Workload-Aware Revenue Maximization in SDN-Enabled Data Center

Haitao Yuan[1], Jing Bi[2], Jia Zhang[3], Wei Tan[4], and Keman Huang[5]
[1]School of Software Engineering, Beijing Jiaotong University, Beijing, China
[2]School of Software Engineering, Beijing University of Technology, Beijing, China
[3]Department of Electrical and Computer Engineering, Carnegie Mellon University, Moffett Field, USA
[4]IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA
[5]Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA
htyuan@bjtu.edu.cn, bijing@bjut.edu.cn, jia.zhang@sv.cmu.edu, wtan@us.ibm.com, keman@mit.edu

***Abstract*—Nowadays many companies and organizations choose to deploy their applications in data centers to leverage resource sharing. The increase in tasks of multiple applications, however, makes it challenging for a data center provider to maximize its revenue by intelligently scheduling tasks in software-defined networking (SDN)-enabled data centers. Existing SDN controllers only reduce network latency while ignoring virtual machine (VM) latency, thus may lead to revenue loss. In the context of SDN-enabled data centers, this paper presents a workload-aware revenue maximization (WARM) approach to maximize the revenue from a data center provider's perspective. The core idea is to jointly consider the optimal combination of VMs and routing paths for tasks of each application. Comparing with state-of-the-art methods, the experimental results show that WARM yields the best schedules that not only increase the revenue but also reduce the round-trip time of tasks of all applications.**

***Keywords*-Cloud data center, revenue maximization, software-defined networking, task scheduling, metaheuristic**

I. Introduction

Currently it is common for heterogeneous applications to run in data centers and deliver services to users around the world, through resource sharing. As a core technique, virtualization is widely implemented to efficiently utilize server and network resources in data centers. Each physical server is pooled and divided into multiple virtual machines (VMs) in a virtualized infrastructure. To ensure system scalability and stability, each application may concurrently run in either heterogeneous or homogeneous VMs in data centers. Admitted tasks of each application in data centers typically have to go through data center network before arriving at VMs. Therefore, the round trip time (RTT) of each task comprises of the network latency needed in network and the VM latency needed in VMs. The latency of each task has a great impact on users' experience and may bring the loss of revenue to a data center provider if it becomes too large. It is reported that a half-second latency leads to 20% loss of users' traffic in Google [1].

Software-defined networking (SDN) is able to provide the centralized network control with OpenFlow-enabled forwarding devices (routers or switches) [2]. It supports the separation of switches in a data plane and controllers in a control plane. The network control and management is moved to a remote controller that determines globally optimized routing for the whole network in data centers. OpenFlow is a standard protocol to exchange information between the data and control planes. An SDN controller manages network devices with OpenFlow APIs, and provides multiple functions such as load balancing, forwarding, network virtualization, and quality of service (QoS) evaluation.

An SDN controller periodically fetches network information (such as utilization and available bandwidth of each link). Therefore, SDN enables data centers to meet tasks' requirements in a real-time and scalable manner. Thus, fine-grained task scheduling and high-performance network management for multiple applications can be achieved. Recently, Google has successfully applied SDN to manage its multiple data centers and realized flexible and efficient traffic engineering [3]. It is shown that network utilization is improved greatly and task loss is significantly reduced with SDN.

However, existing SDN controllers can only obtain network information in data centers, and therefore can only optimize the network latency of tasks by controlling switches. It should be noted that VM latency is also very important for users' experience. Here, VM latency means the response time of each task of an application on a VM. For example, if tasks are scheduled to run on a VM that is already overloaded, significant VM latency may occur. In addition, routing methods in the current SDN controllers cannot well jointly consider the minimization of the network and VM latency. For example, the Floodlight controller [4] chooses the shortest path to select VMs. It may lead to large RTT because of network congestion and large VM latency. Typically, RTT of each task is transformed into its corresponding revenue based on the items defined in service-level agreements (SLAs). Users contribute corresponding revenue to a data center provider based on the execution of their tasks. Clearly, revenue maximization is the most important for a data center provider [5].

To tackle the aforementioned issues, this work investigates the revenue maximization problem for SDN-enabled data center providers. A workload-aware revenue maximization (WARM) approach is presented. By considering workload in an SDN-enabled network, VMs, and SLAs, WARM can

2159-6190/17 $31.00 © 2017 IEEE
DOI 10.1109/CLOUD.2017.12

effectively increase the revenue of the data center provider by specifying the optimal combination of VMs and routing paths for tasks of each given application. To evaluate the effectiveness of the WARM, publicly available task data in Google data centers [6] is used. Comprehensive comparisons show that WARM outperforms several existing task routing methods in terms of revenue and RTT.

The main contributions of this work are summarized in three-fold. First, an architecture of a workload-aware SDN controller is presented that consists of an SDN controller and a cloud controller. The architecture enables the controller to periodically update information in network links and the workload in VMs. Second, based on the architecture, WARM is developed to maximize the revenue of a data center provider by jointly considering network workload, VM status, and SLAs. WARM can well specify the optimal combination of VMs and routing paths for tasks of each application, and realize intelligent scheduling of tasks of all applications. Third, this work makes comprehensive comparison between WARM and its two peers, i.e., Open Shortest Path First [7], and Round Robin [8].

The remaining of this paper is organized as follows. Related work is reviewed in Section II. Section III shows the architecture of an SDN-enabled data center. The workload-aware revenue maximization problem for a data center provider is formulated in Section IV. Section V describes the WARM method in detail. Section V evaluates and compares WARM with peers based on a widely used network emulator Mininet [9] . Section VII concludes the paper.

II. RELATED WORK

This section gives a summary of existing studies related to the research topic in this paper, and further shows the similarities and differences between the proposed WARM method and existing studies.

A. Traffic engineering

Several SDN studies focus on the traffic engineering problem [10]–[12]. For example, an ISP-internal service and traffic management mechanism are presented in [10]. Several SDN-based designs in the network layer are introduced to achieve load balancing of traffic. In this way, a fine-grained traffic engineering that supports high-efficient multicast load balance inside ISP networks is enabled. In [11], several challenging research problems about traffic engineering in SDN-enabled networks are discussed. These research issues include traffic analysis, fault tolerance, flow scheduling, and topology change. It is demonstrated that a global network view (e.g., flow characteristics and network status) can be exploited to realize better traffic management and control. In [12], a power management method based on dynamic voltage and frequency scaling (DVFS) is presented to realize energy-efficient routing of tasks in control and data planes.

The aforementioned approaches only consider traffic engineering in a network, thus can only optimize the network latency of tasks. However, for the tasks of each application, the VM latency may also have a great impact on tasks' RTT experienced by consumers. The reason is that large VM latency may lead to large RTT of tasks, thus decrease the revenue of the data center providers.

B. Revenue optimization

Several existing studies have presented different approaches to achieve revenue maximization for data center providers [13]–[15]. In [13], a controller is proposed to support a dynamic fine-grained resource provisioning method in a non-equilibrium states VCDC. Based on this model and SLAs, a hybrid meta-heuristic algorithm is developed to determine the allocation of CPU and I/O resources, where the revenue of application services is maximized and machine-level energy expenditure is minimized. In [14], a multiserver system is treated as an M/M/m queuing system such that the expected revenue brought by the execution of each task is calculated. The optimal configuration of a multiserver system is studied and determined based on applications' characteristics. In [15], a revenue-based resource allocation mechanism is implemented to dynamically distribute VM resources in a server, such that the total revenue generated based on SLAs is maximized.

These studies typically model and analyze tasks' response time needed in servers or VMs in data centers, and further achieve revenue maximization for the data center providers by converting the response time into corresponding revenue according to SLAs. However, these studies ignore the network latency that may also have great impact on the revenue of the data center providers. Different from these studies, our work explicitly provides the mathematical modeling of RTT including both network and VM latency. Based on the RTT modeling and pre-defined SLAs, the revenue optimization problem for a data center provider is formulated and further solved by a hybrid meta-heuristic algorithm, i.e., WARM.

C. Load balance

Load balance in data centers has been a challenging topic [16]. In [16], performance and power-constrained load distribution approaches in large-scale cloud data centers are presented. In these methods, optimal load distribution and power allocation in heterogeneous multicore processors are addressed to realize high-efficient utilization of resources in clouds. The work in [17] presents an approach to guarantee the scalability of data centers by adopting multiple SDN controllers. This approach alleviates the overload of a single controller and enhances the system's performance by considering load balance among multiple controllers in mega data centers. In [18], a sampling-based load balancing protocol is designed to provide high-performance load balancing for data planes. This protocol periodically collects sampling data in several paths to every switch, and schedules flows to the path with the maximum bandwidth resource.

Nevertheless, these studies focus on load balance with the aim of reducing tasks' response time or enhancing the system's performance. In contrast, this paper can realize load balance and achieve revenue maximization for the data center provider

by jointly considering workload in SDN-enabled network, VMs, and SLAs.

III. Architecture overview

This section shows the architecture of a data center that enables a workload-aware SDN controller. As illustrated in Fig. 1, this controller can achieve revenue maximization for a data center provider by intelligently scheduling the tasks of all applications. The architecture contains workload-aware SDN controllers where the proposed WARM method is executed. WARM jointly considers the workload in both network links and VMs, and determines a VM and routing path for the tasks of an application. In this way, WARM can maximize the revenue of the data center provider by smartly scheduling all the tasks of multiple applications.

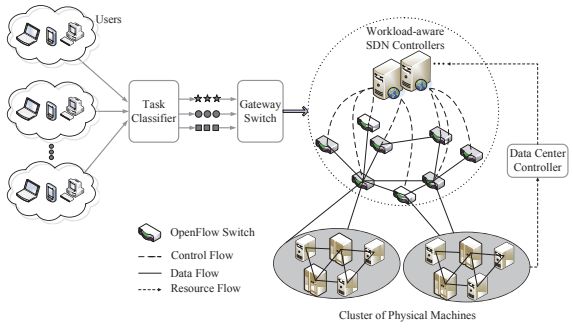

Fig. 1. Architecture of the SDN-enabled data center.

In a traditional SDN architecture, the SDN controller manages traffic information in network links, and it can only control OpenFlow-enabled switches in a network. Thus, the traditional SDN controller can only optimize the task routing in the network and decrease the time needed in the network for tasks of each application. However, the time needed in a VM, i.e., the VM latency, may also have large impact on users' experience. For example, arriving tasks might experience relatively large VM latency if they are unsuitably scheduled to an already overloaded VM. In this case, RTT of tasks is relatively large, and the revenue brought by their execution may be reduced or even become negative based on SLA.

As shown in Fig. 1, the architecture of an SDN-enabled data center contains a cloud controller and a collection of SDN controllers. The SDN controllers periodically collect the network information including topology and link utilization, and determine the scheduling strategy for the tasks of given applications. The cloud controller periodically collects the VM information (e.g., VM utilization and remaining tasks in each VM) and sends it to the SDN controllers. In this way, such two types of components build a workload-aware SDN controller. Currently, to guarantee stability and scalability of a data center, each application is typically deployed in multiple heterogeneous or homogeneous VMs. Thus, there are multiple available VMs that can execute the tasks of an application. Based on this architecture, given tasks of a specific type of applications, WARM first determines a VM to execute the tasks, and then specifies a routing path to that VM by installing the corresponding flow entries in each switch in the path.

The finally determined combination of a VM and routing path can maximize the revenue of a data center provider. This can tackle the shortcomings of routing strategies in currently existing SDN controllers that ignore VM information. For example, the Floodlight controller only selects routing paths based on the shortest path first strategy. However, this strategy may lead to large congestion in network links or large latency in VMs. Therefore, this leads to large RTT, and thus decreases the revenue of a data center provider according to SLAs specified between consumers and the provider.

IV. Problem formulation

According to the proposed architecture, this section formulates the workload-aware revenue maximization problem for a data center provider. For clarity, the main notations in this paper are summarized in Table I.

TABLE I
Symbols used in problem formulation

Notations	Definition
J	Number of application types
α_t	Routing path for type t tasks
ψ_t	VM for type t tasks
λ_t	Arriving rate of type t tasks
ζ_t	Priority of type t tasks
$\Omega_{ts\psi_t}$	Set of reachable paths for type t tasks from VM ψ_t to gateway switch s
$S_{ts\psi_t}$	Set of admissible paths for type t tasks from VM ψ_t to gateway switch s
C_l	Bandwidth capacity of link l
Υ_l	Set of application types admitted to traverse link l
$\mu_t^{\psi_t}$	Service rate of type t tasks in VM ψ_t
$n_t^{\alpha_t}$	Number of switches in path α_t for type t tasks
s_t	Average size of type t task
$\kappa_t^{\alpha_t i}$	Allocated bandwidth for type t tasks in switch i in path α_t
$\gamma_t^{\psi_t}$	Number of queueing type t tasks in VM ψ_t
$T_N^{\alpha_t}$	Time required in network links in routing path α_t for a type t task
$T_V^{\psi_t}$	Time required in VM ψ_t for a type t task

It is assumed that each VM only processes the tasks of one application. For a type t task, its RTT is calculated by summing up the time in network $T_N^{\alpha_t}$ and the time in a VM $T_V^{\psi_t}$. It is assumed that the time required in network links from and to a VM is the same. Then,

$$RTT_{\alpha_t,\psi_t} = T_N^{\alpha_t} + T_V^{\psi_t} \tag{1}$$

Let λ_t denote the arriving rate of type t tasks. The time required in network from (to) a VM can be calculated by summing up the time required in each switch in α_t, i.e., $\sum_{i=1}^{n_t^{\alpha_t}} \frac{1}{\kappa_t^{\alpha_t i} - \lambda_t}$. Therefore, $T_N^{\alpha_t}$ can be calculated as follow:

$$T_N^{\alpha_t} = 2\left(\sum_{i=1}^{n_t^{\alpha_t}} \frac{1}{\kappa_t^{\alpha_t i} - \lambda_t}\right) \tag{2}$$

$T_V^{\psi_t}$ can be obtained by calculating the time required to execute the remaining tasks and new arriving tasks in the VM.

Let $\gamma_t^{\psi_t}$ denote queueing tasks of type t at VM ψ_t. Let s_t denote the average size of each type t task. Let $\mu_t^{\psi_t}$ denote the service rate of type t tasks in VM ψ_t. Then,

$$T_V^{\psi_t} = \frac{\gamma_t^{\psi_t} + s_t}{\mu_t^{\psi_t}} \tag{3}$$

For type t tasks, there are multiple available VMs to execute the tasks. For each VM ψ_t, there are multiple available routing paths connecting gateway switch s and ψ_t. Let α_t denote the routing path for type t tasks. A typical SLA specified between data center providers and customers defines a utility function that transforms RTT of tasks of each application into the corresponding revenue brought to a data center provider. This paper adopts the SLA that is used in [5] and shown in Fig. 4. Let $u\left(RTT_{\alpha_t,\psi_t}\right)$ denote the corresponding revenue if RTT of type t tasks is RTT_{α_t,ψ_t}. the optimal combination of α_t and ψ_t can be determined by maximizing $\hbar$ that denotes the total revenue brought by the execution of tasks of all applications. The optimization problem can be formulated as follows.

$$RTT_{\alpha_t,\psi_t} = 2\left(\sum_{i=1}^{n_t^{\alpha_t}} \frac{1}{\kappa_t^{\alpha_t i} - \lambda_t}\right) + \frac{\gamma_t^{\psi_t} + s_t}{\mu_t^{\psi_t}} \tag{4}$$

$$\underset{\alpha_t,\psi_t}{\textbf{Max}} \left\{\hbar = \sum_{t=1}^{J} u\left(RTT_{\alpha_t,\psi_t}\right)\right\}$$

s.t.

$$\sum_{t\in\Upsilon_l} \kappa_t^{\alpha_t i} \leq C_l \tag{5}$$

$$\lambda_t \leq \kappa_t^{\alpha_t i} \tag{6}$$

Let $\Omega_{ts\psi_t}$ denote the set of reachable paths for type t tasks from VM ψ_t to gateway switch s. Let $S_{ts\psi_t}$ denote the set of admissible paths for type t tasks from VM ψ_t to gateway switch s. Let Υ_l denote the set of application types admitted to traverse link l. Let $n_t^{\alpha_t}$ denote the number of switches in path α_t for type t tasks. Constraint (5) ensures that for each link $l \in \alpha_t$, the total bandwidth allocated to the tasks of applications that are admitted to traverse this link l does not exceed its bandwidth capacity, C_l. $S_{ts\psi_t}$ denotes the set of admissible paths for type t tasks from VM ψ_t to gateway switch s. Constraint (6) guarantees the stability of the task queue of application t in switch i in path α_t.

V. Workload-aware revenue maximization method

Based on the proposed architecture of a workload-aware SDN controller, this section proposes WARM to intelligently schedule tasks of each application. The proposed WARM can achieve revenue maximization for a data center provider, by jointly considering workload in network links and VMs. WARM is executed periodically in the proposed architecture.

There are several deterministic algorithms (e.g., sequential quadratic programming) that can solve the formulated problem. However, most of them rely on special structures of constrained optimization problems. Besides, the quality of final solutions is relatively low and the search process tends to be slow. Current meta-heuristic algorithms are able to avoid drawbacks of deterministic algorithms, and therefore they are commonly used to solve optimization problems because of their easy implementation and robustness. However, they also have their disadvantages and advantages. For example, simulated annealing (SA) can find a high-quality solution by escaping from locally optimal solutions with moves worsening the value of an objective function. Yet, the convergence speed of SA is relatively unsatisfying [19]. Besides, another typical particle swarm optimization (PSO) is also widely used because of quick convergence. However, PSO often traps into locally optimal solutions within its searching process. Therefore, PSO's final solutions to difficult problems are often unacceptable [20].

Chaotic search can increase the possibility of finding globally optimal solutions in a quicker way than the stochastic search in a standard PSO. In the chaotic search, the ergodic and dynamic characteristics in chaotic sequences are able to enable moves that can escape from locally optimal solutions. In a typical chaotic PSO (CPSO), the chaotic search that is derived from the logistic equation is commonly used to realize local search around final solutions determined by PSO. Therefore, to solve the formulated problem, WARM is designed based on a hybrid meta-heuristic algorithm named Hybrid Chaotic Simulated-annealing PSO (HCSP). WARM aims to determine the optimal combination of VM ψ_t and path α_t ($1 \leq t \leq J$) to maximize the profit of a data center provider.

Algorithm 1 WARM

1: Randomly initialize position and velocity of each particle
2: Calculate fitness value of each particle
3: Change Θ and Γ
4: Initialize parameters related to SA, and chaotic PSO
5: $\ell \leftarrow t^0$, $w \leftarrow w_U$, $i \leftarrow 0$
6: **while** $S \leq 95\%$ and $i \leq g^{max}$ **do**
7: Change velocity and position of each particle according to acceptance criterion of Metropolis
8: Update fitness value of each particle in new swarm
9: Change Θ and Γ
10: $t \leftarrow t * d$
11: $w \leftarrow w_U - \frac{i}{g^{max}}\left(w_U - w_L\right)$
12: Update χ^{i+1} based on (7)
13: Update q_j^{i+1} and V_j^{i+1}, S
14: $i \leftarrow i + 1$
15: **end while**
16: Output Θ

Algorithm 1 shows the pseudo code of WARM. Notations used in Algorithm 1 are first introduced here. Θ and Γ denote globally optimal particle's position and locally optimal particles' positions. The upper velocity bound of all particles is denoted by v_U. Particles' velocities are restricted to $[-v_U, v_U]$. Maximum percentage of particles with the same fitness values in each swarm is denoted by S. Let g^{max} denote the maximum

number of iterations. Let w denote inertia weight whose lower and upper bounds are denoted by w_L and w_U, respectively. Line 1 randomly initializes the position and velocity of each particle. Lines 2–3 calculate the fitness value of each particle, and change Θ and Γ. Line 4 initializes parameters related to SA, and chaotic PSO. Let ℓ and d denote the temperature and its cooling rate in SA, respectively. Line 5 initializes ℓ and w with t^0 and w_U, respectively.

The **while** loop stops if $S \leq 95\%$ and $i \leq g^{max}$. Line 1 randomly initializes the position and velocity of each particle according to Metropolis acceptance rule. Lines 8–9 update the fitness value of each particle in a new swarm, and change Θ and Γ. Line 10 decreases temperature t. Line 11 linearly decreases inertia weight w. Lines 7–13 update χ^{i+1} based on the logistic equation (7) typically adopted in chaotic PSO, and q_j^{i+1}, V_j^{i+1}, and S. In (7), χ^i denotes a chaotic parameter whose range is [0, 1] in iteration i. Besides, ϵ is a control parameter whose range is [0, 4]. To guarantee the stochastic characteristic of a chaotic system, $\chi^0 \in [0,1] - \{0, 0.25, 0.5, 0.75, 1\}$ and $\epsilon = 4$. At last, Θ is chosen as the final output that is converted into ψ_t, and α_t ($1 \leq t \leq J$).

$$\chi^i = \epsilon \chi^{i-1}(1 - \chi^{i-1}) \tag{7}$$

Algorithm 2 Transform $\Omega_{ts\psi_t}$ to $S_{ts\psi_t}$

Input: $\Omega_{ts\psi_t}$
Output: $S_{ts\psi_t}$
1: **for all** $t \in J$ **do**
2: **for all** $p_t \in \Omega_{ts\psi_t}$ **do**
3: **for all** $l \in p_t$ **do**
4: $\Upsilon_l = Knapsack(l)$
5: **if** $t \notin \Upsilon_l$ **then**
6: $\Omega_{ts\psi_t} = \Omega_{ts\psi_t} - \{p_t\}$
7: $S_{ts\psi_t} = \Omega_{ts\psi_t}$
8: **end if**
9: **end for**
10: **end for**
11: **end for**

Algorithm 2 shows the approach of transforming $\Omega_{ts\psi_t}$ to $S_{ts\psi_t}$. The $Knapsack$ function in Line 4 determines the assignment of tasks of different applications to each link l. The task assignment is modeled as a knapsack problem (KP). In each link l, the tasks of multiple applications with different bandwidth requirements share limited bandwidth of this link. Tasks of different applications have different priorities. Let ζ_t denote the priority of type t tasks. The bandwidth requirement of high-priority tasks needs to be met first. Let P_l denote the sum of priorities of applications that are admitted to go through link l. *Knapsack* function aims to maximize P_l by determining the set of application tasks that are admitted to go through link l. Thus, KP is a typical 0-1 KP that belongs to combinatorial optimization and solved by *Knapsack* function. Specifically, link l's bandwidth capacity denotes the knapsack's volume. Each application denotes an item to be put in the knapsack. Each application's priority and bandwidth requirement denote the benefit and the volume of an item, respectively. The 0-1 KP in link l can be formulated as follows:

$$max\ P_l = \sum_{t=1}^{J} x_t \zeta_t$$

s.t.

$$\sum_{t=1}^{J} x_t \kappa_t^{\alpha_t i} \leq C_l \tag{8}$$

$$x_t = \begin{cases} 1, & t \in \Upsilon_l \\ 0, & t \notin \Upsilon_l. \end{cases} \tag{9}$$

Constraints (8) and (9) show that the allocated bandwidth of application tasks admitted to go through link l is less than or equal to C_l. The 0-1 KP belongs to NP-hard problems, and several existing methods (e.g., branch and bound, and dynamic programming) can not solve it well. Nevertheless, as an efficient meta-heuristic algorithm, genetic algorithm (GA) is widely used to solve complex 0-1 KP by obtaining close-to-optimal solutions from many potential candidates. Typically, GA starts with a population of potential chromosomes that correspond to solutions. Afterwards, to obtain a higher-quality population, a new population is generated based on the chromosomes in the old one according to their fitnesses. The chance of reproducing for solutions with higher quality is larger. GA repeats this process until termination conditions are met. GA's basic operations include selection based on fitnesses of chromosomes, crossover to obtain new population, and mutation of chromosomes in new population.

In GA's implementation, the roulette-wheel selection and the elitism are combined to increase the possibility of obtaining global optima. Here, Roulette-wheel is a widely used method where the selection of chromosomes is proportionate to their fitnesses. The elitism preserves some of the best chromosomes in new population. Besides, binary encoding is adopted in GA's implementation, and a string of 0's and 1's denotes a chromosome. The crossover of a single point is performed with a specific probability. In addition, mutation is done with a low probability in each bit of each chromosome to avoid falling into local optima.

It is worth noting that in the *Knapsack* function, the initiation of chromosomes in the first population has the complexity of O(J). The fitness calculation, crossover, and mutation operations have complexities of O(J). Thus, the complexity of GA in this work is O(J). In addition, 0-1 KP in each link can be independently solved in parallel. The work shows that the adoption of parallelism can greatly increase performance of an SDN controller, and its maximum processing ability can reach 2×10^7tasks/sec. In other words, the proposed WARM method will not cause too much performance overhead.

VI. Performance Evaluation

This section evaluates the proposed WARM with a widely used network emulator called Mininet [9]. This simulation

adopts the realistic task sampled in real-world Google data centers [1]. Fig. 2 illustrates arriving rates of tasks corresponding to three types of applications for 24 hours in May 2011. The length of each epoch is 5 minutes in the simulation. It is assumed that task arriving rates are already known in advance because there are existing studies that can realize high-precision workload prediction. Similarly, information about network and VMs is updated every 5 minutes.

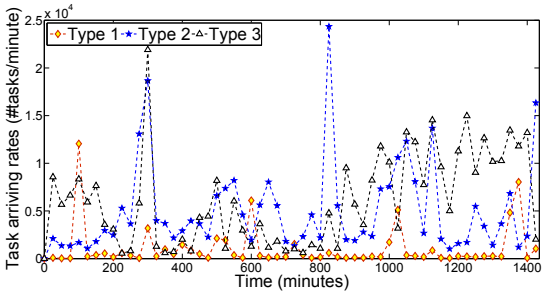

Fig. 2. Task arriving rates of three applications in Google's data center.

A. Experimental setting

This work adopts a typical network topology in data centers called Fat-tree [21]. It is worth noting that the proposed WARM method can be applied to any other network topology. This section evaluates WARM with the widely adopted pod-4 Fat-tree illustrated in Fig. 3. Fat-tree topology typically includes three tiers that are core tier, aggregation tier, and edge tier, respectively. The three tiers connect a gateway switch with VMs in a data center. In Fig. 3, multiple VMs are available to execute tasks of each application. It is assumed that each VM only executes tasks corresponding to a specific application. The bandwidth between each VM and each switch in the edge tier is 1Gbps. Besides, the bandwidth among switches in adjacent tiers is 1Gbps. In addition, to avoid the overload of the data center, the initial number of VMs is determined based on the expected maximum task arriving rate of each application. Then, VMs continuously run in the data center and execute the incoming tasks of each application. As shown in Fig. 3, this topology includes 16 VMs and 20 switches that support the OpenFlow protocol.

There are different types of applications in current data centers. Therefore, this simulation adopts a realistic mixture of large and small tasks. Based on the studies [5], [22], parameters are set as follows: s_1=$2KB$, s_2=$10KB$, s_3=$50KB$, μ_1=2.5×10^3 KB/minute, μ_2=1×10^4 KB/minute, and μ_3=8×10^4 KB/minute.

B. Simulation experiment

It is assumed that a data center provides three types of applications. Fig. 4 illustrates the proposed SLA model where three types of SLAs corresponding to three applications are defined. Three types of applications bring revenue according to a nonlinear pricing model that is widely used in several

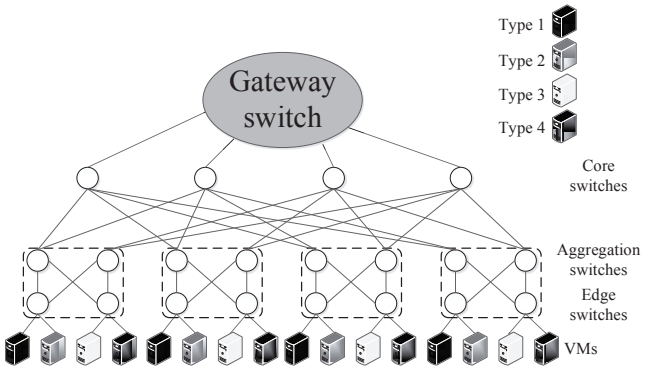

Fig. 3. Fat-tree topology.

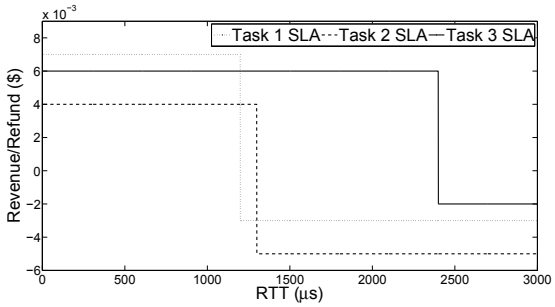

Fig. 4. Service-level agreements (SLAs).

existing studies [5]. This model specifies the money that customers need to pay based on the RTT of each application's tasks. If RTT is less than corresponding threshold, a revenue is delivered to the data center provider. Otherwise, a corresponding penalty is charged. The thresholds in type 1-3 SLAs are set to 1200 μs, 2400 μs, and 1300 μs.

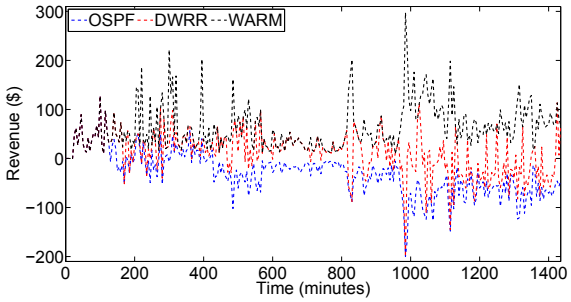

Fig. 5. Revenue comparison.

To demonstrate the effectiveness of the proposed WARM, this paper compares it with other two typical routing methods, including Open Shortest Path First (OSPF) [7] and Dynamic Weight Round Robin (DWRR) [8]. These two algorithms have been proven effective to realize task scheduling, and therefore they are widely used by existing studies in SDN-enabled data centers [7], [8]. Therefore this paper chooses them to show the advantages brought by WARM. Here the distance is simply the

number of network links from the gateway switch to a VM. In the simulation, it is assumed that OSPF schedules tasks of each application to the VM in the bottom left. DWRR schedules tasks of each application to VMs in a round robin fashion.

The revenue comparison among OSPF, DWRR and WARM is shown in Fig. 5. It is shown that WARM achieves larger revenue than OSPF and DWRR do in each epoch. Compared to OSPF and DWRR, the revenue with WARM is increased significantly. RTT of tasks of each application with OSPF is much larger than that with DWRR and WARM. Thus, the revenue corresponding to OSPF is the least according to SLAs of applications. Similarly, the revenue corresponding to DWRR is less than that corresponding to WARM. Therefore, WARM performs better than OSPF and DWRR in terms of the revenue. This result demonstrates that the proposed WARM can intelligently schedule tasks, such that the revenue brought by the execution of these tasks of all applications is maximized. The reason why WARM can increase the revenue of a data center provider is that WARM jointly considers workload in network links and VMs, and SLAs of all applications.

Figs. 6-8 show the RTT of tasks of types 1-3 with OSPF, DWRR, and WARM, respectively. It is shown that WARM outperforms OSPF and DWRR in terms of RTT. Specifically, RTT of tasks of three applications with OSPF is much larger than that corresponding to OSPF and DWRR. The reason is that OSPF always schedules tasks of three applications to the VM in the bottom left, and this causes the largest RTT in each epoch. Besides, RTT of tasks of three applications with DWRR is also larger than that corresponding to WARM. The reason is that WARM jointly considers workload in network and VMs, and can decrease RTT of tasks of each application.

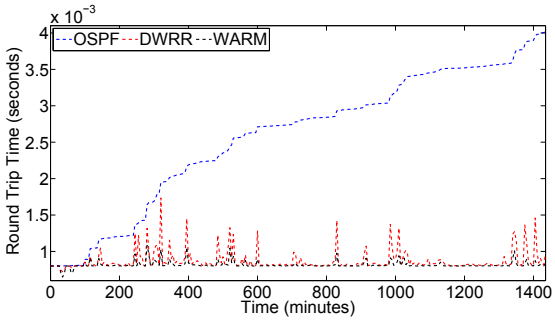

Fig. 6. RTT of type 1 tasks.

TABLE II
COMPARISON OF FOUR ALGORITHMS

Algorithms	Average revenue($)	Average execution time(seconds)
WARM	50.52	2.03×10^{-5}
CPLEX	50.55	3.04×10^{-5}
SA	47.54	1.43×10^{-4}
PSO	42.62	1.01×10^{-5}

This paper compares WARM with CPLEX that is a typical mathematical optimization solver and can determine the optimal solution in theory. Therefore, the comparison between WARM and CPLEX can show the precision of WARM's final solution. Here, we adopt IBM ILOG CPLEX 12.0. Besides, this paper compares WARM with two typical meta-heuristic algorithms including SA and PSO. Table II shows the comparison of four algorithms including WARM, CPLEX, SA, and PSO in the 30th epoch (150–155 minutes). In Table II, each of four algorithms is executed for 10 times.

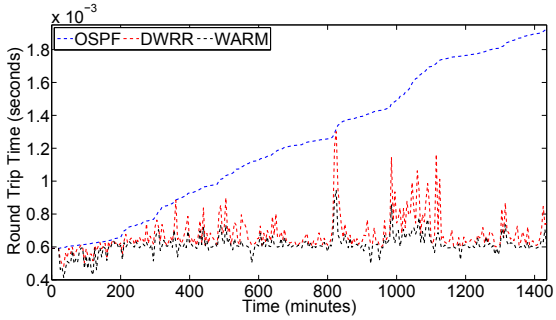

Fig. 7. RTT of type 2 tasks.

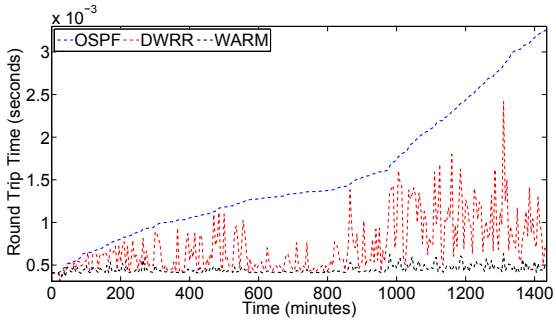

Fig. 8. RTT of type 3 tasks.

It is shown that WARM's revenue, 50.52$, is much close to that of CPLEX, 50.55$. However, the average execution time of CPLEX, 3.04×10^{-5}, is 1.5 times larger than that of WARM, 2.03×10^{-5}. The revenue of WARM's final solution is 50.52$, which is 1.06 times larger than that of SA, 47.54$, and 1.18 times larger than that of PSO, 42.62$, respectively. However, the average execution time of SA is 1.43×10^{-4} seconds, which is about 7 times larger than that of WARM, 2.03×10^{-5} seconds. Though the average execution time of PSO is only 1.01×10^{-5}, its revenue is the least among WARM, SA, and PSO due to its quick trap into local optima. This result demonstrates that WARM can find a solution with higher quality compared with SA and PSO. In addition, it is shown that WARM's average execution time is 2.03×10^{-5} seconds that is less than 5% of RTT of tasks of each application, and therefore the execution time overhead of WARM is negligible.

Fig. 9 illustrates the variance of average VM utilization corresponding to each application with OSPF, DWRR and WARM. The utilization of multiple VMs corresponding to the same application varies with time. Thus, this simulation calculates the variance of average VM utilization for each

application's tasks with OSPF, DWRR and WARM. It can be seen that compared with OSPF and DWRR, the variance with WARM is the least for each application. The reason is that WARM optimally specifies network path and the VM by jointly considering workload in network and VMs, and SLAs of all applications. DWRR schedules tasks in a round robin fashion, and therefore the revenue with DWRR is larger than that with OSPF in each epoch. OSPF always schedules tasks of each application to the VM in the bottom left, and therefore its revenue is the least.

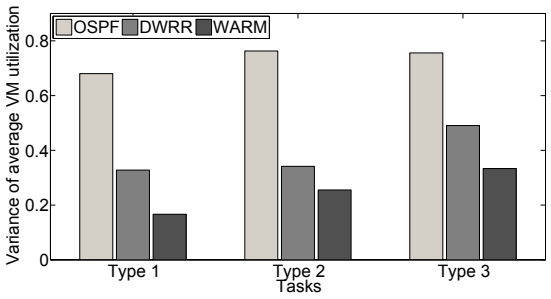

Fig. 9. Variance of average VM utilization.

VII. Conclusion and future work

Revenue maximization is important to a data center provider because it tries to deliver services to users' tasks in the most economic and fastest way. The emergence of software-defined networking (SDN) enables the provider to intelligently schedule the tasks of all applications. Existing controllers in an SDN architecture can only decrease the network latency but fail to consider the virtual machine (VM) latency, therefore leading to large round trip time (RTT). In this paper, a workload-aware revenue maximization (WARM) is proposed to smartly schedule all the tasks by determining the optimal combination of a VM and routing path for each application. Simulation results show that compared with its two peers, i.e., Open Shortest Path First and Round Robin, WARM can effectively increase the revenue of the data center provider and reduce the RTT of tasks drastically.

Acknowledgment

We thank all the reviewers for their insightful comments. This work was supported in part by a grant from the China Postdoctoral Science Foundation (No. 2016M600912).

References

[1] J. Zhu, Z. Zheng, and M. R. Lyu, "DR2: Dynamic Request Routing for Tolerating Latency Variability in Online Cloud Applications," in *Proc. IEEE 6th International Conf. Cloud Comput.*, Santa Clara Marriott, CA, USA, 2013, pp. 589–596.

[2] T. Mizrahi, E. Saat, and Y. Moses, "Timed Consistent Network Updates in Software-Defined Networks," *IEEE/ACM Trans. on Networking*, vol. 24, no. 6, pp. 3412–3425, Dec. 2016.

[3] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving High Utilization with Software-driven WAN," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 15–26, Aug. 2013.

[4] A. Lara, A. Kolasani, and B. Ramamurthy, "Network Innovation using OpenFlow: A Survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 493–512, Aug. 2013.

[5] M. Ghamkhari and H. Mohsenian-Rad, "Energy and Performance Management of Green Data Centers: A Profit Maximization Approach," *IEEE Trans. on Smart Grid*, vol. 4, no. 2, pp. 1017–1025, Jun. 2013.

[6] H. Yuan, J. Bi, W. Tan, and B. H. Li, "Temporal Task Scheduling With Constrained Service Delay for Profit Maximization in Hybrid Clouds," *IEEE Trans. on Automation Science and Engineering*, vol. 14, no. 1, pp. 337–348, Jan. 2017.

[7] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: verifying network-wide invariants in real time," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 467–472, Oct. 2012.

[8] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks," in *Proc. 2014 IEEE Network Operations and Management Symposium*, Krakow, Poland, 2014, pp. 1–8.

[9] A. Zabrovskiy, E. Kuzmin, E. Petrov, and M. Fomichev, "Emulation of dynamic adaptive streaming over HTTP with Mininet," in *Proc. 18th Conf. of Open Innovations Association and Seminar on Information Security and Protection of Information Technology*, St. Petersburg, Russia, 2016, pp. 391–396.

[10] J. Rückert, J. Blendin, R. Hark, and D. Hausheer, "Flexible, Efficient, and Scalable Software-Defined Over-the-Top Multicast for ISP Environments With DynSdm," *IEEE Trans. on Network and Service Management*, vol. 13, no. 4, pp. 754–767, Dec. 2016.

[11] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Network*, vol. 30, no. 3, pp. 52–58, May 2016.

[12] H. Huang, S. Guo, J. Wu, and J. Li, "Green DataPath for TCAM-Based Software-Defined Networks," *IEEE Communications Magazine*, vol. 54, no. 11, pp. 194–201, Nov. 2016.

[13] J. Bi, H. Yuan, Y. Fan, W. Tan, and J. Zhang, "Dynamic Fine-Grained Resource Provisioning for Heterogeneous Applications in Virtualized Cloud Data Center," *IEEE the 8th International Conf. on Cloud Computing*, New York, USA, June 27-July 2, 2015, pp. 429–436.

[14] J. Cao, K. Hwang, K. Li, and A. Zomaya, "Optimal Multiserver Configuration for Profit Maximization in Cloud Computing," *IEEE Trans. on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.

[15] S. Kundu, R. Rangaswami, M. Zhao, A. Gulati, and K. Dutta, "Revenue Driven Resource Allocation for Virtualized Data Centers," in *Proc. IEEE International Conf. on Autonomic Computing*, Grenoble, France, 2015, pp. 197–206.

[16] J. Cao, K. Li, and I. Stojmenovic, "Optimal Power Allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers," *IEEE Trans. on Computers*, vol. 63, no. 1, pp. 45–58, Jan. 2014.

[17] X. Gao, L. Kong, W. Li, W. Liang, Y. Chen, and G. Chen, "Traffic Load Balancing Schemes for Devolved Controllers in Mega Data Centers," *IEEE Trans. on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 572–585, Feb. 2017.

[18] P. Wang, G. Trimponias, H. Xu, H. Liu, and Y. Geng, "Luopan: Sampling based load balancing in data center networks," in *Proc. IEEE 24th International Conf. on Network Protocols (ICNP)*, Singapore, 2016, pp. 1–2.

[19] F. Rodriguez, C. Garcia-Martinez, and M. Lozano, "Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test," *IEEE Trans. on Evolutionary Computation*, vol. 16, no. 6, pp. 787–800, Dec. 2012.

[20] L. Liu, S. Yang, and D. Wang, "Particle Swarm Optimization With Composite Particles in Dynamic Environments," *IEEE Trans. on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 6, pp. 1634–1648, Dec. 2010.

[21] Z. Guo, J. Duan, and Y. Yang, "On-Line Multicast Scheduling with Bounded Congestion in Fat-Tree Data Center Networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 102–115, Jan. 2014.

[22] H. Yuan, J. Bi, W. Tan, and B. Li, "CAWSAC: Cost-aware workload scheduling and admission control for distributed cloud data centers," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 976–985, 2016.