서버의 부하가 설정 임곗값 이상 초과되었을 때 이전과 같이 컨트롤러로 이벤트 패킷을 전송한다. 패킷을 받은 컨트롤러는 이전과 같은 작업을 반복하여 사용자의 요청을 각 서버로 할당한다.

3.2 서버의 동작

시스템 시작 서버는 컨트롤러로 기본정보가 담긴 패킷을 전송하여 서버 목록에 등록한다. 최초 라운드 로빈 방식에 따라 사용자의 요청을 처리하게 되며. 이후 부하율이 설정된 임곗값을 초과하거나, 초과했던 부하율이 임곗값 이하로 내려갈 경우 컨트롤러로 이벤트 패킷을 전송한다. 이후 컨트롤러로부터 서버 상태 확인을 위한 요청 패킷을 받게 되면, 자신의 가용 CPU 사용률과, 메모리의 사용률을 응답 패킷을 통해 컨트롤러로 전송하며, 컨트롤러의 가중치 부여에 따른 사용자의 요청을 할당받아 처리한다. 서버의 부하에 관한 판단은 아래 식(1)[14]을 따른다.

$$SL_i = \alpha \times C_i + \beta \times M_i \quad (1)$$

if($SL_i > TH$) then

Server State1 = EVENT1(OVERLOAD) (2)

else

Server State0 =EVENT0(NOMAL) (3)

위 식에서 i는 서버 번호, SLi은 각 서버의 자원에 대한 부하 상태, Ci는 i번 서버의 CPU 사용율, Mi는 i번 서버의 메모리 사용율, α, β는 부하 판단 요소별 가중치, TH는 서버의 부하 판단 임곗값을 나타낸다. 위 식 (1)에 따라 각 서버는 CPU와 메모리의 사용률에 가중치를 적용한 해당 계산의 결괏값에 따라 부하 여부를 판단하게 되며, 식(2)는 SL_i 값이 TH을 초과하게 되면 컨트롤러로 EVENT1(OVERLOAD) 메시지를 전송하게 되고 이후 컨트롤러의 가중치 계산에 따라 사용자 요청을 처리하고 식(3)에 따라 SL_i 값이 TH값 미만으로 내려갈 경우 EVENT0(NOMAL) 메시지를 컨트롤러로 전송한다.

Ⅳ.실험 및 결과분석

4.1 시스템 구현 및 환경

서버 부하 임곗값과 그에 따른 가중치를 부여하고, 제안기법의 부하분산 효과를 확인하기 위해 패킷 생성기 역할을 하는 요청자와 서버 클러스터의 부하상태 정보를 수집하고 부하분산 정책을 적용하는 컨트롤러, 수신받은 부하분산 정책을 통해 요청자의 요청을 포워딩하는 스위치, 요청자의 요청을 수신하고 처리하는 서버를 Riverbed Modeler로 구축하였다.

제안하는 부하분산 기법을 가상 네트워크 환경에서 구현하여 효과를 평가하였다. 실험환경은 CPU Intel (R) Core(TM) 8250U CPU @ 1.80 GHz, OS Windows 10, RAM 8GB, Riverbed Modeler 18.9.0을 사용하였다.

4.2 실험방법

실험은 서버의 효율적인 부하율과 균등한 사용률 확인을 위해 제안기법(PP)과 대조기법(RR, Round Robin)의 서버별 부하율 비교와 서버와 컨트롤러 간의 응답시간 측정을 통해 제안기법의 효율을 확인한다. 본 논문에서는 서버의 CPU와 메모리중 특정 자원의 비중을 중점으로 두지 않고 서로 같게 설정하였으며, 서버의 TH(Threshold, 임곗값)값은 70%로 설정하였다.

첫 번째 방법은 서버 부하율 측정은 대조기법과 제안기법(PP)의 부하분산 효율을 비교하기 위해 서버 클러스터의 평균 부하율이 최대 80% 수준으로 유지되도록 요청자가 지속적으로 요청 패킷을 생성하여, 각 서버의 부하율을 측정한다. 측정된 각각의 서버의 부하율과 최대 부하율과 표준편차값으로 제안기법과 대조기법의 서버 부하율을 비교한다.

두 번째 방법으로 제안기법과 대조기법의 서버와 컨트롤러 간의 평균 응답시간을 기준을 측정하여 비교한다.

4.3 실험결과 분석

제안기법과 대조기법의 서버별로 부하율 비교한 실험 결과 그림 3 및 표 1과 같다. 제안기법과 대조기법의 서버별 최대 부하율의 평균값은 각각 85.34%과 88.49%으로 제안기법이 3.06%가량 향상된 성능을 볼 수 있으며, 서버별 Std Dev값(서버별 표준편차)의 종합평균값은 제안기법이 16.01 %. 대조기법은 17.74 %으로 제안기법이 대조기법 보다 좀 더 균등한 서버 부하율을 달성한 것을 볼 수 있다. 이는 제안기법의 경우 일정 임계치에 도달한 서버가 발생했을 경우 클러스터 내의 모든 서버의 가용 능력을 파악하여, 이에 부합되는 가중치를 부여하여, 사용자 요청을 할당함에 따라 대조기법 보다 효율적이고, 균등한 부하율 달성하게 된다.

표 1. 제안기법과 대조기법의 서버별 부하율 측정 결과
Table 1. Load for server of the proposed and contrast method

Object	Minimum		Average		Maximum		Std Dev	
	PP	RR	PP	RR	PP	RR	PP	RR
server1	0.1315	0.1334	0.7300	0.7201	0.8598	0.8967	0.1590	0.1779
server2	0.1348	0.1322	0.7270	0.7184	0.8546	0.8778	0.1620	0.1761
server3	0.1338	0.1328	0.7243	0.7167	0.8503	0.8747	0.1593	0.1793
server4	0.1338	0.1296	0.7190	0.7143	0.8489	0.8904	0.1599	0.1762
Total Avg	0.1335	0.1320	0.7251	0.7174	0.8534	0.8849	0.1601	0.1774

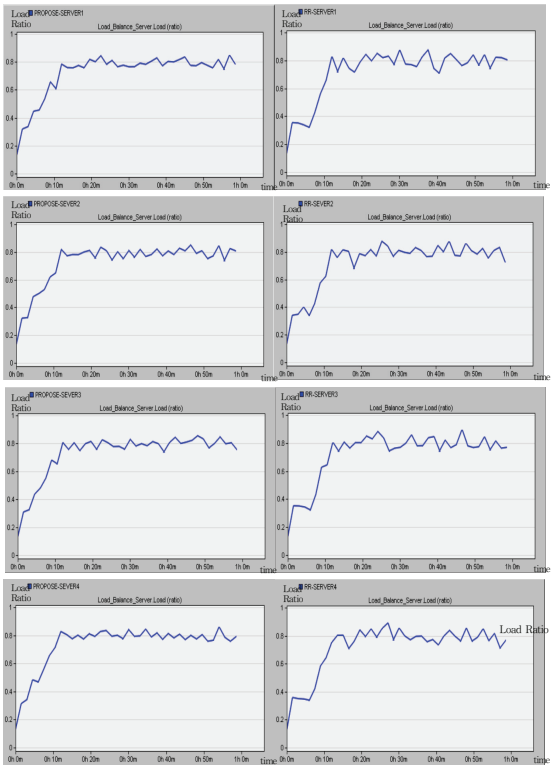

그림 3. 제안기법(좌)과 대조기법(우)의 서버별 부하율
Fig. 3 Load rate by server of proposed and contrast method

4.4 응답시간 분석

제안기법과 대조기법의 서버 - 컨트롤러간의 응답시간 측정시간은 그림 4와 표 2와 같다. 실험결과 대조기법이 제안기법보다 평균적으로 약 0.1초 정도 응답시간이 빠른 것으로 나타났으며, 부하과중이 증가하는 15분 이후부터 대조기법이 제안기법보다 좀 더 나은 응답시간 결괏값을 볼 수 있다. 이는 컨트롤러가 서버 상태를 확인하지 않는 대조기법과 달리 제안기법의 경우 부하 발생 시 보고 패킷과 컨트롤러가 서버의 상태를 알기 위한 확인 패킷을 주고받기 때문에 발생하는 필연적 지연으로 해석할 수 있지만, 지연의 정도가 0.1초로 아주 근소한 차이임을 알 수 있다.

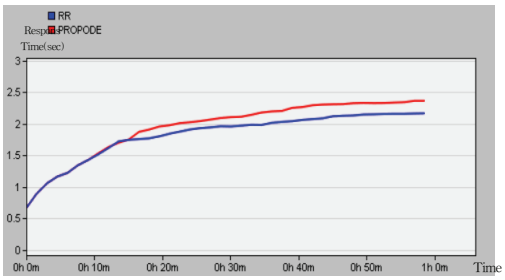

그림 4. 제안기법과 대조기법의 응답시간 측정
Fig. 4 Response time of proposed and contrast method

표 2. 제안기법과 대조기법의 응답시간 측정 결과
Table 2. Response time of proposed and contrast method

sec

Category	Proposed	Round Robin
Minimum	0.74342	0.74342
Average	1.56596	1.46891
Maximum	2.38850	2.19440

Ⅴ. 결론

본 논문에서는 SDN 환경에서 서버 클러스터의 특정 서버에서 부하의 정도가 설정된 임곗값을 초과하거나, 초과되었던 부하가 임곗값 이하로 내려가는 이벤트가 발생했을 때 컨트롤러가 전 서버의 가용상태를 확인하고, 이를 기반으로 가중치를 부여하여 부하분산을 실시하는 기법을 제안하였다. 실험결과 제안기법이 대조기법인 RR(: Round Robin)기법과 비교하여, 보다 3%가량 효율적인 부하분산으로 서버의 최고 부하율을 낮추고, 균등한 사용율을 달성하는데 효과적이었다. 반면, 응답시간 측정 실험에서는 대조기법인 RR기법이 제안기법에 비해 0.1초 가량 더 빠른 응답시간 측정값을 가지는 것을 알 수 있었지만, 그 값이 아주 근소한 차이임을 확인하였다.

이를 통해 제안기법이 규모가 큰 서버 클러스터에서 기존의 기법인 RR보다 서버의 가용능력을 명확히 파악하여 그에 맞는 가중치 부여로 균등한 부하 달성에 효과적일 것으로 기대되며, 기존 네트워크 체계의 한계점으로 거론되었던 경직성을 타파할 수 있는 SDN환경에서 구현되었기 때문에 설정 임곗값을 수정하는 등 관리자에 의해 서버 클러스터가 필요로 하는 환경에 맞춰 유연하게 대응할 수 있을 것으로 기대된다.

References

[1] C. Jeong “A Study on the Method of Implementing an AI Chatbot to Respond to the POST COVID-19 Untact Era” *Journal of Information Technology Services,* vol. 19 no. 4, 2020, pp. 21-47

[2] Martin McKeay, “Adapting to the Unpredictable,” *Akamai, State of the Internet,* vol. 7, Issue 1, 2021.

[3] B. Yoon, “Future Networking Technology of SDN,” *Electronics and Telecommunications Trends*, vol. 27, no. 2, 2012, pp. 129 - 136.

[4] IDC, “Worldwide Data center Software-Defined Networking Forecast, 2019-2023,”Nov, 2019.

[5] J. Jang, "SDN/NFV Military application plan (Based on the Defense Integrated Data Center(DIDC)),” *The Korea Institute of Electronic Communication Sciences,* vol. 15, no. 4, 2020, pp. 687-694.

[6] M. LEE, "Implementation of a Platform for the Big Scientific Data Transfers,” *The Korea Institute of Electronic Communication Sciences,* vol. 13, no. 4, 2018, pp. 881-886.

[7] J. Holusha, "Commercial Property/Engine Room for the Internet; Combining a Data Center With a 'Telco Hotel,” New York Times, 2019

[8] D. Min, “Market Trends of SDN/NFV Supply and Demand,” *Electronics and Telecommunications Trends, Electronics and Telecommunications Research Institute,* vol 31 Issue 2, 2016, pp. 28-40

[9] H. Zhong, Y. Fang, and J. Cui, “LBBSRT: An efficient SDN loadbalancing scheme based on server response time,” *Future Generation Comput. Syst.* vol. 68, 2017, pp. 183-190.

[10] J. Kim, "Efficient Load Balancing Technique Considering Data Generation Form and Server Response Time in SDN,” *The Korea Institute of Electronic Communication Sciences,*vol. 15, no. 4, 2020, pp. 679-686.

[11] J. LEE, "Efficient Load Balancing Technique

through Server Load Threshold Alert in SDN," *The Korea Institute of Electronic Communication Sciences,* vol. 16 no. 5, 2021, pp. 817-824.

[12] P. Deepalakshmi, "D-Serv LB: Dynamic server load balancing algorithm," *International Journal of Communication Systems*, vol. 32, Issue. 1, 2019, pp. 293-310.

[13] M. Yang, "Research on Load Balancing Algorithm Based on the Unused Rate of the CPU and Memory," *International Conference on Instrumentation and Measurement, Computer, Communication and Control(IMCCC),* Qinhuangdao, China, 2015, pp. 542-545.

[14] S. Kiarash, "SD-WLB: An SDN aided mechanism for web load balancing based on server statistics,"*Electronics and Telecommunications Research Institute,* vol. 41, Issue. 2, 2018, pp. 197-206.

저자 소개

이경한(Kyoung-Han Lee)

2015년 건양대 군사학과 졸업(군사학, 문학사)
2021년 ~ 현재 국방대학교 대학원 컴퓨터공학과 석사과정

※ 관심분야 : 네트워크, SDN

권태욱(Tae-Wook Kwon)

1986년 육군사관학교 컴퓨터공학과 졸업(공학사)
1995 미국 해군대학원 컴퓨터공학(공학석사)
2001년 연세대학교 컴퓨터공학과(공학박사)
2007년 ~ 현재 국방대학교 컴퓨터공학과 교수

※ 관심분야 : 네트워크, Sensor Networking, CCN, SDN, NFV