https://doi.org/10.7840/kics.2024.49.11.1566

SDN 환경에서 서버상태에 따른 가변적인 임계값을 적용한 부하분산 기법

이 강 욱◆, 권 태 욱°

Study of Load Balancing Technique with Variable Threshold on Server Status in SDN Environment

Kang-uk Lee◆, Tae-wook Kwon°

요 약

AI, 빅데이터, 딥러닝, 클라우드 등 현대의 네트워크 환경은 복잡하고 다양한 데이터의 흐름이 필연적이게 되었다. 이러한 환경속에서 사용자들의 다양한 데이터의 처리는 하나의 서버가 아닌 수많은 서버로 구성된 데이터센터의 효율적인 네트워크 구성이 필요하게 되었다. 그러나 기존의 네트워크 방식에서는 벤더에 종속되어 각각의 서버를 관리하는 구조이기 때문에 데이터센터의 서버에 대한 관리 소요가 많아지게 되었다. 이를 효율적으로 관리하기 위해 SDN 기술이 주목받고 있으며 이 기술을 통해 요청되는 데이터에 대해 처리할 때 서버들에 대한 부하분산 기술이 중요해졌다. 본 논문에서는 이러한 지연 문제를 해결하기 위해 서버가 일정 부하도가 걸리는 정도를 확인하여 임계값에 도달하게 될 경우 부하도가 상대적으로 낮은 다른 서버에서 데이터를 처리하게 하여 모든 서버의 가용성을 높이는 부하분산 기법을 사용하게 되었다. 특히 모든 서버의 평균 부하도를 고려하여 임계값이 자동으로 계산되어 가변적으로 적용함으로써 현재 네트워크 트래픽에 대한 최적의 부하분산이 이루어질 수 있는 방식을 제안하였다. 이를 실험을 통해 서버의 부하율이 임계값을 넘어 과부하 되지 않는 것을 확인하였다. 대조기법의 경우 고정된 임계값으로 많은 패킷이 유입 시 한 서버가 부하율이 96.39%까지 증가하였지만 제안기법의 경우 모든 서버가 90% 이하의 부하율을 기록하였다. 이를 통해 관리자가 최적의 임계값을 찾을 필요 없이 최적의 부하분산이 이루어진다고 볼 수 있다.

키워드 : 소프트웨어 정의 네트워크, 부하분산, 임계값
Key Words : Software Defined Network(SDN), Load Balancing, Threshold

ABSTRACT

AI, big data, deep learning, and cloud technologies have made modern network environments complex, necessitating the flow of diverse data. In such environments, handling users' diverse data requires an efficient network configuration of data centers composed of numerous servers, rather than a single server. However, traditional network methods are vendor-dependent, managing each server individually, which increases the management overhead for data center servers. To address this efficiently, SDN technology has gained attention, emphasizing the importance of load balancing techniques for handling data requests across servers. This paper proposes a load balancing technique that addresses these delay issues by monitoring server load levels and

◆ First Author : Korea National Defense University Department of Computer Science, fogis1989@naver.com, 정회원
° Corresponding Author : Korea National Defense University Department of Computer Science, kwontw9042@naver.com, 종신회원
논문번호 : 202406-122-C-RU, Received June 23, 2024; Revised July 18, 2024; Accepted July 29, 2024

redirecting data processing to other servers with relatively lower loads when a server reaches a certain threshold, thereby enhancing the availability of all servers. Specifically, by automatically calculating and dynamically applying thresholds based on the average load of all servers, the proposed method aims to achieve optimal load balancing for the current network traffic. Through the experiment, it was confirmed that the server's load rate did not exceed the threshold and become overloaded. In the case of the comparison method, with a fixed threshold, one server's load rate increased to 96.39% when a large number of packets were received. However, with the proposed method, all servers recorded a load rate of below 90%. This demonstrates that optimal load balancing is achieved without the administrator needing to find the optimal threshold.

Ⅰ. 서 론

인공지능(Artificial Intelligence, AI), 빅데이터, 딥러닝, 클라우드 등 다양한 기술이 발전되며 인터넷에는 수많은 데이터가 축적되고 있다. 이러한 데이터를 활용하기 위해 사용자들의 인터넷 사용 시간도 지속적으로 증가하는 추세이다. 세계적인 데이터 통계기관인 DataReportal에 의하면 매년 인터넷 사용 시간이 증가하는 것을 확인할 수 있다. 2024년 1월 기준으로 인터넷 사용시간[1]이 5347 백만 시간으로 이는 5년 전 2019년과 비교하여 약 28.9% 증가하였다. 이는 인터넷에 대한 사용량이 매년 증가하며 앞으로도 계속해서 증가할 것으로 예상할 수 있다.

인터넷 사용량이 많아진다는 것은 사용자들이 서버에 요청하는 데이터의 양이 많아진다는 것을 의미하게 된다. 따라서 데이터센터에서는 이러한 사용자들의 요청된 데이터를 처리하기 위해 여러 서버들에 대한 부하분산이 데이터처리 속도에 큰 영향을 미치게 된다. 기존의 네트워크에서는 관리자가 네트워크 정책을 적용하기 위해 각 장비별로 설정해야 하는 제한사항이 존재했다. 하지만 SDN 기술이 등장하며 모든 장비에 정책을 일괄 적용하여 통일된 정책으로 효율적인 관리[2]가 가능하게 되었다.

소프트웨어 정의 네트워크(SDN)[3] 기술은 소프트웨어로 연결된 각 벤더들을 중앙집중적인 네트워크로 구성함으로써 통일된 정책의 적용이 가능하다. 이를 통해 데이터센터의 서버에 대한 확장 및 축소 등 구성변동이나 정책적용의 유연성 등에 대한 강점이 있다. 또한 SDN을 활용하여 부하분산 정책을 적용하게 됨으로써 데이터 처리효율을 향상된 상황이다.

이미 이전의 SDN 기반의 부하분산 기법과 관련된 연구들이 이루어졌으며 다양한 방식을 제안해오고 있다. 하지만 기존 정책들은 관리자가 특정 조건들을 설정해야 하는 제한점이 있어 유동적인 데이터 트래픽의 환경에서 실시간으로 최적의 부하분산을 시키기에는 어려움이 존재하였다.

이에 본 논문에서는 데이터센터의 특정 서버에만 부하가 가중되는 것을 방지하기 위해 설정된 임계값을 자동으로 조정됨으로써 현재의 트래픽 상황에서 최적의 부하분산이 이루어지도록 하는 방식을 제안하였다. 이를 통해 관리자가 임계값을 수동으로 조정해야만 하는 불편함과 부정확성을 감소시킬 수 있을 것으로 기대된다.

Ⅱ. 관련 연구

2.1 SDN(Software-Defined Network) 개요

SDN의 주요 특징[4]은 제어부와 전송부의 분리다. 기존의 네트워크 벤더는 제어기능과 전송기능이 하나의 벤더에 종속되어 있지만 SDN에서는 제어기능이 네트워크로 연결된 컨트롤러로 분리되면서 각각의 벤더는 전송기능만 남게된다. 즉, SDN의 계층은 그림 1.과 같이 구성된다.

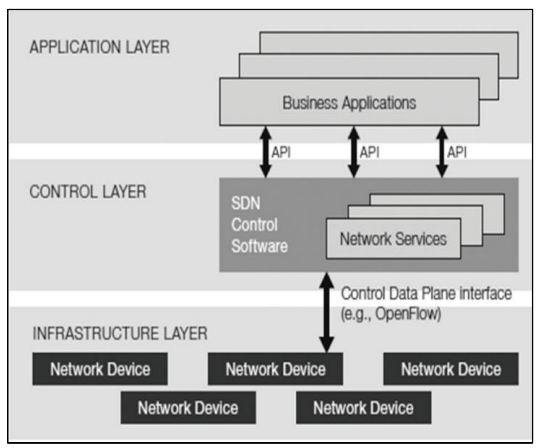

그림 1. SDN 계층구조
Fig. 1. Layer Structure of SDN

SDN은 Application Layer, Control Layer, Infrastructure Layer 3개 계층으로 구성된다. 각 벤더는 데이터를 전송하는 전송부 역할을 수행하며 이는 Infrastructure Layer이다. 모든 벤더와 네트워크로 연결된 컨트롤러가 Control Layer가 되며 각 벤더에 대한 정책을 설정해 준다. 이때 관리자는 컨트롤러에 대한 다양한 기능을 설정하게 되는 Application Layer가 존재한다.

각 계층간에는 다양한 인터페이스로 연결되는데 Application Layer와 Control Layer 간의 인터페이스를 Northbound API라 부르며 Control Layer와 Infrastructure Layer 간의 인터페이스를 Southbound Layer라 부른다. Southbound API는 대표적으로 개방형API인 OpenFlow[5]가 있다. 이러한 API는 각 계층간 네트워크로 연결되어 유지되게 된다.

2.2 기존 부하분산 연구

P. Deepalakshmi의 연구[6]에서는 각 서버의 CPU, 메모리, 연결 가능한 세션 수에 따라 부하를 계산하여 서버에 가중치를 부여하였다. 컨트롤러는 이러한 부하도의 계산을 정기적으로 실시하여 모니터링한다. 서버들의 부하 정도에 따라 서버의 상태를 3단계로 구분하여 단계별 패킷 분배량의 차이를 주는 방식의 부하분산 기법을 제안하였다.

J. Kim의 연구[7]에서는 데이터센터에 유입되는 패킷의 유형을 분석하고 동일한 유형의 패킷이 폭주할 때 미들박스를 이용하여 분류된 폭주하는 유형의 패킷들을 응답시간이 빠른 서버에 할당하여 빠르게 처리하는 부하분산 개념을 제시했다.

J. Lee의 연구[8]에서는 서버의 정보를 위해 주고받는 소요를 줄여 패킷량을 감소시키기 위한 방식을 구상하였다. 컨트롤러는 주기적으로 서버의 상태를 확인하는 것이 아니라 서버의 부하도가 특정 임계값에 도달한 경우에만 서버가 부하상태를 컨트롤러에 전송하는 방식이다. 그러면 컨트롤러는 보고된 서버를 제외하고 다른 서버로 패킷을 분배하는 방식이다.

K. Lee의 연구[9]에서는 J. Lee의 연구와 마찬가지로 서버의 부하율이 임계값에 도달하면 컨트롤러에 보고하게 된다. 보고된 서버를 포워딩 목록에서 제외하고 다른 서버들은 부하 정도에 따라 가중치를 부여하여 덜 할당되도록 하며 부하율이 회복값에 도달하면 다시 포워딩 목록에 포함하여 패킷이 할당되도록 하는 기법을 제안했다.

J. Lee의 연구[10]에서는 K. Lee의 방식과 동일하게 서버의 부하율이 임계값을 초과하거나 회복값 이하로 변경됫을 경우에만 컨트롤러에 보고하여 부하도에 따른 패킷 분배의 가중치를 부여하였다. 다른점은 임계값을 초과하여 회복값 이하로 내려가기 전까진 포워딩 목록에서 해당 서버를 제외하는 대신 부하도에 따른 가중치의 절반만 분배되도록 하는 기법을 제안했다.

기존의 연구들은 특정 서버에 과부하가 발생하는 것을 부하도를 계산하여 지정된 임계값에 도달한 것을 이용하여 계산하였다. 하지만 이러한 방식은 유입되는 전체 패킷량이 일부 서버가 임계값에 도달한 상황에서만 효율을 보이며 모든 서버가 임계값에 도달하는 상황에서는 효율적인 부하분산이 이루어지기 어렵다. 서버에 요청되는 패킷량을 사전에 분석하여 특정 임계값을 관리자가 지정해야 효율적으로 운영이 가능하다는 제한점이 존재한다.

Ⅲ. 제안 기법

3.1 기본 개념

시스템이 시작되면 컨트롤러는 연결되는 서버들로부터 기본 정보를 받게 되고 이를 등록한다. 이후 컨트롤러는 이벤트가 발생되는 기준인 임계값과 회복값을 서버에게 지정되도록 패킷을 보낸다. 이후 요청되는 데이터 패킷을 등록된 서버에 정해진 부하분산 방식으로 할당하게 된다.

서버는 컨트롤러로부터 받은 임계값과 회복값을 현재 부하상태와 비교하게 된다. 이때 서버의 부하 판단은 J. Lee의 연구[10]에 사용된 계산을 이용하였으며 그 식은 아래와 같다.

$$SL_i = \alpha \times C_i + \beta \times M_i \tag{1}$$

SL_i는 I번째 서버의 부하율을 의미하고 C_i는 I번째 서버의 CPI 사용률, M_i는 서버의 메모리 사용률을 의미한다. 이후 부하율 판단 가중치에 따라 α, β 값을 지정해준다. 위와 같이 계산된 부하도를 컨트롤러로부터 받은 임계값(Threshold, TH)과 회복값(Recovery Threshold, RTH)과 비교하여 아래와 같은 조건에 해당될 경우 이벤트 패킷을 생성하여 컨트롤러로 전송한다.

$$\begin{aligned} &\text{if}(SL_i > TH_i) \quad server\,state = 1; \\ &else\,\text{if}(SL_i < RTH_i \,\|\, server\,state = 1) \\ &\quad server\,state = 0; \end{aligned} \tag{2}$$

이때 '1'은 서버의 상태가 과부화 된 것을 의미하며 '0'은 서버의 상태가 과부화 되지 않은 상태를 의미한다.

3.2 세부 절차

제안하는 기법은 특정 서버가 과부화 되는 것을 방지하도록 부하분산이 이루어지도록 한다. 이를 위해 각 서버는 부하율을 계산하며 모든 서버들의 부하율 평균이 낮아지는 것과 임계값에 도달하는 횟수가 줄어드는 것을 목표로 구상하였다.

패킷의 분배방식은 아래 그림 2.와 같이 크게 두 가지로 구분할 수 있다. 첫 번째는 모든 서버가 임계값에 도달하기 전으로 라운드로빈 방식으로 패킷이 들어오는 순서대로 각 서버에 분배하게 된다. 두 번째는 일부 서버의 부하율이 임계값을 초과하였을 경우 초과된 서버를 제외하고 다른 서버들만 라운드로빈 방식으로 분배하게 한다. 이를 통해 임계값에 도달했던 서버는 부하도가 낮아지게 되며 부하도가 특정 값에 도달되면 다시 해당 서버를 포함하여 라운드로빈 방식으로 분배하게 된다.

컨트롤러는 스위치에 유입되는 이벤트 패킷을 처리하기 위한 순서도는 아래 그림 3.과 같다. 유입되는 패킷이 클라이언트가 요청한 데이터 패킷의 경우 현재의 부하분산 정책에 의해 서버에 할당하게 된다.

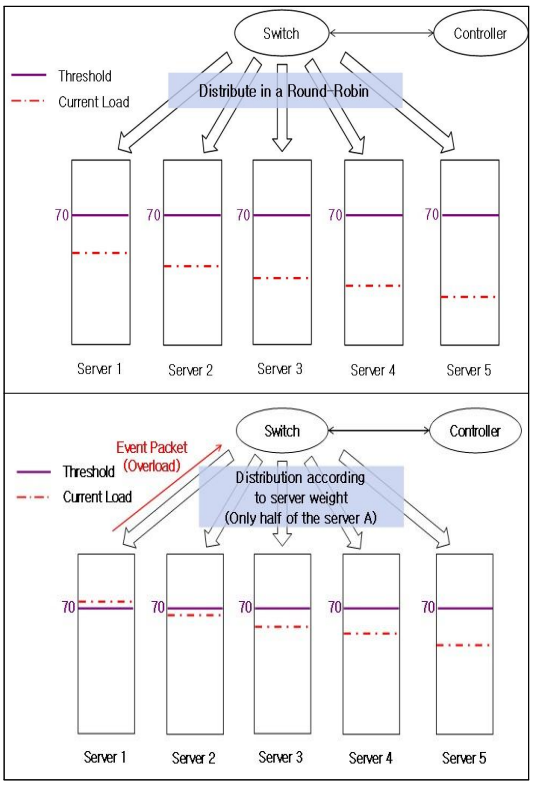

그림 2. 제안기법 분배 방식
Fig. 2. Proposed method of distribution

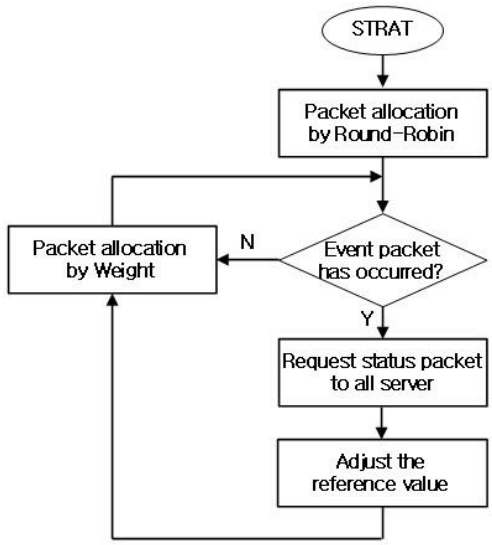

그림 3. 순서도
Fig. 3. Flowchart

발생하는 주요 이벤트는 두 가지가 있다. 첫 번째는 서버의 부하율이 임계값에 도달한 경우이다. 이때 서버는 이벤트 패킷을 컨트롤러로 전달하게 되고 이를 수신한 컨트롤러는 해당 서버를 포워딩 목록에서 제외한다. 이와 동시에 컨트롤러는 모든 서버의 부하율을 보고 받고 전체 서버의 부하율 평균값을 계산하게 된다. 계산된 값만큼 임계값을 조정한다. 조정되는 식은 아래와 같다.

$$TH_{after} = TH_{before} + \frac{1}{10}(TH_{before} - SL_{avg}) \quad (3)$$

TH_{after} : 조정 후 임계값
TH_{before} : 조정 전 임계값
SL_{avg} : 서버 부하율의 평균

두 번째는 서버의 부하율이 회복값에 도달한 경우이다. 이 경우에 서버는 컨트롤러에 보고하게 되고 보고받은 컨트롤러는 해당 서버를 포워딩 목록에 다시 포함한다.

Ⅳ. 실험 및 분석

4.1 실험 환경

제안하는 부하분산 기법을 네트워크 모델을 통해 구현하기 위해 Riverbed Modeler 17.5.pl6 버전을 사용하였다. 모델에는 부하분산 정책을 적용하는 컨트롤러와 컨트롤러의 정책에 따라 데이터를 포워딩하는 스위치,

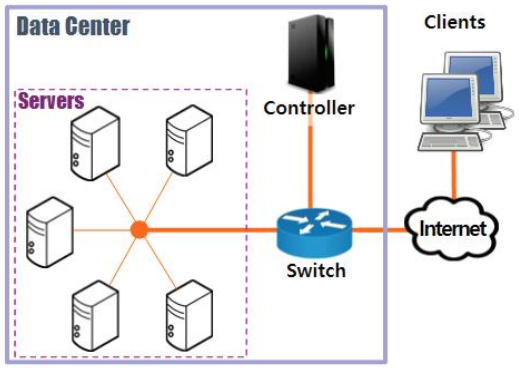

그림 4. 시스템 설계
Fig. 4. System Design

스위치에서 분배된 클라이언트 패킷을 수신하고 처리하는 서버, 지속적으로 패킷을 생성하는 클라이언트로 구성하였다.

4.2 실험방법

제안하는 기법의 효율성을 확인하기 위해 각 서버의 부하율과 임계값(Threshold)에 도달한 횟수를 기존 부하분산 방식과 비교하였다. 본 논문에서는 서버간의 성능차이를 부여한 상태에서 지속해서 유입되는 패킷에 따라 부하도를 측정하였다. 이때 최초의 임계값은 70%로 설정하였으며 회복값은 50%로 설정하였다. 서버에서 이벤트 패킷이 발생하면 임계값 및 회복값을 자동으로 조정하여 부하분산을 실시하였다. 이때 임계값과 회복값은 모든 서버에 동일한 값으로 적용한다.

서버는 5대로 구성하였으며 각 서버의 부하율의 차이를 주기 위해 서버별 성능을 다르게 구성하였다. 서버1의 데이터처리 속도를 100% 기준으로 하였을 때 서버2, 서버3, 서버4, 서버5를 95%, 90%, 85%, 80%로 구성하였다.

대조기법은 부하분산 시 서버의 부하율이 70% 고정된 임계값을 가지며 부하율의 50%로 고정된 회복값을 사용하며 서버별 부하율에 따른 패킷분배의 가중치를 사용한 J. Lee의 연구[10]의 방식을 비교하였다. 이를 통해 임계값이 가변적으로 변하는 경우 효율이 있는지를 확인하였다.

4.3 서버 부하율 분석

대조기법과 제안기법을 이용하여 서버별 부하율을 측정한 결과는 그림 5, 그림 6, 표 1, 표 2.와 같다. 대조기법과 제안기법의 부하율의 평균은 50.54%와 50.55%로 제안기법과 대조기법에 차이가 없었다. 하지만 최대 부하율의 경우 제안기법의 경우 모든 서버가 90% 미만이였지만 대조기법에서 서버5가 96.39%가 나왔다.

이러한 결과가 나온 이유로는 대조기법의 경우 패킷의 유입량과 상관없이 임계값이 고정되어 있기 때문이다. 대조기법의 경우 많은 패킷을 처리하게 되면서 모든 서버가 임계값을 넘게되면 임계값을 넘어 절반만 할당받는 서버가 없게 된다. 서버 부하율에 따른 가중치가 있더라도 처리속도를 능가하여 할당되게 되는 성능이 낮은 서버에 과부하가 발생한 것이다.

제안기법의 경우 대조기법과는 다르게 임계값에 도달한 서버가 있는 경우 서버 부하율의 평균값을 고려하여 임계값을 조정함으로써 특정 서버가 임계값을 초과하여 부하가 걸리는 현상을 예방할 수 있게 된다. 그로 인해 임계값에 도달한 서버는 패킷을 절반만 할당받게 되어 다른 서버보다 부하율을 낮출 수 있는 기회를 갖게 되는 것이다.

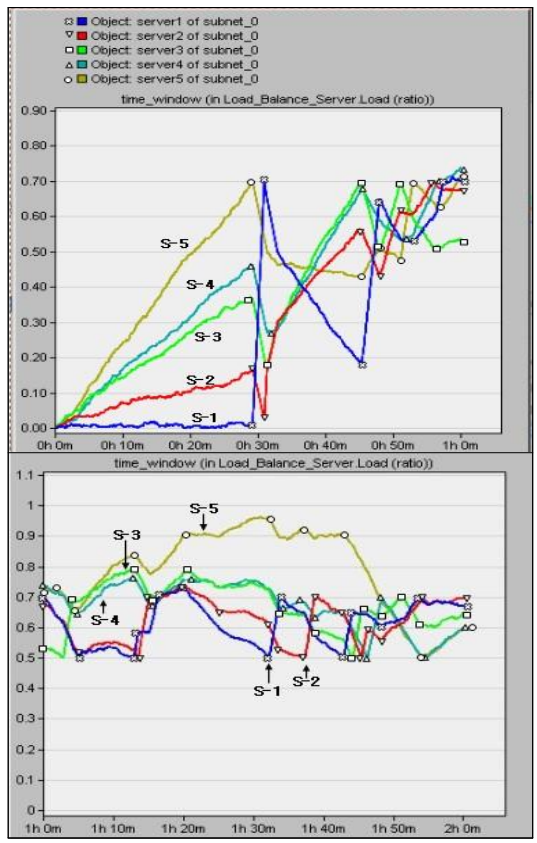

그림 5. 대조기법 서버별 부하율 및 부하율 평균
Fig. 5. Contrast Technique Server Load and Avg. Load

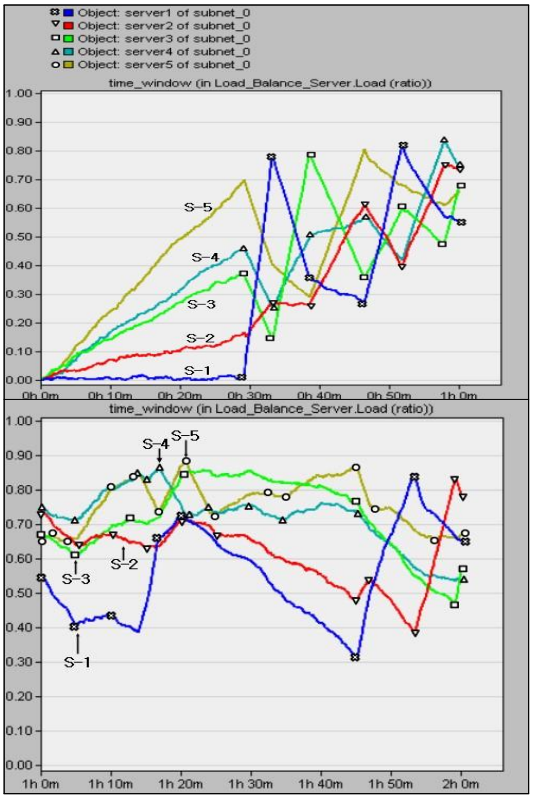

그림 6. 제안기법 서버별 부하율 및 부하율 평균
Fig. 6. Proposal Technique Server Load and Avg. Load

표 1. 대조기법 서버별 부하율 측정 결과
Table 1. Result of Contrast Technique Server Load

Object	Maximum	Average	Std. Dev.
Server1	0.7332	0.4263	0.2612
Server2	0.7386	0.4536	0.2387
Server3	0.7959	0.5131	0.2172
Server4	0.7677	0.5253	0.2098
Server5	0.9639	0.6085	0.2392
Total Avg.	-	0.5054	-

표 2. 제안기법 서버별 부하율 측정 결과
Table 2. Result of Proposal Technique Server Load

Object	Maximum	Average	Std. Dev.
Server1	0.8472	0.4048	0.2603
Server2	0.8397	0.4406	0.2401
Server3	0.8601	0.5314	0.2450
Server4	0.8646	0.5445	0.2301
Server5	0.8811	0.6060	0.2190
Total Avg.	-	0.5055	-

4.4 서버의 임계값 도달 횟수 분석

대조기법과 제안기법의 임계값 도달 횟수는 그림 7, 그림 8, 표 3.과 같다. 대조기법은 임계값에 도달한 총 횟수는 17번이고 제안기법의 임계값에 도달한 총 횟수는 12번이다. 제안기법이 대조기법보다 약 29.41% 가량 감소하였다.

서버의 부하율이 임계값보다 계속해서 높은 경우 횟수는 증가하지 않는다. 하지만 대체로 서버들의 부하율이 임계값에 가까워질수록 임계값 도달 횟수가 많아지게 된다. 대조기법의 경우 서버5가 임계값을 넘은 상태로 일정 유지됐음에도 불구하고 제안기법보다 횟수가 많이 나온 것은 그만큼 모든 서버가 임계값에 거의 근접하여 임계값을 넘은 서버가 발생할 확률이 높아졌다고 판단된다.

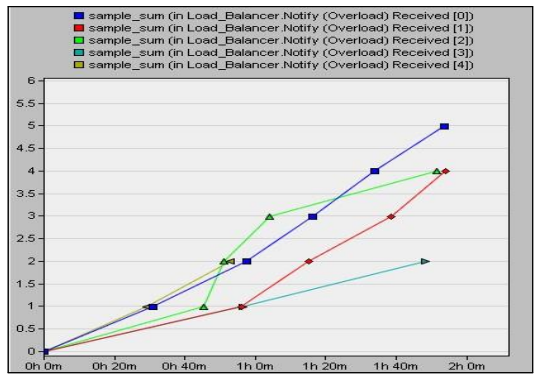

그림 7. 대조기법 임계값 도달 횟수
Fig. 7. Contrast Technique Threshold Reach Count

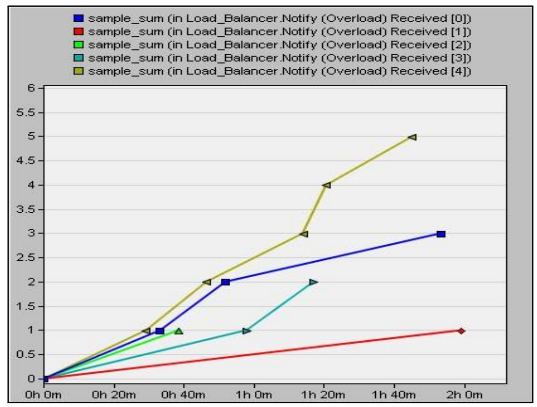

그림 8. 제안기법 임계값 도달 횟수
Fig. 8. Proposal Technique Threshold Reach Count

표 3. 대조기법과 제안기법의 임계값 도달 횟수
Table 3. Number of Threshold Reaches for Contrast and Proposal Technique

Object	Contrast	Proposal
Server1	2	5
Server2	2	2
Server3	4	1
Server4	4	1
Server5	5	3
Total	17	12

4.5 임계값 조정에 따른 부하율 분석

위 실험에서는 제안기법의 경우 임계값 조정 비율을 서버부하량 평균과 임계값 차이의 $\frac{1}{5}$ 만큼 조정하였다. $\frac{1}{5}$ 값이 적당하다고 판단되어 설정하였지만 이는 얼마든지 변경이 가능하다. 이 값을 변수로 조정하였을 때 서버 부하율의 값을 조정하는 식은 아래와 같다.

$$TH_{after} = TH_{before} + \alpha(TH_{before} - SL_{avg}) \quad (3)$$

TH_{after} : 조정 후 임계값

TH_{before} : 조정 전 임계값

SL_{avg} : 서버 부하율의 평균

α : 임계값 조정 변수

α 값에 따른 차이를 확인하기 위해 추가로 α 값을 3가지로 설정하여 실험하였다. α의 값을 $\frac{1}{5}$, $\frac{1}{10}$, $\frac{1}{15}$일 때 실험결과는 그림 9, 그림 10, 그림 11, 표 4.와 같다.

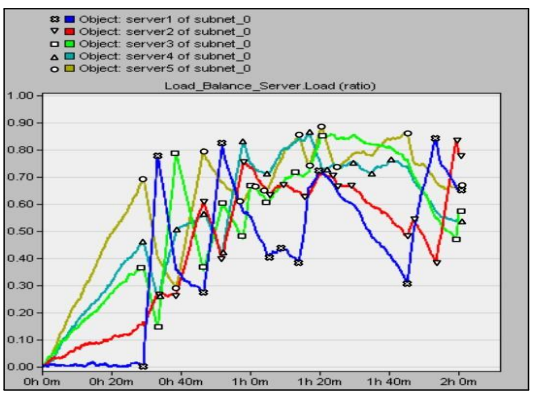

그림 9. 임계값 조정변수 1/5
Fig. 9. Threshold adjustment variable 1/5

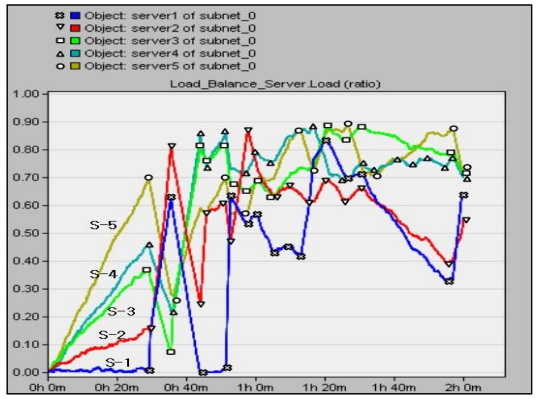

그림 10. 임계값 조정변수 1/10
Fig. 10. Threshold Adjustment Variable 1/10

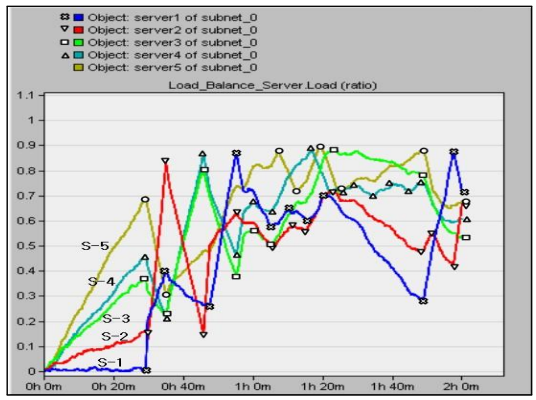

그림 11. 임계값 조정변수 1/15
Fig. 11. Threshold adjustment variable 1/15

표 4. 임계값 조정변수에 따른 서버부하율
Table 4. Server Load Rate According to Threshold Adjustment Variable

Adjustment Variable	Maximum Avg.	Average	Std dev Avg.
1/5	0.8585	0.5055	0.2389
1/10	0.8755	0.5197	0.2527
1/15	0.8773	0.5054	0.2409

위 실험 결과를 통해서 임계값 조정변수를 변경함에 따라 전체적인 서버의 부하율에는 차이가 없는 것을 알 수 있었다. 하지만 조정변수에 따라 서버 간의 부하율 최댓값이 다른 것을 확인하였다. 이는 조정변수가 부하율이 높은 서버의 회복 속도와 연관이 될 수 있음을 의미한다. 따라서 서버 간의 성능 차이와 패킷 유입량의 변동에 따라 적절한 조정변수의 설정이 필요하다고 판단된다.

V. 결 론

본 논문에서는 데이터센터의 SDN 환경에서 서버의 부하분산을 효율적으로 하기 위해 각 서버의 부하율을 이용한 가변적인 임계값을 사용하는 기법을 제안하였다. 가변적인 임계값을 적용함으로써 대조기법과 비교하여 서버 부하율의 평균을 낮추어 효율적으로 부하분산을 하는데 효과적임을 알 수 있었다.

제안기법은 데이터센터의 서버들의 성능에 대비하여 클라이언트의 요청량에 따라 평균 부하량이 계산되어 최적의 임계값으로 자동으로 조정된다는 의미를 가지고 있다. 최적의 임계값 조정변수에 대한 연구는 추가적으로 필요하지만 제안기법의 효과는 앞으로 더욱 커질 것으로 기대되며 부하분산 기법의 효과적인 하나의 선택지가 될 것이다.

References

[1] Simon Kemp, "*Digital 2024: Global overview report*," Article of DataReportal, Jan. 31, 2024,

[2] B. Yoon, "Future networking technology of SDN," *Electr. and Telecommun. Trends*, vol. 27, no. 2, pp. 129-136, 2012. (https://doi.org/10.22648/ETRI.2012.J.270214)

[3] IDC, "*Worldwide data center software-defined networking forecast*," Report, pp. 2019-2023, Nov. 2019.

[4] S. Kang, Y. Kim, and S. Yang, "SDN core technology and evolution prospect analysis," *Inf. & Commun. Mag.*, vol. 30, no. 3, 2013.

[5] Y. Seo and M. Lee, *Understanding OpenFlow using open source Introduction to SDN*, Seoul, YOUNGJIN.COM, 2014

[6] P. Deepalakshmi, "D-serv LB: Dynamic server load balancing algorithm," *Int. J. Commun. Syst.*, vol. 32, no. 1, pp. 293-310, 2019. (https://doi.org/10.1002/dac.3840)

[7] J. Kim and T. Kwon, "Efficient load balancing technique considering data generation form and server response time in SDN," *J. Korea Inst. Electr. Commun. Sci.*, vol. 15, no. 4, pp. 679-686, 2020. (https://doi.org/10.13067/JKIECS.2020.15.4.679)

[8] J. Lee and T. Kwon, "Efficient load balancing technique through server load threshold alert in SDN," *J. Korea Inst. Electr. Commun. Sci.*, vol. 16, no. 5, pp. 817-814, 2021. (https://doi.org/10.13067/JKIECS.2021.16.5.817)

[9] K. Lee and T. Kwon, "Server state-based weighted load balancing techniques in SDN environments," *J. Korea Inst. Electr. Commun. Sci.*, vol. 17, no. 6, pp. 1039-1046, 2022. (https://doi.org/10.13067/JKIECS.2022.17.6.1039)

[10] J. Lee and T. Kwon, "Study of load Balancing technique based on step-by-step weight considering server status in SDN environments," *J. Korea Inst. Electr. Commun. Sci.*, vol. 18, no. 6, pp. 1087-1094, 2022. (https://doi.org/10.13067/JKIECS.2023.18.6.1087)

이 강 욱 (Kang-uk Lee)

2013년 2월 : 해군사관학교 전산과학과 졸업

2023년 3월~현재 : 국방대학교 컴퓨터공학과 석사과정

<관심분야> 통신공학, 인공지능

[ORCID:0009-0001-8413-0130]

권 태 욱 (Tae-wook Kwon)

1986년 2월 : 육군사관학교 컴퓨터공학과 졸업

1995년 6월 : 미 해군대학원 컴퓨터공학과 석사

2001년 2월 : 연세대학교 컴퓨터공학과 박사

<관심분야> 컴퓨터통신 & 네트워크, 센서네트워크, CCN, SDN & NFV, 가상현실과 증강현실

[ORCID:0000-0003-2880-9058]