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 task 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], traffic generated by clouds has been increasing exponentially, the majority of which is within datacenters (DCs) [3]. Hence, DC networks (DCNs) are facing challenges to accommodate such enormous traffic cost-efficiently with sustainable technologies. Meanwhile, it is known that compared with electrical packet switching (EPS), optical circuit switching (OCS) provides larger bandwidth capacity and consumes less power [4–6]. 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 [5, 7]. The top-of-rack (ToR) switches in an HOE-DCN are interconnected by two inter-rack networks based on EPS and OCS, respectively, as shown in Fig. 1(a).

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 February 20, 2019.

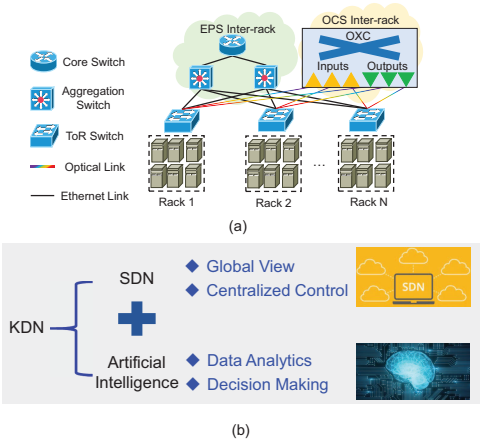


Fig. 1. Illustrations of (a) HOE-DCN, and (b) KDN.

Therefore, the delay-sensitive and/or transient inter-rack traffic can be forwarded by the high-speed Ethernet switches, while the inter-rack traffic that is bandwidth-intensive and/or long-lasting can be routed through an optical cross-connect (OXC).

Although the HOE-DCNs are promising, they cannot work around one of the most challenging demands from the traditional 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 [8–10]. This demand actually becomes even more challenging in an HOE-DCNs, since the addition of the OCS-based inter-rack network complicates its network control and management (NC&M) [11, 12]. 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.

The emerging of software-defined networking (SDN) [13, 14] provides new enabling technologies to architect more programmable, effective and reliable NC&M for DCNs. However, the NC&M enabled by SDN is still reactive, which means that the controller always makes decisions based on the current network status. This would limit the 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 require the involvement of knowledge-defined networking (KDN) [15, 16], which is essentially the

symbiosis of SDN and artificial intelligence (AI) in Fig. 1(b). Specifically, with the centralized control in SDN, the operator visualizes the 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.

Although the idea of applying KDN to HOE-DCNs sounds promising, the actual implementation still needs to solve 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 between the HOE-DCN's configuration and the network applications running in it, and makes wise decisions based on the high-level knowledge.

In this work, we develop a KD-NO system for HOE-DCNs by extending our preliminary study in [12], to address the problems mentioned above. Specifically, the KD-NO system follows the predictive analytics in human behaviors, which means that it first forecasts the 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. Our KD-NO system first 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, we study how to leverage the low-level knowledge for network orchestration. The problem is formulated as a mixed integer linear programming (MILP) model, and we prove its $\mathcal{NP}$-hardness. Hence, we decide to solve it with DRL, and design a DRL-based online scheme that 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 experimentally demonstrate it in a small-scale but real HOE-DCN testbed. The experiments run Hadoop applications [17] 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. Therefore, the task completion time of the Hadoop applications is reduced effectively.

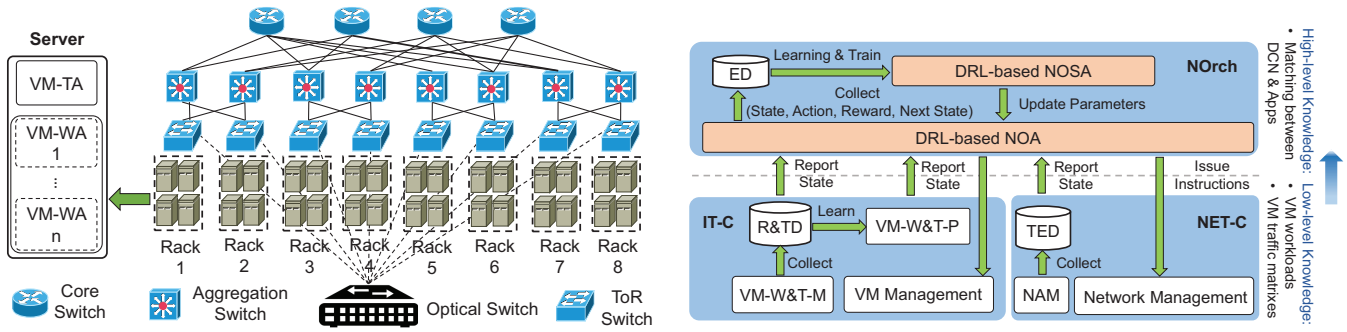
The rest of the paper is organized as follows. Section II reviews the related work. In Section III, we introduce the architecture of our KD-NO system. The designs for fetching the low-level knowledge about an HOE-DCN are discussed in Section IV, while how to extract the high-level knowledge and use it 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

Previously, to seamlessly integrate the advantages of both EPS and OCS for cost-efficient data transfers, a few HOE-DCN architectures were proposed in [5, 18–20]. 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 [8, 21–23]. Specifically, the network orchestration 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 [5, 19, 20] considered how to configure the OCS and EPS inter-rack networks in an HOE-DCN, for satisfying VM traffic demands. However, these approaches still have several limitations. Firstly, they either optimized the configuration reactively based on current network status, or just predicted future traffic matrixes roughly without considering the temporal and spatial correlations of VM traffic, which is known to be harmful for maintaining the prediction accuracy [24, 25]. 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 [26] 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 [27–29]. 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 KDN to realize proactive and agile network orchestration. More recently, people try to incorporate DL or DRL in the network orchestration of DCNs [30, 31]. Chen *et al.* [30] proposed a DRL-based algorithm and incorporated to optimize the traffic scheduling in DCNs. The study in [31] considered an HOE-DCN as the background and discussed how to leverage DL to realize automatic (re)configuration of the OCS-based inter-rack network in it. However, these studies neither covered the full spectrum of KD-NO nor introduced multiple AI modules to solve the three problems in network orchestration in a collaborative manner.



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

Fig. 2. 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.

III. System Architecture

In this section, we explain the system architecture of the HOE-DCN with KD-NO, which is illustrated in Fig. 2.

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

Fig. 2(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' applications [32]. 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. 2(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. 2(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. 2(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

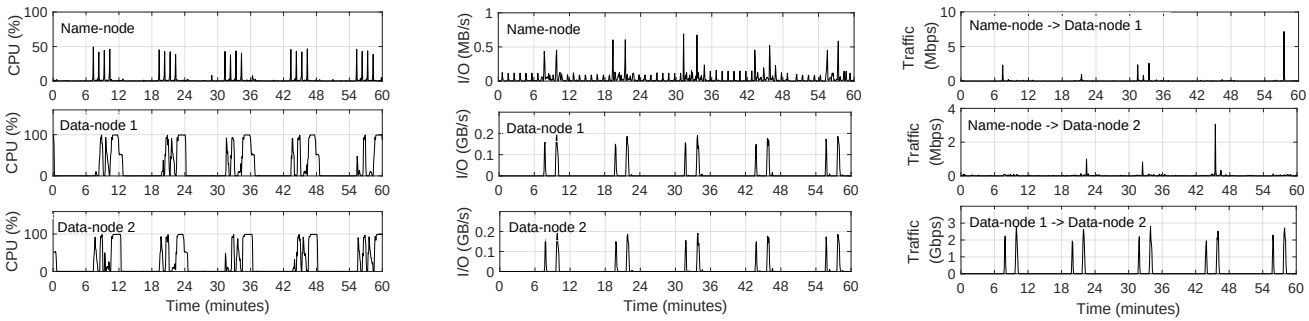
We allocate a VM-WA on each VM to monitor its workloads in terms of CPU and I/O usages. The collected telemetry data is analyzed by the IT-C for predicting future VM workloads and extracting low-level knowledge. Note that, there might be numerous VMs in the HOE-DCN, and thus analyzing and predicting 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 [17] 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 [33] 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 tasks generated according to the data traces released by Google [34]. In the experiment, the original diurnal pattern of the tasks (*i.e.*, for 24 hours) gets scaled down to 12 minutes. The VW-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. 3(a) and 3(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 80.78%, while that of their I/O workloads is 94.32%. 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 [35] of the IT workloads, and Figs. 4(a) and 4(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



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

Fig. 3. Telemetry data collected for a Hadoop cluster that includes three VMs.

of *Data-node* 1 repeat themselves approximately after certain durations and thus are predictable.

Since the IT workloads are predictable and the prediction technique for a time series is quite mature [36], 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 the HOE-DCN, we also implement transfer learning in the VM-W&T-P to enable online training during operation. 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, and in the process, 95% of the historical data is put in the training set while the remaining 5% is the testing set. After that, the predictor is put in the VM-W&T-P and enters the online training. During operation, the predictor forecasts the VM workload in the next provisioning period using the trained LSTM-NN, while after the actual VM workload has been collected, it adds this newly-collected data in its training set to retrain and update its LSTM-NN.

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 20 provisioning periods 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 I
RESULTS ON IT WORKLOAD PREDICTIONS

	Offline Training			
	CPU		I/O	
	Prediction Accuracy	Training Time (sec)	Prediction Accuracy	Training Time (sec)
Data-node 1	99.6%	22.26	89.4%	20.53
Data-node 2	90.9%	–	88.7%	–
	Online Training			
Data-node 1	99.6%	1.34	89.4%	1.42
Data-node 2	89.1%	1.43	87.7%	1.38

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

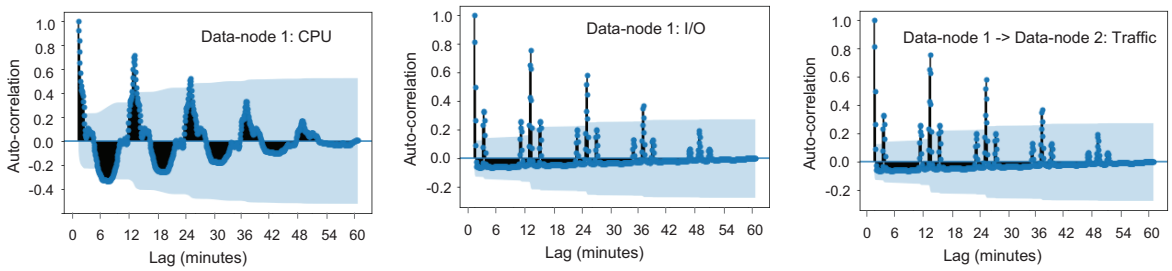
Fig. 2(a) indicates that the VM-TA on each server collects the traffic matrix of the local VMs and reports it to the IT-C. The proposal discussed in the previous subsection can also be applied to fetch low-level knowledge from the traffic data. And for the experiment, the measurements on the traffic among the VMs are plotted in Fig. 3(c). Note that, the inter-rack traffic between a VM pair can be affected by its routing scheme. Specifically, since the capacities of EPS/OCS-based inter-rack links are different (*i.e.*, set as 1 Gbps and 10 Gbps, respectively, in our HOE-DCN testbed), the peak throughput of the traffic could change. To resolve this issue, we preprocess the traffic data by replacing the real-time data-rate with data-transfer bursts, each of which is modeled as the start-time and total data-transfer volume. This is because the inter-rack traffic is usually bursty (as shown in Fig. 3(c)) and the data-transfer bursts would not be affected by the routing scheme.

The screening process determines that in Fig. 3(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. 4(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. Table II summarizes the results regarding the traffic predictor.

Note that, we also define a few discrete levels to quantify the CPU and I/O workloads and inter-rack traffic, and thus they can also be represented by a much more abstracted information model to restrict the optimization space for the NOrch.

TABLE II
RESULTS ON TRAFFIC PREDICTION

	Offline Training	
	Prediction Accuracy	Training Time
Data-node 1 $\rightarrow$ *Data-node* 2	93.5%	29.80 sec
	Online Training	
Data-node 1 $\rightarrow$ *Data-node* 2	93.5%	1.42 sec



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

Fig. 4. Auto-correlations of *Data-node* 1's IT workloads and traffic between *Data-nodes* 1 and 2.

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 overloaded server.
- 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

[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.

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.

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

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

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

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

$$\sum_{G \in \mathbf{G}} y_G = 1. \quad (6)$$

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

$$z_{v,v'}^{r,p} \leq \frac{y_G + x_{v,s} + x_{v',s'}}{3},\ \forall v, v' \in V,\ r, r' \in R,\ s \in S_r, \quad (7)$$
$$s' \in S_{r'},\ G \in \mathbf{G},\ p \in P_{G,r,r'} :\ v \neq v', r \neq r', w_{v,v'}^T > 0,$$

$$\sum_{p \in P_{G,r,r'}} z_{v,v'}^{r,p} \leq 1,\ \forall v, v' \in V,\ r, r' \in R, \quad (8)$$
$$s \in S_r,\ s' \in S_{r'},\ G \in \mathbf{G} :\ v \neq v', r \neq r', w_{v,v'}^T > 0,$$

$$\sum_{p \in P_{G,r,r'}} z_{v,v'}^{r,p} \geq y_G + x_{v,s} + x_{v',s'} - 2,\ \forall v, v' \in V,\ r, r' \in R, \quad (9)$$
$$s \in S_r,\ s' \in S_{r'},\ G \in \mathbf{G} : v \neq v', r \neq r', w_{v,v'}^T > 0.$$

Eqs. (7)-(9) ensure that if there is traffic between two VMs (*i.e.*, $w_{v,v'}^T > 0$) 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.

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

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

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

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

$$M \cdot g_s \geq \left(\sum_{v \in V} x_{v,s} \cdot w_v^C \right) - W_s^C, \quad \forall r \in R,\ s \in S_r, \quad (12)$$

$$M \cdot g_s \geq \left(\sum_{v \in V} x_{v,s} \cdot w_v^I \right) - W_s^I, \quad \forall r \in R,\ s \in S_r. \quad (13)$$

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

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

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

$$M \cdot g_{r,p} \geq \left(\sum_{v,v' : v \neq v'} z_{v,v'}^{r,p} \cdot w_{v,v'}^T \right) - b_p \cdot L_p, \quad \forall r \in R,\ p \in P_r. \quad (15)$$

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 [37]. 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

$$\text{Minimize} \quad \sum_{v} g_v, \quad (16)$$

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 $\{w_v^C, w_v^I\}$, and a set of servers $s \in S$, each of which also has a 2D resource capacity $\{W_s^C, W_s^I\}$, 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 [37]. 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 space and action space are relatively large in our DRL model, we leverage the deep deterministic policy gradient (DDPG) [38] to design its operation procedure.

DDPG uses the empirical values in Q-learning to promote the efficiency of parameter update in policy gradient. Specifically, 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. 2(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). When 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.

Algorithm 1: DRL-based Network Orchestration in NOrch

```
1  Thread I: Online Operation
2      for each service cycle do
3          NOA receives state S_i from IT-C and NET-C;
4          NOA takes action A_i according to Eq. (17);
5          NOA legalizes A_i according to certain rules;
6          NOA calculates reward R_i;
7          NOA stores {S_{i-1}, A_{i-1}, R_{i-1}, 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 {θ^{p'}, θ^{Q'}} with Eqs. (18)-(20);
14         if certain iterations have been passed since the
             last update then
15             NOA updates {θ^p, θ^Q} with Eq. (21);
16         end
17     end
18 End
```

Fig. 5 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. 5, 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.

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 three racks, hardware-based OpenFlow switches, and a reconfigurable optical switch, as shown in Fig. 6. 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 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.

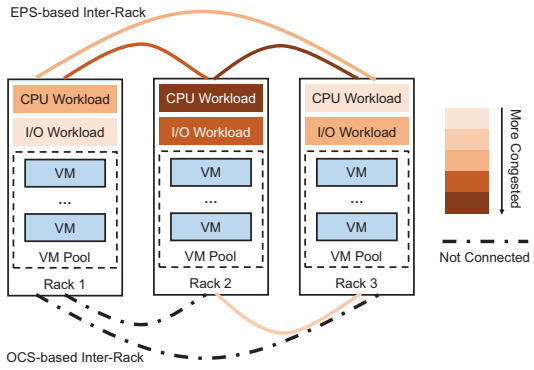


Fig. 5. Example on high-level knowledge regarding the matching degree between HOE-DCN's configuration and applications running in it.

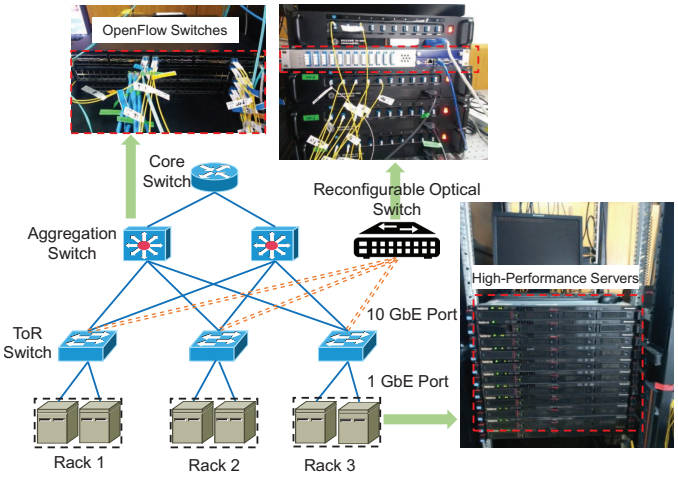


Fig. 6. Experimental setup of the HOE-DCN prototype with KD-NO.

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 [39] and sFlow [40], and realize VM-W&T-P by referring to the LTSM-NN. On the other hand, the NET-C is implemented by modifying ONOS [41] 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

The experiments run Hadoop applications in the testbed and carry out three experimental scenarios. In each scenario, we first create 9 VMs on the three racks and group them into three Hadoop clusters, each of which has one name-node and two data-nodes, and then run CPU- or/and I/O-bound tasks on the clusters. To emulate the practical situations, we generate the tasks according to the real data traces released by Google [34], and scale the original diurnal pattern of the tasks (*i.e.*, for 24 hours) down to 12 minutes, for saving experiment time. Each service cycle of network orchestration is set as 2 minutes. We use the following three experimental scenarios to demonstrate the effectiveness of our proposed KD-NO system.

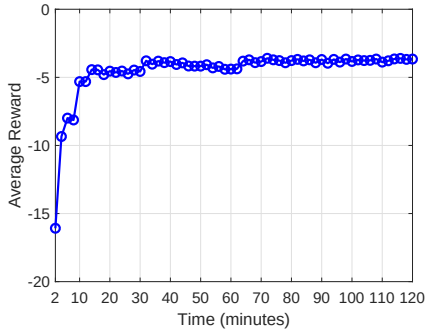


Fig. 7. Convergence performance of NOrch's online training in experiments.

- *Network orchestration without DCN reconfiguration* (**NO w/o DCN Reconfiguration**): it just runs the generated tasks, 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 shows convergence performance of the online training that our DRL-based NOrch performs in the HOE-DCN testbed. The results indicate that starting as an untrained AI module, the NOrch can output relatively high rewards after the experiment runs for $\sim$20 minutes (*i.e.*, 10 service cycles).

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 service cycle. 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, the server 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 service cycle, respectively. The overload threshold on the I/O workload of a sever is set as 3, which is equivalent to a total I/O workload of more than 9 GB in a service cycle 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 switch ports is set as 1. For the optimization objective in Eq. (1), we set all the values of $\{\alpha, \beta, \gamma\}$ as 1 to give them equal importance. Meanwhile, for

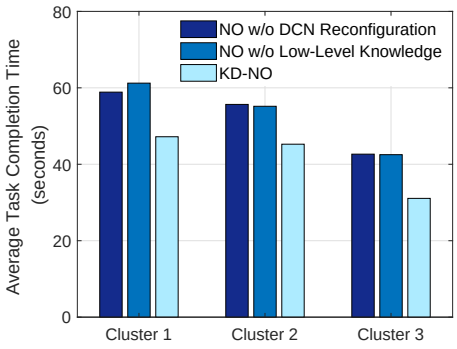


Fig. 8. Average task completion time (first experiment: CPU-bound tasks).

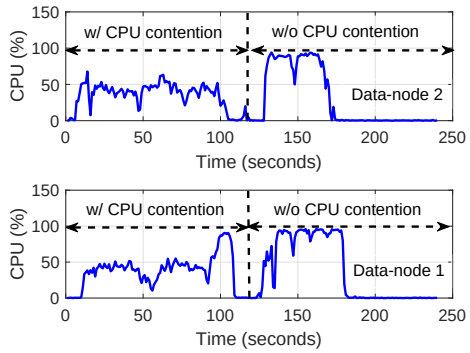


Fig. 9. KD-NO's effect to remove CPU contentions (first experiment: CPU-bound tasks).

each overloaded server, we record the actual degraded duration of each VM on it in a service cycle and use it to evaluate the degree of server congestion more accurately.

C. Experiment with CPU-bound Tasks

The first experiment only considers CPU-bound tasks, 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 task 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 compare the average task completion time achieved by the three experimental scenarios. It can be seen that "KD-NO" provides much shorter task 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 tasks' 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. Specifically, Fig. 9 shows the CPU workloads of two data-nodes before and after the network orchestration conducted by "KD-NO", where before the orchestration, CPU contentions are happening on both data-nodes and result in longer task 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.

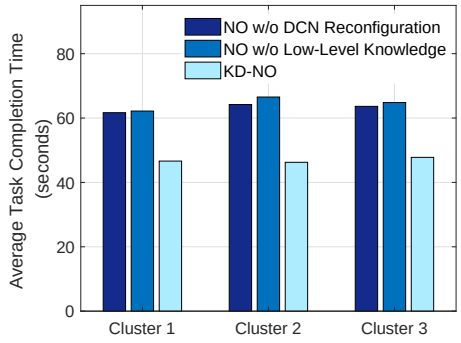


Fig. 10. Average task completion time (second experiment: I/O-bound tasks).

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 an hour. Here, the average total CPU usage refers to the average value of the total CPU usage (in levels) on all the 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 the network orchestration decisions from "KD-NO".

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

	Average Total CPU Usage (in levels)	Total VM Migrations
NO w/o DCN Reconfiguration	11.3333	0
NO w/o Low-Level Knowledge	10.6667	20
KD-NO	3.3333	20

D. Experiment with I/O-bound Tasks

Our second experiment only considers I/O-bound tasks 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 task completion time[2]. Fig. 10 compares the average task 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

[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.

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 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 tasks can be shortened significantly.

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

	Total Congested ToR Ports	Total Inter-rack Reconfigurations
NO w/o DCN Reconfiguration	30	0
NO w/o Low-Level Knowledge	30	30
KD-NO	0	30

E. Experiment with Mixed Tasks

Finally, the third experiment run both mixed tasks (*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 task 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 any HOE-DCN reconfiguration, "NO w/o DCN Reconfiguration" may need to face simultaneous CPU contentions and traffic congestions in the HOE-DCN, which can be proved by the higher CPU usage and the larger number of congested ToR ports in Table V, when we compare its corresponding results with those of "KD-NO". This would hinder the processing of both types of tasks and make its average task completion time longer, as shown in Fig. 11. On the other hand, without the proactive principle and low-level knowledge, it would be difficult for "NO w/o Low-level Knowledge" to make right network orchestration decisions for the future. Therefore, even though it also invoke 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 task completion time is longer than that of "KD-NO".

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.

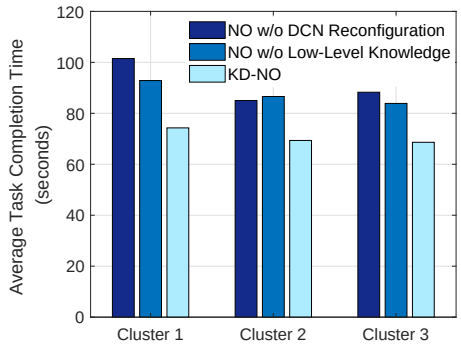


Fig. 11. Average task completion time (third experiment: mixed tasks).

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 achieving 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 task 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).

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] 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.
[5] 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.
[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] W. Lu *et al.*, "AI-assisted knowledge-defined network orchestration for energy-efficient datacenter networks," *IEEE Commun. Mag., in Press*, 2018.
[8] 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.

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

	Average Total CPU Usage (in levels)	Total Congested ToR Ports	Total VM Migrations	Total Inter-rack Reconfigurations
NO w/o DCN Reconfiguration	2.6667	30	0	0
NO w/o Low-Level Knowledge	2.6667	30	15	20
KD-NO	0	0	15	25

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

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

[18] A. Greenberg *et al.*, "VL2: a scalable and flexible data center network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 51–62, Oct. 2009.

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

[20] 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.

[21] 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.

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

[23] 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.

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

[25] 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.

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

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

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

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

[34] G. Amvrosiadis *et al.*, "Bigger, longer, fewer: what do cluster jobs look like outside google?" *Tech. Rep. CMU-PDL-17-104*, Oct. 2017. [Online]. Available: http://www.pdl.cmu.edu/PDL-FTP/CloudComputing/CMU-PDL-17-104.pdf

[35] D. Montgomery, C. Jennings, and M. Kulahci, *Introduction to Time Series Analysis and Forecasting*. John Wiley & Sons, 2015.

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

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

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

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

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

[41] "ONOS." [Online]. Available: https://onosproject.org/