SLA-Aware and Energy-Efficient Dynamic Overbooking in SDN-Based Cloud Data Centers

Jungmin Son, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, and Rajkumar Buyya, *Fellow, IEEE*

Abstract—Power management of cloud data centers has received great attention from industry and academia as they are expensive to operate due to their high energy consumption. While hosts are dominant to consume electric power, networks account for 10 to 20 percent of the total energy costs in a data center. Resource overbooking is one way to reduce the usage of active hosts and networks by placing more requests to the same amount of resources. Network resource overbooking can be facilitated by Software Defined Networking (SDN) that can consolidate traffics and control Quality of Service (QoS) dynamically. However, the existing approaches employ fixed overbooking ratio to decide the amount of resources to be allocated, which in reality may cause excessive Service Level Agreements (SLA) violation with workloads being unpredictable. In this paper, we propose dynamic overbooking strategy which jointly leverages virtualization capabilities and SDN for VM and traffic consolidation. With the dynamically changing workload, the proposed strategy allocates more precise amount of resources to VMs and traffics. This strategy can increase overbooking in a host and network while still providing enough resources to minimize SLA violations. Our approach calculates resource allocation ratio based on the historical monitoring data from the online analysis of the host and network utilization without any pre-knowledge of workloads. We implemented it in simulation environment in large scale to demonstrate the effectiveness in the context of Wikipedia workloads. Our approach saves energy consumption in the data center while reducing SLA violations.

Index Terms—Cloud computing, software defined networking, energy efficient, resource overbooking

1 Introduction

Offering subscription-oriented cloud computing services has attracted great deal of attention in both industry and academia. One of the major concerns in cloud computing is tremendous electrical power consumption in cloud data centers. According to the US Natural Resources Defense Council [1], data centers in the U.S. consumed about 91 billion kilowatt-hours of electricity in 2013, which is roughly twice of the electricity consumption in New York City. Moreover, the electricity consumption of data centers is subjected to rise to about 140 billion kilowatt-hours annually until 2020. With the ever-growing increase in energy consumption of data center infrastructure, energy management has been a center of attention in cloud research such as in cooling systems [2], [3], [4] and in servers [5], [6], [7], [8], [9], [10].

Over-provisioning of resources (hosts, links and switches) is one of the major causes of power inefficiency in data centers. As they are provisioned for peak demand, the resources are under-utilized for the most time. For example, the average utilization of servers reported to be between 10-30 percent for large data centers [11], [12], which results in a situation where considerable capacity of data center is idle. Therefore, VM placement, consolidation, and migration techniques have been effectively applied to improve the server power efficiency [13] for the servers which are not energy proportional.

Similarly, provisioning of network capacity for peak demand leads to energy waste, which can be reduced through the effective use of Software-Defined Networking (SDN). With SDN, now cloud data centers are capable of managing their network stack through software and consider network as one of the key elements in their consolidation techniques. SDN enables the isolation of network's control and forward planes. This way, routing and other control-related issues are set via a software controller, enabling the forward plane to quickly react and adapt to changes in demand and application requirements [14]. The software controller lies between applications and the infrastructure, and performs tasks that, before SDNs, were performed at individual hardware level (switches, routers). With the emergence of SDN, each individual traffic flow between VMs can be controlled and thus network traffics can be consolidated to less number of links by an overbooking strategy.

While overbooking strategies can save energy, they also increase the chance of SLA violation when either host or network is overloaded. If the consolidated VMs or traffics reach the peak utilization at the same time, insufficient amount of resources would be allocated which will delay the workload processing. The main objective of our approach is to ensure both SLA satisfaction and energy saving without compromising one for the other. We aim to reduce SLA violation rate while increasing energy savings.

In this paper, we *propose* dynamic overbooking algorithm for joint host and network resource optimization that, in

- *J. Son and R. Buyya are with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, Vic 3010, Australia. E-mail: jungmins@student.unimelb.edu.au, rbuyya@unimelb.edu.au.*
- *A.V. Dastjerdi is with PwC, Melbourne, Australia. E-mail: amir.vahid@pwc.com.*
- *R.N. Calheiros is with the School of Computing, Engineering and Mathematics, Western Sydney University, NSW 2751, Australia. E-mail: R.Calheiros@westernsydney.edu.au.*

Manuscript received 12 Sept. 2016; revised 28 Apr. 2017; accepted 3 May 2017. Date of publication 8 May 2017; date of current version 13 July 2017. (Corresponding author: Jungmin Son.)
Recommended for acceptance by C. Dobre, G. Mastorakis, C.X. Mavromoustakis, and F. Xhafa
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.2017.2702164

comparison to previous works, has three novelties. First, our approach employs a dynamic overbooking strategy that dynamically adapts to the workload instead of using a fixed percentile. Second, it is designed to work without the prior knowledge of the workload. Lastly, we consider initial placement and consolidation strategies together to find the most effective combination for energy saving and SLA satisfaction.

The rest of the paper is organized as follows. We explain the detailed background of SDN and its usage in the context of cloud computing in Section 2, and the state-of-the-art approaches for energy savings in cloud data centers in Section 3. Section 4 formulates the power model and the energy optimization problem. Section 5 depicts the overall framework and its three components: Resource Utilization Monitor, Initial Placement Policy, and Migration Policy. In Section 6, we explain the strategies for SLA-aware and energy efficient dynamic overbooking. Section 7 presents experiment environment and evaluation results. Finally, Section 8 concludes the paper with future directions for energy optimization in heterogeneous clouds.

2 Background

The current computer networks have reached a point where they are cumbersome to manage and not scaling to requirements of cloud data centres. Utilizing Software Defined Networking in cloud data centres is a new way of addressing the shortcomings of current network technologies. In traditional network, distributed routers are core controllers for network management. While the routers can cooperate with each other by communicating network information, the decision is made by a single router with its discrete control logic without consideration of the entire network.

In contrast, SDN has a centralized controller capable of seeing the global view of the entire network. Therefore, traffic consolidation can be performed by the centralized control logic in consideration of energy consumption and SLA satisfaction comprehensively for the entire data center network. Information collected from the entire network is considered for traffic consolidation, and the overall impact on the whole data center is estimated in the control logic. This was not feasible in traditional network as the control logic in the distributed router has limited information and considers only local impact of the control decision.

The centralized control logic in SDN also allows to have both VM and network traffic at the same time for data center optimization. Instead of consolidating VM and network separately, both can be jointly taken into account. Before SDN, network was not considered in VM consolidation process since network cannot be centrally controlled with the global view of the entire data center.

SDN also brings dynamic configuration of the network by separating the control plane from the forward plane. In SDN, the software controller manages overall network through the control plane in each network device, while the forward plane is in charge of forwarding data packets according to forwarding rules set up by the control plane. As the control plane can be dynamically configured by the central controller, network can be quickly adjusted to the current network condition. For example, dynamic bandwidth allocation for a specific flow is enabled with SDN which can help improve Quality of Service (QoS) of the network intensive applications.

In short, SDN offers more opportunities for traffic consolidation and energy-saving in data center networks. As proposed in our previous work [15], SDN-enabled cloud data center can make QoS enforcement more convenient in data center networks with responding to the rapidly changing network traffic. Joint optimization of hosts and networks is feasible in SDN-enabled data center.

3 Related Work

There are several works that have explored energy-efficient cloud resource management with conventional networking [16]. In tihs paper, we are only focusing at those works in the context of the use of SDN-based virtualized clouds.

ElasticTree [17] is an OpenFlow based network power manager which dynamically change the data center data traffic and adjust network elements for power saving. ElasticTree consolidates network flows to a minimum number of links, and the unused switches are turned off to save more energy consumption. Authors also considered robustness of the network that can handle traffic surges. Although ElasticTree addressed network power savings, VM placement optimization was not considered.

Abts et al. [18] argued that DCN can be energy proportional to the amount of data traffic as like CPU of a computer that consumes less power when it is in low utilization. They proposed link rate adaptation that changes dynamic range depending on the predicted traffic load. They showed that energy proportional networking is feasible by dynamically changing individual link rate. However, they did not address the approach that consolidates traffic and turning off links.

CARPO [19] is a similar approach to ElasticTree and saves data center network power consumption. For traffic consolidation, CARPO adapted correlation analysis between traffic flows so that if the traffic flows are less correlated, those flows can be consolidated into the same network link and more energy savings can be achieved. Additionally CARPO considered link rate adaptation that alters the link speed of each port depending on the traffic amount. When the traffic is decreasing, link speed slows down to save more energy.

Recently, researchers started to consider both DCN and host optimization simultaneously. Jiang et al. [20] investigated VM placement and network routing problem jointly to minimize traffic cost in data center. VM placement and routing problem are formulated and solved using on-line algorithm in dynamically changing traffic loads. The proposed algorithm leveraged Markov approximation to find near optimal solution in feasible time.

Jin et al. [21] also considered both host and network factors jointly to optimize energy consumption. They formulated the joint host-network problem as an integer linear program, and then converted the VM placement problem to a routing problem to effectively combine host and network optimization. Finally the best host for placing VM is determined by depth-first search. Prototype is implemented on OpenFlow based system with fat-tree topology and evaluated with massive test cases via both simulation and real implementation.

VMPlanner [22] is presented by Fang et al. that optimizes VM placement and network routing. They addressed the problem with three algorithms: traffic-aware VM grouping, distance-aware VM-group to server-rack mapping, and power-aware inter-VM traffic flow routing [22]. VMPlanner groups VMs with higher mutual traffic and assigns each VM group to the same rack. Then, traffic flow is aggregated to minimize the inter-rack traffic so that the unused switches can be powered off.

PowerNetS [11] is presented by Zheng et al. and finds the optimal VM placement considering both host and network resources using correlation between VMs. Also, detailed power model is introduced which includes power consumptions of chassis, switch, each port as well as the idle and maximum power consumption of a server. PowerNetS measures correlation coefficients between traffic flows and applies them for VM placement and traffic consolidation.

Unlike these techniques, our proposed work uses dynamic overbooking ratio which dynamically changes based on the workload in real-time. This ensures that, with the changes in workload, data center status, and user requirement, our approach can both save energy and maintain SLA satisfaction.

4 Problem Formulation

The energy efficient host-network resource allocation problem can be formulated as a multi-commodity problem [17]. The objective of the problem is to minimize the power consumption of hosts, switches and links in a data center.

4.1 Power Models

The following notations are used for the problem formulation.

- s_i : The ith switch in the data center;
- l_i : The ith link in the data center;
- h_i : The ith host in the data center;
- $vm_{j,i}$: The jth virtual machine on host i;
- $C(h_i)$: The capacity of host i;
- $C(l_i)$: The capacity of link i;
- $rd(vm_{j,i})$: The resource demand of the $vm_{j,i}$;
- $f_{j,i}$: The flow j on link i;
- $d(f_{j,i})$: The data rate of flow j on link i;
- $|VM|$: The total number of VMs in the data center;
- $|H|$: The total number of hosts in the data center;
- $|L|$: The total number of links in the data center;
- σ_i : The number of VMs placed on host i;
- n_i : The number of flows assigned to link i;
- $CC(X, Y)$ The Correlation Coefficient between two variables X, Y;
- $P(h_i)$: Power consumption of host i;
- $P(s_i)$: Power consumption of switch i;
- P_{idle} : Idle power consumption of host;
- P_{peak} : Peak power consumption of host;
- u_i : CPU utilization percentage of host i;
- P_{static} : Power consumption of switch without traffic;
- P_{port} : Power consumption of each port on switch;
- q_i : The number of active ports on switch i;

Power consumption of host i is modelled based on the host CPU utilization percentage [23]

$$P(h_i) = \begin{cases} P_{idle} + (P_{peak} - P_{idle}) \cdot u_i & \text{if } \sigma_i > 0, \\ 0 & \text{if } \sigma_i = 0. \end{cases}$$

Idle power consumption is constant factor consumed by hosts no matter how much workload it received. It can be reduced only if the host is turned off. Meanwhile a host consumes more energy when it processes more workload which leads to higher CPU utilization. In this research we adopted linear power model described in [23]. As hosts are homogeneous, power consumption of a host will be same to another if the CPU utilization is same.

Power consumption of switch i is calculated based on the active ports [19]

$$P(s_i) = \begin{cases} P_{static} + P_{port} \cdot q_i & \text{if } s_i \text{ is on}, \\ 0 & \text{if } s_i \text{ is off}. \end{cases}$$

Similar to host's energy consumption, a switch also has static part in its power usage regardless of its network traffic. On top of the static consumption, it consumes more energy when more ports are active with a traffic passing through the switch. We use linear model addressed in [19], where energy consumption of a switch is proportional to the number of active ports in the switch.

4.2 Problem Formulation

The problem is to optimize the host and network energy consumption jointly in each time period as described below. $|VM|$ VMs are placed in $|H|$ hosts for the time period where $|L|$ links are connected

$$\textbf{minimize} \sum_{i=1}^{|H|} P(h_i) + \sum_{i=1}^{|S|} P(s_i)$$

and **minimize** $SLA\ violation$

subject to:

$$\sum_{i=1}^{|H|} \sigma_i = |VM| \tag{1}$$

$$\forall h_i \in H, \sum_{j=1}^{\sigma_i} rd(vm_{j,i}) \leq C(h_i) \tag{2}$$

$$\forall l_i \in L, \sum_{j=1}^{n_i} d(f_{j,i}) \leq C(l_i) \tag{3}$$

$$\forall i, \sum_{j=1}^{|VM|} \theta_{i,j} = 1, \text{where } \theta_{i,j} = \begin{cases} 1 & \text{if } vm_{j,i} \text{ is placed in } h_i \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The objectives are to minimize total energy consumption (energy consumed by hosts and switches) in a data center, and at the same time to minimize the SLA violations. As the two distictive objectives have different measurements, we measure them separately to minimize both objectives at the same time. In this work, SLA violation is quantified to the percentage of the requests exceeding the expected response time. We measured the response time of each request with a baseline algorithm without overbooking, and used it as the expected response time to count the number of requests violating SLA.

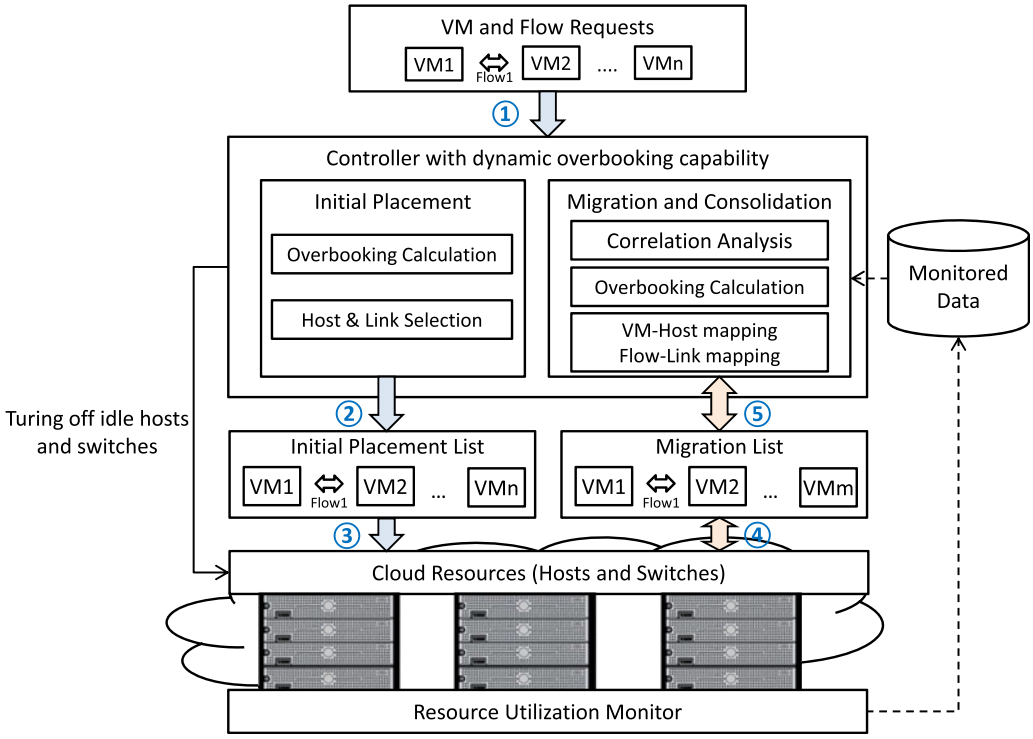


Fig. 1. Resource allocation architecture.

The constraints are that resources given to VMs in a host cannot exceed the capacity of the host, and the total data flow rate in a link cannot exceed the capacity of the link, and each VM is placed only once in a host.

5 Resource Allocation Framework

The architecture aims to minimize SLA violation and maximize energy saving at the same time without pre-knowledge of the workload. Our proposed architecture is illustrated in Fig. 1, which benefits from overbooking through SLA-aware VM and flow consolidation. *Overbooking Controller* is in charge of controlling the initial placement and consolidation process. One of the main components is initial placement policy which decides where to place a VM when it is admitted to the data center and creates the initial placement list. Another top-most component is the migration policy that decides a destination host for a VM when the current host is overloaded. It refers a migration list created based on the monitored data and decides which host to migrate to.

For both components, proper overbooking ratio is identified using link information and correlation analysis between VMs' resource utilization. Then a host is discovered which can provide the identified overbooked capacity. Correlation analysis uses monitoring data collected from hosts, VMs and network traffic. This data is also used to build a migration list which consists of highly utilized VMs in the overloaded hosts to be migrated to another host decided by the Migration policy. The consolidation policy uses current link traffic and host utilization for VM and flow placement and consolidation.

Resource Utilization Monitor. This component is in charge of monitoring the utilization levels of resources. Each physical resource can monitor its utilization by itself, such as CPU utilization of each host or bandwidth utilization of the link between switches. The utilization metrics monitored by each physical resource are collected at this component to provide relevant history data to the migration policy. It also collects utilization data of VMs and virtual links to decide the most suitable host for the VM.

Initial Placement. When VM and virtual link creation requests arrived at the cloud data center, Initial Placement decides where to create the new VM. At this stage no history or workload information is provided to the data center. Instead, only initial VMs and their connection configurations are available to decide where to place the VM. If VMs are created by the same user at the same time, for example, those VMs have a higher probability to generate traffic between each other. Using this information in addition to the VM configuration, this component decides a host that has sufficient host and network resource to serve the request.

Migration and Consolidation. In case of overloading, some VMs in the overloaded host must be migrated to another host in order to minimize SLA violation. Otherwise, VMs in the overloaded host can provide poor performance in computation or network which results in severe customer dissatisfaction. Migration and Consolidation component selects VMs to be migrated in overloaded hosts and decides where to migrate by analyzing historical utilization data of hosts, links and VMs. At first, migration list composing of VMs to be migrated is created based on the monitoring data collected from VMs, hosts and switches. Once the migration list is ready, it analyzes correlation level to other VMs and hosts using historical utilization data. This data is used to pick a migrating host in consideration of overbooking capacity and energy savings.

6 Resource Allocation Strategy

In cloud data center, consolidation of VMs and network traffics into a smaller set of physical resources leads to

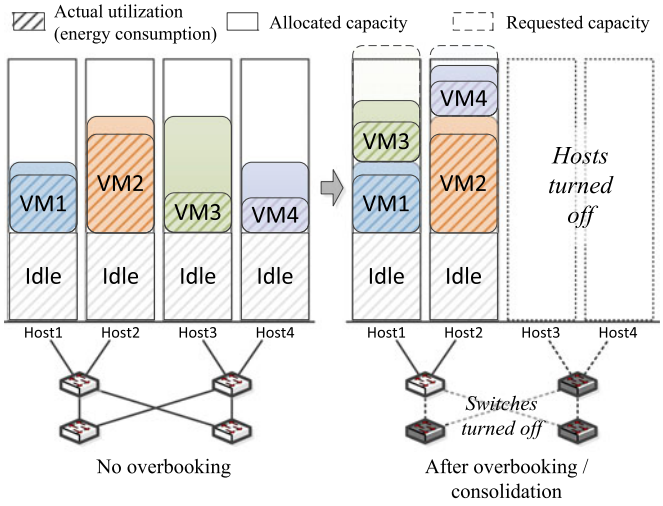


Fig. 2. Example of consolidation with overbooking.

saving power consumption by turning off unused hosts and switches when the physical devices are homogeneous. Although large-scale data centers may consist of heterogeneous devices with different batches of servers and switches, within a single rack, devices are mostly homogeneous. Therefore, we focus on homogeneous configuration to simplify the problem and develop our strategies. A VM is placed to a host by allocating the requested amount of resources, such as CPU cores, memory, disk, and network bandwidth. As in most cases resources are over-provisioned, allocating less resource than requested can help consolidate more VMs and traffics. For clear description, a simple example of overbooking and consolidation in concept is illustrated in Fig. 2.

Before overbooking and consolidation, VM1-4 are placed in four hosts separately and connected through four switches. If all the four VMs have data traffic, all the switches should be active and consume electrical power along with the four hosts. For VM3 and VM4, we can see that the actual utilization is far lower than the allocated capacity. After overbooking, less amount of resource is allocated to VM3 and VM4 which now can be consolidated to Host1 and Host2. After migration of VM3 and VM4 to Host1 and Host2 respectively, the hosts without VMs can be turned off, and the connected switches also can be switched off.

We tackled the resource allocation problem described in Section 4 through two stages: (1) initial placement stage, and (2) migration and consolidation stage. Initial placement is to find a suitable host for a VM when the VM is created in the cloud data center, whereas VM migration is occurred when the VM needs to be migrated to another host due to a host being either overloaded or underutilized. Note that a different algorithm can be selected for each stage, thus multiple combinations of the two stages are available in the proposed system. The following sections explain different algorithms for each stage.

For the initial placement the following conditions hold:

- We have no prior knowledge regarding the workload, host utilization, VM utilization, and data rates of flows.
- Although we have no information regarding correlation coefficient between two VMs, it is likely that for the case of web application, workload of connected VMs is correlated.
- If the initial placement strategy places connected VMs on the same host, there is less opportunity for overbooking leading to smaller overbooking ratio. However, this still allows for more saving for network communication cost. Overbooking ratio determines the percentage of original requested resources by users (either in terms of VM or bandwidth).

6.1 Connectivity-Aware Initial VM Placement Algorithms

Initial VM placement algorithms consider connectivity between VMs as explained below.

ConnCons*: Connected VMs to be Consolidated in One Host.*

At the beginning of the placement, the algorithm (pseudo code is shown in Algorithm 1) groups VMs based on their connectivity. Then, it sorts the groups based on their resource requirements (sum of VM's resource demands) in decreasing order. Once the list is ready, it picks a VM ($vm_{k,i}$) from the top of the list. If it is not connected to other VMs or if the connected VMs have not been placed yet, we place it using most-full bin-packing algorithm. Otherwise, it consolidates the VM to the same server (h_i) where the connected VMs are placed if the following constraint can be met:

$$\sum_{j=1}^{\sigma_i}(rd(vm_{j,i})) + IRAR \times rd(vm_{k,i}) < C(h_i), \qquad (5)$$

where Initial Resource Allocation Ratio ($IRAR$) indicates the proportion of the actually allocating resource to the requested resource at initial stage. Note that Resource Allocation Ratio (RAR) can be regarded as the reverse of overbooking ratio, e.g., 70 percent RAR means that the host will allocate 70 percent of the requested resource to the VM. Thus, with lower RAR value hosts allocate less resource to a VM resulting in placing more VMs in a host and higher chance of SLA violation. $IRAR$ is a predefined constant in the system configuration and can be changed manually.

This method is basically derived from CARPO [19] system that correlated VMs are consolidated into the same or nearby host. In addition to the principle of CARPO, we propose dynamic overbooking strategy that changes overbooking ratio dynamically adapting to the workload.

If there exist multiple connected VMs that have already been placed on different hosts, the most-full host will be selected. Otherwise, it searches for a host with the shortest distances from the connected VM hosts. If multiple VMs have already been placed on different hosts, a host with average shortest distance will be selected. Next, if there are multiple choices (hosts with the same distance and network path with same number of hops), the algorithm uses the most-full first bin-packing algorithm for both hosts and candidate links. In addition, the selected candidates have to meet constraints in Equation (5) and constraint in Equation (6) for each selected Links of l_i

$$\sum_{j=1}^{n_i}(d(f_{j,i})) + IRAR \times d(f_{k,i}) < C(l_i). \qquad (6)$$

If the constraint cannot be met, the algorithm selects the next host candidate until all the VMs are placed. Note that for this algorithm the $IRAR$ are likely to be set to a higher value as utilizations of VMs in a server are likely to be correlated.

Algorithm 1. *ConnCons* Initial Placement

Data: $IRAR$: User-defined initial resource allocation ratio constant.
Data: VM: List of VMs to be placed.
Data: F: List of network flows between VMs.
Data: H: List of hosts where VMs will be placed.
$VMG \leftarrow$ list of VM groups in VM based on connections in F;
sort VMG in descending order of the sum of bandwidth requirements in each group;
for each VM group vmg in VMG **do**
  for each vm in vmg **do**
    $H_{conn} \leftarrow$ List of hosts where other VMs in vmg are placed;
    if H_{conn} is empty **or** length(vmg) = 1 **then**
      Place vm in the most-full host in H;
    else
      sort H_{conn} in ascending order of free resources;
      done $\leftarrow$ false;
      for each h in H_{conn} **do**
        $C_h \leftarrow$ free resource in host h;
        $rd \leftarrow$ adjusted resource demand of vm calculated with $IRAR$;
        if $rd < C_h$ **then**
          Place vm in h;
          $C_h \leftarrow C_h - rd$;
          done $\leftarrow$ true;
        end if
      end for
      if done=false **then**
        Place vm in the host in H with average shortest distance from vmg;
      end if
    end if
  end for
end for

ConnDist*: Connected VMs to be Distributed into Different Hosts.*

At the beginning of the placement, the algorithm (pseudo code is shown in Algorithm 2) sorts the VMs based on their resource requirements in decreasing order. Once the list is ready, it picks a VM ($vm_{k,i}$) from the top of the list. Then, if it is not connected to other VMs or if the connected VMs have not been placed yet, it will be placed using most-full bin-packing algorithm. Otherwise, it ignores servers where the connected VMs are placed, and searches for a server with the average shortest distances from the hosts of connected VMs. Next, if there are multiple choices, the algorithm uses the most-full bin-packing algorithm for both host and link candidates which meets constraint in Equations (5) and (6).

If the constraint cannot be met the algorithm selects the next host candidate until all the VMs are placed. Note that for this algorithm the $IRAR$ is likely to be set to lower values as we consolidate connected VMs to different servers. Therefore, utilization of VMs placed on a same server is less likely to be correlated.

Algorithm 2. *ConnDist* Initial Placement

Data: VM: List of VMs to be placed.
Data: H: List of hosts where VMs will be placed.
sort VM in descending order of the resource requirements;
for each vm in VM **do**
  $VM_{conn} \leftarrow$ List of connected VMs of vm;
  $H_{conn} \leftarrow$ List of hosts where other VMs in VM_{conn} are placed;
  if H_{conn} is empty **then**
    Place vm in the most-full host in H;
  else
    $H_{noconn} \leftarrow H - H_{conn}$;
    Place vm in the most-full host in H_{noconn} with the same constraint in Algorithm 1;
  end if
end for

6.2 VM Migration Algorithms with Dynamic Overbooking

Based on the collected information, this algorithm:

- selects overloaded hosts with the utilization over the threshold (e.g., 0.7) and moves the most utilized VM to the migration list,
- selects the underutilized host with the utilization under the threshold (e.g., 0.1) and move their VMs to the migration list,
- selects the overloaded links with the average bandwidth usage over the threshold (e.g., 70 percent of the link capacity) and move the VMs in the link with highest data rates into the migration list.

It is worth mentioning that the migration of flows happens at the final stage. The reason is that over-utilized VM migration can resolve the link congestion.

After that, the algorithm sorts the VMs in the migration list based on their resource requirements in descending order. Then, it picks a VM ($vm_{k,i}$) from the top of the list. For the selected VM, VM migration algorithm selects a candidate host in which the VM can be placed. In the host selection, dynamic overbooking algorithm is applied as a constraint to make sure the VM and its network link fulfil enough capacity to process the workload, and at the same time limit to minimal amount for consolidation. For the host capacity, Equations (7) and (8) is applied

$$\sum_{j=1}^{\sigma_i}(rd(vm_{j,i})) + DRAR_h \times rd(vm_{k,i}) < C(h_i) \tag{7}$$

$$DRAR_h = Min_{DRAR} + \frac{Max_{DRAR} - Min_{DRAR}}{Max_{DRAR}} \times \frac{1}{\sigma_i}\sum_{j=1}^{\sigma_i} CC(vm_{j,i}, vm_{k,i}). \tag{8}$$

In addition to host capacity constraint, network constraint is also applied as constraint to select the target host and link: Equations (9) and (10)

$$\sum_{j=1}^{n_i}(d(f_{j,i})) + DRAR_l \times d(f_{k,i}) < C(l_i) \tag{9}$$

$$DRAR_l = Min_{DRAR} + \frac{Max_{DRAR} - Min_{DRAR}}{Max_{DRAR}} \times \frac{1}{n_i}\sum_{j=1}^{n_i} CC(d(f_{j,i}), d(f_{k,i})). \quad (10)$$

As you can see from Equations (8) and (10), in this algorithm we dynamically calculate the Dynamic Resource Allocation Ratio ($DRAR$) based on the correlation of VMs in the host. As explained in the previous section, Resource Allocation Ratio is a term that defines the percentage of actually allocated resource compared to the requested resource. It can be regarded as a reserve of overbooking ratio. $DRAR$ is applied not only as constraints of the VM admission, but also actual resource allocation for the migrating VM. This will allow us to save more energy by resource overbooking and honoring more SLA by dynamically changing overbooking ratio.

$DRAR$ is applied as constraints to decide the admission of the migration and to allocate resources to VMs. For example, for 100 percent $DRAR$, the host allocates 100 percent of the requested resource to the VM. If $DRAR$ decreased to 70 percent in another host, it gives only 70 percent of the requested resource thus the host can consolidate more VMs.

To determine $DRAR$, we use correlation coefficient derived from historical utilization data, VM's average utilization of the previous time frame, and preliminarily defined variables to decide the portion of each parameter. Correlation between VMs is calculated with Pearson Correlation Coefficient, which ranges between -1 and 1. Lower the coefficient, lower the correlation. If the coefficient is closer to 1, it indicates the VMs are more correlated. As it ranges from -1 to 1, we use Equation (11) to normalize the range between 0 and 1

$$CC(X,Y) = \left(\frac{\mathbf{Cov}(X,Y)}{\sqrt{\mathbf{Var}(X)\mathbf{Var}(Y)}} + 1\right) \Big/ 2. \quad (11)$$

Additionally, minimum and maximum Dynamic Resource Allocation Ratio (Min_{DRAR} and Max_{DRAR}) are defined to limit the range of the $DRAR$. Min_{DRAR} is differentiated with the average utilization of the VM for the previous time window. Thus, $DRAR$ is affected by not only the correlation, but also the actual utilization of the VM. In order to decide the share of each parameter, α and β are defined in

$$Min_{DRAR} = \alpha \times U(vm_{k,i}) + \beta, \quad (12)$$

where α specifies the portion of the utilization of the VM and β specifies the guaranteed proportion of the requested resource to be allocated to the VM. α and β are defined in the experiment configuration along with the $IRAR$. On the other hand, Max_{OR} is configured to 1.0 in the implementation to make it possible to assign 100 percent of the requested resources.

Algorithm 3 shows overall migration procedure for each VM to a candidate host with the constraints using $DRAR$ calculation. The complexity of the alogorithm is $O(|VM_h|)$ which can run for at most the number of hosts in the data center if all hosts are overloaded. Thus, the overall complexity of the migration process is $O(|VM| \cdot |H|)$ for entire data center which is reasonable for online decision.

Algorithm 3. VM Migration with Dynamic Overbooking

```
1:  Data: α: User-defined constant for historical utilization fraction in Min_OR.
2:  Data: β: User-defined constant for minimum Resource Allocation Ratio.
3:  Data: t_start, t_end: Start and end time of the previous time window.
4:  Data: vm_mig: A VM to be migrated.
5:  Data: h: A candidate host.
6:  function MIGRATE(vm_mig, h)
7:      VM_h ← all VMs in the host h;
8:      u_mig ← utilization matrix of vm_mig in (t_start, t_end);
9:      S_corr ← 0;
10:     for each vm_i in VM_h do
11:         u_i ← utilization matrix of vm_i in (t_start, t_end);
12:         S_corr ← S_corr + CC(u_mig, u_i);
13:     end for
14:     DRAR_h ← calculate with Equation (8);
15:     C_h ← free resource in host h;
16:     rd ← requested resource of VM vm_mig;
17:     rd_DRAR ← DRAR_h × rd;
18:     migrated ← false;
19:     if rd_DRAR < C_h then
20:         DRAR_l ← calculate with Equation (10);
21:         C_l ← free resource of the link of host h;
22:         d ← requested resource of the flow of vm_mig;
23:         d_DRAR ← DRAR_l × d;
24:         if d_DRAR < C_l then
25:             Migrate vm_mig to h;
26:             C_h ← C_h − rd_DRAR;
27:             C_l ← C_l − d_DRAR;
28:             migrated ← true;
29:         end if
30:     end if
31:     returnmigrated
32: end function
```

With the base of dynamic overbooking constraints explained above, three consolidation algorithms are proposed with different host selection methods: most correlated, least correlated, and most underutilized host to be chosen. Consolidating a VM to the most correlated host can make a higher chance to reduce network traffic since the VMs in the host have more correlation on network traffic to the migrating VM. However, it will increase the chance of overloading of the host, as the correlated VMs will have higher possibility to reach the peak at the same time. For this reason, we also propose an approach that consolidates to the least correlated host. If the workload has less network traffic but more computational processing, migration to the least correlated host will reduce the chance of the host overloading. For comparison, migration to the most underutilized host without consideration of correlation is also tested. Note that the dynamic overbooking constraint is applied to every algorithm but with different host selection preferences. These three algorithms are explained below.

MostCorr: VM to be Migrated to the Host Holding the Linked VMs. If the VM to be migrated is not connected to other VMs or if the connected VMs in the list have not been placed yet, it will be placed using most-full bin-packing algorithm. Otherwise, it consolidates the VM to the same server where the connected VMs are placed and if the aforementioned constraints

can be met. If not, it searches for a host with the shortest distances from the connected VMs' hosts. If there are multiple choices, it uses bin-packing (most-full first) both for candidate links and hosts to choose the destination if the aforementioned constraints can be met. Details are described in Algorithm 4.

Algorithm 4. *MostCorr* Migration Algorithm with Dynamic Overbooking

```
1: Data: VM: Selected migration VM list.
2: Data: H: List of hosts.
3: sort VM in descending order of requested CPU resources;
4: for each vm in VM do
5:     VM_conn ← List of connected VMs of vm;
6:     H_conn ← List of hosts where other VMs in VM_conn are placed;
7:     if H_conn is empty then
8:         Migrate vm to the most-full host in H with the constraints in Algorithm 3;
9:     else
10:        sort H_conn in ascending order of free resources;
11:        migrated ← false;
12:        for each h in H_conn do
13:            migrated ← MIGRATE(vm, h);
14:            if migrated=true then
15:                break
16:            end if
17:        end for
18:        if migrated=false then
19:            Migrate vm to the most-full host in H with the constraints in Algorithm 3;
20:        end if
21:    end if
22: end for
```

LeastCorr: *VM to be migrated to the least correlated host*. The algorithm is similar to the previous consolidation strategy, but selects the host with the lowest average correlation coefficient between the migrating VM and the VMs in the host. It calculates correlation coefficient for each host with at least one VM and sorts the list in ascending order. Then, the least correlated host is selected with the $DRAR$ constraints (3) applied. If no host is found among non-empty hosts, it selects the first one from empty hosts. With this algorithm, the connected VMs are likely to be placed into a separate host which will incur more network communication, whereas the chance of host overloading will be reduced.

UnderUtilized: *VM to be Migrated to the Underutilized Host.* In this algorithm the migrating VM is placed to the least utilized host. First underutilized hosts list is prepared among non-empty hosts, and the first VM in the migration list with the highest utilization is placed to the first host in the list which is the most underutilized. Same as the previous algorithms, it also dynamically calculates $DRAR$ based on the number of VMs in the host. $DRAR$ calculated from correlation is applied as constraints to check whether the host can accept the migration or not. In short, the most utilized VM in the migration list is to be placed in the least utilized host.

6.3 Baseline Algorithms

We compare our approach with the baseline algorithms explained below.

NoOver: *No Overbooking Without Any Migration*. The algorithm is a non-overbooking that allocates 100 percent of the requested resource to all VMs and network. It uses Most Full First bin-packing algorithm for VM placement, which allocates the VM to the most full host that has enough resource to serve the VM. When selecting a host, it does not consider any connectivity or correlation between VMs. Therefore, VMs can be allocated in any host regardless of their connectivity, e.g., the connected VMs can be randomly placed in the same host or in different hosts depending on the available resource of each host at the moment. Migration is not implemented in this algorithm as a host will not exceed its capacity. This is used as a baseline at evaluation to calculate SLA violation rate and energy saving percentage.

ConnNone: *Connectivity Agnostic Overbooking*. For initial placement, this algorithm overbooks resources without consideration of the connectivity between VMs. This algorithm allocates less amount of resources to VMs and uses the Most Full First algorithm for VM allocation regardless of VM links. For example, ConnNone 70 percent is to allocate only 70 percent of the requested resource to the VM and place it to the most full host which can serve the 70 percent of the requested resource. Similarly, ConnNone 100 percent is to allocate 100 percent of the requested resource which is in fact same as NoOver algorithm.

StaticMigration: VM to be Migrated to the Most Correlated Host Without Dynamic Overbooking. Similar to MostCorr, this algorithm also selects the correlated host first for a migrating VM with the same constraints described in Section 6 except for $DRAR$. Instead of using dynamically calculated $DRAR$, this algorithm uses a static overbooking ratio for the constraints of the host selection and the resource allocation. As a result, the new host will allocate the same amount of the resource to the migrating VM. This algorithm is implemented in order to refer to PowerNetS [11].

7 Performance Evaluation

The proposed algorithms are evaluated in simulation environment. We implemented the proposed methods in addition to other algorithms including non-overbooking and PowerNetS [11], and measured a response time of the workload and total energy consumption in the data center. SLA violation is checked through the response time of the workload. We measured the response time of each workload with a baseline algorithm without overbooking, and use them to compare with the response time of the proposed algorithms. If the response time of a workload with a proposed algorithm is longer than the baseline one, the workload is as a SLA violation. Energy consumption is also compared with the no overbooking baseline algorithm.

7.1 Testbed Configuration

In order to evaluate our approach, we implement the algorithms in CloudSimSDN [24]. CloudSimSDN is a CloudSim [25] based simulation tool which supports various SDN features such as dynamic network configuration and programmable controller. We add monitoring components to the simulator to gather utilization information of VMs, hosts, and network traffics to be used at dynamic overbooking methods described in Section 6.

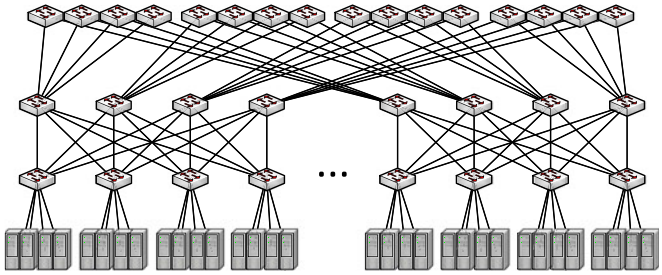


Fig. 3. Network topology used in simulation.

The cloud data center simulated for our experiment consists of 128 hosts, each with 8 CPU cores, connected with Fat-Tree topology [26].

Fig. 3 shows eight-pod Fat-Tree topology which we adopt in the experiment. Each pod consists of 16 hosts, 4 edge switches and 4 aggregation switches. On top of all pods, 16 core switches enables communication between pods by connecting aggregation switches in each pod. Other resource requirements such as memory and storage are not considered in the experiments to eliminate the complexity affecting the results.

Unless noted, initial placement overbooking algorithms (ConnNone, ConnCons, and ConnDist) have Initial Resource Allocation Ratio value set to 70 percent in all experiments. For dynamic overbooking migration algorithms, we set α value being 12 percent and β value being 40 percent (Equation (12)). Thus, Dynamic Resource Allocation Ratio is guaranteed to be at least 40 percent and dynamically changing up to 100 percent which is affected by the previous utilization average for 12 percent and correlation analysis for the rest 48 percent. MAX_{OR} is set 100 percent to make sure VMs can receive the full resource when necessary.

For experiments with migration policy, the monitoring interval is set to 3 minutes to collect the utilization of VMs, hosts, flows, and links. Dynamic time window to run migration policy is configured to 30 minutes, thus migration is attempted every 30 minutes with the utilization matrix of 10 monitored points. These parameters are selected in consideration of the workload and the migration costs, but can be changed arbitarily for different workloads.

7.2 Workload

In a typical data center traffic varies hourly, daily, weekly, and monthly. Traffic characterization of a data center would allow us to discover patterns of changes that can be exploited for more efficient resource provisioning. To achieve our objectives for this activity, we have focused on Wikipedia data center analysis. We decided to investigate a Wikipedia workload by looking into *Page view statistics for Wikimedia projects* which are freely and publicly available. For each day and for all of Wikipedia's projects, the traces consist of hourly dumps of page view counts. To gain insight of the traffic for each project for the whole day (Sept. 1, 2014 chosen for this case) we need to analyze traces which consist of 24 compressed files each containing 160 million lines (around 8 GB in size). We have utilized MapReduce to calculate number request per hour for each project more effectively and faster.

As Fig. 4 shows, we can observe that workload varies per hour and that not all workload are reaching their peaks at

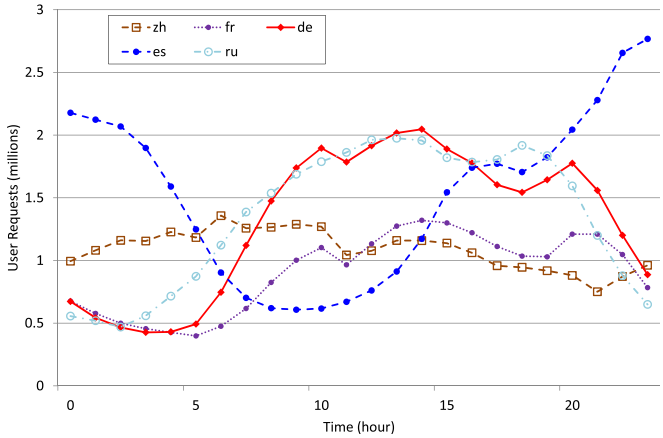


Fig. 4. Wikipedia workload for different projects collected for 1st of Sep. 2014.

the same time. We can assume that each project is hosted by a set of virtual machines; there exist VMs that their utilizations are not correlated. This observation can help us to propose more efficient resource provisioning cloud data center by placing non-correlated VMs in one host and thus accomplishing effective overbooking.

When the workload is supplied to the configured test bed, we can see the utilization of each VM follows the actual workload as depicted in Fig. 5.

7.3 Initial Placement

In this experiment set, we compare the initial placement algorithms without implementing any migration policy. We compare the algorithms in terms of SLA violation rates, energy consumption in hosts and network devices, and the energy savings.

Investigating the Impact of Static Overbooking on Energy Efficiency and SLA Violation. The aim of these experiments is showing the essence of designing dynamic overbooking algorithms that in comparison to static overbooking strategies reduce not only energy consumption but also SLA violations. First, we have conducted experiments to show how much energy we can save when we use static overbooking. Fig. 6 shows SLA violation percentage and energy consumption of ConnNone which does static overbooking for initial placement. In ConnNone, resources are allocated to

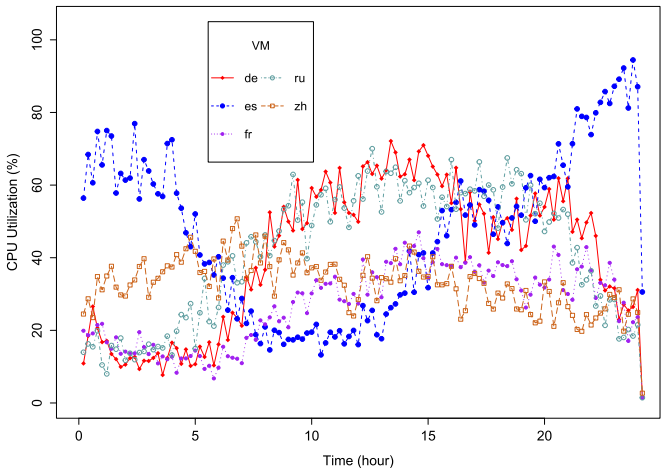
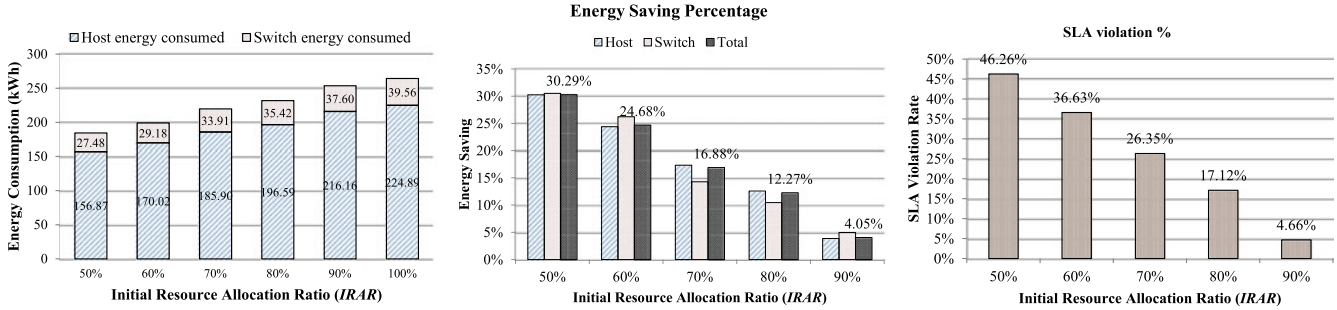


Fig. 5. CPU utilization of VMs with Wikipedia workload.



(a) Energy consumption. (b) Energy saving compared to baseline algorithm (NoOver). (c) SLA violation percentage.

Fig. 6. Energy consumption and SLA violation results of initial placement without consideration of connectivity.

VMs with the fixed $IRAR$ without any consideration of connection between VMs. As shown in Fig. 6a, overall energy consumption linearly decreases as $IRAR$ decreases. Allocating less resource lets hosts to accept more VMs, which leads to less number of active hosts. Network energy consumption also decreases with higher $IRAR$ value, because fewer hosts are communicating through less number of switches. Fig. 6b shows the energy saving percentage of the static overbooking methods compared to the one without overbooking. For an extreme case when only 50 percent of the requested resources are allocated to VMs and networks, it can save 30.29 percent of the energy consumption in total. With 70 percent $IRAR$, the energy saving percentage reduced to 16.88 percent as less VMs can be placed in a single host with the higher $IRAR$.

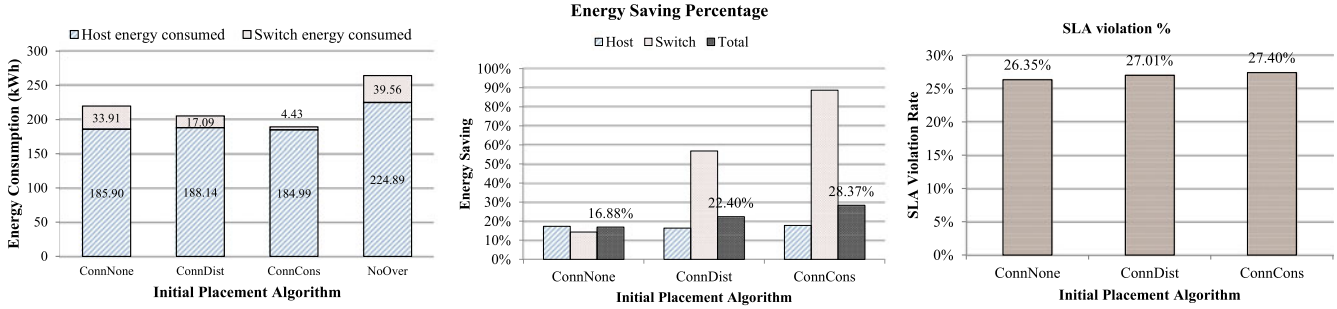
However, as Fig. 6c shows, SLA violation increases significantly with lower $IRAR$ (note that the lower the $IRAR$ means that less resources were allocated to VM and vice versa). This is because the strategy has no consideration of either correlation or migration. As less resources are given to each VM, hosts and network are more frequently overloaded, which leads to slower response of the workload and SLA violation. While the static overbooking with lower $IRAR$ can save more energy, it also increase the SLA violation rate. Therefore, we need an algorithm that considers the trade-off between energy saving and SLA violation while dynamically adjusts overbooking ratio.

Investigating the Performance of Different Initial Placement Algorithms Using Static Overbooking. The aim is to compare the effects of placing connected VMs into a same host (ConnCons) with placing them in a different host (ConnDist). We expect that the connected VMs (especially for the 3 tier web applications) would have correlated workload, hence if they are placed in the same host, there is less chance for overbooking. However, if they are placed in the same host, network traffic and energy consumption would reduce since most network traffic between the connected VMs could be served within the host through memory instead of external network devices.

As shown in Fig. 7a, energy consumption of the switches is significantly reduced under both ConnDist (connected VMs are placed in close distances but not to the same host) and ConnCons (connected VMs to the same host) compared to ConnNone which does overbooking but placing connected VMs to random hosts. Especially, in ConnCons only 4.43kWh electricity was consumed in switches which is almost one fourth of the switch energy consumption in ConnDist algorithm. It shows that network energy consumption is further reduced when connected VMs are placed in the same host.

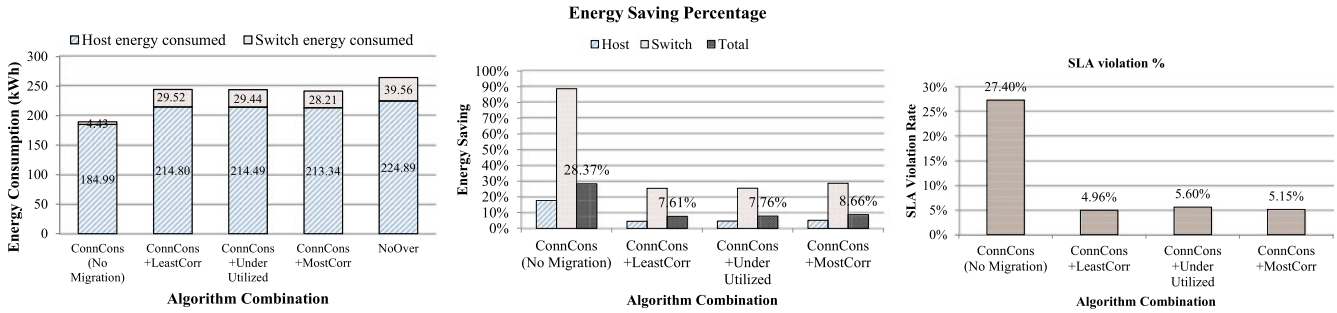
Fig. 7c shows SLA violation percentage of the ConnNone, ConnDist, and ConnCons algorithms with $IRAR$ setting at 70 percent. SLA violation percentages in ConnDist and ConnCons are still as high as ConnNone algorithm reaching at around 25 percent with 70 percent $IRAR$. Although ConnCons was expected to have less chance for overbooking that should result in more SLA violations, the experiment result shows that ConnDist results in a slightly more SLA violations than ConnCons algorithm. This is due to the characteristics of the workload that has less potential to the chance of overbooking.

Fig. 6b shows the energy saving percentage compared to the one without overbooking. As we discussed above, with ConnDist algorithm we can save over 50 percent of switches



(a) Energy consumption. (b) Energy saving compared to baseline algorithm (NoOver). (c) SLA violation percentage.

Fig. 7. Energy consumption and SLA violation results of different initial placement algorithms.



(a) Energy consumption. (b) Energy saving compared to baseline algorithm (NoOver). (c) SLA violation percentage.

Fig. 8. Energy consumption and SLA violation results of different migration strategies implemented on ConnCons (connected VMs in the same host) initial placement algorithm.

power usage, and with ConnCons the saving percentage reaches at almost 90 percent compared to non-overbooking method. Overall power saving is also increased in both ConnDist and ConnCons algorithms.

7.4 Migration Policy

Migration policy plays important role when a host or network encounters overloading. Since overloading happens due to the lack of resources, migration of VMs from an overloaded host to a free host can resolve the overloading issue that might cause significant SLA violation. Several migration policies have been experimented with dynamic overbooking ratio.

Investigating the Impact of Migration Strategies. At first, we tested different migration strategies in the combination of ConnCons initial placement method. In this experiment we aim to find the effectiveness of the different migration strategies under the same initial placement. Figs. 8a and 8b respectively show the total energy consumption and the percentage of energy saving of different migration algorithms. With any migration algorithm, energy consumption of hosts and switches increases compared to the one without migration. While ConnCons without migration can save 28.37 percent of power consumption, three algorithms with migration policies (ConnCons+LeasCorr, ConnCons+UnderUtilized, and ConnCons+MostCorr) can save between 7 and 8 percent of the total energy (Fig. 8b). In detail, three migration algorithms use almost same amount of energy at both hosts and switches, and they still consume less power than the algorithm with no overbooking at all (Fig. 8a).

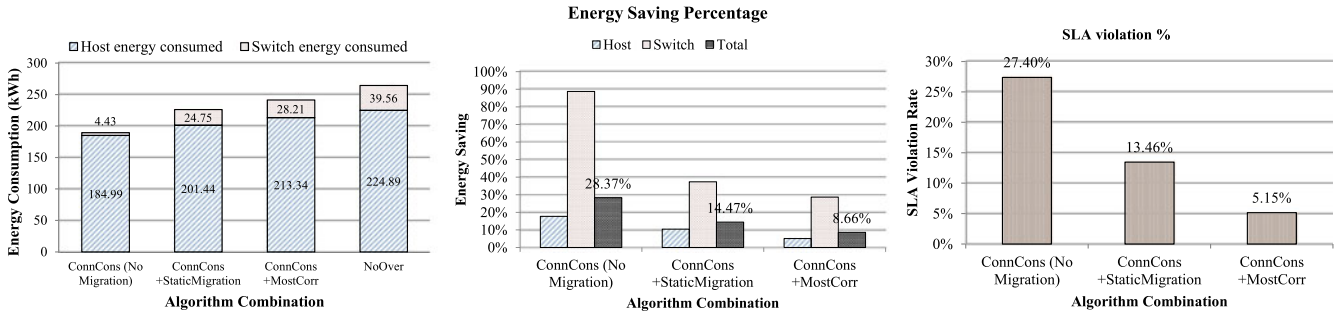
However, as shown in Fig. 8c, SLA violation decreases significantly when migration policies are implemented. While 27.40 percent of workloads violated SLA under ConnCons with no migration policy, just about 5 percent of workloads violated SLA when any migration algorithms was combined. In detail LeaseCorr migration algorithm results in the least SLA violation rate at 4.96 percent, and UnderUtilized policy results in 5.60 percent SLA violation rate, which is far less than the one without migration policy.

The results show the effectiveness of the dynamic overbooking strategy. As the migrating VM has been allocated to the host with dynamic overbooking ratio depending on the VMs in the host, it prevents highly correlated VMs to be consolidated into the same host. All three dynamic overbooking migration algorithms (MostCorr, LeastCorr, and UnderUtilized) show the similar results which significantly reduce SLA violation rate although they use various host selection methods to prioritize the candidate hosts. As we expected, dynamic overbooking can reduce the chance that all VMs in the host hit the peak at the same time, thus VMs acquire enough resources to process their workloads.

Investigating the Impact of Dynamic Overbooking Ratio. Next, we investigated the impact of dynamic overbooking by comparing with a static overbooking strategy under the same overbooking condition. The aim is to compare the effectiveness of our approach with a static overbooking algorithm similar to PowerNetS [11] which also implements overbooking with the consideration of correlation. Direct comparison to PowerNetS is not feasible because our approach is online algorithm without any prior knowledge of the workload, while PowerNetS acquired correlation of workloads in advance. Therefore, we use the results of ConnCons+StaticMigration combination which is the most analogous to PowerNetS. Both of them initially place connected VMs into closer hosts and migrate overloaded VMs to the nearest host where the connected VMs are placed. Note that ConnCons+StaticMigration algorithm is different from PowerNetS in the aspect that StaticMigration algorithm does not consider correlation threshold constraint which PowerNetS did implement. Thus ConnCons+StaticMigration algorithm would result in higher SLA violation rate than PowerNetS. We compare ConnCons+StaticMigration with ConnCons+MostCorr algorithm.

Fig. 9 presents the difference of static overbooking and dynamic overbooking algorithms. As shown in Fig. 9a and 9b, the static overbooking approach (ConnCons+StaticMigration) consumed slightly less energy than the dynamic method (ConnCons+MostCorr). In detail, 56.55 kWh is consumed across the whole data center for both hosts and network in the static overbooking method while 60.39 kWh is consumed in the dynamic overbooking which is 6.79 percent more than the static method. With the static algorithm, the overbooking ratio of the migrating VM is not changed in the new host when the in overloaded host is migrated to another host. Thus, regarding the entire data center more VMs can be placed in a host compared to dynamic overbooking which would allocate more resource for the migrating VM if correlated VM is in the migrating host.

The effectiveness of the dynamic overbooking can be clearly seen in SLA violation percentage presented in Fig. 9c. SLA violation rate of the static overbooking



(a) Energy consumption.

(b) Energy saving compared to baseline algorithm (NoOver).

(c) SLA violation percentage.

Fig. 9. Energy consumption and SLA violation results of dynamic and static overbooking algorithms.

algorithm (13.46 percent) is far higher than the dynamic algorithm (5.15 percent). Although our dynamic overbooking method consumed 6.79 percent more power, it dramatically reduced SLA violation rate by 61.74 percent.

7.5 Analysis of Energy Consumption

In this section, we investigate further details of the power consumption in different algorithm combinations. At first, we analyzed the origin of the energy savings where it comes from by showing the proportion of the saved energy in hosts and switches. Fig. 10 shows the ratio of energy saving by hosts and switches. We tested ConnNone initial placement without migration in various $IRAR$ values (70, 80, and 90 percent) as well as the algorithms evaluated in the previous section. For each algorithm, energy consumption at hosts and switches is measured to calculate the saved energy compared to NoOver algorithm. For ConnNone algorithm, about 80 to 90 percent of the energy saving results from hosts while less than 20 percent from switches regardless of $IRAR$ value. However, when ConnCons initial placement (correlated VMs placed in closer hosts) is applied, energy saving ratio at switches increases significantly reaching at a half of the energy saving regardless of migration policy. This is because the consolidation of VMs can reduce significant amount of network traffic which leads to reducing the number of active switches. Interestingly, for ConnDist algorithm energy saving ratio of switches is lower than ConnCons but higher than ConnNone. As VMs in the same host are less likely peak at the same time in ConnDist algorithm, one host can hold more number of VMs than ConnCons algorithm which also affect the dynamic overbooking ratio adjusted by the correlation. In ConnDist initial placement, VMs in the same host would be less correlated which makes more VMs to be placed in one host at the migration stage, as Dynamic Resource Allocation Ratio increases with lower correlation coefficient.

7.6 Dynamic Overbooking Ratio

In order to investigate the impact of dynamic overbooking in energy consumption, we explored the power consumption of the whole data center (Fig. 11a), energy consumption by hosts (Fig. 11b), and by switches (Fig. 11c) over the time. Compared to the baseline (NoOver), static overbooking method (ConnNone) uses constantly less amount of energy. Correlation-aware algorithms such as ConnCons+MostCorr and ConnDist+LeastCorr have less energy consumption in the beginning, but it converges to the baseline once time passes especially after the whole data center is highly loaded. For the network energy consumption, almost no energy is used with ConnCons algorithm at the beginning when most linked VMs are placed within the same host. However, as hosts get overloaded over the time, more switches are utilized which leads to consuming more energy. This result shows that how our algorithm reduces energy consumption and converges over time.

In this experiment, we investigated how overbooking ratio changes dynamically in correspondence with the workload. We randomly chose one sample VM from the previous experiments, and measured its CPU workload and Resource Allocation Ratio (RAR) in different algorithms. Fig. 12 presents the CPU utilization level and the Resource Allocation Ratio of the VM changing over time. The first Fig. 12a shows the CPU utilization of the VM without overbooking in correspondence with its workload. It is obvious that the VM consumes more CPU resource when there is more load. Figs. 12b, 12c, and 12d show the Resource Allocation Ratio of the VM in different overbooking algorithms. For the static overbooking method without migration (ConnNone), the VM acquires only 70 percent of requested resource all the time constantly as the RAR sets to 70 percent without dynamic overbooking strategy. However, with our proposed dynamic overbooking algorithms (ConnCons +MostCorr and ConnDist+LeastCorr), RAR continuously changes over time following the actual workload. As we set up Initial Resource Allocation Ratio (IRAR) to 70 percent, the RAR starts at 0.7 in both algorithm, and dynamically fluctuates over time following the actual CPU utilization shown in Fig. 12a. The result shows that the overbooking ratio reflects the real workload, so that the VM acquires more resources when necessary.

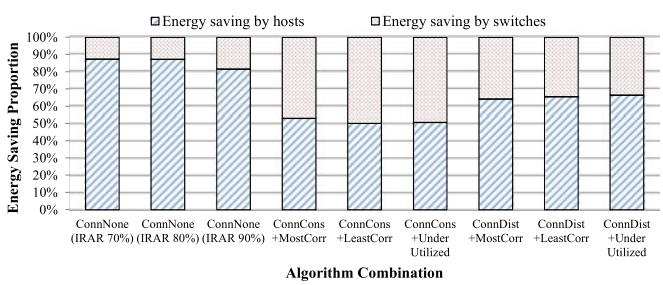
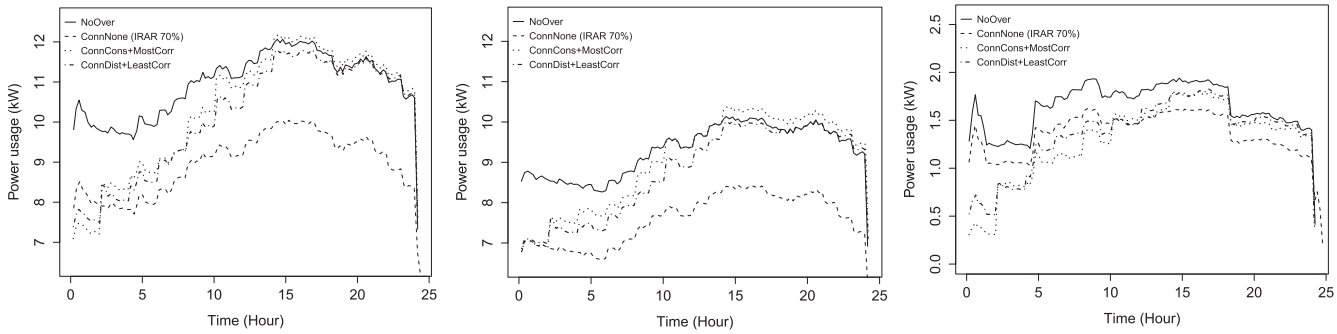
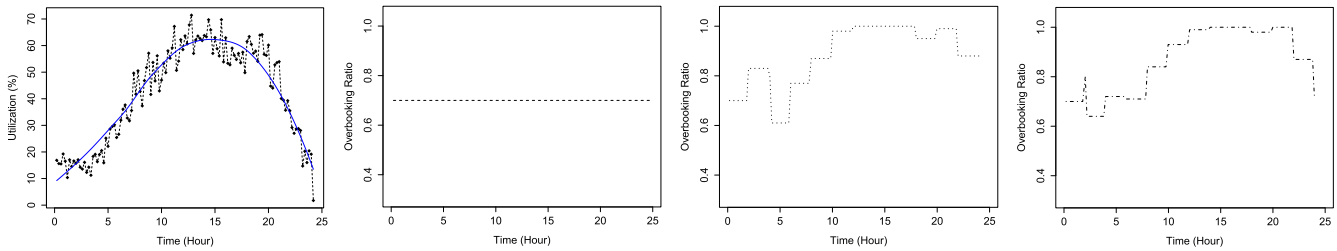


Fig. 10. Energy saving origination.



(a) Overall energy consumption. (b) Energy consumed by hosts. (c) Energy consumed by switches.

Fig. 11. Energy consumption observation over time.



(a) CPU Utilization of the VM. (b) ConnNone (RAR 70%). (c) ConnCons+MostCorr. (d) ConnDist+LeastCorr.

Fig. 12. CPU utilization and resource allocation ratio of a sample VM.

8 Conclusions and Future Work

In this paper, we presented dynamic overbooking strategies that allocate host and network resources dynamically adapting based on the utilization. The variety of workloads affects the dynamic overbooking ratio in real-time through correlation analysis of the VMs and network utilization. By leveraging the dynamic overbooking method, tightly suitable amount of resources can be allocated to VMs which will maximize energy cost savings by reducing the waste of over-provisioned resource, and at the same time minimize SLA violation by allocating enough resource for the actual workload. With the extensive experiments, we demonstrated that our approach can effectively save energy consumption of the cloud data center while reducing SLA violation rates compared to the baseline.

In the future, we plan to include prediction component for correlation analysis that uses a time-series forecasting engine to predict both VM utilization and data rates of flows for the next time slots. When calculating the dynamic overbooking ratio, predicted values obtained from learning algorithm can be used in addition to the historical utilization data, which will give more accurate forecast than the current method. Moreover, adapting the algorithm to heterogeneous environment can be achieved by extending resource allocation strategy and optimization method. This allows the algorithm to work in larger scale data centers with various hardware specifications in use for hosts and switches. It is also extendable to multiple providers with various SLAs if the algorithm includes heterogeneous network cost between different data centers.

References

[1] Natural Resources Defense Council, "America's data centers consuming and wasting growing amounts of energy," (2014). [Online]. Available: https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IB.pdf

[2] L. Li, W. Zheng, X. Wang, and X. Wang, "Coordinating liquid and free air cooling with workload allocation for data center power minimization," in *Proc. 11th Int. Conf. Autonomic Comput.*, Jun. 2014, pp. 249–259.

[3] D. J. Schumacher and W. C. Beckman, "Data center cooling system," U.S. Patent 6 374 627, Apr. 23, 2002.

[4] E. Pakbaznia and M. Pedram, "Minimizing data center cooling and server power costs," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Des.*, 2009, pp. 145–150.

[5] I. Rodero, J. Jaramillo, A. Quiroz, M. Parashar, F. Guim, and S. Poole, "Energy-efficient application-aware online provisioning for virtualized clouds and data centers," in *Proc. Int. Green Comput. Conf.*, Aug. 2010, pp. 31–45.

[6] H. Goudarzi and M. Pedram, "Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jul. 2011, pp. 324–331.

[7] H. Goudarzi, M. Ghasemazar, and M. Pedram, "SLA-based optimization of power and migration cost in cloud computing," in *Proc. 12th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, May 2012, pp. 172–179.

[8] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in *Proc. 9th Int. Conf. Autonomic Comput.*, 2012, pp. 145–154.

[9] T. M. Lynar, Simon, R. D. Herbert, and W. J. Chivers, "Reducing energy consumption in distributed computing through economic resource allocation," *Int. J. Grid Utility Comput.*, vol. 4, no. 4, pp. 231–241, 2013.

[10] Q. Xilong and X. Peng, "An energy-efficient virtual machine scheduler based on cpu share-reclaiming policy," *Int. J. Grid Utility Comput.*, vol. 6, no. 2, pp. 113–120, 2015.

[11] K. Zheng, X. Wang, L. Li, and X. Wang, "Joint power optimization of data center network and servers with correlation analysis," in *Proc. 33rd Annu. IEEE Int. Conf. Comput. Commun.*, 2014, pp. 2598–2606.

[12] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, San Rafael, CA, USA: Morgan and Claypool Publishers, vol. 4, no. 1, pp. 1–108, 2009.

[13] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "Network-aware virtual machine placement and migration in cloud data centers," *Emerging Res. Cloud Distrib. Comput. Syst.*, vol. 42, pp. 42–91, 2015.

[14] Open Network Foundation, "Software-defined networking: The new norm for networks," 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/ white-papers/wp-sdn-newnorm.pdf
[15] R. Buyya, R. N. Calheiros, J. Son, A. V. Dastjerdi, and Y. Yoon, "Software-defined cloud computing: Architectural elements and open challenges," in *Proc. 3rd Int. Conf. Advances Comput. Commun. Informat.*, 2014, pp. 1–12.
[16] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances Comput.*, vol. 82, pp. 47–111, 2011.
[17] B. Heller, et al., "ElasticTree: Saving energy in data center networks," in *Proc. 7th USENIX Conf. Netw. Syst. Des. Implementation*, 2010, pp. 17–17.
[18] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2010, pp. 338–347.
[19] 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.
[20] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2876–2880.
[21] H. Jin, et al., "Joint host-network optimization for energy-efficient data center networking," in *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process.*, 2013, pp. 623–634.
[22] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Comput. Netw.*, vol. 57, no. 1, pp. 179–196, 2013.
[23] S. Pelley, D. Meisner, T. F. Wenisch, and J. W. VanGilder, "Understanding and abstracting total data center power," in *Proc. Workshop Energy-Efficient Des.*, June 2009.
[24] 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.
[25] 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.
[26] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Oct. 2008.

Jungmin Son received the BE degree from Ajou University, South Korea, and the M.InfoTech degree from the University of Melbourne, Australia. He is currently working towards the PhD degree in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne. His research interests include SDN-enabled cloud computing, resource scheduling, and energy efficient data centers.

Amir Vahid Dastjerd received the PhD degree and worked as a research fellow with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. He is a manager at PwC Australia, Melbourne. His current research interests include big data analytics, cloud service coordination, scheduling, and resource provisioning using optimization, machine learning, and artificial intelligence techniques. He is a member of the IEEE.

Rodrigo N. Calheiros is a lecturer in the School of Computing, Engineering and Mathematics, Western Sydney University, Australia. He has worked in the field of cloud computing and related areas since 2008, and he has carried out R&D supporting research in the area. His research interests also include big data, internet of things, fog computing, and their application.

Rajkumar Buyya is a Fellow of IEEE, a professor of computer science and software engineering and director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the university, commercializing its innovations in grid and cloud computing. He served as a future fellow of the Australian Research Council during 2012-2016. He has authored over 550 publications and four textbooks including *Mastering Cloud Computing* published by McGraw Hill and Elsevier/Morgan Kaufmann in 2013 for Indian and international markets, respectively. He is one of the highly cited authors in computer science and software engineering worldwide. Microsoft Academic Search Index ranked him as #1 author in the world (2005-2016) for both field ratings and citation evaluations in the area of distributed and parallel computing. For further information, please visit: http://www.buyya.com