Predictive Analytics based Knowledge-Defined Orchestration in a Hybrid Optical/Electrical Datacenter Network Testbed

Hongqiang Fang, Wei Lu, Qinhezi Li, Jiawei Kong, Lipei Liang, Bingxin Kong, and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—For datacenter networks (DCNs), it is always important to have an effective network orchestration scheme that can coordinate the usages of IT and bandwidth resources timely. In this work, we consider the hybrid optical/electrical DCNs (HOE-DCNs) and propose a knowledge-defined network orchestration (KD-NO) system for them. The KD-NO system follows the predictive analytics in human behaviors, which includes forecasting based on memory and decision making based on knowledge. To explain the design of our KD-NO system, we first discuss how to fetch low-level knowledge from the telemetry data about the resource utilization in an HOE-DCN. Then, we describe how to optimize the HOE-DCN's configuration for network orchestration. Specifically, we design an online scheme based on deep reinforcement learning (DRL), and make sure that it can extract high-level knowledge from the low-level input and come up with optimal HOE-DCN configurations on-the-fly. We prototype the proposed KD-NO system and demonstrate it in an HOE-DCN testbed. The experiments run Hadoop applications in the testbed and show that our KD-NO system can make timely and correct decisions in different experimental schemes by leveraging the two-level knowledge, maintain a high matching degree between the HOE-DCN's configuration and the applications running in it, and thus effectively reduce the job completion time.**

***Index Terms*—Datacenter networks (DCNs), Network orchestration, Knowledge-defined networking (KDN), Deep learning, Deep reinforcement learning, Artificial intelligence (AI).**

I. Introduction

RECENTLY, due to the rapid development of cloud computing and Big Data analytics [1, 2], cloud traffic has been increasing exponentially, the majority of which is within datacenters (DCs) [3, 4]. Hence, DC networks (DCNs) are facing challenges to accommodate such enormous traffic cost-efficiently with sustainable technologies. Compared with electrical packet switching (EPS), optical circuit switching (OCS) provides larger bandwidth capacity and consumes less power [5–8]. To this end, the hybrid optical/electrical DCN (HOE-DCN) architecture that can seamlessly integrate the benefits of EPS and OCS has attracted intensive interests from both academia and industry [9, 10].

The top-of-rack (ToR) switches in an HOE-DCN are interconnected by two inter-rack networks based on EPS and OCS, respectively. Although HOE-DCNs are promising, they cannot work around one of the most challenging demands for DCNs, *i.e.*, an effective network orchestration scheme that can coordinate the usages of IT and bandwidth resources proactively and timely for ensuring various quality-of-service (QoS) requirements [11–14]. This demand actually becomes even more challenging in an HOE-DCN, since the addition of the OCS-based inter-rack network complicates its network control and management (NC&M) [15, 16]. In other words, an HOE-DCN operator has more heterogeneous network elements (NEs) to coordinate for maintaining a high matching degree between the HOE-DCN's configuration and the huge volume of network applications running in it.

With software-defined networking (SDN) [17–19], people can architect more programmable, effective and reliable NC&M for DCNs. However, the NC&M is still reactive, which means that the controller always makes decisions based on the current network status. This would limit a DCN operator's capability of guaranteeing various and stringent QoS demands. Therefore, we expect the automation and agility of the NC&M in HOE-DCNs to involve knowledge-defined networking (KDN) [20, 21]. KDN is essentially the symbiosis of SDN and artificial intelligence (AI). Specifically, with the centralized control in SDN, the operator visualizes its HOE-DCN by collecting rich telemetry data proactively, and then, it leverages AI-assisted data analytics to abstract knowledge from the data through deep learning (DL) or deep reinforcement learning (DRL) and uses the knowledge to reach smart decisions for automatic and agile NC&M.

The actual implementation of KDN in HOE-DCNs still faces a few open and challenging problems. First of all, to satisfy the QoS requirements of applications, the KDN-based scheme needs to orchestrate the IT and bandwidth resources in an HOE-DCN, *i.e.*, managing not only the NEs in the EPS/OCS-based inter-rack networks but also the virtual machines (VMs) and the servers. This escalates the KDN-based NC&M to knowledge-defined network orchestration (KD-NO). Secondly, the amount of telemetry data could be large, while the useful information buried in it is usually sparse. How to extract knowledge from it would be an interesting but challenging problem. Finally, the knowledge extracted from telemetry data, *i.e.*, the future trends of IT and bandwidth utilizations in the HOE-DCN, is still fragmented and in the low-level, and thus it cannot be directly used for network orchestration. In other words, the KD-NO needs to further abstract high-level knowledge regarding the matching degree

H. Fang, W. Lu, Q. Li, J. Kong, L. Liang, B. Kong, and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: zqzhu@ieee.org).

Manuscript received on May 16, 2019.

between the HOE-DCN's configuration and the network applications running in it, to make wise decisions.

In this work, we develop a KD-NO system for HOE-DCNs by extending our preliminary study in [16], to address the aforementioned problems. The KD-NO system follows the predictive analytics in human behaviors. Specifically, it first forecasts VMs' future demands on IT and bandwidth resources (*i.e.*, forecasting based on memory) and then finds the optimal HOE-DCN configuration based on the predictions (*i.e.*, decision making based on knowledge). Here, the HOE-DCN configuration refers to how to place the VMs in server racks and how to route the VM traffic through the EPS/OCS-based inter-rack networks. To accomplish the predictive analytics, our KD-NO system collects telemetry data regarding the resource utilizations in the HOE-DCN, analyzes the spatial and temporal correlations of the data, and predicts future workloads and traffic matrix of VMs with several DL modules (*i.e.*, fetching the low-level knowledge). Then, it leverages the low-level knowledge for network orchestration, which is formulated as a mixed integer linear programming (MILP) model and is proven as an $\mathcal{NP}$-hard problem. Hence, we design a DRL-based online scheme to solve it, which extracts high-level knowledge from the low-level input and gets optimal HOE-DCN configurations on-the-fly.

We prototype the proposed KD-NO system and experimentally demonstrate it in a small-scale but real HOE-DCN testbed. The experiments run Hadoop applications [22] in the testbed to evaluate our proposal. The results show that: 1) the KD-NO system can accurately predict the workloads and traffic matrix of the VMs running Hadoop applications, *i.e.*, fetching the low-level knowledge successfully, 2) it can coordinate the IT and bandwidth resources in the HOE-DCN timely without disturbing the active applications, and 3) taking the low-level knowledge as inputs, the DRL-based online scheme can extract and learn the high-level knowledge quickly, and make wise decisions proactively to maintain a high matching degree between the HOE-DCN's configuration and the applications running in it.

The rest of the paper is organized as follows. Section II reviews the related work. In Section III, we introduce the architecture of our KD-NO system. The designs for fetching the low-level knowledge about an HOE-DCN are discussed in Section IV, while how to extract the high-level knowledge and use it for proactive network orchestration are presented in Section V. We show the experimental demonstrations in Section VI. Finally, Section VII summarizes the paper.

II. Related Work

The architectures of HOE-DCNs were proposed in [9, 23, 24] to seamlessly integrate the advantages of both EPS and OCS. These proposals can potentially address the bandwidth crunch and ever-increasing energy consumption in DCNs, and the transition from the traditional DCNs to them can be conducted smoothly. Meanwhile, to fully explore the advantages of HOE-DCNs, an effective network orchestration scheme would be required to coordinate the IT and bandwidth resources in them [11, 25–27], which generally needs to worry about three problems, *i.e.*, routing the inter-rack traffic among VMs, configuring the OCS and EPS inter-rack networks, and placing VMs in the server racks.

The studies in [9, 23, 24] considered how to configure the OCS and EPS inter-rack networks in an HOE-DCN, for satisfying VM traffic demands. However, they still have limitations. Firstly, they either configured an HOE-DCN reactively based on current network status, or just predicted future traffic roughly without considering its temporal and spatial correlations, which is known to be harmful for maintaining the prediction accuracy [28, 29]. Secondly, they did not discuss how to reconfigure the inter-rack networks to address dynamic traffic demands. Lastly but most importantly, they did not tackle the VM placement problem [30] and thus the IT resource usages in the HOE-DCNs might not be well coordinated.

The joint optimization of VM placement and inter-rack traffic routing in traditional EPS-based DCNs has been addressed in [31–33]. Nevertheless, due to the absence of the OCS-based inter-rack network, their solutions cannot be directly applied to HOE-DCNs, and more importantly, they did not tried to leverage DL/DRL to realize proactive and agile network orchestration. More recently, people try to incorporate DL or DRL in the network orchestration of DCNs [34, 35]. However, these studies neither covered the full spectrum of KD-NO nor introduced multiple AI modules to optimize the network orchestration collaboratively. In terms of the system architecture of KD-NO, the open network automation platform (ONAP) [36] laid out the modular design to coordinate the IT and bandwidth resources in a DCN to follow the loop of collection, analysis, and decision making. This provides us the blueprint to architect our KD-NO system. But as ONAP put its emphasis on orchestrating virtual network functions, we extend it to consider general DCN applications and to leverage multiple AI modules and multi-level knowledge, for making smart decisions to configure an HOE-DCN.

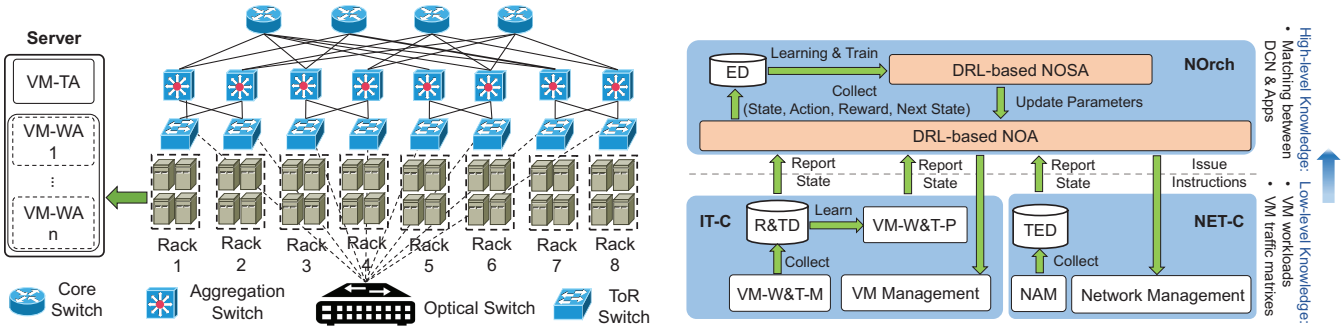
III. System Architecture

In this section, we explain the system architecture of the HOE-DCN with KD-NO, which is illustrated in Fig. 1.

A. Hybrid Optical/Electrical DCN (HOE-DCN)

Fig. 1(a) shows the data plane of the HOE-DCN, where the VMs in server racks can have inter-rack communications through either a hierarchical EPS-based inter-rack network built with ToR, aggregation, and core switches, or a flat OCS-based inter-rack network that interconnects ToR switches with an optical switch. Hence, for the inter-rack traffic between a VM pair, we have the flexibility to route it through one of the two inter-rack networks, depending on its characteristics and the current and foreseeable status of the inter-rack networks.

In addition to the network part, the data plane also contains servers organized in racks, where VMs can be deployed to run applications. The VMs consume IT resources, such as CPU cycles, memory, and disk space. Where to place the VMs is an interesting problem in the network orchestration, since its solution affects not only the IT and bandwidth resource usages in the HOE-DCN but also the QoS metrics of the VMs'



(a) Data plane of HOE-DCN (b) Knowledge-defined control plane

Fig. 1. System architecture of HOE-DCN with KD-NO, VM-TA: VM traffic agent, VM-WA: VM workload agent, IT-C: IT controller, R&TD: IT resource and traffic database, VM-W&T-M: VM workload and traffic monitor, VM-W&T-P: VM workload and traffic predictor, NET-C: network controller, TED: traffic engineering database, NAM: network abstraction module, NOrch: network orchestrator, NOA: network orchestration agent, ED: experience database, NOSA: network orchestration substitute agent.

applications [37, 38]. For example, if we place two VMs with a huge bandwidth demand in between in two racks whose inter-rack bandwidth capacity is very limited, they would have difficulty to run their service while the communication bottleneck caused by them would also disturb other VMs in the two racks. Therefore, a high matching degree between the HOE-DCN's configuration (*i.e.*, the VM placement and traffic routing scheme) and the applications running in VMs is vital for achieving effective network orchestration. This essentially motivates us to propose the predictive analytics based KD-NO. The KD-NO requires us to collect and predict the VMs' workloads and the traffic matrix among them. Hence, we deploy a VM traffic agent (VM-TA) on each server to collect the matrix of the traffic going in/out the local VMs, while for each VM on the server, we assign a VM workload agent (VM-WA) to monitor its workload.

B. Knowledge-Defined Control Plane

We design the knowledge-defined control plane to include three modules, *i.e.*, the IT controller (IT-C), network controller (NET-C), and network orchestrator (NOrch) in Fig. 1(b). Here, the IT-C manages the VMs deployed in servers, while the NET-C configures the inter-rack networks to route traffic through them. Meanwhile, the IT-C and NET-C collect telemetry data regarding the data plane, *i.e.*, the IT and bandwidth usages and information about VM traffic, and extract knowledge from the data. This step of knowledge extraction is necessary because the amount of telemetry data is usually large while the useful information buried in it is sparse.

However, as the IT-C and NET-C cover different parts of the data plane, the extracted knowledge from them is still fragmented and thus cannot be directly utilized for making wise decisions on configuring the HOE-DCN. Therefore, we place the NOrch on top of them for coordination, which analyzes the low-level knowledge from the IT-C and NET-C to obtain the high-level knowledge about the matching between the HOE-DCN's configuration and active applications. Then, the NOrch instructs the IT-C and NET-T to coordinate the IT and bandwidth resources in the HOE-DCN. Here, the low/high-level knowledge provides us with not only historical information but also forecasts. Specifically, the IT-C incorporates two DL-based AI modules to predict future VM workloads and traffic matrix, based on historical telemetry data, while the NOrch takes the predictions and current network state as inputs and leverages two DRL-based AI modules to determine the optimal HOE-DCN configurations on-the-fly.

The details regarding the three modules are as follows.

1) IT-C: The VM workload and traffic monitor (VM-W&T-M) in it receives the telemetry data collected by the VM-TAs and VM-WAs in Fig. 1(a), and stores the data in the IT resource and traffic database (R&TD). Then, based on the historical data in R&TD, the VM workload and traffic predictor (VM-W&T-P) forecasts future workloads and traffic matrixes for the VMs. The IT-C then reports the current and predicted VM resource demands to the NOrch for it to orchestrate the HOE-DCN, and will implement the corresponding instructions with the VM management module.

2) NET-C: The network abstraction module (NAM) in it abstracts the topology of the inter-rack networks, monitors the bandwidth usages on switch ports, and stores the results in the traffic engineering database (TED). Meanwhile, the TED also collects the information about traffic routing through the network management module. Then, the NET-C sends the network status in TED to the NOrch for it to make network orchestration decisions, and will implement the corresponding instructions with the network management module.

3) NOrch: The DRL-based network orchestration agent (NOA) in it periodically receives the low-level knowledge about the data plane from the IT-C and NET-C, and puts it in a trained neural network for abstracting the high-level knowledge, which can be utilized to make wise decisions on configuring the HOE-DCN, *i.e.*, reconfiguring the inter-rack networks, migrating VMs, and rerouting the inter-rack traffic. To train the DRL-based NOA for intelligent decision making, we incorporate both an experience database (ED) and a DRL-based network orchestration substitute agent (NOSA) in the NOrch. Here, the NOA and NOSA use the same architecture for their neural networks. During operation, the NOA stores the network state, action, reward and next state as an entry of experience in the ED, after each decision making. When a batch of enough experience entries has been stored in the

ED, the NOSA trains itself with them, and then updates the neural network in the NOA with the training result. By doing so, the NOrch utilizes the NOA for making decisions in an online manner, while the NOSA works in the background to train its neural network simultaneously.

IV. Fetching Low-Level Knowledge about HOE-DCN

In this section, we elaborate on how the NET-C and IT-C collect telemetry data in the HOE-DCN and extract useful information (*i.e.*, the low-level knowledge) from it.

A. Collection of Network Status in NET-C

As traffic collection is handled by the VM-TAs deployed on the servers (*i.e.*, in Fig. 1(b)), the NET-C needs to abstract the feasible inter-rack topologies, each of which represents a configuration of the EPS/OCS-based inter-rack networks to interconnect the ToR switches, collect the set of available ToR switch ports in each inter-rack topology, and monitor the bandwidth usages on the currently-used ports (*i.e.*, electrical or optical). These three items compose the low-level knowledge from the NET-C to the NOrch. Among them, the first two are the static configurations that help the NOrch model the HOE-DCN, while the third one is dynamic and reflects the current network status. Here, we consider a congested port as a "hot-spot" in the HOE-DCN, and its total number should be minimized in the KD-NO. Hence, we define a few discrete levels to quantify the bandwidth usage of a port, such that the usage is represented by a much more abstracted information model to restrict the optimization space for the NOrch.

B. Collection and Prediction of VM Workloads in IT-C

We allocate a VM-WA on each VM to monitor its workloads in terms of CPU and I/O usages. The collected telemetry data is analyzed by the IT-C for predicting future VM workloads and extracting low-level knowledge. Note that, there might be numerous VMs in the HOE-DCN, and thus analyzing and predicting the workloads of all the VMs would not be a scalable solution. We have to admit that this scalability issue is still an interesting and open question in the KD-NO, but it can be relieved by the following considerations and procedure.

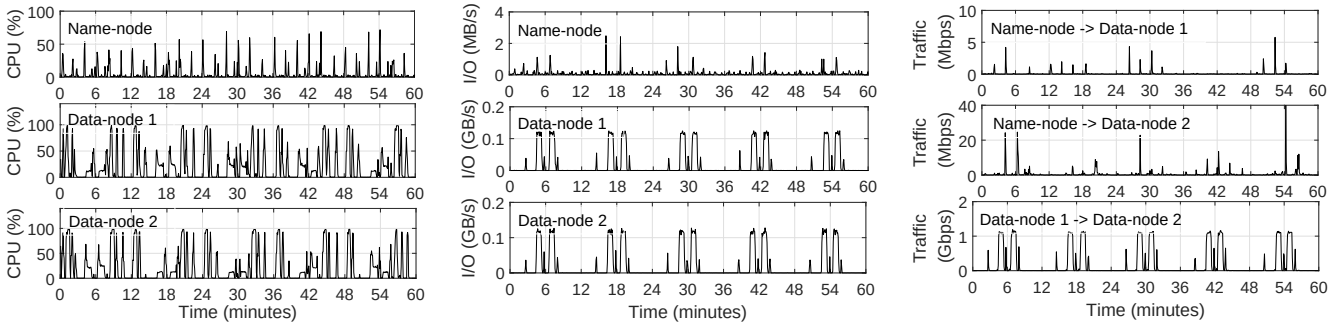
It is obvious that the VMs on a server would not contribute equally to its IT workloads. Actually, the contributions could have difference in magnitude scale. Therefore, we propose a simple "screening" process to only select the major contributors for analysis and prediction. Taking the Hadoop application [22] as an example, the VM that runs its name-node usually has much less CPU and I/O usages than those running its data-nodes. Hence, the screening process would just ignore the name-node VM. Secondly, even for the major contributors, we might not need to train specific DL modules to analyze and predict their IT workloads, considering their correlations. For instance, in the Hadoop application, the data-node VMs can have highly-correlated CPU and I/O usages over time. To this end, for highly-correlated IT workloads, we just train one DL model for all their predictors, apply the predictors in the IT-C, and leverage transfer learning [39] to adjust them for maintaining high forecast accuracy during operation. This further reduces our efforts on developing the IT-C.

We conduct a simple experiment involving Hadoop applications to further explain the aforementioned procedure in details. We have a Hadoop cluster with three VMs, which processes CPU-bound and I/O-bound jobs generated according to the job arrival distributions in the cluster data traces released by Google [40, 41]. In the experiment, the original diurnal pattern of the jobs (*i.e.*, for 24 hours) gets scaled down to 12 minutes. The VM-WAs report the VMs' CPU and I/O workloads every 5 seconds. One VM is the name-node and the other two are the data-nodes.

Figs. 2(a) and 2(b) show the collected CPU and I/O workloads of the VMs in an hour (experiment time). We observe that both the CPU and I/O workloads of the name-node are much lower than those of the two data-nodes, respectively. Therefore, the screening process should only treat the two data-nodes as the major contributors of IT workloads. Meanwhile, by comparing the CPU and I/O workloads of the two data-nodes, we can see that they are highly correlated over time. Specifically, the cross-correlation of the two data-nodes' CPU workloads is 64.60%, while that of their I/O workloads is 99.12%. Hence, the VM-W&T-P's complexity in predicting the IT workloads of the VMs can be reduced by considering the VMs as a correlated VM group, *i.e.*, the DL models trained for predicting the IT workloads of *Data-node* 1 can be reused to forecast those of *Data-node* 2.

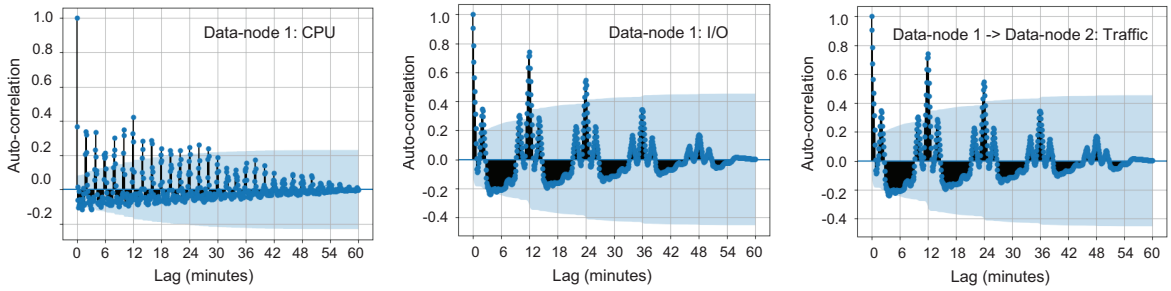
Next, before proceeding to IT workload prediction, we need to confirm that the workloads follow certain patterns and thus are actually predictable. Hence, we calculate the auto-correlations [42] of the IT workloads, and Figs. 3(a) and 3(b) show the results for *Data-node* 1, where the light blue area in each plot denotes the region whose confidence level is below 95%. The auto-correlation results for *Data-node* 2 are similar since the two data-nodes are highly correlated over time, and thus we omit its results here. We observe that the auto-correlations of both CPU and I/O workloads fluctuate with certain pattern, which indicates that the IT workloads of *Data-node* 1 repeat themselves approximately after certain durations and thus are predictable.

As the IT workloads are predictable and the prediction technique for a time series is quite mature [42], we design the VM-W&T-P in the IT-C with long short-term memory based neural networks (LSTM-NNs). To adapt to the dynamic IT workloads in an HOE-DCN, we also implement transfer learning in the VM-W&T-P. Specifically, for each IT workload predictor, we have offline and online training phases. In the offline training, we train the predictor with historical data regarding the corresponding IT workload. To effectively reduce the predictor's complexity and the optimization space in the NOrch, we do not just use the collected time series of the IT workload as its input, but summarize the samples over a certain period (*e.g.*, 1 minute in our experiments) to obtain an aggregated time series and input the aggregated data to the predictor. And in the process, 95% of the aggregated data is used as the training set while the remaining 5% is the testing set. After that, the predictor is put in the VM-W&T-P and it



(a) CPU usages of VMs (b) I/O workloads of VMs (c) Traffic among VMs

Fig. 2. Telemetry data collected for a Hadoop cluster that includes three VMs.



(a) CPU usage (b) I/O workload (c) Traffic between VMs

Fig. 3. Auto-correlations of *Data-node* 1's IT workloads and traffic between *Data-nodes* 1 and 2.

enters the online training. Specifically, the predictor forecasts the volumes of its corresponding IT workload in future service cycles using the trained LSTM-NN, and after the actual IT workload has been collected, it adds this newly-collected data in its training set to retrain and update its LTSM-NN. Note that, we select the period of sample aggregation such that the tradeoff between the predictor's complexity and the NOrch's effectiveness can be balanced well.

TABLE I
RESULTS ON IT WORKLOAD PREDICTIONS

	Offline Training			
	CPU		I/O	
	Prediction Accuracy	Training Time (sec)	Prediction Accuracy	Training Time (sec)
Data-node 1	90.7%	24.08	91.0%	23.75
Data-node 2	85.7%	–	86.5%	–
	Online Training			
Data-node 1	87.2%	1.15	89.4%	1.42
Data-node 2	82.3%	1.23	84.1%	1.35

Table I summarizes the results regarding the LSTM-NNs for IT workload prediction. As we only train the LSTM-NNs for *Data-node* 1's IT workload predictors, the time of the offline training for *Data-node* 2's predictors is absent. For the offline training, the accuracy of *Data-node* 1's predictors is that on its testing set, while the accuracy of *Data-node* 2's predictors is calculated on all of its historical data. We observe that the offline training achieves relatively high prediction accuracy for *Data-node* 1 on both CPU and I/O workloads. In the meantime, if we apply the predictors of *Data-node* 1 to forecast the corresponding IT workloads of *Data-node* 2, the prediction accuracy is still reasonably high, which supports our decision to treat the VMs as a correlated VM group. On the other hand, the online training is conducted for the data-nodes' predictors for six service cycles and we average the results to put in Table I. It can be seen that the online training takes much shorter time to update the LSTM-NNs and the prediction accuracy is still maintained reasonably high.

TABLE II
RESULTS ON TRAFFIC PREDICTION

	Offline Training	
	Prediction Accuracy	Training Time
Data-node 1 → *Data-node* 2	91.6%	29.15 sec
	Online Training	
Data-node 1 → *Data-node* 2	89.8%	1.13 sec

C. Collection and Prediction of VM Traffic Matrix in IT-C

Fig. 1(a) indicates that the VM-TA on each server collects the traffic matrix of the local VMs and reports it to the IT-C. The proposal discussed in the previous subsection can also be applied to fetch low-level knowledge from the traffic data. And for the experiment, the measurements on the traffic among the VMs are plotted in Fig. 2(c). The screening process determines that in Fig. 2(c), the major contributor is the traffic between the two data-nodes. Meanwhile, the auto-correlation of the traffic between *Data-nodes* 1 and 2 in Fig. 3(c) suggests that the traffic is predictable. Hence, we still use an LSTM-NN for the traffic prediction and train it with both the offline and online phases. Similar as the IT workload predictor, the

traffic predictor also aggregates historical traffic samples and use the aggregated data as its input. Table II summarizes the results regarding the traffic predictor. Note that, to restrict the optimization space of the NOrch, we further define a few discrete levels to quantify the predictions of IT workload and inter-rack traffic volume, so that they can be represented with a much more abstracted information model.

V. Abstracting and Utilizing High-Level Knowledge about HOE-DCN

In this section, we discuss how to leverage the low-level knowledge regarding the data plane for network orchestration. We formulate the network orchestration problem as an MILP model, prove its $\mathcal{NP}$-hardness, and then design a DRL-based online scheme to orchestrate the HOE-DCN, which can extract high-level knowledge from the low-level input and come up with optimal HOE-DCN configurations on-the-fly.

A. Optimization in Network Orchestration

The network orchestration to coordinate the IT and bandwidth resources in the HOE-DCN is actually an optimization problem. Specifically, based on the collected and predicted results on VM workloads and traffic matrix, we should optimize the configuration of the inter-rack topology, the locations of the VMs, and the routing scheme of VM traffic, such that the number of hot-spots (*i.e.*, degraded VMs due to overloaded servers[1] and congested ToR switch ports) in foreseeable future can be minimized for ensuring high QoS to the applications. Meanwhile, during the process, the operating expenses (OPEX) of network orchestration (*i.e.*, VM migrations and reconfigurations on inter-rack topology) should be minimized as well. This particular optimization problem is formulated as:

Parameters:

- V: the set of VMs currently running in the HOE-DCN.
- R: the set of ToR switches in the HOE-DCN.
- S: the set of servers in the HOE-DCN.
- S_r: the set of servers under ToR switch $r \in R$.
- P_r: the set of ports in ToR switch $r \in R$ for inter-rack communications, and we assume that each ToR switch has the same number and types of ports.
- b_p: the linerate of inter-rack port $p \in P_r$ in a ToR switch.
- L_p: the threshold on date-transfer time when using port $p \in P_r$ in a ToR switch.
- $\mathbf{G}$: the set of feasible and considered inter-rack topologies, where each $G(R, E) \in \mathbf{G}$ indicates how all the ToR switches are connected with the links in E. Here, a link can be either electrical or optical.
- f_G: the binary indicator that equals 1 if the HOE-DCN is using inter-rack topology G, and 0 otherwise.
- c_v^R: the IT requirement of VM v in terms of RAM usage.
- c_v^C: the IT requirement of VM v in terms of CPU usage.
- c_v^D: the IT requirement of VM v in terms of disk space.
- $l_{v,s}$: the binary indicator that equals 1 if VM $v \in V$ is currently running in server $s \in S$, and 0 otherwise.
- w_v^C: the predicted workload of VM v in terms of CPU usage, during the next provisioning cycle.
- w_v^I: the predicted workload of VM v in terms of I/O demand, during the next provisioning cycle.
- $w_{v,v'}^T$: the predicted traffic from VM v to VM v', during the next provisioning cycle.
- $P_{G,r,r'}$: the set of available ports for the communication from ToR switch r to ToR switch r', when inter-rack topology G is used.
- C_s^R: the total IT resources on server s in terms of RAM.
- C_s^C: the total IT resources on server s in CPU usage.
- C_s^D: the total IT resources on server s in disk space.
- W_s^C: the threshold on CPU workload for server s.
- W_s^I: the threshold on I/O workload for server s.
- α: the parameter that weights the cost of one hot-spot due to either a congested port or a degraded VM.
- β: the parameter that weights the cost of a VM migration.
- γ: the parameter that weights the cost of an inter-rack topology reconfiguration.
- M: a positive number is big enough to ensure that the related inequalities are correct.

Variables:

- $x_{v,s}$: the boolean variable that equals 1 if we run VM v on server s, and 0 otherwise.
- y_G: the boolean variable that equals 1 if inter-rack topology G is selected, and 0 otherwise.
- $z_{v,v'}^{r,p}$: the boolean variable that equals 1 if VM v uses port p in ToR switch r to talk with VM v', and 0 otherwise.
- m_v: the boolean variable that equals 1 if we need to migrate VM v, and 0 otherwise.
- m: the boolean variable that equals 1 if we need to reconfigure the inter-rack topology, and 0 otherwise.
- g_s: the boolean variable that equals 1 if server s is overloaded, and 0 otherwise.
- g_v: the boolean variable that equals 1 if the performance of VM v is degraded, and 0 otherwise.
- $g_{r,p}$: the boolean variable that equals 1 if port p in ToR switch r is congested, and 0 otherwise.

Objective:

For jointly minimizing the number of hot-spots and OPEX due to DCN reconfigurations, the objective is defined as

$$\text{Minimize} \quad \alpha \cdot \left(\sum_v g_v + \sum_{r,p} g_{r,p} \right) + \beta \cdot \sum_v m_v + \gamma \cdot m. \quad (1)$$

Constraints:

$$\sum_{r \in R} \sum_{s \in S_r} x_{v,s} = 1, \quad \forall v \in V. \quad (2)$$

Eq. (2) ensures that each VM is deployed on a server.

$$\sum_{v \in V} x_{v,s} \cdot c_v^R \leq C_s^R, \quad \forall r \in R,\ s \in S_r, \quad (3)$$

$$\sum_{v \in V} x_{v,s} \cdot c_v^C \leq C_s^C, \quad \forall r \in R,\ s \in S_r, \quad (4)$$

$$\sum_{v \in V} x_{v,s} \cdot c_v^D \leq C_s^D, \quad \forall r \in R,\ s \in S_r. \quad (5)$$

[1]For a server that runs multiple VMs simultaneously, if the total CPU or total I/O workloads on it exceeds certain thresholds, respectively, the performance of all the VMs will be degraded, *i.e.*, the server is overloaded.

Eqs. (3)-(5) ensure that the IT resource capacities of each server are not exceeded.

$$\sum_{G\in\mathbf{G}} y_G = 1. \tag{6}$$

Eq. (6) ensures that only one inter-rack topology is selected.

$$z^{r,p}_{v,v'} \le \frac{y_G + x_{v,s} + x_{v',s'}}{3},\ \forall v,v' \in V,\ r,r' \in R,\ s \in S_r,\ s' \in S_{r'},\ G \in \mathbf{G},\ p \in P_{G,r,r'} :\ v \ne v', r \ne r', w^T_{v,v'} > 0, \tag{7}$$

$$\sum_{p\in P_{G,r,r'}} z^{r,p}_{v,v'} \le 1,\ \forall v,v' \in V,\ r,r' \in R,\ s \in S_r,\ s' \in S_{r'},\ G \in \mathbf{G} :\ v \ne v', r \ne r', w^T_{v,v'} > 0, \tag{8}$$

$$\sum_{p\in P_{G,r,r'}} z^{r,p}_{v,v'} \ge y_G + x_{v,s} + x_{v',s'} - 2,\ \forall v,v' \in V,\ r,r' \in R,\ s \in S_r,\ s' \in S_{r'},\ G \in \mathbf{G} : v \ne v', r \ne r', w^T_{v,v'} > 0. \tag{9}$$

Eqs. (7)-(9) ensure that if there is traffic between two VMs (*i.e.,* $w^T_{v,v'} > 0$) and the VMs are located in different racks, a port has be allocated on both end ToR switches to support the inter-rack communication between them.

$$m_v \ge x_{v,s} \cdot (1 - l_{v,s}), \quad \forall r \in R,\ s \in S_r,\ v \in V. \tag{10}$$

Eq. (10) determines whether a VM needs to be migrated.

$$m \ge y_G \cdot (1 - f_G), \quad \forall G \in \mathbf{G}. \tag{11}$$

Eq. (11) determines whether the inter-rack topology needs to be reconfigured.

$$M \cdot g_s \ge \left(\sum_{v\in V} x_{v,s} \cdot w^C_v\right) - W^C_s, \quad \forall r \in R,\ s \in S_r, \tag{12}$$

$$M \cdot g_s \ge \left(\sum_{v\in V} x_{v,s} \cdot w^I_v\right) - W^I_s, \quad \forall r \in R,\ s \in S_r. \tag{13}$$

Eqs. (12) and (13) determine whether a server is overloaded due to CPU over-usage or the exhaustion of I/O resources.

$$g_v \ge g_s + x_{v,s} - 1, \quad \forall r \in R,\ s \in S_r,\ v \in V. \tag{14}$$

Eq. (14) determines whether the performance of a VM is degraded due to server overload.

$$M \cdot g_{r,p} \ge \left(\sum_{v,v':v\ne v'} z^{r,p}_{v,v'} \cdot w^T_{v,v'}\right) - b_p \cdot L_p, \quad \forall r \in R,\ p \in P_r. \tag{15}$$

Eq. (15) determines whether a ToR switch port is congested.

B. Hardness Analysis

Theorem. *The network orchestration problem is $\mathcal{NP}$-hard.*

Proof: We prove the $\mathcal{NP}$-hardness of the problem by restriction, which means that we restrict away certain of its aspects until a known $\mathcal{NP}$-hard appears [43]. We first remove all the constraints except for Eqs. (2), (12)-(14), Then, the network optimization problem becomes a VM placement problem whose optimization objective is

$$\text{Minimize} \quad \sum_v g_v, \tag{16}$$

since the restriction makes variables $g_{r,p}$, m_v and m irrelevant to the optimization. Meanwhile, it is easy to verify that in the restricted problem, the objective $\sum_v g_v$ can only be $0, 1, ..., |V|$.

When we restrict $\sum_v g_v = 0$, which is equivalent to $\sum_s g_s = 0$ (referring to Eq. (14)), the optimization of the restricted problem becomes a decision problem: Given a set of VMs $v \in V$, each of which has a two-dimensional (2D) resource requirement $\{w^C_v, w^I_v\}$, and a set of servers $s \in S$, each of which also has a 2D resource capacity $\{W^C_s, W^I_s\}$, whether there exists a feasible solution to pack all the VMs in the servers such that the servers' resource capacities would not be exceeded (*i.e.,* $\sum_s g_s = 0$ subject to Eqs. (2), (12) and (13))? This decision problem is essentially a 2D knapsack problem, if we treat the VMs and servers as items with 2D sizes and boxes with 2D capacities, respectively. It is known that the 2D knapsack problem is $\mathcal{NP}$-hard [43]. Hence, since the restricted case of the network orchestration problem is the general case of a known $\mathcal{NP}$-hard problem, we prove its $\mathcal{NP}$-hardness. ■

C. DRL-based Network Orchestration

Since the network orchestration problem is $\mathcal{NP}$-hard and a hand-crafted heuristic for it would have difficulty to adapt to the dynamic environment in an HOE-DCN, we decide to solve it by leveraging DRL. An important advantage of our DRL-based approach is that based on its experience, the AI module can make wise KD-NO decisions timely in an online manner. The DRL model is designed as:

- **State**: state $\mathbf{S}_i$ refers to the state of the HOE-DCN at time instant i, which includes the current inter-rack topology, IT resource requirements of VMs in RAM usage, CPU usage and disk space, VM locations, predicted CPU and I/O workloads and traffic matrix of VMs, and ToR switch ports used for VMs' inter-rack communications.
- **Action**: action $\mathbf{A}_i$ is the action taken at time instant i, which includes the selected inter-rack topology, new VM locations, and new ToR switch ports selected for VMs' inter-rack communications. Here, we encode each element in an action as an integer, *e.g.*, the inter-rack network topology is encoded as an integer within $[1, |\mathbf{G}|]$.
- **Reward**: reward $\mathbf{R}_i$ of action $\mathbf{A}_i$ is calculated by taking the opposite number of the result from Eq. (1).

As both the state and action spaces are relatively large in our DRL model, we leverage the deep deterministic policy gradient (DDPG) [44] to design its operation procedure. The rationale for choosing DDPG is two-fold. Firstly, it adopts an Actor-Critic learning strategy to avoid the difficulty of action selection due to the need of traversing the entire action and state spaces whose sizes are relatively large. Secondly, it combines policy gradient with Q-learning and lets them learn from each other, and thus achieves improved learning efficiency. DDPG has a dual neural network architecture, namely, Actor and Critic, respectively. Actor uses a deterministic policy gradient function $\mu_{\theta^p}(\mathbf{S}_i)$ to select an action directly, *i.e.*,

$$\mathbf{A}_i = \mu_{\theta^p}(\mathbf{S}_i), \tag{17}$$

where θ^p is the parameter in Actor. Critic uses a Q-function $Q_{\theta^Q}(\mathbf{S}_i, \mathbf{A}_i)$ to evaluate the Q-value of action $\mathbf{A}_i$ in state $\mathbf{S}_i$,

and transmits the action gradient (*i.e.*, $\nabla_{\mathbf{A}_i} Q_{\theta^Q}(\mathbf{S}_i, \mathbf{A}_i)$) to Actor, for increasing the probability of selecting the action that has a larger Q-value. Here, θ^Q is the parameter in Critic.

Actor optimizes $\mu_{\theta^p}(\mathbf{S}_i)$ using the policy gradient

$$\nabla_{\theta^p} J \approx \frac{1}{N} \sum_{i=1}^{N} X_i \cdot Y_i, \tag{18}$$

where N is the iterations in training, and X_i and Y_i are the gradients of $\mu_{\theta^p}(\mathbf{S}_i)$ and $Q_{\theta^Q}(\mathbf{S}_i, \mathbf{A}_i)$, respectively. The gradients are calculated as

$$\begin{cases} X_i = \nabla_{\theta^p} \mu_{\theta^p}(\mathbf{S}_i), \\ Y_i = \nabla_{\mathbf{A}_i} Q_{\theta^Q}(\mathbf{S}_i, \mathbf{A}_i), \end{cases} \tag{19}$$

where according to Eq. (17), $\mathbf{A}_i$ in Y_i equals $\mu_{\theta^p}(\mathbf{S}_i)$ in Y_i.

Critic optimizes $Q_{\theta^Q}(\mathbf{S}_i, \mathbf{A}_i)$ by minimizing the squared loss between the expected and estimated Q-values, *i.e.*,

$$L = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{R}_i + \kappa \cdot Q_{\theta^Q}(\mathbf{S}_{i+1}, \mathbf{A}_{i+1}) - Q_{\theta^Q}(\mathbf{S}_i, \mathbf{A}_i))^2, \tag{20}$$

where κ is a discount factor and N is the iterations in training.

We realize the DRL-based network orchestration with the two DRL-based agents in the NOrch (*i.e.*, the NOA and NOSA in Fig. 1(b)). Here, the NOA and NOSA have the same neural network architecture that includes both Actor and Critic, and their parameters are $\{\theta^p, \theta^Q\}$ and $\{\theta^{p'}, \theta^{Q'}\}$, respectively. *Algorithm* 1 shows the detailed procedure of the DRL-based network orchestration in the NOrch. There are two threads, which are for the online operation (*Lines* 1-10) and training in background (*Lines* 11-18), respectively. On one hand, in each service cycle, the NOA receives the current state $\mathbf{S}_i$ from the IT-C and NET-C (*Line* 3), takes network orchestration action $\mathbf{A}_i$ according to Eq. (17) (*Line* 4), legalizes $\mathbf{A}_i$ to satisfy the constraints in previous subsection (*Line* 5), and calculates the corresponding reward $\mathbf{R}_i$ by taking the opposite number of the result from Eq. (1) (*Line* 6). Then, the IT-C and NET-C configure the HOE-DCN accordingly to coordinate the IT and bandwidth resources in it (*Line* 8). Before the next cycle starts, the NOA stores $\{\mathbf{S}_i, \mathbf{A}_i, \mathbf{R}_i, \mathbf{S}_{i+1}\}$ in the ED as an entry of experience (*Line* 7). On the other hand, in each background training episode, the NOSA performs training based on the experience in ED and updates parameters $\{\theta^{p'}, \theta^{Q'}\}$ according to Eqs. (18)-(20) (*Line* 13). Meanwhile, after certain training iterations, the NOA updates its parameters $\{\theta^p, \theta^Q\}$ (*Lines* 14-16) with the following rules:

$$\begin{cases} \theta^p = \tau_p \cdot \theta^p + (1 - \tau_p) \cdot \theta^{p'}, \\ \theta^Q = \tau_Q \cdot \theta^Q + (1 - \tau_Q) \cdot \theta^{Q'}, \end{cases} \tag{21}$$

where τ_p and τ_Q are the small coefficients for stable learning.

Fig. 4 gives an illustrative example on the high-level knowledge regarding the matching degree between the HOE-DCN's configuration and the active applications, *i.e.*, a state seen by the NOrch. Here, we assume that there are three racks and each of their ToR switches possesses an optical port that can be used to route traffic through the OCS-based inter-rack network. Hence, only one pair of the ToR switches can use the OCS-based inter-rack network at a time. With the high-level knowledge in Fig. 4, the NOrch can easily identify hot-spots in the HOE-DCN, and with *Algorithm* 1, it makes wise network orchestration decisions to improve the matching degree.

Algorithm 1: DRL-based Network Orchestration in NOrch

1 **Thread I:** Online Operation
2 **for** *each service cycle* **do**
3 NOA receives state $\mathbf{S}_i$ from IT-C and NET-C;
4 NOA takes action $\mathbf{A}_i$ according to Eq. (17);
5 NOA legalizes $\mathbf{A}_i$ according to certain rules;
6 NOA calculates reward $\mathbf{R}_i$;
7 NOA stores $\{\mathbf{S}_{i-1}, \mathbf{A}_{i-1}, \mathbf{R}_{i-1}, \mathbf{S}_i\}$ in ED;
8 IT-C and NET-C configure HOE-DCN;
9 **end**
10 **End**
11 **Thread II:** Training in Background
12 **for** *each training episode* **do**
13 NOSA updates $\{\theta^{p'}, \theta^{Q'}\}$ with Eqs. (18)-(20);
14 **if** *certain iterations have been passed since the last update* **then**
15 NOA updates $\{\theta^p, \theta^Q\}$ with Eq. (21);
16 **end**
17 **end**
18 **End**

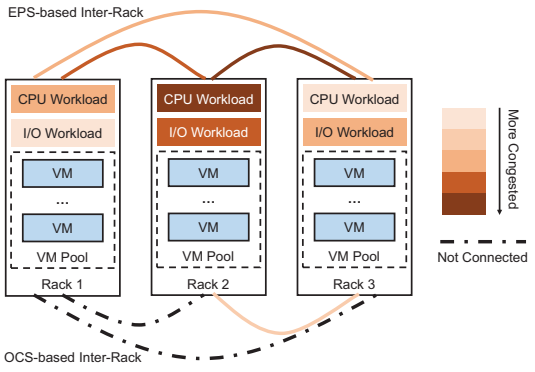


Fig. 4. Example on high-level knowledge regarding the matching degree between HOE-DCN's configuration and applications running in it.

VI. Experimental Demonstrations

In this section, we first briefly explain our system implementation and experimental setup, and then perform three experiments to demonstrate the effectiveness of our proposal.

A. System Implementation

We prototype a small-scale but real HOE-DCN testbed, and implement the proposed KD-NO system in it, which is developed with Python 3.6 and uses the machine learning library of TensorFlow.

1) Data Plane: It is built with a few Linux servers organized in four racks, hardware-based OpenFlow switches, and a reconfigurable optical switch, as shown in Fig. 5. The OpenFlow switches are the ToR, aggregation and core switches in the EPS-based inter-rack network whose connections are based on 1GbE. Meanwhile, the optical switch gets connected to the

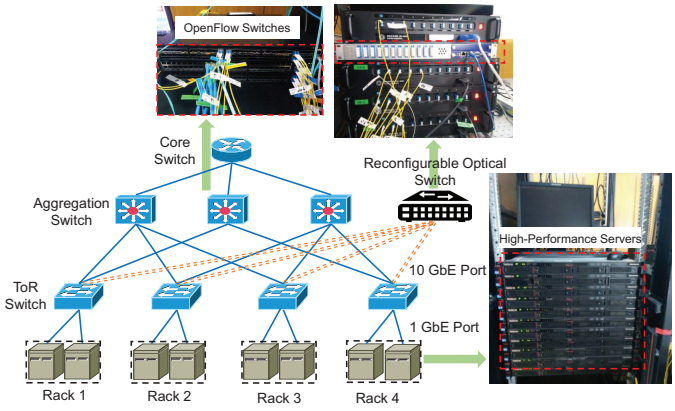


Fig. 5. Experimental setup of the HOE-DCN prototype with KD-NO.

10GbE optical ports on all the ToR switches to form the OCS-based inter-rack network. Each server in a rack is connected to its ToR switch through a 1GbE port. The switching time of the optical switch is around 100 milliseconds, which might still be an issue for applying optical switching in DCNs. However, with accurate knowledge abstraction, our KD-NO system can bypass this issue to certain extent. More specifically, the KD-NO system not only tries to minimize the number of inter-rack reconfigurations, but also leverages the prediction results to conduct inter-rack reconfigurations in advance with the "make-before-break" scheme. Hence, the interruption on inter-rack elephant flows caused by optical switching can be avoided.

2) Control Plane: To realize the IT-C, we leverage the OpenStack cloud platform and develop the VM management module with OpenStack APIs. Moreover, we implement the VM-W&T-M based on collectd [45] and sFlow [46], and realize VM-W&T-P by referring to the LTSM-NN. On the other hand, the NET-C is implemented by modifying ONOS to achieve timely monitoring and management on the NEs in the HOE-DCN. We implement *Algorithm* 1 in the NOA and NOSA, and make them cooperate with each other as designed. Besides, the NOA communicates with the IT-C and NET-C, such that it can receive network status for making timely network orchestration decisions while the decisions can be sent to the IT-C and NET-C for execution.

B. Experimental Setup

We conduct experiments in three scenarios. In each scenario, we run four threads simultaneously to facilitate the operation of our KD-NO system, as illustrated in Fig. 6.

- *Thread* 1: it dynamically generates jobs for the Hadoop clusters running in the HOE-DCN testbed. Specifically, we first create 12 VMs on servers in the testbed and group them into four Hadoop clusters, each of which has one name-node and two data-nodes, and then use this thread to generate CPU- or/and I/O-bound jobs on the clusters. To emulate practical Hadoop workloads, the jobs are generated according to the job arrival distributions in the cluster data traces released by Google [40, 41]. Meanwhile, to save the time for experiments, we scale the original diurnal pattern of the jobs (*i.e.*, for 24 hours) down to 12 minutes. The thread uses the statistical model extracted from the Google traces to create the jobs, but it deviates the peak and valley periods of jobs in the four clusters to introduce certain uncertainty. This is to check whether our proposed KD-NO system can learn the dynamics and differences in resource demands and therefore make right HOE-DCN configuration decisions.
- *Thread* 2: it leverages the NOrch to perform a DCN reconfiguration for network orchestration every 2 minutes. Note that, in a more practical scheme, the duration of this service cycle should be acquired through online learning to adapt to the dynamic workloads in an HOE-DCN. We will consider this in our future work.
- *Thread* 3: it uses the VM-W&T-M to collect telemetry data, including VM workloads and traffic between VMs.
- *Thread* 4: it utilizes the VM-W&T-P to forecast future VM workloads and traffic.

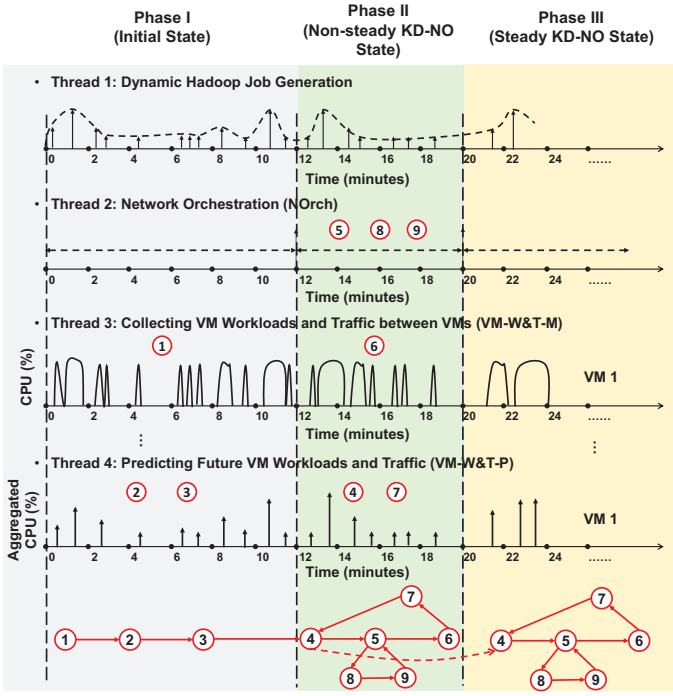


Fig. 6. Operation procedure of our KD-NO system in experiments.

Then, according to whether the DRL-based NOA and NOSA have been involved and converged in the online training, we divide the KD-NO's operation in the HOE-DCN into three phases, *i.e.*, the initial state, and non-steady and steady KD-NO states, respectively. The training of the LTSM and DRL based modules is executed by the aforementioned threads in different phases. In **Phase I** (*i.e.*, the initial state), we run Hadoop jobs in the HOE-DCN testbed (*i.e.*, *Thread* 1), and collect enough telemetry data (*i.e.*, ① marked in *Thread* 3) without any DCN reconfiguration. We use the collected telemetry data to train the LTSM-based predictors offline. Before the offline training, we pre-process the telemetry data by summarizing the data points within every minute to get an aggregate data point (*i.e.*, ② in *Thread* 4), to speed up the training. Next, the VM-W&T-P divides the aggregated data

into training and testing sets, with the ratios of 95% and 5%, respectively, which are then used to train the LTSM-based predictors offline (*i.e.*, ③ in *Thread* 4). The events in **Phase I** happen according to the procedure ①→②→③.

In **Phase II** (*i.e.*, the non-steady KD-NO state), we start to let the DRL-based NOA perform KD-NO and train the DRL-based NOSA in the background. Here, as shown in Fig. 6, we design two online training loops, for the LSTM-NNs in the VM-W&T-P and the DRL-based NOA/NOSA, respectively. At the beginning of each service cycle, the VM-W&T-P that has been trained in **Phase I** forecasts the VM workloads and traffic in coming service cycle(s), and sends the results to the DRL-based NOA as the low-level knowledge input (*i.e.*, ④ in *Thread* 4). Then, the DRL-based NOA makes the network orchestration decision based on its current DRL model (*i.e.*, ⑤ in *Thread* 2). After the HOE-DCN has been reconfigured accordingly, the VM-W&T-M collects the actual VM workloads and traffic (*i.e.*, ⑥ in *Thread* 3), based on which the VM-W&T-P updates its LSTM-NNs on-the-fly with transfer learning (*i.e.*, ⑦ in *Thread* 4). Hence, the online training loop of the LTSM-NNs in the VM-W&T-P is ④→⑤→⑥→⑦→④→$\cdots$ in **Phase II**.

Meanwhile, the DRL-based NOA collects and stores entries of KD-NO experience (*i.e.*, ⑧ in *Thread* 2). After enough experience entries have been collected, the DRL-based NOSA trains itself with them in the background, and updates the DRL-based NOA with the training result periodically (*i.e.*, ⑨ in *Thread* 2). By doing so, the NOrch can utilize the DRL-based NOA for making KD-NO decisions in the online manner, while the NOSA works in the background for online training. The online training loop of the DRL-based NOA/NOSA is ⑤→⑧→⑨→⑤→$\cdots$ in **Phase II**.

Note that, after the training on the DRL-based NOSA has converged, we consider that the KD-NO system has proceeded to **Phase III** (*i.e.*, the steady KD-NO state). In this phase, when the dynamic VM workloads and traffic in the HOE-DCN are about to deviate the testbed from its optimal configuration, the DRL-based NOA will make correct and timely decisions on VM migration, traffic rerouting, and/or inter-rack reconfiguration, to maintain the performance of the HOE-DCN. Therefore, the online training loops for the LSTM-NNs in the VM-W&T-P and the DRL-based NOA/NOSA still operate the same as in **Phase II**. Meanwhile, we would like to point out that reconfigurations on the HOE-DCN can change the fluctuation patterns of VM workloads and traffic, which would decrease the prediction accuracy of the VM-W&T-P. Consequently, this can make the DRL-based NOA/NOSA drift from the optimal state. If this event happens, the KD-NO system will return to **Phase II** shortly, until the DRL-based NOA/NOSA have converged again and moved back to the optimal state. With this cycle, our KD-NO system has the robustness to operate well in dynamic network environment.

We evaluate the KD-NO system in **Phase III** with the following three experimental scenarios, and demonstrate that it can orchestrate the resources in the HOE-DCN correctly and timely, when its DL/DRL modules are trained and operational.

- *Network orchestration without DCN reconfiguration* (**NO w/o DCN Reconfiguration**): it just runs the generated jobs, but does not perform any inter-rack topology reconfiguration, VM migration or routing scheme update.
- *Network orchestration without low-level knowledge* (**NO w/o Low-level Knowledge**): it is a reactive approach that uses an exhaustive search algorithm to determine the HOE-DCN configuration based on the current network status, but does not leverage any low-level knowledge, *e.g.*, the predicted VM workloads and traffic matrix.
- *Knowledge-defined network orchestration* (**KD-NO**): it is our proposed KD-NO scheme.

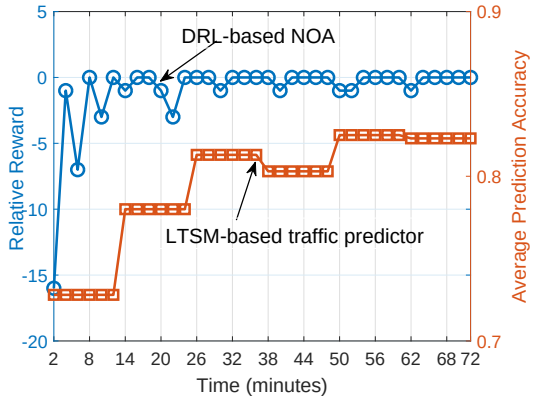


Fig. 7. Convergence performance of NOrch's online training in experiments.

Fig. 7 shows convergence performance of the online training that our DRL-based NOrch performs in the HOE-DCN testbed. Here, the relative reward refers to the difference between the reward of the action made by the NOrch in a network state and the optimal reward in the same state. The results indicate that starting as an untrained AI module, the NOrch can output relatively high rewards after the experiment has been running for $\sim$24 minutes (*i.e.*, 12 service cycles). Meanwhile, under the guidance of the NOrch, the online prediction accuracy of the LTSM-based traffic predictor also increases gradually. Finally, they make the KD-NO system enter the steady KD-NO state.

To quantify the congestions on servers and ToR switch ports, we categorize the IT workloads and traffic of VMs into several levels and set the overload thresholds for them accordingly. For a VM's CPU workload, we categorize it into three levels based on the length of its peak duration (*i.e.*, when the VM's CPU usage is 100%) in a period of sample aggregation. Specifically, the levels of $[0, 1, 2]$ correspond to a peak duration within $[0, 5)$, $[5, 30)$, and $[30, 60)$ seconds, respectively. The overload threshold on the CPU workload of a server is set as 1, which means that when the total CPU workload of all the VMs on the server (in levels) exceeds 1, it is considered as overloaded.

The I/O workload of a VM is qualified into three levels too. Here, the levels $[0, 1, 2]$ correspond to an I/O workload of $[0, 0.1)$, $[0.1, 3]$ and $(3, 6]$ GB in a period of sample aggregation, respectively. The overload threshold on the I/O workload of a sever is set as 24, which is equivalent to a total I/O workload of more than 72 GB in an aggregation period on the server. Using the same quantification criteria as the I/O workload, traffic among VMs are also classified into three levels, and the overload threshold for 1GbE EPS-based ToR

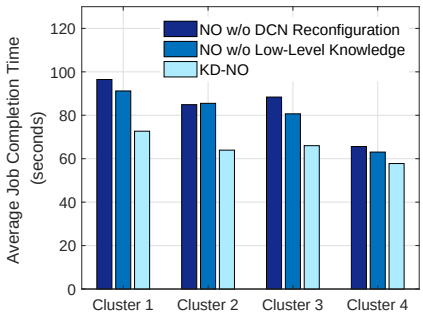


Fig. 8. Average job completion time (first experiment: CPU-bound jobs).

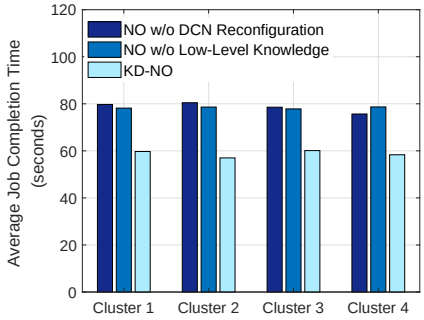


Fig. 10. Average job completion time (second experiment: I/O-bound jobs).

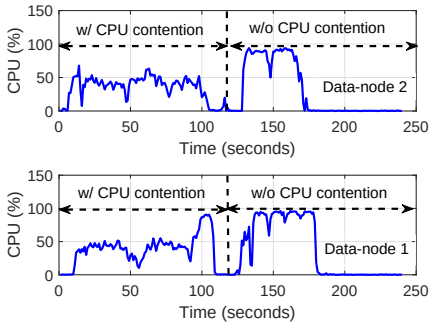


Fig. 9. KD-NO's effect to remove CPU contentions (first experiment: CPU-bound jobs).

switch ports is set as 1. For the optimization objective in Eq. (1), we set the values of $\{\alpha, \beta, \gamma\}$ as $\{1, 0.5, 0.05\}$ to give minimizing the number of hot-spots the highest priority.

C. Experiment with CPU-bound Jobs

The first experiment only considers CPU-bound jobs, which usually require a lot of CPU usages but do not cause high I/O workloads or inter-rack traffic loads. Hence, to ensure high QoS to the Hadoop applications, we need to avoid contentions on CPU usage as many as possible, for reducing the average job completion time. As the I/O workloads and inter-rack traffic loads are relatively low, they become irrelevant in the network orchestration of the HOE-DCN.

Fig. 8 compares the average job completion time achieved by the three experimental scenarios. It can be seen that "KD-NO" provides much shorter job completion time than the two benchmarks. Moreover, the results actually suggest: 1) our proposed KD-NO system can perform timely and correct VM migrations to minimize CPU contentions in the HOE-DCN and thus reduce the jobs' completion time significantly, and 2) without the proactive principle and low-level knowledge, the network orchestration of "NO w/o Low-level Knowledge" might not be a better scheme than "NO w/o DCN Reconfiguration", *i.e.*, some of its VM migrations are based on incorrect decisions. The first statement can be further verified with the results in Fig. 9, which shows the CPU workloads of two data-nodes before and after the network orchestration conducted by "KD-NO". Before the orchestration, CPU contentions are happening on both data-nodes and result in longer job processing time in them, but after the orchestration to invoke correct VM migration(s), CPU contentions are removed and both VMs can process their tasks with much shorter latency.

Table III summarizes the results on the average total CPU usage (in levels) and the number of total VM migrations, which are collected after running the experiment in the HOE-DCN testbed for 18 service cycles. Here, the average total CPU usage refers to the average value of the total CPU usage (in levels) on all the overloaded servers per service cycle. We observe that with the same number of VM migrations, "KD-NO" provides much less overloaded servers than "NO w/o Low-level Knowledge". This confirms the timeliness and correctness of its network orchestration decisions.

TABLE III
METRICS RELATED TO HOE-DCN OPERATION (FIRST EXPERIMENT: CPU-BOUND JOBS)

	Average Total CPU Usage (in levels)	Total VM Migrations
NO w/o DCN Reconfiguration	15.3333	0
NO w/o Low-Level Knowledge	13.1667	12
KD-NO	7.1666	12

D. Experiment with I/O-bound Jobs

Our second experiment only considers I/O-bound jobs that demand for a lot of I/O usages and can generate high inter-rack traffic loads, but would not result in significant CPU usages. Therefore, we need to minimize congested ToR ports as many as possible, for reducing the average job completion time[2]. Fig. 10 compares the average job completion time in the three experimental scenarios, which exhibits the similar trends as those in Fig. 8. Table IV lists the results on the total number of congested ToR ports and the total number of inter-rack reconfigurations, which indicate that "KD-NO" would not require more inter-rack reconfigurations than "NO w/o Low-level Knowledge", but its total number of congested ToR ports is much smaller. All these results further confirm the effectiveness of "KD-NO", and specifically, it can adjust the

[2]We consider the I/O resources on each server is enough, *i.e.*, there would not be any I/O contentions, in the experiment.

inter-rack topology and the VMs' inter-rack communication schemes intelligently according to the traffic distribution in the foreseeable future. Hence, the inter-rack links in the HOE-DCN can be utilized wisely, and thus the completion time of I/O-bound jobs can be shortened significantly.

TABLE IV
METRICS RELATED TO HOE-DCN OPERATION (SECOND EXPERIMENT: I/O-BOUND JOBS)

	Total Congested ToR Ports	Total Inter-rack Reconfigurations
NO w/o DCN Reconfiguration	24	0
NO w/o Low-Level Knowledge	24	12
KD-NO	0	12

E. Experiment with Mixed Jobs

Finally, the third experiment run both mixed jobs (*i.e.*, both CPU- and I/O-bound ones) in the Hadoop clusters, to test the full capability of our KD-NO system. This time, the network orchestration system needs to make decisions to coordinate different types of applications, while without a proper design, it may generate decisions that are in conflict with each other.

The results on average job completion time in Fig. 11 still show the superiority of "KD-NO" over the two benchmarks. The rationale behind the trends in Fig. 11 can be explained with the results in Table V. As it does not invoke HOE-DCN reconfigurations, "NO w/o DCN Reconfiguration" may need to face simultaneous CPU contentions and traffic congestions. This is confirmed by the higher CPU usage and the larger number of congested ToR ports in Table V, when we compare its results with those of "KD-NO". Hence, the processing of both types of jobs would be hindered to lead to longer average job completion time. Without the proactive principle and low-level knowledge, it would be difficult for "NO w/o Low-level Knowledge" to make right decisions for the future. Therefore, although it also invokes a few VM migrations and inter-rack reconfigurations, it cannot avoid future CPU contentions and traffic congestions as "KD-NO" does. As a result, its average job completion time is longer than that of "KD-NO".

F. Discussion

Note that, in a real HOE-DCN, the applications would be much more various and complex than those considered in our experimental demonstrations. For instance, in addition to Hadoop workloads, there cloud be workloads from Spark and TensorFlow too. However, no matter what types of workloads get processed in the HOE-DCN, the operator always needs to schedule IT and bandwidth resources proactively and timely to ensure high QoS. This is exactly what our KD-NO system is designed to do. Therefore, although we only consider Hadoop workloads in the experimental demonstrations, our KD-NO system is not specifically designed for handling them. With sufficient training and minor modifications, it should be able to optimize the QoS of various applications running in the HOE-DCN. In other words, the operation principle of our KD-NO system is to extract the knowledge regarding matching degree between the HOE-DCN's configuration and the applications running in it and to optimize the HOE-DCN on-the-fly, which does not depend on the actual applications.

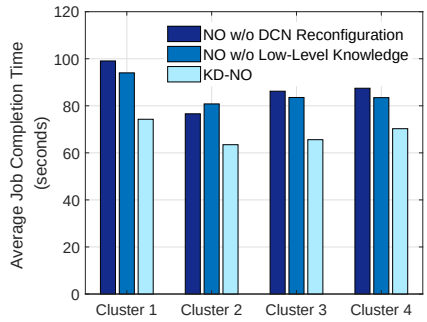


Fig. 11. Average job completion time (third experiment: mixed jobs).

Meanwhile, we also need to admit that due to the limited budget, our experimental demonstrations are still based on a simplified and small-scale HOE-DCN testbed and only consider Hadoop workloads. Although the testbed can be treated as a prototype to confirm the effectiveness of our KD-NO system in several simple use-cases, the generality and scalability of our proposal still deserve further investigations.

VII. CONCLUSION

In this paper, we proposed a KD-NO system for HOE-DCNs, which follows the predictive analytics in human behaviors to first forecast VMs' future demands on IT and bandwidth resources and then find the optimal HOE-DCN configuration based on the predictions. When collecting telemetry data about the resource utilization in an HOE-DCN, our KD-NO system analyzed the spatial and temporal correlations of the data, and predicted future workloads and traffic matrix of VMs with several DL modules to fetch the low-level knowledge. Next, it leveraged the low-level knowledge for network orchestration. We formulated the network orchestration problem as an MILP model, proved its $\mathcal{NP}$-hard, and proposed a DRL-based online algorithm. Specifically, based on the extracted knowledge, the DRL-based AI modules could coordinate the IT and bandwidth resources in the HOE-DCN on-the-fly, so as to achieve a high matching degree between the HOE-DCN's configuration and the applications running in it.

To verify the effectiveness of our proposal, we built an HOE-DCN testbed with KD-NO. The experiments run Hadoop applications in the testbed, and demonstrated that our KD-NO system can make timely and correct network orchestration decisions in different experimental schemes by leveraging the two-level knowledge, and therefore largely reduce the average job completion time. Moreover, we found that even though low-level knowledge cannot be directly used for network orchestration, it is indispensable in the overall KD-NO process.

ACKNOWLEDGMENTS

This work was supported in part by the NSFC Projects 61701472 and 61771445, CAS Key Project (QYZDY-SSW-JSC003), NGBWMCN Key Project (2017ZX03001019-004).

TABLE V
METRICS RELATED TO HOE-DCN OPERATION (THIRD EXPERIMENT: MIXED JOBS)

	Average Total CPU Usage (in levels)	Total Congested ToR Ports	Total VM Migrations	Total Inter-rack Reconfigurations
NO w/o DCN Reconfiguration	4	12	0	0
NO w/o Low-Level Knowledge	3.6771	12	6	18
KD-NO	0	2	10	18

REFERENCES

[1] P. Lu *et al.*, "Highly-efficient data migration and backup for Big Data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[2] K. Wu, P. Lu, and Z. Zhu, "Distributed online scheduling and routing of multicast-oriented tasks for profit-driven cloud computing," *IEEE Commun. Lett.*, vol. 20, pp. 684–687, Apr. 2016.

[3] "Cisco global cloud index: Forecast and methodology, 2016-2021." [Online]. Available: https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[4] Y. Tian, R. Dey, Y. Liu, and K. Ross, "Topology mapping and geolocating for China's Internet," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, pp. 1908–1917, Sept. 2013.

[5] Z. Zhu *et al.*, "RF photonics signal processing in subcarrier multiplexed optical-label switching communication systems," *J. Lightw. Technol.*, vol. 21, pp. 3155–3166, Dec. 2003.

[6] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[7] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[8] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[9] N. Farrington *et al.*, "Helios: A hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 339–350, Oct. 2010.

[10] W. Lu *et al.*, "AI-assisted knowledge-defined network orchestration for energy-efficient datacenter networks," *IEEE Commun. Mag., in Press*, 2018.

[11] H. Bazzaz *et al.*, "Switching the optical divide: Fundamental challenges for hybrid electrical/optical datacenter networks," in *Proc. of SOCC 2011*, pp. 1–8, Aug. 2011.

[12] P. Lu, Q. Sun, K. Wu, and Z. Zhu, "Distributed online hybrid cloud management for profit-driven multimedia cloud computing," *IEEE Trans. Multimedia*, vol. 17, pp. 1297–1308, Aug. 2015.

[13] H. He, Y. Zhao, J. Wu, and Y. Tian, "Cost-aware capacity provisioning for internet video streaming CDNs," *Comput. J.*, vol. 58, pp. 3255–3270, Dec. 2015.

[14] W. Lu *et al.*, "Profit-aware distributed online scheduling for data-oriented tasks in cloud datacenters," *IEEE Access*, vol. 6, pp. 15 629–15 642, 2018.

[15] Z. Zhu *et al.*, "Demonstration of cooperative resource allocation in an OpenFlow-controlled multidomain and multinational SD-EON testbed," *J. Lightw. Technol.*, vol. 33, pp. 1508–1514, Apr. 2015.

[16] W. Lu *et al.*, "Leveraging predictive analytics to achieve knowledge-defined orchestration in a hybrid optical/electrical DC network: Collaborative forecasting and decision making," in *Proc. of OFC 2018*, pp. 1–3, Mar. 2018.

[17] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–98, Apr. 2014.

[18] S. Li *et al.*, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, pp. 12–20, Mar./Apr. 2017.

[19] X. Wang *et al.*, "PNPL: Simplifying programming for protocol-oblivious SDN networks," *Comput. Netw.*, vol. 147, pp. 64–80, Dec. 2018.

[20] A. Mestres *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 47, pp. 2–10, Jul. 2017.

[21] B. Li, W. Lu, S. Liu, and Z. Zhu, "Deep-learning-assisted network orchestration for on-demand and cost-effective vNF service chaining in inter-DC elastic optical networks," *J. Opt. Commun. Netw.*, vol. 10, pp. D29–D41, Oct. 2018.

[22] D. Borthakur, *The Hadoop Distributed File System: Architecture and Design*. Apache Software Foundation, 2007.

[23] G. Wang *et al.*, "c-Through: Part-time optics in data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 327–338, Oct. 2011.

[24] K. Chen *et al.*, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Trans. Netw.*, vol. 22, pp. 498–511, Apr. 2014.

[25] W. Fang *et al.*, "Joint defragmentation of optical spectrum and IT resources in elastic optical datacenter interconnections," *J. Opt. Commun. Netw.*, vol. 7, pp. 314–324, Mar. 2015.

[26] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[27] W. Fang *et al.*, "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks," *IEEE Commun. Lett.*, vol. 20, pp. 1539–1542, Aug. 2016.

[28] S. Kandula *et al.*, "The nature of data center traffic: Measurements & analysis," in *Proc. of ACM SIGCOMM 2009*, pp. 202–208, Aug. 2009.

[29] J. Guo and Z. Zhu, "When deep learning meets inter-datacenter optical network management: Advantages and vulnerabilities," *J. Lightw. Technol.*, vol. 36, pp. 4761–4773, Oct. 2018.

[30] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[31] J. Jiang *et al.*, "Joint VM placement and routing for data center traffic engineering," in *Proc. of INFOCOM 2012*, pp. 2876–2880, Mar. 2012.

[32] W. Fang *et al.*, "VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Comput. Netw.*, vol. 57, pp. 179–196, Jan. 2013.

[33] B. Kong *et al.*, "Demonstration of application-driven network slicing and orchestration in optical/packet domains: On-demand vDC expansion for Hadoop MapReduce optimization," *Opt. Express*, vol. 26, pp. 14 066–14 085, 2018.

[34] L. Chen, J. Lingys, K. Chen, and F. Liu, "AuTo: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proc. of ACM SIGCOMM 2018*, pp. 191–205, Aug. 2018.

[35] S. Salman *et al.*, "DeepConf: Automating data center network topologies management with machine learning," in *Proc. of NetAI 2018*, pp. 8–14, Aug. 2018.

[36] "Open Network Automation Platform (ONAP)." [Online]. Available: https://www.onap.org/

[37] Y. Wang, P. Lu, W. Lu, and Z. Zhu, "Cost-efficient virtual network function graph (vNFG) provisioning in multidomain elastic optical networks," *J. Lightw. Technol.*, vol. 35, pp. 2712–2723, Jul. 2017.

[38] K. Han *et al.*, "Application-driven end-to-end slicing: When wireless network virtualization orchestrates with NFV-based mobile edge computing," *IEEE Access*, vol. 6, pp. 26 567–26 577, 2018.

[39] S. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, pp. 1345–1359, Oct. 2010.

[40] "Google Cluster-Usage Traces." [Online]. Available: https://github.com/google/cluster-data

[41] C. Reiss *et al.*, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Tech. Rep. ISTC-CC-TR-12-101*, pp. 1–21, Apr. 2012. [Online]. Available: http://www.pdl.cmu.edu/PDL-FTP/CloudComputing/ISTC-CC-TR-12-101.pdf

[42] A. Weigend, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Routledge, 2018.

[43] M. Garey and D. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. New York, 1979.

[44] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, Feb. 2016. [Online]. Available: https://arxiv.org/abs/1509.02971

[45] "collected." [Online]. Available: https://collectd.org/

[46] "sFlow." [Online]. Available: https://sflow.org/