

Received June 9, 2020, accepted June 17, 2020, date of publication July 6, 2020, date of current version July 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007469

Dynamic Resource Management for Cloud Spot Markets

FADI ALZHOURI¹, (Member, IEEE), SUHIB BANI MELHEM², ANJALI AGARWAL³, (Member, IEEE), MUSTAFA DARAGHMEH³, YAN LIU³, (Member, IEEE), AND SADEQ YOUNIS³

¹Department of Electrical and Computer Engineering, University of Ottawa, Ottawa, ON K1N 6N5, Canada

²Department of Electrical and Computer Engineering, York University, Toronto, ON M3J 1P3, Canada

³Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G 1M8, Canada

Corresponding author: Fadi Alzhouri (alzhouri@yahoo.com)


ABSTRACT Resource management for cloud computing environments that are characterized by many layers emerges as a critical task for cloud computing providers. Such providers are compelled by the demands and strategies of stochastic customers to adopt dynamic resource management for the top-bottom scaling of the cloud resources on the basis of variable needs. Resource management in the infrastructure as a service layer relies on virtual machine (VM) characteristics, such as estimated VM classes. Given that a cloud provider offers a variety of VM classes that differ as regards the size of computing resources (e.g., central processing unit, memory, and input/output devices), optimizing cloud resources to maximize cloud revenue is a challenging dilemma. More specifically, the dynamic management of resources in cloud spot markets is confronted with various severe obstacles. In consideration of these issues, this study investigated a dynamic resource management model for cloud spot markets and put forward an efficient model that manages spare resources for the purpose of expanding cloud revenue. The model estimates the available spare capacity of a spot market, evaluates the maximum expected revenue of stagnant VMs on the basis estimated cumulative capacity, and locates the optimum VM combinations that bear complementary workloads and capacities and can coexist in a certain host. Our model also improves the understanding of cloud resource scaling and generates inferences that can be adopted in managing cloud resources for all layers as well as Reserved and On-Demand markets.

INDEX TERMS Resource management, resource allocation, dynamic allocation, cloud, IaaS, spot market, virtual machine, CloudSim.

I. INTRODUCTION

Many cloud computing providers that offer infrastructure as a service (IaaS) implement various pricing schemes, such as on-demand, reserved, and spot pricing. Amazon Elastic Compute Cloud [1] adopts these schemes, as does Microsoft, which provides Reserved and Pay-as-you-go pricing for its Azure virtual machine instances [2]. These schemes are roughly partitioned into two categories. The first, usually called reserved pricing, involves the imposition of a static price-per-time unit for long-term, uninterrupted usage under a service level agreement (SLA). The second category, commonly referred to as spot pricing, entails the implementation of a dynamic price-per-time unit for short-term usage that can be interrupted in some situations and is ungoverned by an SLA. These pricing strategies apparently discriminate

VM prices and features to suit customer demand and willingness to pay. Each pricing scheme is also underlain by potential technical reasons and motivations. Spot pricing for VM instances has raised considerable debate about accompanying techniques and objectives. Such discussion has been directed, for example, toward Amazon and Google cloud services. Amazon sells its spare computing capacity through spot instances [1]. Whereas, Google initially offers Pre-emptible VMs for batch and fault-tolerant jobs, before terminating running instances when underlying resources are needed [3]. Reference [4] identified the necessities and objectives that drive cloud providers to adopt a spot pricing strategy. The authors concluded that these companies offer VM instances for a limited time at a dynamic price, depending on the state of cloud resources and regardless of claims that they provide cost-saving services to customers. The condition of cloud resources varies according to utilization level and the extent of the gap between the computing capacities

The associate editor coordinating the review of this manuscript and approving it for publication was Chunsheng Zhu .

of aggregate VMs and physical machines (PMs). This gap represents a set of unutilized VMs called stagnant VMs. As can be seen, maximizing cloud revenue necessitates not only the adoption of an appropriate spot pricing model but also creation of an effective resource management model that can be optimized for spot markets.

The above-mentioned problems can be described as tantamount to finding the optimum mapping from VMs to PMs so as to reduce resource consumption and, hence, substantially augment total expected revenue. The key role of resource allocation is to adjust a given set of assets in response to the law of supply and demand, for the purpose of maximizing utilization levels on one hand and minimizing resource consumption on the other. These tasks are accomplished ultimately in service of expanded cloud profit. Tackling such challenging problems requires overcoming many significant obstacles, which include the following:

- Recognizing the underlying factors that affect supply in a cloud
- Estimating the prospective computing capacity for sale
- Selecting a favourable market in which to sell VMs related to prospective capacity
- Assessing the expected revenue from this capacity
- Finding an allocated VM algorithm that maximizes cloud revenue

Most studies regarding cloud resource management investigated this issue as a general supply chain optimization problem and failed to highlight the underlying relationship between resource management and pricing as well as the technical factors behind supply operations. For instance, [5] made the best use of each running PM to reduce the consumption of cloud power, but all the cloud's resources were handled under the same conditions. Likewise, [6] sought to save as much power as possible by estimating the capacity required in a succeeding time interval, subject to a probabilistic SLA. However the proposed strategy is suitable only for a private cloud and a certain form of contract.

Motivated by the need to address the above mentioned challenges and accordingly formulate techniques that can be used to improve resource management, we conducted this research with a view to providing the following contributions:

- We put forward a new systematic approach to managing spare cloud resources to expand cloud revenue. The proposed method, which we call the dynamic resource allocation for spot markets (DRASM), does not rely on reducing power consumption in the cloud by maximizing use in each host or estimating the capacity demanded by customers. Instead, our approach uniquely leverages the positive differences between the cost and revenue associated with active resources.
- We characterized the relationships between VM virtualization and migration on one hand and between VM virtualization and VM spot markets on the other hand. Identifying and analyzing these relationships contributes to enhancing spot pricing schemes, such as those adopted by [4] and [7].

- We validated our approach by comparing its performance with those presented by well-known research in this area, namely [8].

The rest of the paper is organized as follows: Section II reviews the related literature, and Section III describes the proposed model. Section IV introduces the technical specifications and simulation tool used in this work, after which it discusses the numerical results. Section V examines the effect of assumptions. Section VI concludes the paper.

II. RELATED WORK

This section presents and discusses works related to resource allocation and management for IaaS cloud resources such as VMs. Lee *et al.* [9] proposed a priority-based resource scheduling algorithm called the dynamic priority scheduling algorithm (DPSA) to handle service request scheduling problem. In DPSA, user requests are received, analyzed, and categorized on the basis of particular requirements into task units for the direct scheduling of adequate resources and the provision of effective services that accord with user demands. Mandal and Khilar [10] put forward a VM scheduling algorithm to reduce the time spent on VM allocation to a server and optimize resource utilization. The algorithm represents a list of resources in a binary search tree (BST) instead of presenting them in a queue. The use of the algorithm involves creating a BST for VM specification and sending this BST to a VM scheduler. The BST also contains a list of servers. Using the BSTs of the servers and VMs, the VM scheduler then selects the VM with the maximum requirements and searches for a server that best satisfies the aforementioned machine's needs. Beloglazov and Buyya [11] developed a modified best fit decrease scheduling algorithm for VM resource reallocation, in which all VMs are sorted in descending order with respect to the current utilization of a central processing unit (CPU). Each VM is allocated to a host that generates the least increase in power consumption due to the allocation. This heuristic algorithm, which is based on the traditional greedy algorithm, can optimize VM allocation, but it easily reverts to the local optimum and difficultly achieves the global optimum with a single point of a search strategy. Teng and Magoules [12] created an equilibrium-based resource scheduling technique that predicts the prospective prices of resources under the absence of bidding information on competitors. In this approach, the proportion of Nash equilibrium allocation is received by users; deadlines are fulfilled, and budget constraints are addressed through the implementation of the technique on CloudSim. Tomás and Tordsson [13] proposed a VM placement framework that involves monitoring and profiling applications to predict their behaviours and the types of resource usage in which they engage. The best location for deploying an application is determined via a smart overbooking scheduler. This approach is more suitable for a cloud environment that offers software as a service (SaaS).

Xiao *et al.* [14] developed a virtualization-based dynamic resource allocation mechanism that improves data

center utilization. Resource prediction is grounded in the past behaviors of VMs and the use of the exponentially weighted moving average (EWMA). Then, a skewness algorithm is employed to estimate disproportion in the multidimensional use of a processor through hotspot mitigation. This approach performs effectively in hotspot migration and load balancing but does not allow for live migration. Reference [15] proposed a joint VM provisioning approach in which multiple VMs are consolidated and provisioned on the basis of an estimation of their aggregate resource needs. In designing this method, the authors exploited statistical multiplexing among the workload patterns of multiple VMs. Reference [16] provided Stackelberg game to model the pricing and resource allocation problem between the IaaS and SaaS cloud providers. The study concluded that the dynamic pricing mechanism is more profitable than the fixed pricing for IaaS and SaaS providers alike. Moreover, the SLA is a major factor in the model, and IaaS cloud resources are fully available for SaaS, the main purpose of which is to increase social welfare. All previous works sought to manage cloud resources considered merely identical characteristics for all VMs from an engineering perspective, namely cloud resources that provide VMs tailored for quality of service (QoS) and SLA. Our approach classifies VMs into two categories (R-VMs and S-VMs) according to their characteristics, and hence, to the appropriate pricing strategy and market. The proposed dynamic resource allocation for the spot markets (DRASM) strives to maximize marginal revenue for cloud providers targeting the spot market.

III. PROPOSED APPROACH

Cloud resource management (or resource allocation) that considers the entirety of cloud resources has been extensively researched [13], [17], [18], and [19]. Therefore, we confirm here that the proposed model is not intended to increase the revenue of the entire resources of cloud, but rather to take advantage of active sources that consume expenses. This is called increasing the marginal revenue in the economic literature. This section vividly discusses the various aspects of our cloud resource management via spot pricing model, which is anchored in the spare IaaS cloud resources regarded as targets in this work.

Spare IaaS resources are non-virtualized and unutilized capacities of powered-on resources, that is, capacities that incur costs as they are operated. We refer to these resources as stagnant resources (or stagnant capacities) in this paper. Stagnant capacities are typically distributed over multiple classes of hosts (PMs) that are each made up of many components [e.g., CPU, random access memory (RAM), storage devices, network interface cards, basic input/output system]. These resources are created by imposed conditions, namely, those stipulated in SLAs or features that are provided to customers. In other words, stagnant capacities can be considered the “side effects” of elasticity, reliability, availability, and security. Studies indicated that these unused resources are, for most of their lifecycle, idle without any revenue generated

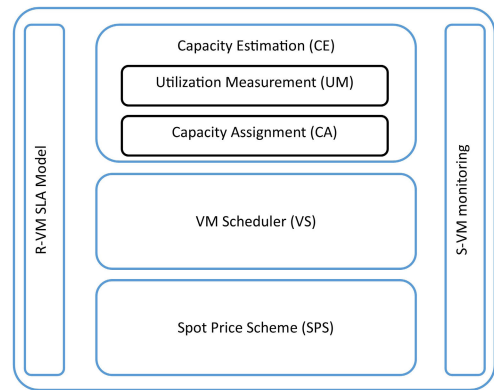


FIGURE 1. Block diagram of dynamic resource allocation composed of Capacity Estimation, VM Scheduler, R-VM SLA model, S-VM Monitoring and Spot Price Scheme.

from them; the majority of cloud resources generally suffer from underutilization [13], [20], and [21]. Amazon, a pioneer of spot market-based cloud computing, invented a dynamic pricing scheme for its spot instances to drive resource use to high levels [20]. In a similar vein, our model is designed to identify optimum ways of resource management for stagnant capacities on the grounds of spot pricing and with a view to earning the most substantial cloud revenue possible. We emphasize that the proposed dynamic resource management method is meant exclusively for stagnant resources through which the greatest return is earned in a spot market. Achieving this goal necessitates addressing two main issues:

- When to activate the model for the reconfiguration of sources
- How to manage both hosts and VMs, that is, the transition from current configurations to new ones

As illustrated in the block diagram in Fig. 1, our proposed dynamic resource allocation approach is composed of three core components. In brief, the approach put forward in this work is implemented by estimating the accumulated capacity of all hosts through a capacity estimation (CE) unit. This unit accomplishes estimation in two stages. The first is carried out by the utilization measurement (UM) unit, where usage levels are determined. The second stage involves mapping the optimal distribution of VMs on a cluster’s hosts through a capacity assignment (CA) unit. Subsequently, a VM scheduler (VS) allocates VM combinations that achieve high marginal revenue. The VS unit represents the core of our model. It specifies the host that must be set in sleep mode if an increase in stagnant capacity occurs, and it selects the VM that must be terminated if a shortage in stagnant capacity arises. The VS unit also prompts the CE unit to generate a new distribution, namely, a new allocation map, for VMs to obtain optimal allocation whenever necessary. Finally, a spot price scheme (SPS) refers to the optimum decision, migration, provision, or elimination of VMs when applicable. The CE reacts to a predetermined change in stagnant capacity to estimate the stagnant candidate capacity from all PMs. The VS decides to turn some PMs into sleep mode to shut down later to reduce expenses or supply their resources for the spot market,

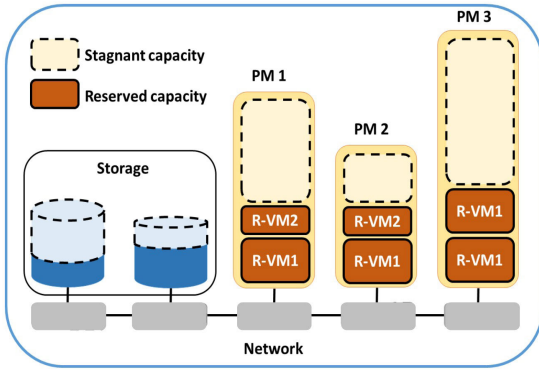


FIGURE 2. State of the cloud after deploying R-VMs.

depending on the rewards earned. The VS accomplishes its mission by cooperating with the SPS unit that determines the dynamic price of each type of S-VM in the spot market. Details regarding the proposed approach are as follows.

A. CAPACITY ESTIMATION

To estimate the available spare capacity that can be supplied to a spot market, we adopted the technique in [15], which relies on aggregate capacity for this purpose. However, [15] applied the aggregate capacity technique for VMs that depend on reserved or on-demand pricing schemes. That is, reserved VMs are used in conjunction with SLAs to supply the needs of a static price market. We refer to such VMs as R-VMs to distinguish them from spot VMs that are called S-VMs. S-VMs are characterized by dynamic pricing and are not bound by SLAs. They can therefore be terminated at any time, and the resources that they hold can be relinquished to R-VMs. To achieve this target, we split the task into two phases. In the first, UM is implemented to scan all hosts and calculate the usage level of each host, in addition to the aggregate utilization level of all hosts in a cluster. These measurements are then used in the second phase, CA, to detect the optimal quantity and allocation of S-VMs throughout the cluster.

The state of a cloud is depicted in Fig. 2. The state of cloud resources that are currently powered on shows that R-VMs consume part of these resources, and the rest are represented as unutilized assets, as denoted by the dotted line border in Fig. 2. The CE unit estimates and applies stagnant resources for a spot market. Note that managing cloud resources for a reserved market is beyond the scope of this research. That is, R-VM distribution is not covered by our system, and CE reacts to any change in stagnant capacity that occurs from the management of R-VMs and S-VMs alike. In other words, alterations to stagnant capacity can happen either to power on a new VM or to power off VMs for customers from both reserved and spot markets. The former reduces stagnant capacity, whereas the latter increases it.

Throughout this section, let κ denote the set of hosts in a cluster, where the number of hosts is $N \triangleq |\kappa|$, and let β represent the set of VMs located in a single host, where the number of VMs is $M \triangleq |\beta|$. For each host $i \in \kappa$,

let $c_{ij}^r(t)$ refer to the average utilization level of virtual machine R-VM_j that is located in physical machine PM_i at time t , the term utilization refers to CPU utilization. On the basis of these definitions, the UM unit calculates capacity, that is, the average utilization level, consumed by all R-VMs that are located in PM_i as follows:

$$rClC_i^r(t) = \frac{\sum_{j=1}^M c_{ij}^r(t)}{M}. \tag{1}$$

Likewise, the unit calculates the stagnant capacity of each powered-on PM_i and the aggregate stagnant capacity of all powered-on hosts in the cluster thus:

$$C_i^s(t) = C_i - MC_i^r(t) \tag{2}$$

and

$$C^s(t) = \sum_{i=1}^N C_i^s(t), \tag{3}$$

where C_i is the entire capacity of PM_i.

That is, $C_i^s(t)$ refers to the aggregate stagnant capacity for PM_i and $C^s(t)$ represents the aggregate remaining capacity from all compatible PMs after reserved instances R-VMs consume $C^r(t)$ out of the entire capacity.

The problem concerning packing m S-VMs into stagnant capacity $C^s(t)$ that is distributed on N host can be formulated as a knapsack problem, for which the goal is to maximize revenue f by provisioning the most worthy S-VMs. That is,

$$\begin{aligned} \max f &= \sum_{i=1}^N \sum_{j=1}^M p_j y_{ij} \\ \text{s.t. } &\sum_{j=1}^M c_j^s y_{ij} \leq C_i^s \quad \forall i \\ &\sum_{i=1}^N \sum_{j=1}^M c_j^s y_{ij} > C_i \quad \forall i \\ &y_{ij} \in Z_0^+ \end{aligned} \tag{4}$$

where

$$y_{ij} = \begin{cases} \text{number of S-VM}_j & \text{that are provisioned into host } i \\ 0 & \text{otherwise,} \end{cases}$$

To achieve the most revenue from $C^s(t)$, each capacity $C_i^s(t)$ must be packed with the S-VM_j that features the most profitable price p_j and acceptable capacity c_j^s . The first constraint maintains the stagnant capacity of each host i , whereas the second restricts the entire stagnant capacity offered to the spot market. We supposed that S-VM_j's capacity c_j^s has two dimensions, which represent computing capacity and memory capacity. Without loss of generality, we assumed that S-VMs are sorted so that

$$\frac{p_1}{c_1^s} \geq \frac{p_2}{c_2^s} \geq \dots \geq \frac{p_m}{c_m^s}.$$

Correspondingly, to solve such an integer linear programming (ILP) problem, particularly in a way that sustains large packages, the CA unit employs a column generation approach,

as was done in [4] and [22]. More specifically, a branch and bound technique is adopted. The use of the column generation approach in resolving this issue emphasizes its necessity. This scheduling problem is NP-hard, and reaching the global optimum point of an ILP is unsustainable and time-consuming. The technique works by decomposing the problem into two sub-problems—a master LP problem and an auxiliary LP problem (referred to as the pricing problem). Initially, a possible set of patterns must be identified. These patterns refer to a distinct combination of complementary S-VMs from each class that each PM can host in a cluster. Subsequently, the pricing problem suggests new promising candidates or patterns for the master problem. If the newly proposed patterns improve the objective underlying the master problem, they will be appended to the pattern set as a new column. Interested readers are referred to [4] and [23] for more details on the aforementioned column generation technique. Algorithm 1 illustrates the detailed steps taken by the CE Unit to achieve its purpose. The outcomes of CE are fed to the SPS unit to assess the spot price of each S-VM in the next horizon, namely, the number of S-VMs from each class that can be involved in the spot market. It is worth mentioning that each cluster consists of compatible PMs regardless of their capabilities (e.g., Hz for CPU and byte for RAM). The intention is to alleviate complexity where the proposed approach can work for inhomogeneous clusters separately.

Algorithm 1 Estimating S-VMs

```

1: Initialize:
2:  $C_i^f(t) = 0$  for all PMs
3: for each PM  $i, i = \{1, \dots, N\}$  do
4:   for each R-VM  $j, j = \{1, \dots, M\}$  do
5:      $C_i^r(t) = C_i^r(t) + c_{ij}^r(t)$ 
6:   end for
7:    $C_i^f(t) = C_i^f(t)/M$ 
8: end for
9: for each PM  $i, i = \{1, \dots, N\}$  do
10:   $C_i^s(t) = C_i - MC_i^r(t)$ 
11: end for
S-MAP
12:  $\leftarrow$  Using (4), find the optimum number and allocation
    for S-VMs
13: return S-MAP
  
```

As mentioned previously, this capacity results from technical constraints and the need to satisfy customer requirements. One of the most crucial of such requirements is the SLA between parties. Therefore, the manner by which $C^s(t)$ capacity is sold in a spot market and its use through S-VMs must not violate the SLA that governs R-VMs.

B. R-VM SLA MODEL

We define an *SLA-violate* function $V(\cdot)$ that is used as basis for determining the possibility of any potential violation of an R-VM SLA as a result of abandoning part of current capacity

to reduce operational expenses. An SLA violation occurs when the following inequality holds for PM_i :

$$rCIC^s(t+1) \geq (C^s(t) - C_i)\alpha \quad (5)$$

where $\alpha \in [0, 1]$ denotes a threshold parameter imposed by an SLA, and $C^s(t+1)$ refers to the new stagnant capacity in the succeeding period after the capacity of PM_i is dispensed. Therefore,

$$V(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

through which the S-VM scheduler can decide to arrange any PM to sleep mode. If the PM switches to sleep mode to a shutdown later, the new stagnant capacity will decrease in the next time interval by subtracting the capacity of PM from the current stagnant capacitance. The model provides a percentage, defined by α , of new resulting stagnant capacity to the spot market rather than providing the full stagnant capacity to maintain SLA and QoS for R-VMs. In other words, the new stagnant capacity should not exceed a certain limit, especially in case of high demand and workload for R-VMs in reserved markets.

C. S-VM SCHEDULER

To use stagnant resources and derive the highest expected return, we divided the VS into two separate session decisions, with the first decision referring to a computing session and the subsequent decision pertaining to a networking session. Any noticeable change in stagnant capacity initiates the VS unit to configure and adjust both PMs and VMs to the optimum status from a cloud provider's perspective; here, "noticeable change" is equivalent to the minimum-size class of VM and is called k in Algorithm 2. The computing session elects candidate hosts and S-VMs for termination in consideration of the governing SLA. Hence, the networking session chooses the most convenient host from the candidates in accordance with the status of network congestion. Note that work during the networking session is deferred for future operations. We therefore assumed that networking resources are smoothly available. Detailed information on VS tasks and the corresponding technique is discussed in the following subsections. Changing aggregate stagnant capacity $C^s(t)$ refers to the demands and deals that are held under the auspices of a reserved market in time t .

First, releasing or terminating an R-VM in a reserved market increases the $C^s(t)$ amount, which in turn, induces the VS to search for any possible host, PM, without neither R-VM nor S-VM consuming its computing power, if any. If such a PM is captured, whether it can be discarded from a set of stagnant resources is determined to reduce the expenses incurred by a cluster. The rationale of this decision is that it sets the proposed PM to sleep mode for later power off if the abandonment of this redundant capacity does not adversely affect the SLA of R-VM customers. Otherwise, this PM is retained to supply a spot market. Providing a spot market with

Algorithm 2 Scheduling S-VMs

```

1: Given:
2:  $C^s(t)$ : the current aggregate stagnant capacity
3:  $C^s(t-1)$ : the aggregate stagnant capacity at time t-1
4: Monitoring: if the change exceeds the threshold  $k$ 
5: while ( $|C^s(t) - C^s(t-1)| \geq k$ ) do
6:   if ( $C^s(t) - C^s(t-1) > 0$ ) then
7:     if (Is there any PM without any VM) then
8:       if (terminating the PM violates the SLA) then
9:         Keep the PM and initiate Algorithm 1
10:      else
11:        Convert the PM to the sleep mode
12:      end if
13:    else
14:      if (is there any PM with only S-VMs) then
15:        if PM's revenue < PM's cost then
16:          if (terminating the PM violates the SLA)
17:            then
18:              Keep the PM and initiate Algorithm 1
19:            else
20:              Terminate all PMs' S-VMs
21:              Convert the PM to the sleep mode
22:            end if
23:          end if
24:          Keep the PM and initiate Algorithm 1
25:        end if
26:      else
27:        Initiate Algorithm 3
28:        Initiate Algorithm 1
29:      end if
30:    end while

```

additional capacity stimulates the CE unit to compute (1), (2), and (4) using Algorithm 1. Such calculation is intended to estimate and pack potential S-VMs. Conversely, when the VS reaches a PM that hosts only S-VMs, the VS evaluates the revenue generated from hosting these S-VMs and then preserves the PM in its current state and activates the CE unit to estimate new possible S-VMs when the approximated revenue exceeds the host's operational costs. However, if the return gained from these S-VMs is insufficient (i.e., the profit is non-positive), the VS checks whether reducing stagnant capacity by placing the PM in sleep mode has negative effects on an SLA. The negative effects on an SLA prevents the VS from abandoning the PM. In the reverse case, the VS retains the targeted PM and again initiates Algorithm 1 to update the S-MAP.

Second, decreasing $C^s(t)$ refers to a situation wherein a new R-VM is provisioned in a reserved market or to a host, PM, in which failure prevents the availability of the host's resources. This incident drives the VS to notify the S-VM monitoring unit to terminate S-VMs that compete against the R-VMs on a PM's resources. The selection process for

Algorithm 3 S-VM Monitoring

```

1: for each PM  $i, i = \{1, \dots, N\}$  do
2:    $L_i \leftarrow$  workload for each PM $_i$ 
3: end for
4: for each PM  $i, i = \{1, \dots, N\}$  do
5:   while ( $L_i > C_i\alpha$ ) do
6:     for each S-VM  $j, j = \{1, \dots, \bar{m}\}$  do
7:       TS-VM  $\leftarrow$  S-VM with minimum price
8:     end for
9:     Terminate TS-VM
10:   end while
11: end for

```

such termination is based on price, wherein the lowest priced S-VM is terminated. To this end, the S-VM monitoring unit applies Algorithm 3, after which the VS notifies the CE unit to reconfigure the stagnant capacity and relocate the pool of S-VMs over to PMs to achieve optimal placement.

D. S-VM MONITORING

The S-VM monitoring unit is responsible for keeping track of the workload and utilization of each host in a cloud. If the workload of any host exceeds a predefined threshold, this unit first terminates the S-VM that generates the lowest revenue to reduce the workload and so on until the workload of the target PM falls in the allowable range. If the price of a new request for S-VM exceeds the current assigned price, and stagnant capacity is available, the request is fulfilled. Otherwise, the S-VM Monitoring unit terminates the lower price S-VM and fulfills the higher price S-VM request. The rationale behind this step is the allocation of PM resources to worthy S-VMs, that is, those that generate high revenue, or to R-VMs in case of PM failures or maintenance.

E. SPOT PRICE SCHEME

The main purpose of the SPS unit is to assign an initial dynamic price to an S-VM in a spot market. First, the CE provides an estimated volume for each capacity c_j^s , and SPS unit uses this estimate to set initial price p_j . The initial price is determined on the basis of the average of many values obtained from the SPS unit for the same capacity state, where initial price p_j is inversely proportional to stagnant capacity. Note that there is no known distribution of VM [24] prices specifically in a spot market. This absence is attributed to the fact that the spot price is actually the bidding price, which is influenced by the information available to both consumers and providers. Nevertheless, the SPS unit scales the price up and down on the grounds of supply and demand. To accomplish this task, we employed the dynamic pricing scheme from [25] to control prices over the horizon; the time look ahead is one hour. The dynamic pricing scheme is formulated as a finite horizon stochastic Markov decision process.

On the other hand, the VS induces the SPS unit to provide updated spot prices for the purpose of terminating undesirable S-VMs and freeing up space for the most profitable S-VMs.

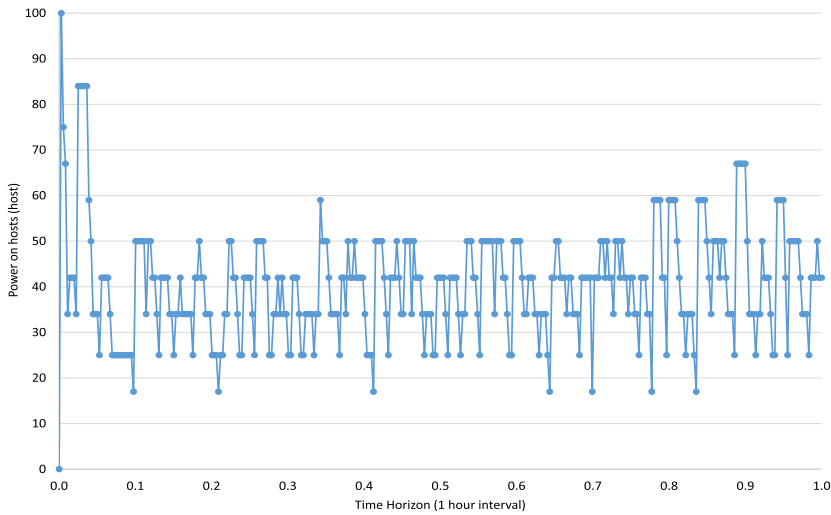


FIGURE 3. Active hosts.

TABLE 1. Hosts specifications.

Host	Features
HP ProLiant ML110 G4	Xeon 3040 1860 MHz, 2 cores, 4 GB
HP ProLiant ML110 G5	Xeon 3075 2660 MHz, 2 cores, 4 GB

TABLE 2. VMs specifications.

Virtual machine	MIPS	RAM
S-VM1	2500	1870
S-VM2	2000	1740
S-VM3	1000	1740
S-VM4	500	613

IV. EXPERIMENTS

This section discusses the empirical evaluation findings. We closely followed the steps and techniques involved in DRASM that were discussed in Section III to demonstrate the effects of dynamic allocation in view of aggregate capacity.

A. TECHNICAL SPECIFICATIONS

In this assessment, we implement Cloudsim [26] as a simulation tool and modelling framework for our provisioning technique. Further, we use a workload dataset from Google’s Compute Clusters [27], which represent resource consumption for both CPU and memory for seven hours. The cloud data center is characterized by 100 hosts (or PMs) and a dynamic workload normally distributed with mean value equal to the normal workload in real data centers. Without loss of generality, we assume that the cloud contains two types of hosts and the specifications for the hosts are given in Table 1. The PMs consolidate 100 VMs of four classes with different configurations, as shown in Table 2. We also considered the spot prices of Amazon T2 spot instances and set all bidding prices for S-VMs as equal to or greater than setting-up prices.

B. NUMERICAL RESULTS

Figure 3 illustrates the findings derived from an average of 10 executions for our model using CloudSim. The results showed that the average number of powered-on hosts within an hour’s interval is approximately 41, but this figure at

the beginning of the period is 100. This finding indicates a capacity savings of 59%. An interesting point about the results is that the significant savings originates not only from the S-VMs pool but also from the activities of R-VMs, with the latter consuming a mere 25% of cloud capacity. Reference [15] demonstrated that the average capacity savings is around 40%, but this calculation takes into account only R-VMs. This difference is attributed to the fact that S-VMs are volatile (when a host contains highly loaded R-VMs) compared with R-VMs. Let us move on to the utilization levels of powered-on hosts. The result on the monitoring of host utilization is shown in Figure 4, which reflects the average utilization of all hosts (i.e., Each time slot represents the utilization of 100 hosts, from host-1 to host-100.) within one hour. A high utilization level is observed at $t = 0.16$, $t = 0.27$, and $t = 0.64$, indicating that some of the hosts are overloaded. The explanation for this result is that some S-VMs attempt to consume a much greater quantity of their aggregate capacity, yet the individual capacities of these S-VMs enable them to consume more resources than that allowable in aggregated capacity. In subsequent time slots, utilization levels fall below 80%, thereby validating our algorithm. That is, the cloud must terminate the least profitable S-VM to release more capacities for use by either R-VMs or S-VMs. Furthermore, we observed that utilization levels peak under a small number of hosts (e.g., between 25 and 50 hosts). This case reflects that a cloud provider reduces the resources supplied to a spot market.

Figure 5 illustrates the distribution of S-VM classes over cloud hosts during the analysis period (one hour). In order to clarify the comparison process, we performed this experiment using 12 hosts and 12 R-VMs. In an effort to maximize the marginal revenue of a cloud through spot marketing, we computed the capacity available for each host, then identified the best hosting configurations for all S-VM classes on the basis of price. The outcome showed that at time $t = 0.10$, the most favorable hosting configurations are 13 units of S-VM1, two units of S-VM2, three units of S-VM3, and three units

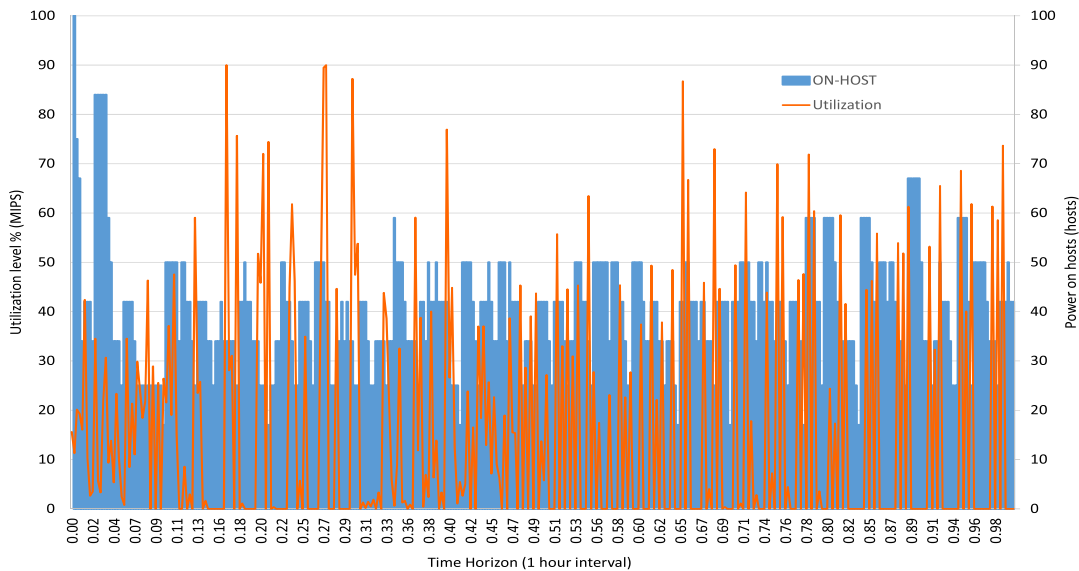


FIGURE 4. Utilization levels.

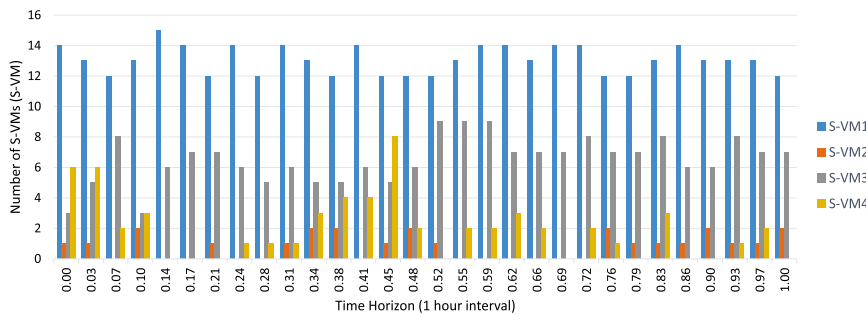


FIGURE 5. S-VMs hosting configurations.

of S-VM4. At time $t = 0.17$, the best hosting configurations are 14 units of S-VM1 and seven units of S-VM3.

C. PERFORMANCE EVALUATION

We do not claim that the aggregate capacity technique and dynamic resource allocation are contributions of this research. Nevertheless, we deemed it necessary to pinpoint that to the best of our knowledge, our strategy for dynamic resource allocation in a cloud is the first that considers separate categories of VMs (S-VMs and R-VMs). Therefore, comparisons of performance is not possible at the current time. Consequently, we could evaluate DRASM only from the utilization perspective of all VM categories. We thus compared DRASM with the MPC presented by [8].

For this purpose, we used a workload dataset from Google’s Compute Clusters [27], as mentioned earlier, which represent resource consumption for both CPU and memory for seven hours. We simulated IaaS cloud infrastructure that is similar to that used to implement MPC, which consists of 7000 PMs that all contain identical CPUs of 10640 MHz capacity. We also adopted three classes of VMs that match the specifications in [8]. Figure 6 depicts the first hour of utilization for all VM classes on the basis of the servicing of 173,751 distinct requests for VMs. At time $t = 0.5$,

the average utilization levels are 53% and 39% for MPC and DRASM, respectively. This variance is ascribed to MPC’s allocation of fixed capacity to each class of VMs over the established horizon compared with DRASM’s allocation of aggregate capacity to each class of VMs. Another interesting point about these findings is that in DRASM, 3000 PMs were placed in sleep mode during the experiment; hence, the utilization of 39% refers to usage by 4000 hosts instead of 7000 hosts, as in the experience with MPC. After time $t = 0.7$, however, the utilization of both approaches reach very close levels a status that occurs when each VM is highly overloaded. In other words, the aggregate capacity of VMs is equates with the cumulative maximum capacity of such machines. Indeed, the number of hosts and their utilization level has an essential role in reducing hosting costs. Neutralizing 3000 hosts results in 53% energy saving. In addition, [28], referring to the relationship between the utilization level and the energy consumption of the HP ProLiant G5, indicates that 30% and 50% of utilization incur 105 and 116 watt/h respectively. Finally it should be noted that the indicated energy is only due to the host’s consumption regardless of any other energy consumed in the data center such as cooling and others.

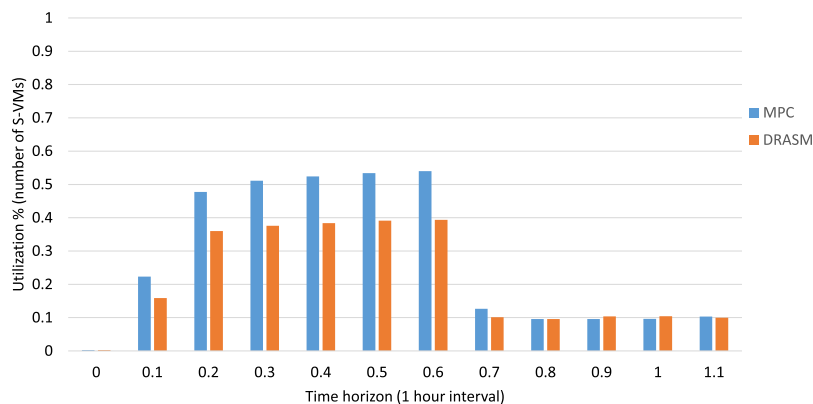


FIGURE 6. DRASM vs MPC, the Utilization depicts the proportion of S-VMs operating in the cluster.

V. THREATS TO VALIDITY

It is obvious that the experiments' findings have achieved substantial progress and opened the door to new insights in managing profitable cloud sources. The findings result from feeding DRASM with real requests and workload. However, it must be noted here that the proposed approach has been restricted by some assumptions. It is important to clarify the extent and sensitivity of these assumptions to the results. The model adopts aggregate stagnant capacity, as indicated in Equation 2, which is based on average utilization levels. However, at a very high peak load, the proposed stagnant capacity exceeds the actual stagnant capacity. To avoid the effect of such circumstances, the cloud should estimate an appropriate value for the threshold parameter α in Equation 5. Monitoring the history of the α values over thoughtful time periods or using other prediction methods to estimate the exact value of α improves system efficiency. Moreover, VS decisions, in Section IV-C, only relate to computational resources. Despite the critical importance of computational resources in the cloud, network resources also have an impact on imperial evaluations (e.g., choosing a PM with a high level of network congestion requires a queue, another PM choice, or the use of hardware technique). The inclusion of the network congestion issue requires additional network-level consideration that has been deferred for future work.

VI. CONCLUSION

This research presented a new dynamic resource allocation approach for spot markets called DRASM. The study was aimed at minimizing hosting costs and maximizing cloud revenue through the creation of an efficient resource management technique that consolidates a spot market's VMs. First, the CE unit determines the prospective capacity for sale in a spot market, after which it creates the optimal mapping between S-VMs and PMs in view of revenue generation. This mapping results from applying the column generation approach. Second, the VS unit monitors and controls the S-VM pool to increase the utilization of a cloud's hosts and reduce hosting costs. This work leveraged and applied several techniques to fulfil the desired tasks, aggregate capacity, dynamic resource allocation, and dynamic price.

The empirical evaluations based on actual requests and workloads exhibited promising hosting savings. As for future work, we have initiated efforts to realize dynamic collaboration between dynamic resource management and dynamic pricing schemes for spot markets on one hand and resource management for reserved markets on the other.

REFERENCES

- [1] *Amazon EC2 Pricing*. Amazon Web Services Inc. Seattle, WA, USA, 98108-1226. Accessed: May 2018. [Online]. Available: <http://aws.amazon.com/ec2/pricing/>
- [2] *Azure Pricing*. Microsoft Azure. Redmond, WA, USA, 98052-6399. Accessed: May 2018. [Online]. Available: <https://azure.microsoft.com/en-gb/pricing/details/virtual-machines/linux/>
- [3] *Preemptible Virtual Machines*. Google Inc. Mountain View, CA, USA. Accessed: May 2018. [Online]. Available: <https://cloud.google.com/preemptible-vms/>
- [4] F. Alzhouri, A. Agarwal, Y. Liu, and A. S. Bataineh, "Dynamic pricing for maximizing cloud revenue: A column generation approach," in *Proc. 18th Int. Conf. Distrib. Comput. Netw.*, 2017, p. 22.
- [5] A. J. Younge, G. von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, "Efficient resource management for cloud computing environments," in *Proc. Int. Conf. Green Comput.*, Aug. 2010, pp. 357–364.
- [6] R. Wolski and J. Brevik, "QPRED: Using quantile predictions to improve power usage for private clouds," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 179–187.
- [7] F. Alzhouri and A. Agarwal, "Dynamic pricing scheme: Towards cloud revenue maximization," in *Proc. IEEE 7th Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Nov. 2015, pp. 168–173.
- [8] Q. Zhang, Q. Zhu, and R. Boutaba, "Dynamic resource allocation for spot markets in cloud computing environments," in *Proc. 4th IEEE Int. Conf. Utility Cloud Comput.*, Dec. 2011, pp. 178–185.
- [9] Z. Lee, Y. Wang, and W. Zhou, "A dynamic priority scheduling algorithm on service request scheduling in cloud computing," in *Proc. Int. Conf. Electron. Mech. Eng. Inf. Technol.*, Aug. 2011, pp. 4665–4669.
- [10] S. K. Mandal and P. M. Khilar, "Efficient virtual machine placement for on-demand access to infrastructure resources in cloud computing," *Int. J. Comput. Appl.*, vol. 68, no. 12, pp. 6–11, 2013.
- [11] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1366–1379, Jul. 2013.
- [12] F. Teng and F. Magoulés, "Resource pricing and equilibrium allocation policy in cloud computing," in *Proc. 10th IEEE Int. Conf. Comput. Inf. Technol.*, Jun. 2010, pp. 195–202.
- [13] L. Tomás and J. Tordsson, "Improving cloud infrastructure utilization through overbooking," in *Proc. ACM Cloud Autonomic Comput. Conf. (CAC)*, 2013, pp. 1–10.
- [14] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.

- [15] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," in *Proc. 7th Int. Conf. Autonomic Comput. (ICAC)*, 2010, pp. 11–20.
- [16] Z. Zhu, J. Peng, K. Liu, and X. Zhang, "A game-based resource pricing and allocation mechanism for profit maximization in cloud computing," *Soft Comput.*, vol. 24, no. 6, pp. 4191–4203, Mar. 2020.
- [17] C. Papagianni, A. Leivadreas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1060–1071, Jun. 2013.
- [18] E. Oppong, S. Khaddaj, and H. E. Elastriss, "Cloud computing: Resource management and service allocation," in *Proc. 12th Int. Symp. Distrib. Comput. Appl. Bus., Eng. Sci.*, Sep. 2013, pp. 142–145.
- [19] H. Xu and B. Li, "Anchor: A versatile and efficient framework for resource management in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1066–1076, Jun. 2013.
- [20] *Cloud Computing, Server Utilization, & the Environment*. Amazon Web Services Inc. Seattle, WA, USA, 98108-1226. Accessed: Apr. 2018. [Online]. Available: <https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment/>
- [21] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," Berkeley, CA, USA, Tech. Rep. UCB/EICS-2009-28, 2009.
- [22] S. Vakili, B. Heidarpour, and M. Cheriet, "Energy efficient resource allocation in cloud computing environments," *IEEE Access*, vol. 4, pp. 8544–8557, 2016.
- [23] J. V. de Carvalho, "LP models for bin packing and cutting stock problems," *Eur. J. Oper. Res.*, vol. 141, pp. 253–273, Sep. 2002.
- [24] B. Javadi, R. K. Thulasiramy, and R. Buyya, "Statistical modeling of spot instance prices in public cloud environments," in *Proc. 4th IEEE Int. Conf. Utility Cloud Comput.*, Dec. 2011, pp. 219–228.
- [25] F. Alzhouri, A. Agarwal, and Y. Liu, "Maximizing cloud revenue using dynamic pricing of multiple class virtual machines," *IEEE Trans. Cloud Comput.*, early access, Oct. 25, 2018, doi: [10.1109/TCC.2018.2878023](https://doi.org/10.1109/TCC.2018.2878023).
- [26] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.
- [27] J. L. Hellerstein. (Jan. 2010). *Google Cluster Data*. [Online]. Available: <http://googleresearch.blogspot.com/2010/01/google-cluster-data.html>
- [28] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Comput., Pract. Exper.*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012.



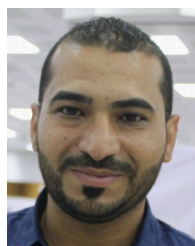
FADI ALZHOURI (Member, IEEE) received the M.Sc. degree in computer engineering from Kuwait University, in 2007, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada, in 2018. He studied on natural language processing and artificial intelligence. He is currently holding a postdoctoral position with the University of Ottawa. His research interests include cloud computing, the Internet of Things, artificial intelligence, operations research, and optimization.



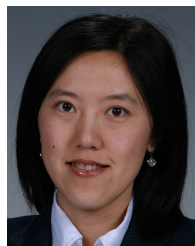
SUHIB BANI MELHEM received the bachelor's and M.Eng. degrees in computer engineering from the Jordan University of Science and Technology, Jordan, and the M.Eng. and Ph.D. degrees in electrical and computer engineering from Concordia University, Canada. He is currently a Postdoctoral Fellow with the Department of Electrical and Computer Science, York University, Toronto, ON, Canada, collaboration with the National Research Council Canada, Canada. His current research interests include cloud computing, resource management for virtual machine live migration, decision algorithms, cybersecurity for the Internet of Things, and load balancing for 5G small-cell networks.



ANJALI AGARWAL (Member, IEEE) received the B.E. degree in electronics and communication engineering from the Delhi College of Engineering, India, in 1983, the M.Sc. degree in electrical engineering from the University of Calgary, Alberta, in 1986, and the Ph.D. degree in electrical engineering from Concordia University, Montreal, in 1996. She has worked as a Lecturer with IIT Roorkee. She has also worked as a Protocol Design Engineer and a Software Engineer in industry. She is currently a Professor with the Department of Electrical and Computer Engineering, Concordia University. Her current research interests include cloud networks, heterogeneous networks, and wireless networks, including security and the virtualization of cognitive radio networks.



MUSTAFA DARAGHMEH received the bachelor's degree in computer science from Al-Balqa' Applied University, Al-Huson University, Irbid, Jordan, in 2009, and the master's degree in computer science from the Jordan University of Science and Technology, Irbid, in 2014. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Concordia University, Montreal, QC, Canada. His current research interests include cloud computing, resource management, and multiagent systems.



YAN LIU (Member, IEEE) received the Ph.D. degree from The University of Sydney, Australia. She was a Senior Scientist with the Pacific Northwest National Laboratory, Washington State, where she led Research and Development Projects on data intensive computing for a wide range of domains. She was a Senior Researcher with National ICT, Australia. She is currently an Associate Professor with the Electrical and Computer Engineering Department, Concordia University, Canada. Her research interests include big data stream processing, cloud software architecture, distributed computing, software performance engineering, and adaptive systems. She is a member of ACM.



SADEQ YOUNIS received the B.S. degree in engineering technology from Yarmouk University, Jordan, in 2011, and the M.S. degree in electrical and computer engineering from Concordia University, Montreal, QC, Canada, in 2013. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the Gina Cody School of Engineering and Computer Science, Concordia University. He is also a Researcher with the Cloud Computing Research Laboratory, Concordia University.

...