SDN Lullaby: VM Consolidation for SDN using Transformer-Based Deep Reinforcement Learning

Eui-Dong Jeong, Jae-Hyoung Yoo, and James Won-Ki Hong

Department of Computer Science and Engineering, POSTECH, Pohang, Korea
{justicedong, jhyoo78, jwkhong}@postech.ac.kr

***Abstract*—This study introduces Virtual Machine (VM) Consolidation using a Transformer-based Deep Reinforcement Learning (DRL) method, to address the complexity and inefficiency in operating Software Defined Networks-enabled Network Function Virtualization (SDN-enabled NFV). The distribution of Virtual Network Functions (VNFs) as VMs across servers often leads to energy loss due to irregular deployment. The proposed approach enhances energy efficiency while maintaining the performance of Service Function Chains (SFCs). By refining the VM consolidation process and leveraging a more sophisticated DRL method, this approach promises a more efficient solution to VM consolidation in SDN-enabled NFV environments.**

***Index Terms*—SDN, NFV, Energy Efficiency, VM Consolidation, Deep Reinforcement Learning**

I. Introduction

Today's networks often exhibit overly complicated structures due to the increasing number of devices dedicated to specific functions, developed to meet various business requirements. The network infrastructure that facilitates the connection and configuration of these devices has grown increasingly complex. It leads to diminished flexibility and challenges in responding to dynamic market demands [1].

While Software Defined Network-enabled Network Function Virtualization (SDN-enabled NFV) emphasizes network flexibility and efficiency, achieving this in practice is not straightforward. Typically, Virtual Network Functions (VNFs) are distributed in the form of containers or Virtual Machines (VMs), and they are allocated to servers, utilizing resources such as CPU, memory, and bandwidth. If VNFs are not properly distributed across servers, energy loss will occur by maintaining unnecessary servers [2]. Management features such as Auto Scaling can cause VNFs to be deployed irregularly and inefficiently, especially in the late night and early morning hours, when the overall traffic reduced.

Several methods have been proposed to address this problem in both hardware and software. Hardware solutions have made significant progress in improving energy efficiency. However, they have not effectively addressed the inefficient distribution of software-based VNFs. In other words, a software-based method is needed to effectively manage and operate VNFs. One such method is VM Consolidation, which optimally distributes VMs across physical servers to enhance energy efficiency. Previous studies have explored rule-based [2], Machine Learning (ML), and Reinforcement Learning (RL) [3] [4] approaches for VM Consolidation. However, these VM Consolidation studies have several limitations:

- Loss of information due to the conventional VM Consolidation procedure.
- Use of insufficient methods to account for VM relationships.

To address these limitations, this study proposes VM Consolidation using a Transformer-based Deep Reinforcement Learning (DRL) approach and refining the VM Consolidation procedure. The proposed method aims to overcome the aforementioned challenges and enhance the efficiency of VM distribution and management in SDN-enabled NFV.

II. Related Works

Various methods have been proposed to improve server energy efficiency (Fig. 1). Initially, hardware-based approaches such as Cooling Systems or Power Controllers were introduced. Also, a method such as Dynamic Voltage and Frequency Scaling (DVFS) was proposed [5]. This has benefited from the server's power supply, but it operates on a per-server basis rather than per-VM basis. However, VM Consolidation presents system control on a VM basis [3] [4] [6].

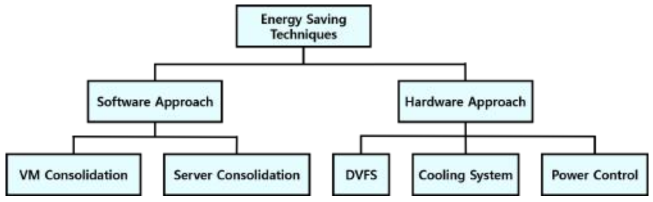

Fig. 1. Energy Saving Techniques

VM Consolidation aims to distribute and schedule VMs effectively in multi-server environments, such as Cloud Data Centers (CDCs), to minimize server energy consumption while maintaining the quality of services running inside each VM. Conventionally, VM Consolidation encompasses a three step decision making process (Fig. 2) [6]. The first step, Server Selection (or Host Detection), determines which server to extract the VM from. The second step, VM Selection, identifies the VM to be extracted from the chosen server. Finally, in the VM Placement step, a server is selected for the VM to be relocated to. Several studies have proposed which algorithm to use at each step [7]. However, these VM Consolidation approaches is not suitable for managing VMs in the SDN-enabled NFV environment, despite being effective for commercial services in CDCs. This is because it does not consider the performance of the service function chain (SFC) at all. For instance, the overall processing time of an SFC can be reduced by placing all VNFs belong to single SFC to the same server or edge. Therefore, VM Consolidation for SDN-enabled NFV have to take these requirements into account.

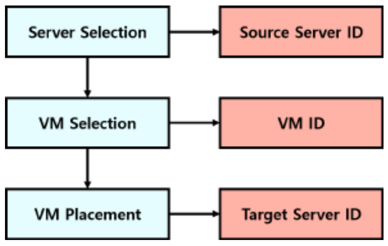

Fig. 2. VM Consolidation Procedure

Clearly, a series of studies have attempted to adapt VM consolidation to the SDN-enabled NFV environment. Initially, rule-based methods were employed; however, due to the NP-hard nature of VM Consolidation in SDN-enabled NFV environments, this approaches were computationally expensive and suboptimal [2]. Consequently, several high-performance methods utilizing Deep Reinforcement Learning (DRL) have been proposed [3] [4] [6]. The commonly applied methodology utilizes the simple deep learning network, which learns the value of taking actions in specific states to find the optimal policy. However, this methodology has limitations, so we incorporates Self-Attention Mechanism (SAM), components of Transformer [8], that can reflect the relationship between multi elements data like VNFs or Servers.

Furthermore, this study proposes the elimination of the Server Selection step from the conventional three step VM Consolidation process. The Server Selection step provides only reference information for the VM Selection step, as designing a system that favors VMs belonging to a specific server may introduce biases due to information loss. If the Server Selection step can consistently provide optimal results, then including it will be a good choice for performance. However, based on my current understanding, existing research cannot provide assurance regarding this matter.

III. Implementation

A. Requirements

In this study, we aim to optimize SDN-enabled NFV during low-traffic periods, such as dawn, by redistributing VNFs implemented through VMs. Therefore, we named the implementation **SDN Lullaby**. Before presenting the specific implementation, we define the following requirements:

1) VM Consolidation is performed on edge-by-edge basis. This means that we don't consider about moving VNFs to other edge.
2) This study assumes a low-traffic period with minimal services and no deployment of new VNF during VM Consolidation.
3) Computing resources of each server are measured based on CPU and memory usage.
4) We have two goals. First is to minimize the number of servers hosting VMs with VNFs installed to reduce energy consumption. Second is maximizing the network processing performance of SFCs. For this, this system considered to reduce processing time for SFC. Simply, if we make all VNFs belong to a specific SFC position to same server, we can reduce the SFC processing time. So, if the number of SFCs with all VNFs installed on the same server increase, then SFCs could perform better.
5) The reward in this system considers both energy efficiency and SFC processing time. In other words, the objective is to maximize the number of servers without any VNFs installed and maximize the number of SFCs with all VNFs on the same server.

B. Environments

Before introducing our DRL method, we explain the environment in which the model operates.

1) State: We defined four types of information that can be acquired on Edge Networks.

- Edge Information: Edge's CPU/Memory Capacity, and CPU/Memory Load
- Each Server Information: Each Server's Id, CPU/Memory Capacity, and CPU/Memory Load
- Each SFC Information: Each SFC's Id, CPU/Memory Request
- Each VNF Information: Each VNF's Id, CPU/Memory Request, installed Server Id, and included SFC Id

As a result, we obtained the following structured State data:

$$\begin{aligned} &State = \{\\ &\quad \text{Edge information},\\ &\quad \text{List of Each Server Information},\\ &\quad \text{List of Each SFC Information},\\ &\quad \text{List of Each VNF Information}\\ &\} \end{aligned}$$

2) Action: What the system wants is to move a particular VM to a particular server. Accordingly, in the corresponding environment, the action is expressed as a pair (VM ID, Server ID). This is possible because unlike the existing VM Consolidation method discussed earlier, it consists of only two steps instead of three steps. In previous studies, VM was selected after identifying the server to remove VM, but in this paper, VM is directly selected without the Server Selection step (**VM Selection**). And, the server to install VM is also selected immediately (**VM Placement**).

$$\text{Action} = \{\text{VM ID}, \text{Server ID}\}$$

3) Rewards: The purpose of the system is to maximize the number of servers that do not have any VNFs installed, and maximize the number of SFCs with all VNFs on the same server. Therefore, we use a simple reward formula defined below, that the total number of servers is N_{Server}, the number of servers without any VNFs installed is N^0_{Server}, and total number of SFCs is N_{SFC}, $N^{\forall}_{\text{SFC}}$ is the number of SFCs with all VNFs installed on the same server.

$$\text{Reward} = N^0_{\text{Server}}/N_{\text{Server}} + N^{\forall}_{\text{SFC}}/N_{\text{SFC}} \qquad (1)$$

$$0 \leq \text{Reward} \leq 2$$

In this formula, energy efficiency and SFC performance were considered with the same weight as follows, but it is also possible to design them to give higher weights to specific value by changing them.

4) Emulation: Training DRL in a real SDN-enabled NFV system is extremely expensive. Adding VNFs, deleting them, and redirecting SFCs again excessively increases the learning time at each step. Therefore, learning through an emulation environment is suitable for overall training. So, we directly implemented this emulation environment for training and evaluation.

C. Architecture

Our agent has 3 components, preprocessor, DRL model, and postprocessor. The preprocessor simply reformats the state to DRL model's input, and the DRL model outputs values of each VNF/server. Then, finally, the postprocessor chooses an action based on these values. (Fig. 3)

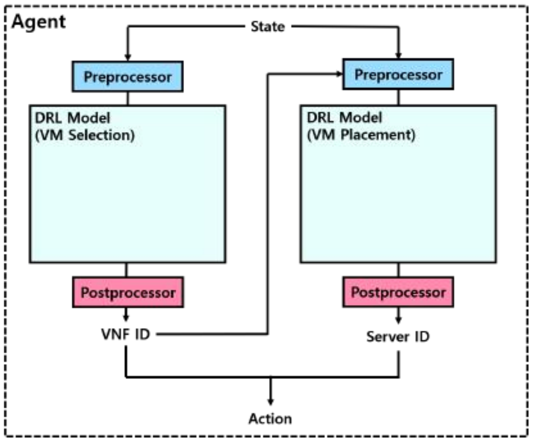

Fig. 3. Agent Architecture

1) Preprocessing: The preprocessor reformats the state, as defined in the Environment Section III-B, to serve as input for the DRL model. Since the Agent performs two decision-making steps (VM Selection and VM Placement), separated input formatting is required for each step. In the VM Selection step, the preprocessor extract information of edge, each VM, and each server from state. In the VM Placement step, the preprocessor extract information of edge, target VM, and each server from state and result of VM Selection. Therefore, as shown in TABLE I, input is transmitted in three dimension tensor data.

	VM Selection	VM Placement
total Dimension	3	3
Dimension 1	Batch Size	Batch Size
Dimension 2	Number of Maximum VNF	Number of Server
Dimension 3	11	11

TABLE I
SDN LULLABY INPUT DIMENSION

In particular, in order to deliver information on each VNF and server in Dimension 3, 11 data were extracted from the State in the form presented in TABLE II. And zero padding was applied to VM that do not exist for batch learning because the number of VMs for each episode continues to change.

VM Selection	VM Placement
VNF CPU Request	Server CPU Capacity
VNF Memory Request	Server CPU Load
Placed Server CPU Capacity	Server Memory Capacity
Placed Server CPU Load	Server Memory Load
Placed Server Memory Capacity	Selected VNF CPU Request
Placed Server Memoy Load	Selected VNF Memory Request
Placed SFC ID	Selected VNF's SFC ID
Edge CPU Capacity	Edge CPU Capacity
Edge CPU Load	Edge CPU Load
Edge Memory Capacity	Edge Memory Capacity
Edge Memory Load	Edge Memory Load

TABLE II
SDN LULLABY INPUT DATA

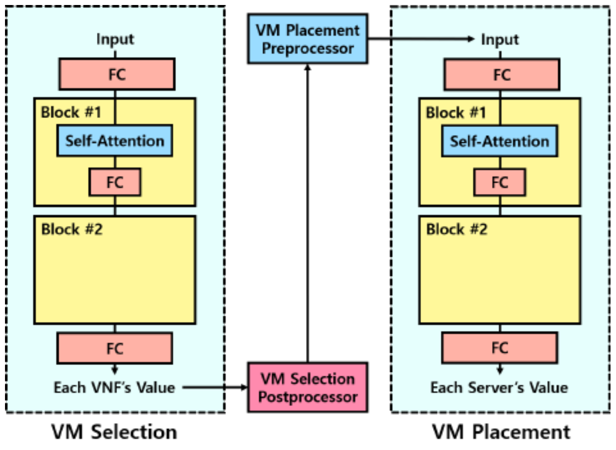

Fig. 4. DRL Model Architecture

2) DRL model: The overall network structure is shown in Fig. 3. As described in the previous section, the step of VM Consolidation was compressed from step 3 to step 2. Therefore, the entire system was expressed through two steps, VM Selection and VM Placement. First, in the VM Selection, a neural network estimate the value of each VNF. In the same way, the VM Placement has the same structure and performs it. The only difference is they output each server's value.

In Fig 4 architecture, each network was divided into three parts: input layer, hidden blocks, and output layer. In the input layer, a single Fully Connected (FC) layer was used to perform simple encoding on data to the input for the hidden blocks. The hidden blocks utilized the SAM presented in Transformer Architecture. Compared to Multi Layer Perceptron (MLP), SAM has the advantage of being able to estimate the hidden value in consideration of correlation between each VNF/server that cannot be performed in MLP. In other words, given information on all VNFs/servers included in the edge, it is advantageous to use the relationships between them to select the most appropriate VNF to move and find the server to locate. In addition, it can bring greater advantages in inference time and learning time in that parallel processing is possible compared to Recurrent Neural Network (RNN). In the process of applying Self-Attention, there is a process of performing position encoding to convey the position information of each sequence, but this information is not important in this system and does not need to be considered. In addition, we choose Query, Key, and Value for SAM all took the same value. Finally, in the output layer, the data input from each VNF/server is compressed into one value through the FC layer. This value can be thought of as a value for each VNF/server.

3) Postprocessing: Lastly, this study goes through the process of filtering and selecting the output of the DRL model. The default is to select the VNF/server with the highest value, but the following impossible or meaningless actions are prevented from being performed through filtering. This method is one of the ways to handle impossible actions in DRL.

- If no server has sufficient capacity and cannot move a VNF to any server, it prevents the selection of that VNF. (VM Selection)
- If the server is original location of selected VNF, it prevents the selection of that server. (VM Placement)
- If the capacity of the server is not sufficient to move the VNF, prevent the selection of that server. (VM Placement)

In other words, after extracting the value to select each VNF and server extracted through the network, in case of an impossible action, 0 was masked in the output value through filtering to prevent the corresponding value from being selected.

IV. Experiments and Evaluation

A. Environments

We conducted performance evaluation of **SDN Lullaby** in the following experimental environment. The experiments were performed using a Quadro RTX 5000 GPU. The emulation environment, as described in Section III-B4, was utilized for the experiments. The parameters presented in TableIII were used as inputs for the emulation process.

Each Server CPU capacity	Each Server Memory capacity	Number of Server	Number of SFC
12	32	8	8

TABLE III
Emulation Environment Parameters

Further implementation details and code contents have been uploaded to the public GitHub repository. (https://github.com/euidong/sdn-lullaby)

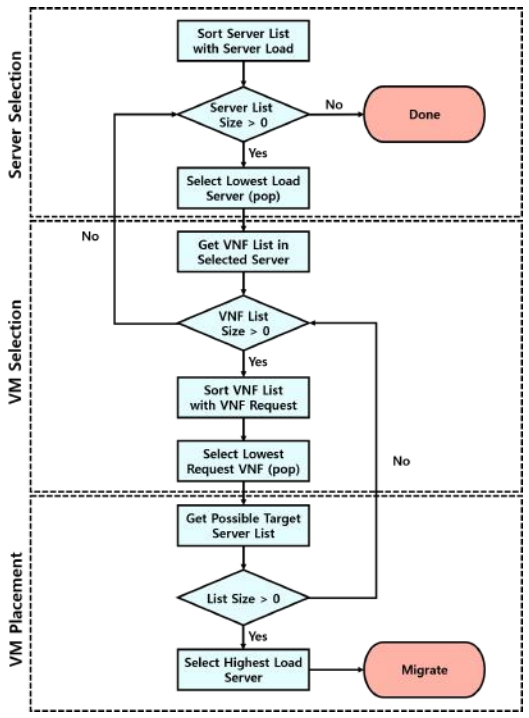
Server Selection
Sort Server List with Server Load
Server List Size > 0
No
Done
Yes
Select Lowest Load Server (pop)
VM Selection
No
Get VNF List in Selected Server
VNF List Size > 0
Yes
Sort VNF List with VNF Request
Select Lowest Request VNF (pop)
No
VM Placement
Get Possible Target Server List
List Size > 0
Yes
Select Highest Load Server
Migrate

Fig. 5. Rule-based Approach Procedure

B. Compare with baselines

First, we will show the results of performance comparison with the existing system. We use a baseline rule-based model. It was implemented to follow Fig. 5. This is an slight variation of V.Eramo et al [2].

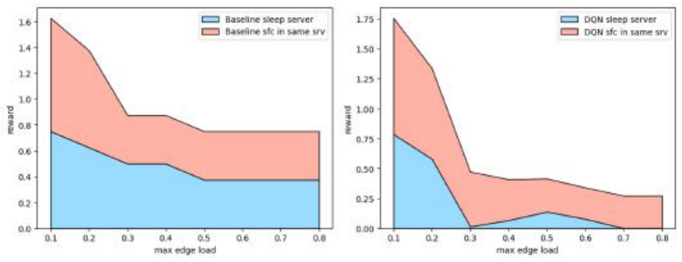

Fig. 6. Reward according to edge load, (left) is baseline agent's result, and (right) is DQN agent's result.

Fig. 6 illustrates how the reward of each method changes as the edge load varies. For edge loads of 0.1, and 0.2, the overall performance of the DRL implementation is higher than that of the rule-based method. However, as the edge load increases, it can be observed that the rule-based method outperforms the DRL methods. This performance difference can be attributed to the conflicting requirements of energy efficiency and SFC performance. However, for requirements during low-traffic periods when edge load is minimal, this performance difference is not significant.

C. Effect of Step Compression

This evaluation compares the results of actually removing one step from VM consolidation. Here, we set the maximum edge load to 0.2 and compared the DRL model with and without the server selection step of the baseline model. In Figure 7, we can see that the overall training time is reduced by removing steps. So we can see that two step inference is much faster than three step inference. Additionally, the two step model reached rewards above 1.4, while the three step model only reached around 1.2. It can be seen that performance has improved as information loss between each step can be reduced.

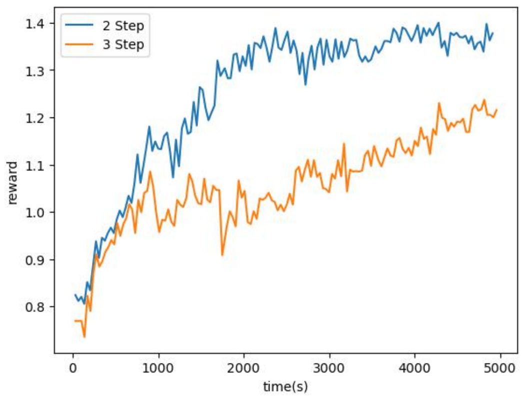

Fig. 7. Step Compression Effect

D. Effect of Self Attention Mechanism

This evaluation shows the performance comparison of the SAM method and LSTM [9] widely used in existing papers (Fig. 8). First of all, it can be seen that the method utilizing LSTM takes remarkably long training time because parallel processing is impossible due to structural limitations. Processing time to process the same step was required more than 10 seconds per 200 episodes. In addition, it can be seen that SAM is more advanced in terms of performance. The SAM based model reached a reward of 1.4 or higher, but the LSTM based model only reached around 1.3.

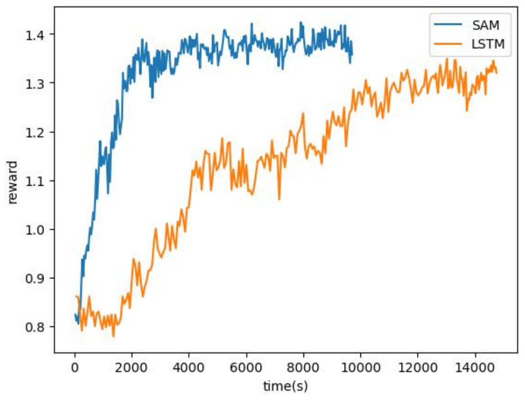

Fig. 8. SAM and LSTM Performance

V. Conclusion

The proposed approach aims to increase energy efficiency while preserving or enhancing network quality. By operating entirely based on DRL without the need for manual intervention, it offers the following contributions:

- Reducing the VM Consolidation process from three steps to two steps, which not only reduces inference and training time but also offers advantages in terms of performance.
- Providing an alternative Full DRL approach to designing SDN-enabled NFV management.

Although this study makes important contributions, there are some limitations that should be considered. First, the current implementation relies on a single FC layer to encode the input data. Utilizing advanced encoding methods can potentially improve performance. Second, the evaluation is limited to the currently implemented emulation environment. In further research, the proposed approach should be applied to a real SDN-enabled NFV environment to demonstrate its effectiveness through concrete experimental results.

Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (2018-0-00749, Development of Virtual Network Management Technology based on Artificial Intelligence) and Smart HealthCare Program(www.kipot.or.kr) funded by the Korean National Police Agency(KNPA, Korea) [Project Name: Development of an Intelligent Big Data Integrated Platform for Police Officers' Personalized Healthcare / Project Number: 220222M01]

References

[1] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262, 2016.

[2] V. Eramo, A. Tosti, and E. Miucci. Server resource dimensioning and routing of service function chain in nfv network architectures. *Journal of Electrical and Computer Engineering*, 2016:7139852, Apr 2016.

[3] Jing Zeng, Ding Ding, Kaixuan Kang, HuaMao Xie, and Qian Yin. Adaptive drl-based virtual machine consolidation in energy-efficient cloud data center. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2991–3002, 2022.

[4] Rachael Shaw, Enda Howley, and Enda Barrett. Applying reinforcement learning towards automating energy efficient virtual machine consolidation in cloud data centers. *Information Systems*, 107:101722, 2022.

[5] Peyman Mokaripoor and M Hosseini Shirvani. A state of the art survey on dvfs techniques in cloud computing environment. *J. Multidiscip. Eng. Sci. Technol*, 3(5):4740–4743, 2016.

[6] Bhagyalakshmi Magotra, Deepti Malhotra, and Amit Kr. Dogra. Adaptive computational solutions to energy efficiency in cloud computing environment using vm consolidation. *Archives of Computational Methods in Engineering*, 30(3):1789–1818, Apr 2023.

[7] Zhihua Li, Xinrong Yu, Lei Yu, Shujie Guo, and Victor Chang. Energy-efficient and quality-aware vm consolidation method. *Future Generation Computer Systems*, 102:789–809, 2020.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.