Priority-Aware VM Allocation and Network Bandwidth Provisioning in Software-Defined Networking (SDN)-Enabled Clouds

Jungmin Son, *Student Member, IEEE* and Rajkumar Buyya, *Fellow, IEEE*

Abstract—In the multi-tenant architecture of cloud computing, different applications have different requirements and priorities. In the case of network congestion within a data center, failure of transferring data on time may cause a significant performance degradation of an application. It may result in a severe failure for a critical application that needs a certain level of QoS satisfaction, therefore, efficient resource provisioning techniques are vital to ensure transferring the high-priority data over the other traffics even in network congestion. In Software-Defined Networking (SDN)-enabled clouds, this is possible by reconfiguring network flows dynamically adapting to network traffics. In this paper, we propose priority-aware resource placement algorithms considering both host and network resources. Our priority-aware VM allocation (PAVA) algorithm places VMs of the high-priority application to closely connected hosts to reduce the chance of network congestion caused by other tenants. The required bandwidth of a critical application is also guaranteed by bandwidth allocation with a configuration of priority queues on each networking device in a data center network managed by SDN controller. Our experiment results show that the combination of proposed approaches can allocate sufficient resources for high-priority applications to meet the application's QoS requirement in a multi-tenant cloud data center.

Index Terms—Cloud computing, software-defined networking, resource management, energy efficiency

1 Introduction

Majority of the recent Internet applications utilize cloud computing infrastructure to provide elastic and cost-effective services [1]. Cloud providers provision computing, memory, storage, and networking resources for tenants to serve their applications with different requirements. These applications require different amounts of resources with a different priority. Compute-intensive applications such as large scientific application require more computing and memory power than networking resources, while network-intensive applications need more networking bandwidth than computing power. Cloud providers should allocate and provision resources efficiently in the data center to satisfy various requirements.

Cloud providers also need to guarantee the Service Level Agreement (SLA) for customers to ensure a different level of reliability and Quality-of-Service (QoS) requirements. For example, QoS-critical (higher-priority) applications such as medical software for cyber surgery, IoT applications for real-time disaster management, or deadline constrained scientific applications, need more strict policies in cloud resource management, while normal (lower-priority) applications may have a loose requirement. With the mixed applications with different QoS requirements sharing the same data center, providers should provision the resources efficiently to satisfy various QoS requirements of different applications.

However, it is difficult to guarantee such QoS in clouds because resources in a data center are shared by multiple tenants and applications, which are often over-booked to save the operational cost of a cloud provider [2]. Network resources, in particular, are oversubscribed in many data center designs to reduce the total cost of the design [3]. A simple method for providers to guarantee the QoS is to assign dedicated hosts and networks solely for a certain tenant, but it is deprived of all the benefits of clouds such as elasticity, low-cost, and dynamic provisioning of the resources. Although the dedicated resources can maximize the application performance as the allocated resources are only utilized for a specific application, it is a costly solution losing many advantages of cloud computing paradigm.

Unless assigning dedicated physical networking solely for certain applications, clouds tenants shall share the network resource. In cloud data centers, there are trade-offs between network proportionality, minimum guarantee, and high utilization [4]. This limitation lets cloud providers incapable of guaranteeing a minimum bandwidth for a certain tenant while targeting network proportionality and high utilization at the same time. For example, when a provider guarantees a minimum bandwidth for certain traffic by reserving bandwidth, the overall network utilization would be decreasing due to the unused reserved bandwidth. Thus, most public cloud providers including Amazon, Microsoft

• *The authors are with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Melbourne, Victoria 3010, Australia.*
E-mail: jungmins@student.unimelb.edu.au, rbuyya@unimelb.edu.au.

Manuscript received 17 Oct. 2017; revised 26 Apr. 2018; accepted 27 May 2018. Date of publication 30 May 2018; date of current version 6 Mar. 2019.
(Corresponding author: Jungmin Son.)
Recommended for acceptance by M. Qiu and S.-Y. Kung.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSUSC.2018.2842074

and Google do not guarantee the network bandwidth and only provides best-effort performance in their networking service. However, a certain type of applications in need of timely responsiveness (e.g., medical and disaster management applications) require more reliable networking performance in clouds.

The emergence of Software-Defined Networking (SDN) enables a fulfillment of network QoS satisfaction by the introduction of dynamic network reconfiguration based on the network traffic. SDN has brought many opportunities in networking with centralized manageability and programmable control logic. In SDN, a controller oversees the entire network by gathering all information of every network device and manages the network traffics dynamically with the customized control logic. SDN integration in a cloud data center has shown to be effective to improve the energy efficiency [5], [6], [7], the network performance [8], [9], [10], the network availability [11], and the security [12]. It also enables network slicing and dynamic bandwidth allocation which can be exploited for QoS satisfaction [13], [14].

In this work, we propose a novel VM and network allocation approach (PAVA+BWA) in the combination of a priority-aware VM allocation algorithm (PAVA) considering network connection between VMs on the application level with a network bandwidth allocation algorithm (BWA) to differentiate the higher-priority flows over normal network traffics. These algorithms are to allocate enough resources for QoS-critical applications in cloud environments where the computing and networking resources are shared with other tenants. We distinguish such applications to give the higher priority over the other tenants for resource provisioning and network transmission.

We model an application based on its priority to differentiate in VM capacity and network bandwidth. Our approach can allocate sufficient computing and networking resources for a critical application with high priority even in a busy data center. We employ a network bandwidth allocation strategy enabled by SDN for the critical application traffic so that the application can complete network transmission on time regardless of the network condition. With our approach, QoS-critical applications can be served in-time on clouds, while other applications can still share the resources for the rest of time. It considers host and networking resources jointly to prevent the network delay which can cause QoS failure. The applications' QoS requirements are assumed to be provided to the cloud management at the time of the request. Using the metrics of QoS requirements including computing capacity and network bandwidth, the proposed algorithm determines where to place VMs and flows. The algorithm considers both computing and networking requirements jointly to select the host to place the VM and the links between hosts. After selecting the network links, we use dynamic bandwidth allocation to meet the networking requirement.

The key *contributions* of the paper are:

- a priority-aware VM placement algorithm that places VMs of a critical application into proximity hosts with enough resources;
- a bandwidth allocation method for higher-priority flows to guarantee the minimum bandwidth in overloaded data center networks;
- a system that provisions both compute and network resources jointly to offer quality of service to critical applications in SDN-enabled cloud data centers;
- a performance evaluation of our proposed algorithm that is compared with related approaches and depicted its effectiveness through detailed simulation experiments using both synthetic and real (Wikipedia) workloads.

The rest of the paper is organized as follows. We discuss the state-of-the-art approaches in Section 2. The overall system architecture is explained in Section 3 followed by a detailed explanation of the proposed algorithm along with baselines in Section 4. Section 5 presents experiment configuration and evaluation results. Finally, Section 6 concludes the paper with future directions to the potential extension of the proposed approach.

2 Related Work

Many approaches have been proposed to network-aware VM placement and SDN-based network flow scheduling. Wang et al. proposed MAPLE [15], [16], a network-aware VM placement system that exploits an estimated bandwidth usage of a VM. In MAPLE, authors calculated estimated bandwidth based on empirical traces collected from the servers. The algorithm uses First Fit Decreasing approach in order to determine a host to place the VM. MAPLE focuses on per-VM network requirement and assumes that all VMs have a homogeneous processing requirement.

MAPLE project is extended to MAPLE-Scheduler, a flow scheduling system that dynamically reschedule the network flows based on QoS requirement of each VM [9]. The authors implemented dynamic network flow scheduler to relocate flows based on the estimated bandwidth usage. At the beginning, the system finds a default route using equal-cost multipath (ECMP) protocol [20] which is a widely adopted protocol for multi-path routing that distributes network traffics evenly over multiple paths. Then, the system detects potential flows that can violate the QoS and reschedules them to an alternate path where the QoS can be satisfied.

EQVMP is another VM placement approach that is aware of energy efficiency and QoS [17]. EQVMP partitions VMs into groups based on traffic matrix to reduce the inter-group traffic and balance the load in each group. VM groups are placed onto hosts using bin packing algorithm to reduce the number of hosts and minimize the energy consumption. After the placement, EQVMP performs load balancing which detects over-utilized network link and relocates flows in the link. The authors assume that the capacity requirement of VMs are homogeneous and considers only network requirement.

S-CORE has been proposed for SDN-based VM management that exploits a VM migration technique to minimize the network-wide communication cost for cloud data centers [18]. The system monitors VM traffic flows periodically and migrates VMs with large network traffics into close hosts to reduce the network cost. The prototype is implemented both on a KVM-based test-bed and in a simulation and evaluated with synthetic workloads. Although the system considers network traffics of VMs to minimize the total network cost through the network monitoring capability of SDN, it is incapable of dynamic traffic management in the congested network.

TABLE 1
Summary of Related Works

Work	VM allocation unit	VM placement method	VM type	Traffic management	Parameters	Energy efficiency
MAPLE [9], [15], [16]	Connected VMs	Network-aware	Homogeneous	Dynamic routing	Estimated effective bandwidth	x
EQVMP [17]	Connected VMs	Hop reduction	Homogeneous	Dynamic routing	VM traffic	✓
S-CORE [18]	Not Applicable (NA)	Migration only	Homogeneous	VM migration to close hosts	VM traffic	x
QVR [10]	NA	NA	NA	Dynamic routing & Bandwidth allocation	Delay, jitter, packet loss	x
FCTcon [7]	NA	NA	NA	Bandwidth allocation	Flow completion time	✓
DISCO [19]	NA	NA	NA	Flow consolidation	Traffic correlation, delay	✓
PAVA+BWA	Application level	Priority-aware	Heterogeneous	Bandwidth allocation	Priority, VM capacity, bandwidth requirement	✓

Lin et al. proposed QoS-aware virtualization and routing method (QVR) that isolates and prioritizes tenants based on QoS requirements and allocates network flows dynamically [10]. The system supports fine-grained network virtualization based on tenants and the network requirement. The flows are allocated dynamically onto physical links considering the minimum arrival rate and maximum utilization. The system also uses adaptive feedback traffic engineering to ensure the end-to-end QoS. The proposed approach is applied to a wide area network and compared with other routing protocols in a simulation. Their approach is applicable to different network topology but more focused on wide area network.

FCTcon has been proposed to improve energy efficiency in data center networks by dynamically managing the flow completion time [7]. The proposed method consolidates network flows into a smaller set to save energy usage in a data center. In order to prevent performance degradation, the system calculates an expected flow completion time and dynamically adapts the flow bandwidth to meet the required completion time for delay-sensitive flows. In their system, the controller receives feedback of flow completion time from a monitor constantly monitoring the data center network and updates bandwidth allocation and flow paths to consolidate flows while preventing delay of sensitive flows.

DISCO is also proposed by the same authors to increase the scalability of traffic flow consolidation technique for energy efficient data center networks [19]. In this work, the authors addressed the trade-offs between scalability, network performance, and energy efficiency in a large-scale data center network. Network traffics within a data center can be consolidated into a smaller number of network links and switches to save power consumption by turning off unused switches. However, the high complexity of the consolidation algorithm in SDN's centralized model can result in a scalability issue for a data center. The authors proposed a distributed traffic management system to support a scalable traffic consolidation technique. The proposed system also considers network delay as a constraint to improve the network performance. Nevertheless, only network traffic is taken into account in this work without consideration of computing resources.

Table 1 summarizes related works and compares with our approach (PAVA+BWA). Unlike the aforementioned studies, our approach prioritizes critical applications over the other applications considering application-level resource requirements. Instead of considering individual VMs and flows, a set of multiple VMs consisting of the same application (e.g., web and database servers for a web application, or mappers and reducers for a MapReduce application) are taken into the consideration for resource provisioning. In our model, VMs can be heterogeneous, i.e., each VM can have different computing and networking requirements which are managed by our VM allocation and network bandwidth allocation system. Parameters considered in the proposed method includes priority, VM processing capacity, and flow bandwidth requirements. We also consider the energy efficiency of a data center which may be affected by the implementation of the algorithm.

3 System Architecture

Fig. 1 shows overall architecture of our system along with flows of interaction between different components. In brief, application requests are submitted to the system with a priority information (critical or normal). Each application request consists of an arbitrary number of VMs and flows with detailed specifications. Priority of an application is then analyzed for VM and flow placement to provide sufficient resources for a critical application as well as to prioritize network traffic over normal applications. Based on the analyzed information, the host and link selection algorithm determines where to place VMs and flows. Flow information is also sent to the network manager to communicate with the SDN controller for dynamic bandwidth allocation and flow scheduling. The detail of each component is explained below.

At first, application requests are created by cloud tenants to serve their applications and include VM types and flow information. A VM type consists of the number of processing cores, each core's processing capacity, the amount of memory, and the storage size. In this research, we focus on computing power and ignore memory and storage size in order to reduce the problem complexity. An application request also includes flow specification consisting of a source and destination VM and the required bandwidth. VM and flow specifications are modeled based on commercialized cloud providers such as Amazon AWS and Microsoft Azure who provide predefined VM types and customizable virtual networks. We add an extra entity, application priority, to this model to differentiate applications. Note that either critical or normal application request can be submitted at arbitrary time, and the system

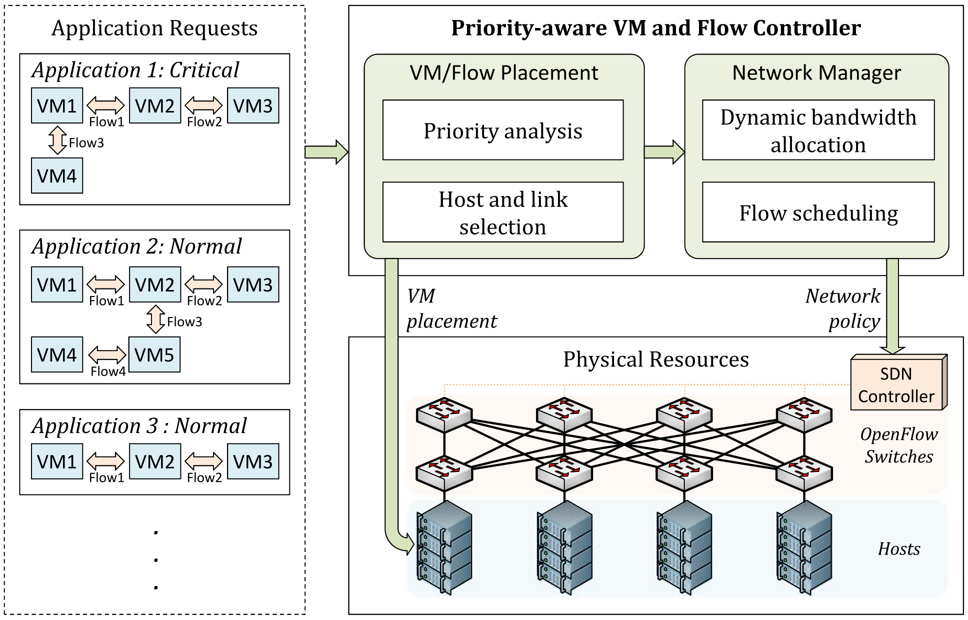


Fig. 1. Architectural design of priority-aware VM and flow management system.

provisions the resources available at the time of the application submission. If a critical application is submitted after a normal one, the system allocates residual resources to the critical one with a different provisioning method.

Given the VM and flow requests, the system analyzes QoS requirements and prioritizes the VMs and flows accordingly. The priority of an application can be determined with various methods by either the cloud provider or the tenants. For example, a global priority from all the tenants can be determined by local prioritization requirements submitted by individual tenants [21]. The amount of the requested resources, such as VM capacity and network requirements, can also be exploited, so that applications requesting more resources can be prioritized over the other applications. Application priorities can be differentiated by the class of tenants [22], e.g., premium tenants paying extra service charge for application prioritization or government tenants in need of running time-critical applications for emergency. Although our system model is capable of any prioritization method, in this paper we assume that the application priority is specified by user in binary value (critical or normal) and supplied to the system along with the application request. This is to simplify the prioritization process and focus on the resource prioritization method.

After the analysis of an application priority, the system selects a host and links to place the requested application by running the VM placement algorithm, which is proposed in this paper that jointly considers computing and networking requirements. Once the selection is done, the result is passed to the network manager for network resource configuration and to the physical resources for actual allocation of the VM and flow.

Network manager ensures to allocate minimum bandwidth for a critical application even in network congestion. It is in charge of SDN controller that dynamically allocates bandwidth for each flow. By default, the bandwidth of a link is equally shared among the flows passing the link, but a higher-priority flow can acquire more bandwidth through implementing a priority queue on switches such as queuing disciplines (*qdisc*) and Hierarchy Token Bucket (*HTB*) implemented in Linux's traffic control (*tc*) suite. In order to control flows and their network bandwidths, the network manager communicates with SDN controller to provision the networking resources dynamically. SDN controller manages all switches in the data center through OpenFlow protocol.

4 Priority-Aware Resource Provisioning Methods

In this section, we explain the strategy for computing and networking resource allocation in aware of application's priority. We model the priority of application discretely. Each application provides its priority as a *critical* (higher-priority) or *normal* (lower-priority) application. It can be manually set up by tenants who submit the application request to a cloud provider or automatically configured by cloud provider based on the application information. We assume that there is a reasonable mixture of critical and normal application requests in a data center so that not all the applications running in a data center are set to be critical applications. If all of them have the same priority, they will be served equally. Resources are allocated without any special consideration for normal applications, whereas critical applications are ensured to get sufficient computing power and prior network transmission over the normal applications.

4.1 Priority-Aware VM Allocation (PAVA)

For VM allocation, we propose Priority-Aware VM Allocation (PAVA) algorithm (shown in Algorithm 1) which allocates a closely connected host for a critical application with the information of network topology. In this method, we consider the priority of application as well as the network connectivity of physical hosts in a data center. Before allocating VMs, the system gathers the network topology of the data center. All hosts in a data center are grouped by their connectivity. For example, hosts in the same rack connected to the same edge switch will be grouped together. VMs are also grouped based on the application.

After grouping hosts, PAVA algorithm is running to find a suitable VM-host mapping to place a VM for a critical application. PAVA tries to place a set of VMs for the critical application onto the host group which has more computing and networking resources. If the group of VMs cannot be fit in a single host group, VMs will be placed across multiple host groups with the closest proximity (e.g., under a different edge switch in the same pod). In this way, VMs in the same application will be closely placed to maintain the minimal number of hops for network transmission. For example, a network traffic between VMs placed within the same host is transferred through the host memory without incurring any traffic to networking devices. Similarly, if the VMs are hosted under the same edge network or in the same pod, a cost for network transmission between those VMs can be reduced with a lower probability of interference by the other applications. PAVA guarantees VMs for the critical application to be placed on not only the host with the sufficient capacity but also within the same or closer hosts in network topology to reduce the communication cost. We use First Fit Decreasing (Algorithm 3) for normal applications.

Algorithm 1. Priority-Aware VM Allocation (PAVA)

```
1: **Input**: $vm$: VM to be placed.
2: **Input**: $rd$: Resource demand of $vm$;
3: **Input**: $app$: Application information of $vm$.
4: **Input**: $H$: List of all hosts in data center.
5: **Output**: VM placement map.
6: $H_{group} \leftarrow$ Group $H$ based on edge connection;
7: $Q_H \leftarrow$ Empty non-duplicated queue for candidate hosts;
8: $placed \leftarrow$ false;
9: **if** $app$ is a higher-priority application **then**
10:    $H_{app} \leftarrow$ list of hosts allocated for other VMs in $app$;
11:    **if** $H_{app}$ is not empty **then**
12:       $Q_H$.enqueue($H_{app}$);
13:       **for each** $h_a$ in $H_{app}$ **do**
14:          $H_{edge} \leftarrow$ A host group in $H_{group}$ where $h_a$ is included ;
15:          $Q_H$.enqueue($H_{edge}$);
16:       **end for**
17:       **for each** $h_a$ in $H_{app}$ **do**
18:          $H_{pod} \leftarrow$ Hosts in the same pod with $h_a$;
19:          $Q_H$.enqueue($H_{pod}$);
20:       **end for**
21:    **end if**
22:    sort $H_{group}$ with available capacity, high to low;
23:    $Q_H$.enqueue($H_{group}$);
24:    **while** $Q_H$ is not empty **and** $placed$ = false **do**
25:       $h_q = Q_H$.dequeue()
26:       $C_h \leftarrow$ free resource in host $h_q$;
27:       **if** $rd < C_{h_q}$ **then**
28:          Place $vm$ in $h_q$;
29:          $C_h \leftarrow C_h - rd$;
30:          $placed \leftarrow$ true;
31:       **end if**
32:    **end while**
33: **end if**
34: **if** $placed$ = false **then**
35:    Use FFD algorithm to place $vm$;
36: **end if**
```

4.2 Bandwidth Allocation for Priority Applications (BWA)

As cloud data center's network infrastructure is shared by various tenants, providing constant network bandwidth is crucial for application's QoS to avoid performance degradation in traffic congestion caused by other tenants. We propose bandwidth allocation (BWA) approach to allocate the required bandwidth for a critical application. This approach utilizes per-flow traffic management feature for virtualized network [23], [24]. SDN controller can manage switches to allocate requested bandwidth by configuring priority queues in switches to ensure privileged traffic transmitting over the other lower-priority traffics.

After VM placement process is completed, the bandwidth requirement and the virtual network information of a critical application are sent to the SDN controller. SDN controller then establishes priority queues (e.g., Linux qdisc and HTB) for the critical application flows on every switch along the link. Network traffic generated from VMs of the critical application will use the priority queue so that the required bandwidth can be obtained for the critical applications. This method is only applied to critical applications in order to prioritize network transmission over the normal traffic in the shared nature of data center networking. It ensures that the critical application can get enough bandwidth even in a congested network caused by the other application. Algorithm 2 explains the detailed procedure of BWA method. For all flows, the algorithm sets the default path using ECMP, which distributes network traffic based on the address of the source and destination hosts. For higher-priority flows, the algorithm sets up an extra flow rule in each switch along the path. The priority queue set for the higher-priority flow can guarantee the minimum bandwidth required by the application. For lower-priority flows, the algorithm only sets a default path using ECMP without a further configuration.

Algorithm 2. Bandwidth Allocation for Critical Applications (BWA)

```
1: **Input**: $F$: List of network flows.
2: **Input**: $topo$: Network topology of the data center.
3: **Output**: Priority queue configuration in switches.
4: **for each** flow $f$ in $F$ **do**
5:    $h_{src} \leftarrow$ the address of the source host of $f$;
6:    $h_{dst} \leftarrow$ the address of the destination host of $f$;
7:    $S_f \leftarrow$ list of switches between $h_{src}$ and $h_{dst}$ in $topo$;
8:    **for each** switch $s$ in $S_f$ **do**
9:       **if** $f$ is a higher-priority flow **then**
10:         $s$.setPriorityQueue($h_{src}$, $h_{dst}$, $f.vlanId$, $f.bandwidth$);
11:      **end if**
12:   **end for**
13: **end for**
```

4.3 Baseline Algorithms

The proposed approaches are compared with three baseline algorithms: exclusive resource allocation, random allocation, and state-of-the-art heuristic.

Exclusive resource allocation (*ERA*) is to allocate dedicated hosts and networks exclusively for a critical application, thus resources are not shared with any other tenants.

The application can fully utilize the capacity of the dedicated resources to process its workloads, as the required computing and networking resources can be obtained without any interference from other applications. However, all the benefits of cloud computing will be lost including elasticity and dynamicity in this method. It is impractical in reality because exclusively allocated resources will result in an extravagant cost for cloud providers which will be passed on to the customers. In this paper, we use this algorithm only for measuring the expected response time of a critical application to calculate QoS violation rate. Details of how to calculate QoS violation rate is explained in Section 5.3.

Algorithm 3. First-Fit Decreasing for Bandwidth Requirement (FFD)

```
1:  Input: $VM$: List of VMs to be placed.
2:  Input: $H$: List of hosts where VMs will be placed.
3:  Output: VM placement map.
4:  for each $vm$ in $VM$ do
5:      sort $H$ with available resource, low to high;
6:      for each $h$ in $H$ do
7:          $C_h \leftarrow$ free resource in host $h$;
8:          $rd \leftarrow$ resource demand of $vm$;
9:          if $rd < C_h$ then
10:             Place $vm$ in $h$;
11:             $C_h \leftarrow C_h - rd$;
12:             $placed \leftarrow$ true;
13:             break;
14:         end if
15:     end for
16: end for
```

Random allocation (*Random*) is to place a VM on a random host capable of providing enough resources for the VM. In this method, a host is randomly selected with no intelligence but solely based on the resource capacity.

The state-of-the-art heuristic baseline algorithm is to place VMs in First Fit Decreasing (FFD) order determined by the amount of required bandwidth, which is combined with a dynamic flow (DF) scheduling method for network traffic management. FFD and DF are derived from MAPLE project [9], [15], [16], where the applications are equally considered for VM allocation and flow scheduling based on their bandwidth requirement regardless of applications' priority. Details of baselines are explained below.

First Fit Decreasing (FFD). FFD searches a host to place the VM in the first fit decreasing order based on the bandwidth. It consolidates more VMs into a host with enough resources and does not distribute them across the data center. Thus, VMs are placed into a smaller set of hosts, whereas other empty hosts can put into an idle mode which can increase the energy efficiency of the entire data center. This baseline is derived from MAPLE [15], [16], but instead of Effective Bandwidth, we use the VM's requested bandwidth to determine the resource sufficiency of a host for a VM. In our system, we do not need to calculate a separate Effective Bandwidth because the required bandwidth is predefined in the application specification.

In addition to the power saving at hosts, FFD can also reduce the energy consumption of switches. When more VMs are placed on the same host, the possibility of in-memory transmission between VMs in the same host is increased which emits the network transmission over the switches. Although the algorithm does not consider the network condition in itself, it can affect the amount network traffic to some extent from the nature of the algorithm. The pseudo-code of the algorithm is presented in Algorithm 3.

Dynamic Flow Scheduling (DF). On multi-path network topology, dynamic flow scheduling is a common approach to find an alternate path for a flow in case of network congestion or link error suggested in multiple studies [9], [17]. This method detects a congested link and relocates the flows in the congested link into an alternate path with more capacity. However, based on our observation, this approach is less effective for short-distance flows in Fat-tree topology due to Fat-tree's architectural advance which can achieve a network over-subscription ratio of up to 1:1 [3]. For an edge switch in Fat-tree, the number of downlinks to the connected hosts are same as the number of up-links to the aggregation switches, and the traffic flows are equally distributed based on the address of the source and the destination host. Thus, the network traffic is already balanced among the links between edge and aggregation switches in the same pod. The algorithm is still effective for inter-pod traffics because the number of links between aggregation and core switches is less than the ones between aggregation and edge switches. When the link to a core switch is congested, DF algorithm can relocate the flows into an alternate link to the other core switch with less traffic.

The pseudo-code of dynamic flow scheduling algorithm (DF) is described in Algorithm 4 which is derived from MAPLE-Scheduler [9]. DF is used as a baseline to compare with our bandwidth allocation approach. The algorithm is applied periodically to find the least busy path for higher-priority traffics, while normal traffics still use the default path determined by ECMP based on the source address.

Algorithm 4. Dynamic flow Scheduling Algorithm (DF)

```
1:  Input: $f$: A network flow to be scheduled.
2:  Input: $h_{src}$: the address of the source host of $f$.
3:  Input: $h_{dst}$: the address of the destination host of $f$.
4:  Input: $topo$: Network topology of the data center.
5:  Output: Network path from $h_{src}$ to $h_{dst}$.
6:  $s \leftarrow$ next hop from $h_{src}$ for flow $f$ in $topo$
7:  while $s$ is not $h_{dst}$ do
8:      $L \leftarrow$ available links on $s$ for flow $f$.
9:      $l_{Next} \leftarrow h_{src}$ mod $L$.size() (default path);
10:     if $f$ is a priority flow then
11:         for each link $l$ in $L$ do
12:             if $l$.utilization() $<$ $l_{Next}$.utilization() then
13:                 $l_{Next} \leftarrow l$;
14:             end if
15:         end for
16:     end if
17:     $s$.updateNextHop($f$, $l_{Next}$);
18:     $s \leftarrow l_{Next}$
19: end while
```

5 Performance Evaluation

The proposed algorithms are evaluated in a simulation environment. Two use-case scenarios are prepared to show the

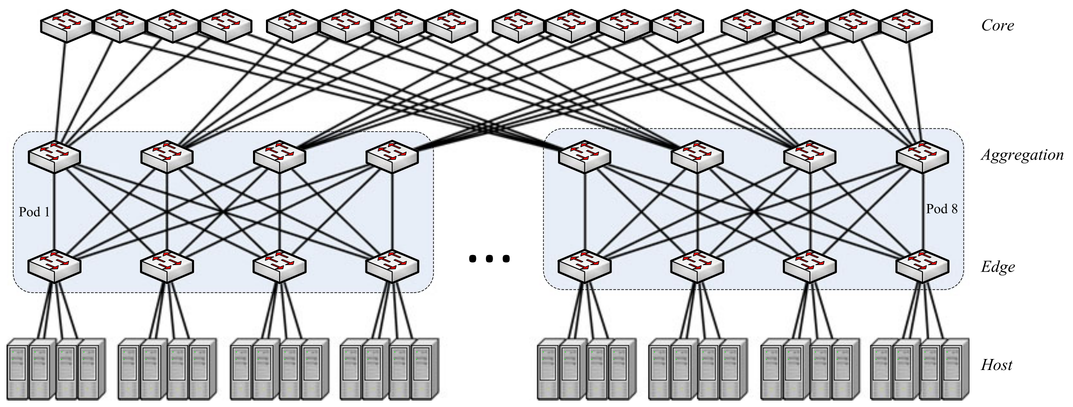


Fig. 2. 8-pod fat-tree topology setup for experiments.

effectiveness of PAVA and BWA: a straightforward network-intensive application and more practical 3-tier application. We measure the response time of workloads to check the impact of the algorithms on both critical and normal application's performance. Energy consumption of the data center and the number of active hosts and their up-time are also measured to consider the cost of cloud providers.

5.1 Experiment Configuration and Scenarios

For evaluation, we implemented the proposed method and baselines on CloudSimSDN [25] simulation environment. CloudSimSDN is an extension of CloudSim [26] simulation tool that supports SDN features for cloud data center networks. CloudSimSDN has been modified and updated to support the application prioritization. The priority information has been supplied along with the application request, which is read and exploited by the modified modules that store and retrieve the application priority. In the simulation, we generated an 8-pod fat-tree topology data center network with 128 hosts connected through 32 edge, 32 aggregation, and 16 core switches. Thus, each pod has 4 aggregation and 4 edge switches, and each edge switch is connected to 4 hosts. Fig. 2 shows the topology of the configured cloud data center for the experiment. All physical links between switches and hosts are set to 125 MBytes/sec.

In the aforementioned simulation environment, we evaluate our approach in two scenarios.

5.1.1 Scenario 1: Synthetic Workload

The first scenario is to place a critical application in an overloaded data center environment. To make the data center overloaded, 15 lower-priority applications consisting of 16 VMs in each application are first placed in the data center that constantly generates network traffics. After these VMs are placed, the higher-priority application consisting of the same number of VMs is submitted to the data center. Once all VMs are placed using our proposed VM placement algorithm (PAVA), synthetically generated workloads are submitted to the critical application, which has both computing and networking loads. The bandwidth allocation method (BWA) is applied to transfer the networking part of the critical application workloads. This scenario is to test the effectiveness of PAVA and, especially, BWA in a condition that the higher-priority application is significantly interfered by other applications.

5.1.2 Scenario 2: Wikipedia Workload

The second scenario reflects a more practical situation where applications are placed on a large-scale public cloud that a massive number of VM creation and deletion requests are submitted every minute. Frequent VM creation and deletion result in a fragmented data center. Network traffics generated by the scattered VMs can increase the overall load of the data center network, which makes the network traffic management more critical in applications' performance.

We create 10 different application requests modeled from three-tier web applications. Each application consists of 2 database, 24 application, and 8 web servers communicating to one another. One out of the 10 applications is set to be critical, while the rest to be normal. The size of VMs are varied based on the tier, e.g., database tier servers are defined having 2 times more processing capacity than application tier servers. Virtual networks are also defined between all VMs in the same application so that any VMs can transfer data to any other VMs in the same application. Required bandwidth for the critical application is set to the half of physical link bandwidth, while the normal application is set to be a fourth of the physical bandwidth to differentiate the priority.

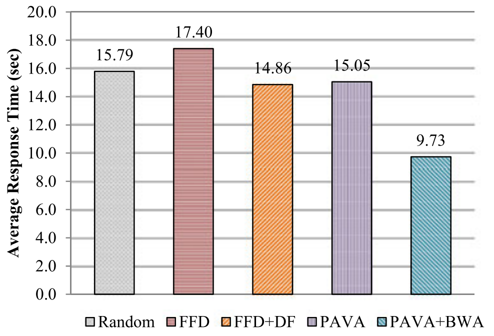
We also generate workloads for the second scenario from three-tier application model [27] based on Wikipedia traces available from *Page view statistics for Wikimedia projects*. Every application receives approximately between 80,000 and 140,000 web requests generated from traces in a different language, each of which consists of processing jobs in VM servers and network transmissions.

For both scenarios, we measure the response time of both critical and normal applications, the QoS violation rate of the critical application, and the power consumption of the data center.

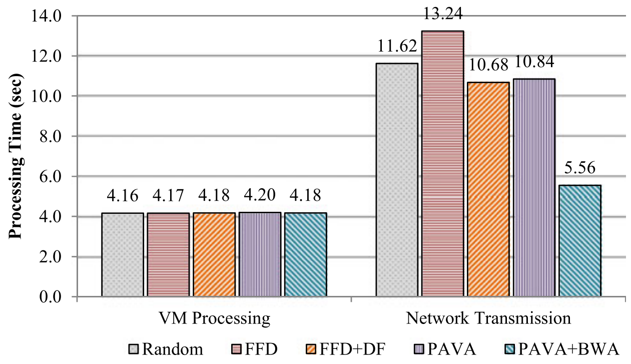
5.2 Analysis of Response Time

At first, we evaluate the performance of the proposed algorithm by measuring the average response time of the critical application, and VM processing and network transmission time in detail with each algorithm. Note that QoS violation is not considered for calculating the averages in this section.

Fig. 3 shows the results in Scenario 1 where the data center network is constantly overloaded by other applications. The average response time (Fig. 3a) is significantly reduced by 38.4 percent in PAVA+BWA (both PAVA and BWA algorithms applied) compared to the Random algorithm, mainly resulting from 52.1 percent reduction in network

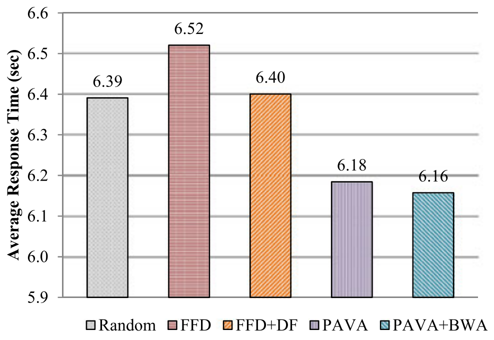


(a) Average response time of the critical application.

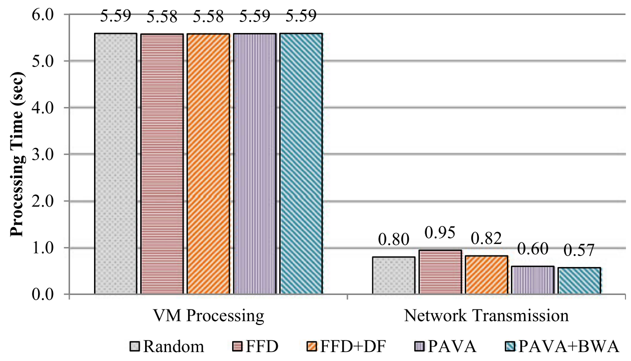


(b) VM processing and network transmission time of the critical application.

Fig. 3. Performance matrices of the critical application in Scenario 1 (synthetic workload).



(a) Average response time of the critical application.



(b) VM processing and network transmission time of the critical application.

Fig. 4. Performance matrices of the critical application in Scenario 2 (Wikipedia workload).

transmission time (Fig. 3b). VM processing time remains same regardless of algorithm combination which shows that VMs acquire enough processing resources.

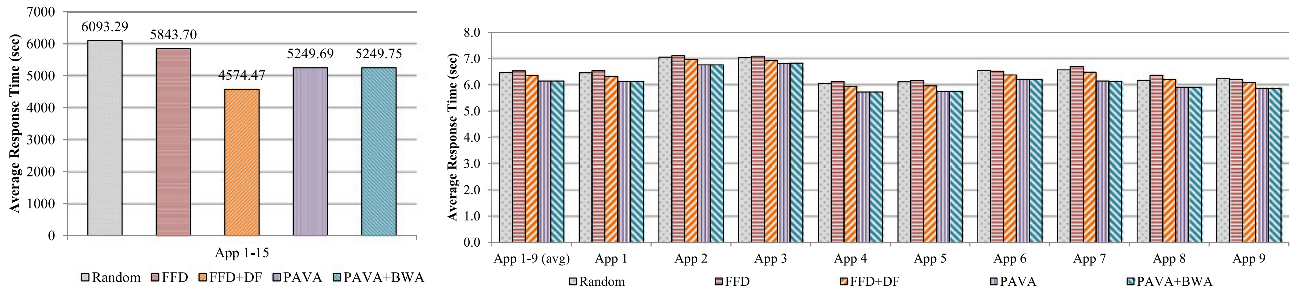
For FFD, the average response time is 10.2 percent increased compared to Random method due to the increased network transmission time. Since FFD consolidates more VMs into a smaller number of hosts without consideration of their connectivity, network transmissions within the critical application are significantly interfered by other applications placed on the same host. Similarly, applying PAVA without network bandwidth allocation cannot improve overall performance substantially due to the consolidation of VMs into shared hosts, although the average response time is still shorter than the one from FFD.

With the implementation of DF in addition to FFD, the average network transmission time is reduced to 10.68 seconds from 13.24 of FFD. Although dynamic flow scheduling can find a less crowded path, it is ineffective in this scenario where all the alternate paths are busy. On the other hand, BWA provides the best result in network transmission time reduced to almost half of all the other methods. This shows that our bandwidth allocation method can significantly improve the critical application's network performance in the overloaded network environment.

Fig. 4 depicts the results from Scenario 2 where more complex applications and workloads are submitted to a large-scale cloud data center. In this scenario, the average response time is reduced by 3.3 percent in PAVA, and BWA is not shown as effective as the previous scenario. Note that the network is not so frequently overloaded in Scenario 2, which limits the effectiveness of BWA method. On the other hand, PAVA becomes more effective on account of the proximity of VMs. As the VMs of the same application have been placed closely with PAVA, the network transmission time between them is reduced by 25 percent from 0.80 seconds in Random to 0.60 seconds in PAVA. The critical application's network workloads pass through only low-level switches (edge and/or aggregation switches) as the VMs are placed under the same edge network or the same pod, and thus not interfered by other traffics.

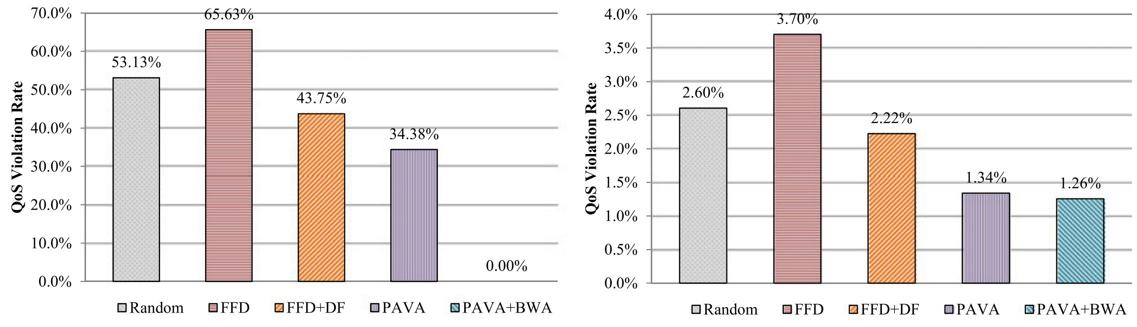
Similar to the previous scenario, FFD increases the average response time due to the VM consolidation to shared hosts. Implementation of DF also reduces the network transmission time, which makes the average response time of FFD+DF become similar to the Random method. VM processing times are almost same no matter which algorithm is being used. In short, the proposed algorithm (PAVA+BWA) improves the response time of the critical application by 34.5 percent for Scenario 1 and 3.8 percent for Scenario 2 compared to the state-of-the-art baseline (FFD+DF). The improvement in Scenario 2 is not as significant as Scenario 1 due to the origin of workloads, e.g., network-intensive workloads can be more beneficial from the proposed algorithm.

Additionally, we measure the average response time of normal (lower-priority) applications to see the effect of our algorithm on the other normal applications. Fig. 5 shows the measured response time of normal applications in both scenarios. Compared to the Random algorithm, PAVA and



(a) Scenario 1 (synthetic workload). (b) Scenario 2 (Wikipedia workload).

Fig. 5. Average response time of normal (lower-priority) applications.



(a) Scenario 1 (synthetic workload). (b) Scenario 2 (Wikipedia workload).

Fig. 6. QoS violation rate of critical application workloads.

PAVA+BWA actually result in improving the performance of lower-priority applications by reducing the average response time by 13.8 and 4.9 percent respectively in Scenario 1 and 2. The baseline algorithm FFD+DF also reduces the response time of lower-priority applications. In short, our algorithm maintains or even improves the performance of lower-priority applications, while improving the performance of a critical application.

5.3 Analysis of QoS Violation Rate

QoS violation rate is calculated by comparing the response time from ERA algorithm with the one from other algorithms. We assume that the expected response time can be fully achieved in ERA because it allocates dedicated servers with enough resource for every VMs required in the application. We compare the response time of each workload and count the QoS violated workload if the response time exceeds the one from ERA. Equation (1) shows the calculation of QoS violation rate (r_v) from workloads set (W), where t_X and t_{ERA} denote the response time of a workload (w_v) measured from the designated algorithm and ERA respectively. It counts the number of workloads whose response time from the designated algorithm is exceeding the response time from ERA and divides by the total number of workloads.

$$r_v = \frac{|\{w_v \in W | t_X(w_v) > t_{ERA}(w_v)\}|}{|W|}. \tag{1}$$

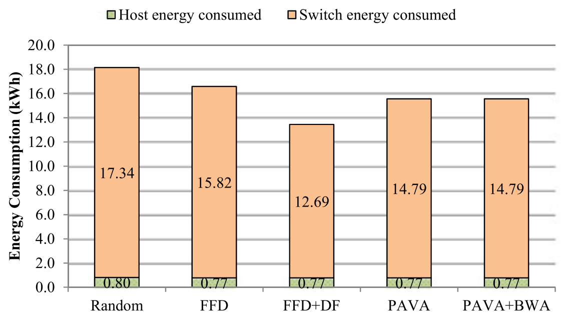
Average QoS violation rate of the critical application is shown in Fig. 6. In Scenario 1, PAVA results in 34.38 percent QoS violation whereas PAVA+BWA has no violation at all (see Fig. 6a). As we discussed in the previous section, BWA is more effective in overloaded networks where other tenants generate heavy traffic loads. The baseline (FFD+DF) also reduce the QoS violation rate from Random's 53.13 to 43.75 percent but not as significant as BWA.

Similar results can be found in Scenario 2 (see Fig. 6b) where PAVA and PAVA+BWA show the lowest QoS violation rate reaching 1.34 and 1.26 percent respectively. Interestingly, overall violation rate is much lower, ranging between 1.26 and 3.70 percent in Scenario 2 compared to between 0 and 65.36 percent of Scenario 1. This is due to the significant degradation of the network performance in Scenario 1 where network overload by other applications interferes the application. Although the QoS violation rate in Scenario 2 is not as high as in Scenario 1, the impact of our algorithm is still significant to improve the violation rate by 51.5 percent reducing from 2.60 to 1.26 percent. It is a crucial improvement for critical applications that should guarantee the QoS requirement. Although BWA is not as beneficial as Scenario 1, it can still reduce the violation rate by 0.08 percent compared to PAVA alone.

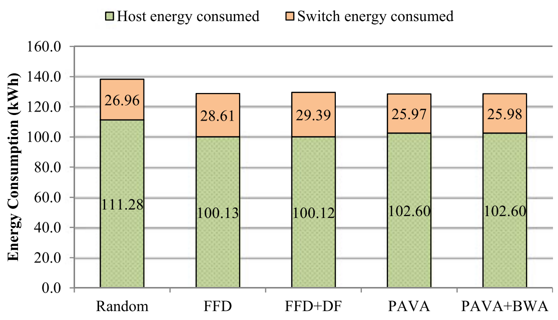
Compared to the state-of-the-art baseline, our proposed algorithm combination, PAVA+BWA, can reduce QoS violation rate from 43.75 to 0 percent for heavy network traffic scenario and from 2.22 to 1.26 percent (reduction by 43.2 percent) for large-scale complex application scenario.

5.4 Analysis of Energy Consumption

Energy consumption is evaluated to find the influence of the proposed algorithm to the operational cost of a cloud data center. We measured the utilization of hosts and switches over time and used power model of hosts [28] and switches [29] respectively to calculate the overall power consumption, using the same model and method exploited in our previous paper [2]. Unused hosts and switches are assumed to be in an idle mode to save energy, and the



(a) Scenario 1 (synthetic workload).



(b) Scenario 2 (Wikipedia workload).

Fig. 7. Detailed power consumption of hosts and switches in a data center.

power consumption of active hosts and switches is calculated based on the utilization of a host and the active ports of a switch.

Fig. 7 shows the measured energy consumption of the entire data center and the detailed power consumption in hosts and switches for both scenarios. In Scenario 1 (see Fig. 7a) both PAVA and PAVA+BWA save 14.2 percent of a total data center energy usage compared to the Random algorithm, whereas FFD and FFD+DF save 8.6 and 25.9 percent of power cost respectively. The difference mainly comes from switches, because workloads in Scenario 1 consist of marginally huge network traffics combined with a small computation load on VMs.

In Scenario 2, PAVA and PAVA+BWA consume the least amount of energy among all algorithms. For host energy consumption, all the four algorithm combinations (FFD, FFD+DF, PAVA, and PAVA+BWA) consume less energy compared to Random, since both PAVA and FFD consolidate VMs into the smaller number of hosts and turn off many unused hosts. For switches, however, FFD (28.61 kWh) and FFD+DF (29.39 kWh) consume more amount of energy compared to Random (26.96 kWh), while the consumption in PAVA (25.97 kWh) and PAVA+BWA (25.98 kWh) is lower than the Random. As those VMs in the same application group are closely placed with PAVA mostly within the same edge network or within the same pod, the network traffics are consequently consolidated passing through fewer switches. Thus, the energy consumption is lowered resulting from the decreased number of active switches.

Nevertheless, the results show that the proposed algorithm at least will not increase the power consumption of the data center. In fact, it can help to reduce the operational cost by consolidating VMs into a smaller number of hosts while providing the required QoS for a critical application. In Wikipedia workload, the energy consumption is even reduced compared to the state-of-the-art baseline.

5.5 Analysis of Algorithm Complexity

We analyze the time complexity of the proposed algorithm. First, the greedy bin-packing FFD algorithm (Algorithm 3) takes $O(|H| \log |H|)$ time to place one VM in a data center with $|H|$ number of available hosts. In order to place $|VM|$ number of VMs, the algorithm takes $O(|VM| \cdot |H| \log |H|)$ which is feasible for online dynamic VM placement.

PAVA is based on FFD algorithm with an extra computation for critical applications. For each VM in critical applications, PAVA needs an additional sorting time, $O(|H| \log |H|)$, to find a closely connected host from the previously placed VMs. Thus, given VMs in critical applications ($VM_c \in VM$), the overall complexity of PAVA including the additional computation for critical VMs along with the basic FFD is:

$$\begin{aligned} &O(|VM_c| \cdot |H| \log |H|) + O(|VM| \cdot |H| \log |H|) \\ &\quad = O(|VM| \cdot |H| \log |H|), \end{aligned}$$

which is same as the time complexity of FFD algorithm.

The time complexity of BWA algorithm is $O(|S| \cdot |F_c|)$ where $|F_c|$ number of flows for critical applications are placed in a data center network consisting of $|S|$ number of switches. This is also a small addition compared to dynamic scheduling algorithm where flows are periodically rerouted by running routing algorithms to find the best route. However, the overhead of BWA may occur at the time of packet forwarding in switches because of extra forwarding rules and queues configured for critical applications. Although we could not simulate this overhead in our evaluation environment, the extra overhead is negligible when the proportion of higher-priority flows is significantly smaller than the data center. The number of switches installing the extra queues will be minimized especially with PAVA where the priority VMs are placed in close hosts, thus network traffics are transmitted through the minimal number of network hops.

6 Conclusions and Future Work

In this paper, we presented a priority-based VM allocation and network traffic management scheme with bandwidth allocation and dynamic flow pathing mechanism. The algorithms are evaluated in a simulation environment with a large-scale fat-tree topology and multiple applications with a different priority. The results show that the proposed priority-aware VM allocation method actually places the critical application into the closer hosts so that it reduced both the energy consumption and the average response time for the critical application. The bandwidth allocation method is specifically effective in the overloaded network scenario where the higher-priority traffic is interfered by other applications. Our algorithm is outperformed the state-of-the-art approaches and able to deal with applications with varied priorities and QoS requirements. Such feature will be useful in critical applications in need of strict timeliness such as a disaster management or other emergency alert applications.

To further extend the algorithm, the priority of application can be modeled more fine-grained values instead of the binary priority. In the current algorithm, priority is set to a binary value, i.e., an application is either critical or not. Instead, we can define the priority in multi-degree values which can allocate differentiated amounts of resources to applications with a different level of priority. The algorithm needs to be modified to support the multi-degree priorities and function to convert from the priority value to the amount of allocated resource. It is also possible to combine the proposed algorithm with a resource overbooking method [2]. For example, a critical application is assigned with lower overbooking ratio so that more resources can be utilized, while lower-priority applications are more overbooked to save the operational cost of cloud providers. The algorithm can further broaden to include multiple QoS parameters such as throughput, latency, and response time to be considered as priority factor to enable the system to determine a proper amount of resources autonomously. As the use of SDN in edge clouds is getting popular, our approach can be utilized by them for supporting mobile applications [30].

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility,” *Future Generation Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[2] J. Son, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, “SLA-aware and energy-efficient dynamic overbooking in SDN-based cloud data centers,” *IEEE Trans. Sustainable Comput.*, vol. 2, no. 2, pp. 76–89, Apr.-Jun. 2017.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2008, pp. 63–74.

[4] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, “Faircloud: Sharing the network in cloud computing,” in *Proc. ACM SIGCOMM Conf. Appl. Technol. Architectures Protocols Comput. Commun.*, 2012, pp. 187–198.

[5] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, “Elastictree: Saving energy in data center networks,” in *Proc. 7th USENIX Conf. Netw. Syst. Des. Implementation*, 2010, pp. 17–17.

[6] X. Wang, X. Wang, K. Zheng, Y. Yao, and Q. Cao, “Correlation-aware traffic consolidation for power optimization of data center networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 992–1006, Apr. 2016.

[7] K. Zheng and X. Wang, “Dynamic control of flow completion time for power efficiency of data center networks,” in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, Jun. 2017, pp. 340–350.

[8] I. Petri, M. Zou, A. R. Zamani, J. Diaz-Montes, O. Rana, and M. Parashar, “Integrating software defined networks within a cloud federation,” in *Proc. 15th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, May 2015, pp. 179–188.

[9] R. Wang, S. Mangiante, A. Davy, L. Shi, and B. Jennings, “QoS-aware multipathing in datacenters using effective bandwidth estimation and SDN,” in *Proc. 12th Int. Conf. Netw. Service Manag.*, Oct 2016, pp. 342–347.

[10] S.-C. Lin, P. Wang, and M. Luo, “Jointly optimized QoS-aware virtualization and routing in software defined networks,” *Comput. Netw.*, vol. 96, pp. 69–78, 2016.

[11] H. Amarasinghe, A. Jarray, and A. Karmouch, “Fault-tolerant iaas management for networked cloud infrastructure with SDN,” in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–7.

[12] A. Chowdhary, S. Pisharody, A. Alshamrani, and D. Huang, “Dynamic game based security framework in SDN-enabled cloud networking environments,” in *Proc. ACM Int. Workshop Security Softw. Defined Netw. Netw. Function Virtualization*, 2017, pp. 53–58.

[13] A. V. Akella and K. Xiong, “Quality of Service (QoS)-guaranteed network resource allocation via software defined networking (SDN),” in *Proc. IEEE 12th Int. Conf. Dependable Autonomic Secure Comput.*, 2014, pp. 7–13.

[14] K. Govindarajan, K. C. Meng, H. Ong, W. M. Tat, S. Sivanand, and L. S. Leong, “Realizing the quality of service (QoS) in software-defined networking (SDN) based cloud infrastructure,” in *Proc. 2nd Int. Conf. Inf. Commun. Technol.*, May 2014, pp. 505–510.

[15] R. Wang, R. Esteves, L. Shi, J. A. Wickboldt, B. Jennings, and L. Z. Granville, “Network-aware placement of virtual machine ensembles using effective bandwidth estimation,” in *Proc. 10th Int. Conf. Netw. Service Manag. Workshop*, Nov. 2014, pp. 100–108.

[16] R. Wang, J. A. Wickboldt, R. P. Esteves, L. Shi, B. Jennings, and L. Z. Granville, “Using empirical estimates of effective bandwidth in network-aware placement of virtual machines in datacenters,” *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 267–280, Jun. 2016.

[17] S.-H. Wang, P. P. W. Huang, C. H. P. Wen, and L. C. Wang, “EQVMP: Energy-efficient and qos-aware virtual machine placement for software defined datacenter networks,” in *Proc. Int. Conf. Inf. Netw.*, Feb. 2014, pp. 220–225.

[18] R. Cziva, S. Jout, D. Stapleton, F. P. Tso, and D. P. Pezaros, “SDN-based virtual machine management for cloud data centers,” *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 212–225, Jun. 2016.

[19] K. Zheng, X. Wang, and J. Liu, “DISCO: Distributed traffic flow consolidation for power efficient data center network,” in *Proc. 16th Int. IFIP TC6 Netw. Conf. Netw.*, 2017, pp. 1–9.

[20] C. Hopps, “Analysis of an equal-cost multi-path algorithm,” *Internet Requests for Comments*, USA: RFC Editor, Nov. 2000. [Online]. Available: https://tools.ietf.org/html/rfc2992

[21] W.-T. Tsai, Q. Shao, and J. Elston, “Prioritizing service requests on cloud with multi-tenancy,” in *Proc. IEEE 7th Int. Conf. E-Bus. Eng.*, Nov. 2010, pp. 117–124.

[22] W. Ellens, M. ivkovic, J. Akkerboom, R. Litjens, and H. van den Berg, “Performance of cloud computing centers with multiple priority classes,” in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 245–252.

[23] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, “Sharing the data center network,” in *Proc. 8th USENIX Conf. Netw. Syst. Des. Implementation*, 2011, pp. 309–322.

[24] V. T. Lam, S. Radhakrishnan, R. Pan, A. Vahdat, and G. Varghese, “Netshare and stochastic netshare: Predictable bandwidth allocation for data centers,” *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 5–11, Jun. 2012.

[25] J. Son, A. V. Dastjerdi, R. N. Calheiros, X. Ji, Y. Yoon, and R. Buyya, “CloudSimSDN: Modeling and simulation of software-defined cloud data centers,” in *Proc. 15th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, May 2015, pp. 475–484.

[26] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, “Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Softw.: Practice Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[27] D. Ersoz, M. S. Yousif, and C. R. Das, “Characterizing network traffic in a cluster-based, multi-tier data center,” in *Proc. IEEE 27th Int. Conf. Distrib. Comput. Syst.*, 2007, pp. 59–59.

[28] S. Pelley, D. Meisner, T. F. Wenisch, and J. W. VanGilder, “Understanding and abstracting total data center power,” in *Proc. Workshop Energy-Efficient Des.*, 2009.

[29] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao, “Carpo: Correlation-aware power optimization in data center networks,” in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1125–1133.

[30] K. Gai, M. Qiu, and H. Zhao, “Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing,” *J. Parallel Distrib. Comput.*, vol. 111, pp. 126–135, 2018.

Jungmin Son received the BEng degree in information and computer engineering from Ajou University, Korea, and the M.Info.Tech. degree from the University of Melbourne, in 2006 and 2013, respectively. He is working toward the PhD degree with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia. He is a recipient of the Australian Postgraduate Award (APA) funded by the Australian Government and awarded by the University of Melbourne. His research interests include SDN-enabled cloud computing, resource provisioning, scheduling, and energy efficient data centers. He is a student member of the IEEE.

Rajkumar Buyya is a Redmond Barry Distinguished professor and director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the university, commercializing its innovations in cloud computing. He served as a Future fellow of the Australian Research Council during 2012-2016. He has authored more than 625 publications and seven text books including *Mastering Cloud Computing* published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese, and international markets, respectively. He also edited several books including *Cloud Computing: Principles and Paradigms* (Wiley Press, USA, Feb 2011). He is one of the highly cited authors in computer science and software engineering worldwide (h-index=116, g-index=255, and 70,100+ citations). Microsoft Academic Search Index ranked him as #1 author in the world (2005-2016) for both field rating and citations evaluations in the area of distributed and parallel computing. "A Scientometric Analysis of Cloud Computing Literature" by German scientists ranked him as the World's Top-Cited (#1) Author and the World's Most-Productive (#1) Author in Cloud Computing. Recently, he was recognized as a "2016 Web of Science Highly Cited Researcher" by Thomson Reuters and a fellow of IEEE for his outstanding contributions to cloud computing. Software technologies for grid and cloud computing developed under his leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world. He has led the establishment and development of key community activities, including serving as foundation chair of the IEEE Technical Committee on Scalable Computing and five IEEE/ACM conferences. His contributions and international research leadership are recognized through the award of "2009 IEEE Medal for Excellence in Scalable Computing" from the IEEE Computer Society TCSC. Manjrasoft's Aneka Cloud technology developed under his leadership has received "2010 Frost & Sullivan New Product Innovation Award". He served as the founding editor-in-chief of the *IEEE Transactions on Cloud Computing*. He is currently serving as co-editor-in-chief of the *Journal of Software: Practice and Experience*, which was established more than 45 years ago. For further information, please visit his cyberhome: www.buyya.com.