IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 14, 2018, accepted July 18, 2018, date of publication August 1, 2018, date of current version September 5, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2861462

Analyzing Energy-Efficiency of Two Scheduling Policies in Compute-Intensive Applications on Cloud

PING KUANG[1], WENXIA GUO[1], XIANG XU[1], HONGJIAN LI[1,3], WENHONG TIAN[1,2], (Member, IEEE), AND RAJKUMAR BUYYA[4], (Fellow, IEEE)

[1]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[2]Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China
[3]Department of Computer Science and Technology, Chongqing University of Post and Telecommunication, Chongqing 400065, China
[4]Department of Computer Science, The University of Melbourne, Melbourne, VIC 3010, Australia

Corresponding author: Wenhong Tian (tianwenhong@cigit.ac.cn and tian_wenhong@uestc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61672136, Grant 61650110513, and Grant 61602434, in part by the Science and Technology Plan of Sichuan Province under Grant 2016GZ0322, and in part by the West Light Plan of Chinese Academy of Science under Grant R51A150Z10.

ABSTRACT One of the key problems facing cloud applications is to reduce their energy consumption, which can increase the working lifetime of a machine, decrease the operation costs of cloud providers, and the environmental impact caused by power consumption. It is very important to design and evaluate an energy-efficient cloud. Recently, two open problems are raised in the literature: 1) what is the optimal solution (the lower bound) for the total energy consumption? and 2) what is the energy-efficiency for a scheduling algorithm? In this paper, we consider two major scheduling policies: 1) always power-on physical machines (PMs) once turning-on and 2) turning-off (hibernating) idle PMs, both with possible virtual machine migrations during evaluation. Focusing on compute-intensive applications on cloud, we propose analytical methods to settle down the two open problems. Our theoretical results are validated by experimental results in different scheduling scenarios and can be applied in cloud computing environments to help energy-efficient design.

INDEX TERMS Cloud data centers, energy-aware resource scheduling, the lower bound, energy efficiency, modified interval scheduling.

I. INTRODUCTION

Cloud computing is one of the key technologies to deploy various applications. It provides computational and storage network resources on demand. Although cloud computing paradigm is rapidly emerging, there exist several open challenges including energy-efficiency management of its resources. As the scale of cloud computing and data center increase, the cost and impact of energy consumption become very important issues. One of the key problems facing cloud applications is to reduce their energy consumption, which can increase the working lifetime of a machine, decrease the operation costs of cloud providers and the environmental impact caused by power consumption.

A simplified VM allocation process in cloud is presented in FIGURE 1. In Cloud, users can send requests through Internet or intranet. The procedure is as follows:

1). User's request initialization: user initiates a request through the Internet;

2). Finding suitable resources: based on the user's identity (such as geographic location, etc.) and the business characteristics (quantity and quality requirements), the schedule center submits the request to an appropriate physical machine(PM), in which the management program submits the request to a scheduling domain and a type of scheduling algorithm is executed and resource is allocated to the user;

3). Sending feedbacks (scheduled results) to users;

4). Scheduling the tasks: executing scheduling tasks and deploying resources;

5). Updating/Optimization: the schedule center executes optimization in the back-end and prepares resources in different PMs based on the optimization objective functions for later use.

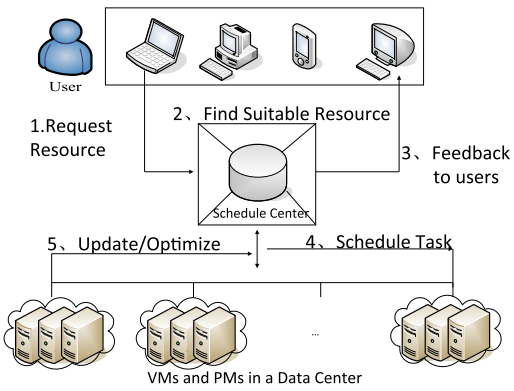

FIGURE 1. VM allocation process in the IaaS cloud.

A. RELATED WORK

This paper focus on compute-intensive applications of Infrastructure as a Service (IaaS) in cloud data centers. Related work in cloud is extensive. The following are closely related to our research. There are generally two types of VM scheduling algorithms, one is offline where a set of VM requests is known in advance and the scheduler can proceed these requests together, another type is online where jobs come one by one and the scheduler can only proceed one request each time. Beloglazov *et al.* [1] proposed offline allocation of VMs to minimize the total number of migrations through modified best-fit bin packing heuristics, which is considered one of the best algorithm for energy-efficiency and we will compare our proposed method against it. Guazzone *et al.* [4] considered two-level control model to automatically allocating resource to reduce energy consumptions for web-service applications, this approach can be discussed. Kolen *et al.* [8] modeled a real-time service as a real-time VM request, and used dynamic voltage frequency scaling schemes to provision VMs in cloud data centers, we will discuss online scheduling as real-time applications and use different algorithms for energy-efficient scheduling. Matthew *et al.* [9] proposed dynamic programming methods for minimizing total energy related costs for offline and online virtual machine allocation in Content Delivery Networks, their approaches are similar to some of our proposed methods. One of the difficulty scheduling problems in cloud data centers is to combine the allocation and migration of Virtual Machines (VMs) with full life cycle constraints, which is often neglected [6]. Flammini *et al.* [13] proposed offline algorithms to minimize the total busy time of all machines through turning-off idle servers, this is one of two scheduling policies we will discuss in this paper. Liu *et al.* [14] explored the balance between server energy consumption and network energy consumption to present an energy-aware joint virtual machine (VM) placement. These research laid foundation for our study. Tian *et al.* [20] consider scheduling parallel jobs formed by a set of independent tasks and consider energy consumption during scheduling, they proposed efficient methods for single job and multiple online jobs respectively by minimizing the total completion time (TCT), which is one of the metrics we also apply. Li *et al.* [21] proposed method for minimizing total busy time of parallel job offline scheduling and also provided trace-driven simulation results, which we can compare our methods in this paper easily. For modern server systems, I/O device and NUMA inter-node buses play important roles, as shown in studies [22], [23]. Tian *et al.* [24] introduce an algorithm to minimize total energy consumption by considering virtual machine reservations where approximation ratio is applied. Since this paper focuses on computing intensive applications, we consider CPU and memory especially with VM migrations in our energy consumption model. We will discuss in the future for other types of applications. Lefèvre and Orgerie [11] provided a framework for designing and evaluating an energy-efficient cloud with Green Open cloud (GOC) architecture. They showed experimental results from real tests without analytical models. They also raised two open problems:

(1) what is the optimal solution (the lower bound) for the total energy consumption?

(2) what is the energy efficiency for a scheduling algorithm?

These two problems are very important for energy consumption design and evaluation. In this paper, we propose analytical methods for evaluating power and energy consumption in compute-intensive application on cloud.

The main **contributions** of this paper are:

- Proposed analytical methods for evaluating total energy consumption for two major scheduling policies: (A). always power-on Physical Machines (PMs) once turning-on and (B).turning-off (hibernating) idle PMs during evaluation, both with possible virtual machine (VM) migrations.
- Offered analytical solutions for two open issues suggested in literature, one is for the lower bound of total energy consumption and another is for evaluating energy-efficiency of a given algorithm.
- Validated theoretical results for different scheduling scenarios by real environment test and simulation.

II. PROBLEM FORMULATION

The problem of real-time scheduling of VMs is modelled as a modified interval scheduling problem. There are detailed explanation and analysis on fixed interval scheduling problems in [7], [8] and references therein. We have following assumptions for our scheduling:

- 1). The required CPU capacity of each VM is a part of the total CPU capacity of a PM. We focus on compute-intensive applications, this is reasonable and applied already in some research.
- 2). The time is discrete in slotted windows. The total time period $[0, T]$ is slotted into equal length (l_0) in discrete time, thus the total number of slots is $K = T/l_0$ (forming a positive integer). The start-time is set as s_0, the interval of a request j is shown in slot format as

[StartTime, EndTime, RequestedCapacity]=[s_j, e_j, d_j] with both start-time s_j and end-time e_j being non-negative integers, d_j is the CPU capacity demand of a VM from a physical machine (PM).

- 3). For all jobs, there are no precedence constraints other than those implied by the start-time and end-time. Preemption is not considered in this paper.

The following key definitions are based on above assumptions:

Definition 1: Traditional fixed processing time interval scheduling. A request set $\{1, 2, \cdots, n\}$ is considered in which the i-th request has an interval of time starting at s_i and finishing at e_i. Each request occupies the whole capacity of a machine during fixed processing time. This paper considers that each request occupies a virtual machine (VM), this is an reasonable scenario. More results on traditional interval scheduling problem (ISP) can be found in Kleinberg *et al.* [5], Kovalyov *et al.* [7] and Kolen *et al.* [8].

Definition 2: Interval scheduling with capacity sharing (ISWCS). The key difference from traditional interval scheduling is that the resource can be shared by different VM requests if at any time the total capacity of all requests is not more than the total capacity the resource can offer.

The organization of remaining contents is: the power and energy consumption models are proposed respectively in Section III and IV; performance evaluation and our theoretical observations are in Section V; The Conclusions and Future Work are provided in Section VI.

III. THE POWER CONSUMPTION MODEL OF A PM

The most of research has found that the overall load of a PM is generally proportional to its CPU utilization (similar to Beloglazov *et al.* [1], Mathew *et al.* [9], Fan *et al.* [15], Mastroianni *et al.* [17] and references therein). This is valid for compute-intensive applications where CPU utilization dominates. This paper focuses on power and energy consumption caused by compute-intensive applications of PMs and VMs. **We consider homogeneous case that all PMs have same configurations unless otherwise specified**. As for heterogeneous case, we separate PMs by homogeneous groups so that homogeneous case can still be applied, this is shown in performance evaluation section. A power consumption model for a PM (or aVM) i is proposed [1], [9], [17]:

$$P_i(U_i(t)) = P_{min} + (P_{max} - P_{min})U_i(t) \tag{1}$$

where P_{max} is the power consumed when the PM is fully utilized, P_{min} is the power consumed when the PM is idle; and $U_i(t)$ is the CPU utilization at time t. There is also power consumption model for a blade server considering disk, memory and network [16]:

$$P_i(t) = 14.5 + 0.2U^t_{cpu} + 0.003U^t_{disk} + 4.5e^{-8}U^t_{mem} + 3.1e^{-8}U^t_{net} \tag{2}$$

One can see that CPU is the dominating factor for power consumption in Equ (2). There are also non-linear model and Dynamic Voltage Frequency Scaling (DVFS) model [6] for the power consumption. However, we concentrate on energy-efficiency analysis of compute-intensive applications so that we take power consumption model in Equ (1). Our results hold for the energy model which is a convex function with time. When using average CPU utilization U_i (arithmetic mean) during [0,t], the average power consumption can be represented as:

$$P_i(U_i) = P_{min} + (P_{max} - P_{min})U_i \tag{3}$$

IV. THE ENERGY CONSUMPTION MODEL OF A PM

Formally, the energy consumption of PM i during time interval $[t_0, t_1]$ can be computed by:

$$E_i = \int_{t_0}^{t_1} P_i(U_i(t))dt \tag{4}$$

When average CPU utilization and power model are applied, the energy consumption of a PM i during $[t_0, t_1]$ is:

$$\begin{aligned} E_i &= P_i(U_i)(t_1 - t_0) = P(U_i)t_i \\ &= P_{min}t_i + (P_{max} - P_{min})U_i t_i \end{aligned} \tag{5}$$

where $t_i = t_1 - t_0$ is the total power-on time of PM i, $P_{min}t_i$ is the energy consumption during the power-on time of PM i which we set as $P_{min}t_i = E_{i_{on}}$, and $(P_{max} - P_{min})U_i t_i$ is the energy consumption by hosting VMs on PM i. Considering that a VM j increases the total utilization of a PM i from U to U' with $U' - U = \Delta U$ and VM j is fully utilized (this is the worst case), then the energy increase ΔE_{ij} after hosting VM j on PM i from time t_0 to t_1 can be defined as:

$$\begin{aligned} \Delta E_{ij} &= (P_{min} + (P_{max} - P_{min})U' - (P_{min} \\ &\quad + (P_{max} - P_{min})U))(t_1 - t_0) \\ &= (P_{max} - P_{min})(U' - U)(t_1 - t_0) \\ &= (P_{max} - P_{min})\Delta U(t_1 - t_0) \end{aligned} \tag{6}$$

Considering the additional energy consumption of VM j migration at slot k. Denote the total number of VM migrations in slot k as M_k. In the following equations, *mig* is abbreviated for migration, *tot* for total, and *tra* for transition. The additional energy consumption of migrating VM j is related to its capacity (especially memory), and can be computed as reported in [14].

$$E^{mig}_j = c_1 d_j + c_2 \tag{7}$$

where c_1, c_2 is constant, d_j is the requested capacity of VM j. For simplicity, we can consider the additional energy consumption by VM migrations is constant with the total number of migrations. Therefore the total energy consumption (E^{mig}_{tot}) by VM migrations in K slots can be computed as

$$E^{mig}_{tot} = \sum_{k=1}^{K} c_0 M_k \tag{8}$$

Considering the additional energy computation of transiting PM i from on state to off (hibernating) state (or vice versa) as

γ (constant) and there are m_k power-on PMs at slot k, then the total energy consumption of PMs transitions can be computed as

$$E_{tot}^{tra} = \sum_{k=1}^{K} \gamma |m_k - m_{k-1}| = \gamma \sum_{k=1}^{K} |m_k - m_{k-1}| \quad (9)$$

where m_0 is the total number of PMs in the off (hibernating) state at the beginning. Set $\alpha = P_{min}$, $\beta = (P_{max} - P_{min})$, I_k^i as a Boolean variable indicates whether PM i is power-on or not in slot k. Let $\rho = \frac{\sum_{i=1}^{m} U_i t_i}{\sum_{i=1}^{m} t_i} = \frac{\sum_{i=1}^{m} U_i t_i}{T}$ as the offered load to the CDC (cloud data center), $T = \sum_{i=1}^{m} t_i$, then $\sum_{i=1}^{m} U_i t_i = \rho T$, and we have the energy consumption (E_{tot}^{on}) by running m PMs during K slots:

$$\begin{aligned} E_{tot}^{on} &= \sum_{i=1}^{m} E_i = P_{min} T + (P_{max} - P_{min}) \sum_{i=1}^{m} U_i t_i \\ &= \alpha T + \beta \rho T = (\alpha + \beta\rho) \sum_{i=1}^{m} t_i \\ &= (\alpha + \beta\rho) \sum_{i=1}^{m} \sum_{k=1}^{K} I_k^i \end{aligned} \quad (10)$$

By considering three parts of energy consumption of running PMs (a), transiting PMs (b) and migrating VMs (c), the total energy consumption of a cloud data center (CDC) can be counted as:

$$\begin{aligned} E_{tot}^{CDC} &= a + b + c = E_{tot}^{On} + E_{tot}^{tra} + E_{tot}^{mig} \\ &= (\alpha + \beta\rho) \sum_{i=1}^{m} \sum_{k=1}^{K} I_k^i + \gamma \sum_{k=1}^{K} |m_k - m_{k-1}| \\ &\quad + c_0 \sum_{k=1}^{K} M_k \end{aligned} \quad (11)$$

Set the total number of VM migrations as M during evaluation period of K slots. The problem of minimizing total energy consumption subject to some Quality of Service (QoS) constraints can be formally stated as

$$\begin{aligned} &\min \sum_{i=1}^{m} E_i \\ &= \min \ ((\alpha + \beta\rho) \sum_{i=1}^{m} \sum_{k=1}^{K} I_k^i \\ &+ \gamma \sum_{k=1}^{K} |m_k - m_{k-1}| + c_0 \sum_{k=1}^{K} M_k) \\ &\text{subject to : (a) } \forall \text{ slot } s \in \{1, 2, \ldots, K\}, \\ &\qquad \sum_{VM_j \in PM_i} d_j \leq g_i \text{ (CPU constraint)} \\ &\qquad \text{(b) } \forall \text{ slot } s \in \{1, 2, \ldots, K\}, \\ &\qquad \sum_{VM_j \in PM_i} VMEM_j \leq MEM_i \text{ (Memory constraint)} \\ &\qquad \text{(c) } \forall j, 0 \leq s_j < e_j \text{ (interval constraint)} \\ &\qquad \text{(d) } \forall i \& \forall t, U_l \leq CPU_i(t) \leq U_u \text{ (CPU bounds)} \\ &\qquad \text{(e) } M \leq M_0 \text{ (Migration constraint)} \end{aligned}$$

where (a) requires that the total CPU capacity of all VMs on PM i cannot be larger than the available CPU capacity (g_i) PM i can provide; (b) requires that the total memory capacity of all VMs on PM i cannot be larger than the available memory capacity (MEM_i) PM i can provide; (c) requires that each job has start-time s_j and end-time e_j; (d) states that at any time the CPU utilization of each PM has a lower threshold (U_l) and a upper threshold (U_u); (e) requires that the total number of migrations during evaluation period should be less than M_0.

Given a set of VM jobs J, the total number of PM status (On/Off) transitions and the total number of VM migrations are constants under strongly divisible capacity configuration comparing to optimal results (explained in the following section). We will prove this in the following section.

A. THE LOWER BOUND OF THE TOTAL ENERGY CONSUMPTION

The theoretical lower bound of the total energy consumption in a Cloud data center is determined by the configuration of jobs (VM requests) and the PMs. Firstly we consider the strongly divisible capacity configuration.

Definition 4: Strongly divisible capacity configuration (SDC): the capacity of all jobs form a divisible sequence, i.e., the sequence of distinct capacities $d_1 \geq d_2 \geq \ldots \geq d_i \geq d_{i+1} \geq \ldots$ taken on by jobs (the number of jobs of each capacity is arbitrary) is such that for all $i > 1$, d_i exactly is divisible by d_{i+1}. A list J of items has divisible item capacity if the capacities of the items in J form a divisible sequence. If J is a list of items and g is the total capacity of a PM, then the pair (J, g) is weakly divisible if J has divisible item capacities and strongly divisible if in addition the largest item capacity d_1 in J exactly divides the capacity g. See paper [3] for the detailed discussion.

Example 1: Set the total CPU capacity of a PM as $g = 8$ which represents the CPU Cores, and each VM requests one of capacities in $\{1, 2, 4, 8\}$, then the sequence of requested capacities forms a strongly divisible capacity (SDC). But the sequence $\{1, 3, 5, 7\}$ does not form a strongly divisible capacity for $g = 8$. If all jobs have unit demand ($d_j = 1$), then the sequence of requested capacities also form a SDC.

In the Following Sections, Unless Otherwise Specified, the SDC Is Considered:

Theorem 1: For a given set of jobs under SDC and the workload is is represented in the slot window format in discrete time, the total power-on time (in slots) is the total number of the minimum number of machines used in each slot. Therefore the lower bound of total energy consumption can be counted easily.

Proof: Consider a set of jobs J under SDC, we can compute the minimum number of PMs needed for each time slot, as $m_1, m_2, \ldots m_K$ for total K slots, where m_i is the minimum number of machines needed for slot i. The minimum number of power-on PMs at slot i, $m_i = \lceil \frac{l_i}{g} \rceil$, and l_i is the sum of load for slot i. An example is shown in FIGURE 2. The total power-on time (in slots) of all machines is the sum of the

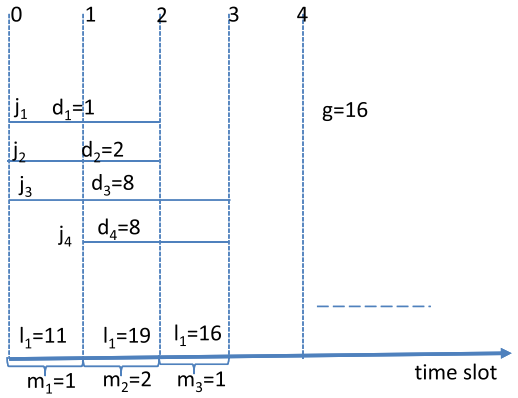

FIGURE 2. Lower bound of energy consumption.

minimum number of PMs in all slots, i.e. $T = \sum_{k=1}^{K} m_i = \sum_{k=1}^{K} \lceil \frac{l_i}{g} \rceil$. This indicates the total power-on time in each slot is the sum of the minimum number of PMs used in each slot. Then the lower bound of total energy consumption can be computed by Equ (5). This completes the proof. □

Remark: The lower bound can be computed easily and used for the solution to the open problem (1). But it is not easy to achieve in the general capacity configuration case because that minimizing the total power-on time of a given set jobs is a NP-complete problem in general case [12], [13].

However, for SDC, the lower bound of the total power-on time can be easily obtained by algorithms such as Best-Fit-Decreasing (BFD) and First-Fit (FF), see [3] for a detailed introduction. Under SDC, the total number of On/Off PMs can be easily computed and the total number of VM migrations is fixed once a set of jobs J is given. As for minimizing the total power-on time, it is also possible to obtain the lower bound with job (VM) migrations.

Observation 1: It is possible to reach the lower bound of the total energy consumption by a constant number (M) of job migrations under SDC.

Proof: By Theorem 1 and Equ (11), one can see that a lower bound exists for the total energy consumption. However without job migration, it is NP-complete in general case while reaching the lower bound is possible with job migration (a job can be migrated from one host machine to another host machine to be continuously proceeded).

The approach is presented in Algorithm 4.1 LB-Min-Migration. Algorithm 4.1 firstly sorts all jobs in non-decreasing order of jobs' start-time (line 1) and computes the load of each slot by the minimum number of PMs used (line 3-4); then it computes the longest continuous interval $[z_1, z_2]$ with same load and divides jobs into two groups (line 5-9); it schedules jobs in each group by First Fit; and migrates the job to an existing PM when the minimum number of PMs will be larger than the slot load (line 12-15); it updates the load of each PM and deletes allocated jobs and repeats the major steps until all jobs are scheduled (line 17-22). If a new schedule passes through an interval that already has the minimum number of PMs used (by the lower bound), then the new allocation will be migrated to an existing PM that still have available capacity in the interval, therefore only the minimum number of PMs is needed for any slot (or interval). Job migration only happens when necessary. Hence the algorithm reaches the lower bound with the minimum total number (M) of job migrations only. □

Algorithm 1 LB-Min-Migration Under SDC

Input: A Job instance $J = \{j_1, j_2, \ldots, j_n\}$, and g, the max capacity g of a machine
Output: The scheduled jobs, the total power-on time of all machines, the total number VM migrations M

```
1  Sort all jobs in non-decreasing order of their start-time
   (s_j for job j), such that s_1 ≤ s_2 ... ≤ s_n, set h = 1,
   M = 0;
2  forall the slots under consideration do
3      Represent load of slot i by the min number of
       machines needed for it, denoted as m_i (take integral
       value by ceiling function)
4  end
5  forall the jobs under consideration do
6      Find the longest continuous interval with same load
       first, denoted as [z_1, z_2];
7      forall the jobs either started or ended in [z_1, z_2] do
8          separate jobs into end-time first and start-time
           first groups, consider the longest job first in the
           same group;
9          if m_i is not reached in all slots of this interval
           then
10             allocate to the first machine if available,
               otherwise open a new machine and set
               h = h + 1;
11             else
12                 forall the those slots that the min
                   number of machines will be more than
                   m_i by new allocation do
13                     the allocation is migrated to an
                       existing machine which still can host
                       in those slots.
14                     M = M + 1;
15                 end
16             end
17         end
18         update load of m_h, remove allocated jobs;
19     end
20     Find total power-on time of all machines;
21     Return the total number of power-on PMs (m),
       the total power-on time of all machines, and the total
       number VM migrations M
22 end
```

Taking the example in FIGURE 2 and assuming that two PMs are at hibernating (energy-saving) state at time 0. Four jobs have following characteristics: $d_1 = 1, d_2 = 2, d_3 = 8,$

$d_4 = 8$, respectively; j_1, j_2 have starting-time at $t = 0$ and end-time at $t = 2$; j_3 has starting-time at $t = 0$ and end-time at $t = 3$; j_4 has start-time at $t = 1$ and end-time at $t = 3$. If $\{j_1, j_2, j_3\}$ are allocated to PM h_1, j_4 is allocated to h_2, the total power-on time of two PMs will be t_1 during [0,3], t_2 during [1,3], i. e., $3+2 = 5$ slots. Applying Algorithm 4.1, with 1 migration at time 2 (the 3rd slot), j_4 is migrated to h_1 in the 4th slot, then the total power-on time of the two PMs will be $3+1 = 4$, which is the lower bound for the total power-on time.

We also have the following Theorem:

Theorem 2: For energy model given in Equ. (11) and a given set of VM requests J, the total energy consumption of all PMs can be exactly computed and is related to the total number of power-on PMs, the total power-on time of all PMs, the total offered load (ρ) by running VMs, and some constant additional energy by PM transitions and VM migrations.

Proof: Let us set $\alpha = P_{min}$ and $\beta = (P_{max} - P_{min})$, we have:

$$\begin{aligned} E_{tot}^{CDC} &= a + b + c = E_{tot}^{On} + E_{tot}^{tra} + E_{tot}^{mig} \\ &= (\alpha + \beta\rho) \sum_{i=1}^{m} \sum_{k=1}^{K} I_k + \gamma \sum_{k=1}^{K} |m_k - m_{k-1}| \\ &\quad + c_0 \sum_{k=1}^{K} M_k \\ &= (\alpha + \beta\rho)T + C_1 + C_2 \end{aligned} \tag{12}$$

where $T = \sum_{i=1}^{m} \sum_{k=1}^{K} I_k$ is the total power-on time of all PMs, C_1, C_2 is the energy consumption of transiting PMs and migrating VMs respectively, which is fixed and negligible compared to E_{tot}^{on} once the set of VM jobs is given by Theorem 1 and Observation 1. We can know that the total energy consumption of all PMs is determined by the total number of power-on PMs (m), the total power-on time (T) of all PMs and total offered load (ρ). Once J is known, then m and T are major factors affecting total energy consumption. □

This paper considers two major scheduling policies: (A) always power-on PMs once turning-on and (B) turning-off (hibernating) idle PMs during evaluation, both with possible VM migrations.

Observation 2: For Policy A, the total energy consumption is affected by the total power-on time of all PMs for a given set of VM jobs.

Proof: From Equ (11), it is known that the total energy consumption can be affected by the total power-on time of all PMs (T), the total number of power-on PMs (m), and total workload (ρ) by hosting all VMs. ρ is fixed once the jobs are given. Therefore, the total energy consumption is affected by the combination of total power-on PMs and their total power-time. □

Observation 3: For Policy B (turning-off (hibernating) idle PMs), once a set of VM jobs are given, the total number of PMs (m) and their total power-on time (T) determine the difference of total energy consumption of any two scheduling algorithms, so that if one algorithm can reduce T, then the total energy consumption can be decreased.

Proof: From Equ (11), it can be seen that the total energy consumption is affected by the total power-on time of all PMs (T) and total workload (ρ) through hosting all VMs when ρ is fixed once the jobs are given. Then the total number of power-on PMs (m) and the total power-on time (T) of all PMs will affect the total energy consumption. So turning-off (or hibernating) idle PMs will reduce T and therefore decrease the total energy consumption. And the reduced energy can be computed easily. □

The practical ways to decrease the total energy consumption include turning-off (or hibernating) idle servers and using workload consolidation to migrate VMs from lower utilization PMs to other PMs and then turning-off (hibernating) idle servers. Having the lower bound, we can evaluate the energy efficiency as follows.

B. ENERGY EFFICIENCY EVALUATION OF SCHEDULING ALGORITHMS

To find a solution to the open problem (2), i.e., to compute the energy-efficiency of a scheduling algorithm, it is needed to compare the result obtained by an algorithm with the lower bound (abbreviated as LB in this paper). In this paper, we set the lower bound as the optimum (abbreviated as OPT). The ratio of the result received by an approximation algorithm over the OPT is called approximation ratio and can be applied to measure the energy-efficiency in this case.

Definition 5: The Approximation Ratio: a deterministic algorithm is a C-approximation for the objective of minimizing the total energy consumption if its total energy consumption is at most C times that of the lower bound (LB).

Example 2: For a given set of VM requests, BFD (Best-Fit-Decreasing) is $(\frac{11}{9})$-approximation in general case regarding the total number of power-on PMs (see [1]), and is 4-approximation for unit-demand regarding the total power-on time (see Flammini *et al.* [13]).

In the following section, we provide performance evaluation for the two major scheduling policies.

V. PERFORMANCE EVALUATION

First, a simple example is given below to explain how power and energy consumptions are computed for (A) PMs are always power-on once turning-on and (B) PMs are turning off when idle.

Example 3: Considering a testing period of [0, 490] seconds, a PM is always turned on during the interval [0, 490], with average CPU utilization 0.50 during [10,130] and [310,490], CPU utilization is zero during [0,10] and [130,310] ; from real tests we have $P_{max} = 300$ watts, $P_{min} = 210$ watts, and its total length of power-on time 490 seconds. The average CPU utilization in [0, 490] is 0.3125.The average power consumption is 210+(300-210)×0.50=255 watts. Then the total energy consumption is:(210+(300-210)×0.3125)×490/3600 = 32.41 watts-hours

while the energy consumption of power-on is 210×490/3600 = 28.58 watts-hours and the energy consumption of hosting VMs is 90×0.3125×490/3600 =3.83 watts-hours.

Example 4: Considering a testing period of [0, 490] seconds, a PM is turned on during two intervals [10, 130], [310, 490] respectively and turned off during [0,10] and [130,310]. The average CPU utilization is 0.50; P_{max} = 300 watts, P_{min} = 210 watts, the total length of power-on time for the PM is 130-10+490-310=300 seconds. The average power consumption is 210+(300-210)×0.50= 255 watts. Then the total energy consumption is:(210+(300-210)×0.50)×300/3600=21.25 watts-hours while the energy consumption of power-on is 210×300/3600=17.5 watts-hours and the energy consumption of hosting VMs is 90×0.5×480/3600=3.75 watts-hours.

We can see that the energy consumption of hosting VMs are the same for case (A) and (B), the difference for (A) and (B) lies in the energy consumption of the power-on time of PMs.

A. TEST IN REAL ENVIRONMENT

In this section, we evaluate our theoretical results in a real environment. We have set-up two-node cloud the same as in [11]. Each VM reservation is a compute-intensive job with a *cpuburn* (consuming mainly CPU resource) running on the VM. **Each node can host up to 7 VMs. All the VMs are identical in terms of memory and CPU configuration**.The two scheduling algorithms considered are:

(1). Round-robin (abbreviated as Round): the 1st job is allocated to the 1st cloud node, the 2nd job to the 2nd one, and so on. When all the nodes are idle, the order of the nodes is changed (do not always attribute the 1st job in the queue to the 1st node).

(2).First-Fit: it puts all the jobs on the 1st cloud node, and if there are still jobs in the queue, the second node is used and so on (the order is changed to balance the roles when all the nodes are idle).

These two scheduling algorithms are well-known and widely applied in large-scale distributed systems management. The two-node PMs are identical with P_{max} = 300 watts and P_{min} = 210 watts. Each On/Off state transition for a PM costs 5 Joules in less than 2 minutes, so $\gamma \ll 10/60 = 0.167$ watt-hour. Similarly, each VM migration costs 10 Joules in less than 1 minutes so $c_0 \ll 0.167$ watt-hour. These are very small and negligible. In the following, there are at most 2 times state transitions or VM migrations for each case, so that we do not consider the additional energy costs by PM state transitions and VM migrations. The job arrival scenario is also same as in [11], the Gantt chart is shown in Fig.3:

At $t = 10$, 3 jobs of length 120 seconds each and 3 jobs of length 20s each;

At $t = 130$, 1 job of length 180s;

At $t = 310$, 8 jobs of length 60s each;

At $t = 370$, 5 jobs of length 120s each, 3 jobs of length 20s each and 1 job of length 120s, in that order.

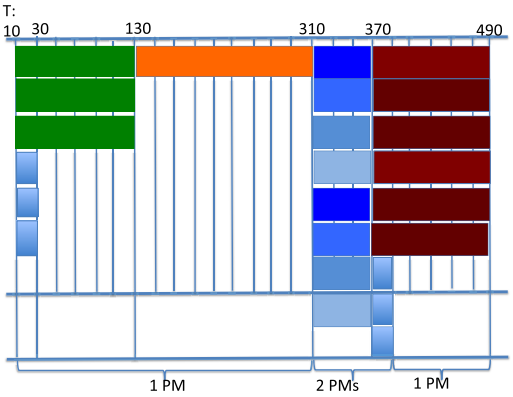

FIGURE 3. The lower bound for 24 jobs instance.

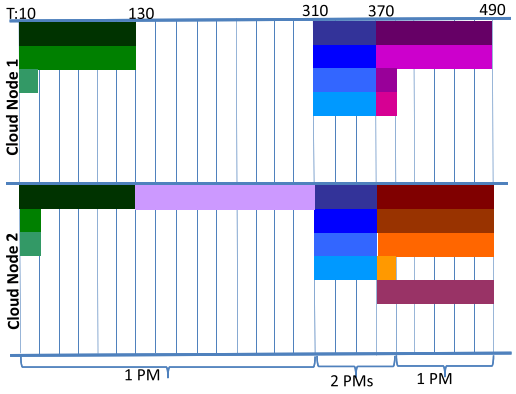

FIGURE 4. Gantt chart for round-robin under policy A.

1) THE LOWER BOUND OF TOTAL ENERGY CONSUMPTION

In FIGURE 3, for the same 24 jobs, we can find the lower bound of total power-on time is 560 seconds: one machine works during [10, 490] with power-on time 480 seconds, another machine works during [310, 390] with power-on time 80 seconds (turned off during other intervals), so the total power-on time is 560 seconds. From these, we can find the lower bound for the total power and energy consumption, LB=35.4912 Watt-hours.

2) POLICY A: ALWAYS POWER-ON PMS ONCE TURNING ON

Example 5: Consider the instance shown FIGURE 4 and FIGURE 5. For Round-Robin, two machines have utilization $U_1 = U_2 = \frac{9}{28}$; the average utilization $U = \frac{31}{112}$.

For unbalanced scheduling, two machines have utilization $\hat{U}_1 = \frac{25}{84}$, $\hat{U}_2 = \frac{43}{168}$; the average utilization $\hat{U} = \frac{31}{112}$.

Since their average utilization are the same and total power-on time also same, so that their average power consumption are the same and their total energy consumption are identical,i.e, $E^A = (210 + (300 - 210) \times \frac{31}{112}) \times 960/3600 = 62.6428$ watts-hours, the result is consistent with results obtained in [11]. This result validate Observation 2. Also the energy efficiency is $\frac{E^A}{LB} = \frac{62.6428}{35.492} \approx 1.76$.

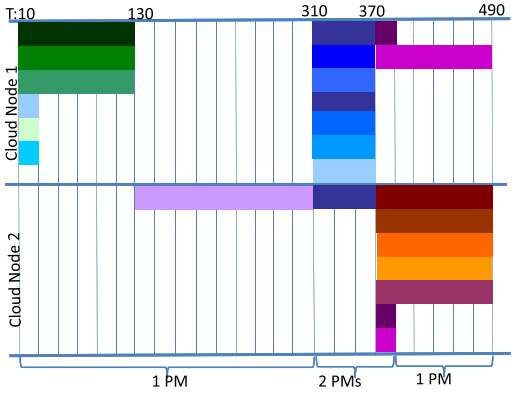

FIGURE 5. Gantt chart for first-fit under policy A.

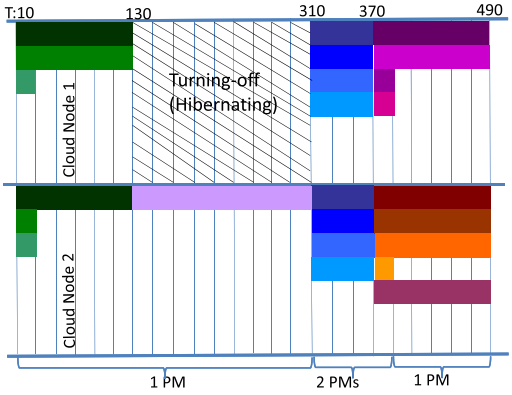

FIGURE 6. Gantt chart for the round-robin under policy B.

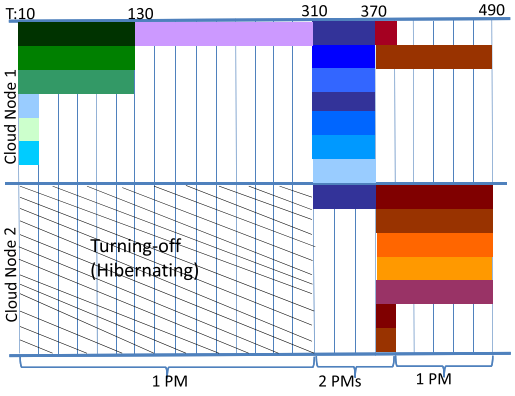

FIGURE 7. Gantt chart for the first-fit under policy B.

3) POLICY B: TURNING-OFF PMS WHEN IDLE

Example 6: Consider the instance shown FIGURE 6 and FIGURE 7. For Round-Robin, two machines have utilization $U_1 = \frac{26}{105}$, $U_2 = \frac{9}{28}$; the average utilization $U = \frac{239}{840}$; the total power-on time of two machines is 780 seconds in this case, and the total energy consumption is $E^B_{round} = 51.048$ Watt-hours, the energy efficiency is $\frac{E^B_{round}}{LB} = \frac{51.048}{35.492} \approx 1.44$.

For First-Fit, two machines have utilization $\hat{U_1} = \frac{29}{84}$, $\hat{U_2} = \frac{5}{9}$; the average utilization $\hat{U} = \frac{227}{504}$; the total power-on time of two machines is 660 seconds in this case, and the total energy consumption is $E^B_{FF} = 45.9315$ Watt-hours, the energy efficiency is $\frac{E^B_{FF}}{LB} = \frac{45.9315}{35.492} \approx 1.29$.

The total energy consumption of balanced Round-Robin scheduling (with total power-on time 780 seconds) is larger than unbalanced First-Fit scheduling (with total power-on time 660 seconds).

We also observe that the total energy-consumption of Policy B is less than Policy A for a given set of jobs and same PM configurations.

Observation 4: The total energy-consumption of Policy B is less than Policy A for a given set of jobs and same PM configurations, and for Policy B, First-Fit is more energy-efficient than Round-Robin.

Next we consider Policy B with VM migrations to further decrease the total energy consumption.

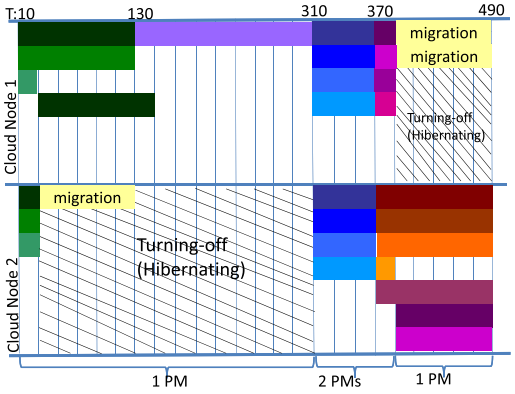

FIGURE 8. Gantt chart for round-robin under policy B with VM migrations.

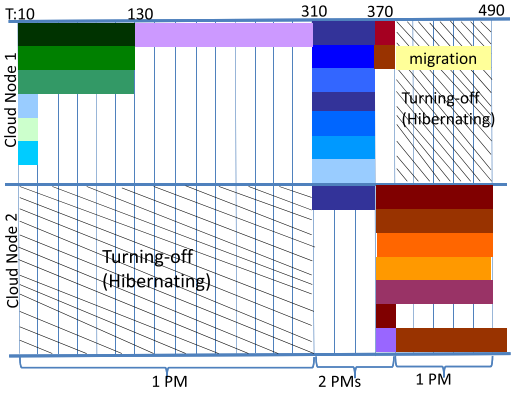

FIGURE 9. Gantt chart for first-fit under policy B with VM migrations.

B. POLICY B: TURNING-OFF PMS WHEN IDLE WITH VM MIGRATIONS

Example 7: Consider the instance shown in Fig.8 and Fig.9. For Round-Robin scheduling, two machines have utilization $U_1 = \frac{43}{153}$, $U_2 = \frac{5}{7}$; the average utilization $U = \frac{23}{51}$; the total power-on time of two machines is 580 seconds in this case, and the total energy consumption is $E^B_{round} = 40.3725$ Watt-hours, the energy efficiency is $\frac{E^B_{round}}{LB} = \frac{40.3725}{35.492} \approx 1.14$.

For unbalanced First-Fit scheduling, two machines have utilization $\hat{U}_1 = \frac{54}{133}$, $\hat{U}_2 = \frac{40}{63}$; the average utilization $\hat{U} = \frac{161}{798}$; the total power-on time of two machines is 560 seconds in this case, and the total energy consumption is $E^B_{FF} = 35.4912$ Watt-hours,the energy efficiency is $\frac{E^B_{FF}}{LB} = \frac{35.4912}{35.492} = 1.0$.

In this case, the total energy consumption of balanced Round-Robin scheduling (with total power-on time 580 seconds) is larger than unbalanced First-Fit scheduling (with total power-on time 560 seconds). These results are consistent with experimental results in [11].

We observe that the total energy consumption is further reduced with VM migrations.

C. SIMULATION USING SYNTHETIC DATA

The configuration of VMs and PMs are given in Table 1 and 2. We consider SDC case. Table 3 also provides different P_{min} and P_{max} for three type of PMs. Notice that VM Type $i-$ can only be allocated to PM Type i, heterogeneous case is transformed to homogeneous case by grouping in this way. It is assumed that all VMs occupy all their requested capacity (the worst case), and all PMs are at power-saving mode (hibernating) at the start-time to save energy consumption. A limited number of VM migrations are also applied by setting CPU threshold ($U_l = 0.2$, $U_u = 0.90$) and VM migrations are triggered when the CPU utilization of a PM is below U_l or above U_u. Each On/Off state transition for a PM costs $\gamma = 0.167$ watt-hour. Similarly, each VM migration costs 10 Joules in 1 minutes so $c_0 = 0.167$ watt-hour.

The metrics include:

1) The total number of PMs used, the actual total number of powered-on PMs during the testing period.

2) The total length of power-on time of all PMs during the testing period.

3) The total energy consumption of a cloud data center.

We considered seven algorithms in this simulation:

- Round-Robin (Round): the round-robin is one of commonly used scheduling algorithms (for example by Amazon EC2 [19]), which allocates VM requests in turn to each PM. The strength of this algorithm is that it is easy to implement and can keep good load balance when VM requests and PMs are homogeneous.
- Offline scheduling without migration (OFWIM): It firstly sorts all requests in non-decreasing order of their start-time and then schedules the request to the 1st available PM, also called First-Fit-Decreasing (FFD). Its computational complexity is of $O(nlogn)$ where n is the total number of requests.
- Offline scheduling with migration (OFWM): This is same as OFWIM except that VM migrations are applied.
- Online scheduling without migration (ONWIM): It firstly allocates the request to the first available PM, also called First-Fit (FF). Its computational complexity is of $O(n)$ where n is the total number of requests.

TABLE 1. 8 Types of virtual machines (VMs) in Amazon EC2.

Memory	CPU (units)	Storage	VM Type
1.875	1 (1 cores x 1 units)	211.25	1-1(1)
7.5	4 (2 cores x 2 units)	845	1-2(2)
15.0	8 (4 cores x 2 units)	1690	1-3(3)
17.1	6.5 (2 cores x 3.25 units)	422.5	2-1(4)
34.2	13 (4 cores x 3.25 units)	845	2-2(5)
68.4	26 (8 cores x 3.25 units)	1690	2-3(6)
1.7	5 (2 cores x 2.5 units)	422.5	3-1(7)
6.8	20 (8 cores x 2.5 units)	1690	3-2(8)

- Online scheduling with migration (ONWM):This is same as ONWIM except that VM migrations are applied.
- Modified Best Fit Decreasing (MBFD) algorithm: is a bin-packing algorithm introduced in [1], it firstly sorts all requests in non-increasing order of their processing time and then allocates the request to the first available PM. It is $\frac{11}{9}$-approximation regarding the total number of PMs. Its computational complexity is of $O(nlogn)$ where n is the total number of requests.
- The lower bound (LB): It is the theoretical lower bound, obtained by the approach proved in Theorem 1. The computational complexity of computing the theoretical lower bound is linear with the total loads on all slots.

In the simulation, the total numbers of arrivals (requests) is 1000, each type of VMs has equal number, i.e., 125 and the total time slots $K = 10000$. Poisson arrival process and have exponential service time are followed by all requests, the mean inter-arrival period is set as 5 slots, the maximum intermediate period is set as 50 slots, the maximum duration of all requests is set as 50, 100, 200, 400, 800 slots, respectively. Each slot is set as 5 minutes. For instance, if the requested duration (service time) of a VM is 20 slots, actually its duration is $20 \times 5 = 100$ minutes. Simulations are run 10 times and all the results shown in this paper are the average of the 10 runs for each set of inputs. For the case where VM migration is adopted, we record the total number of migrations.

1) POLICY A: ALWAYS POWER-ON PMS ONCE TURNING-ON

FIGURE 10 shows the total number of power-on PMs as the maximum duration of VMs varies from 50 to 800 slots while all other parameters are the same. For all cases, Round>ONWIM $\geq$ OFWIM $\geq$ MBFD$\geq$ ONWM$\geq$ OFWM$\geq$ *LB* regarding the total number of power-on PMs. We observe that ONWM and OFWM use almost the same number of PMs, very close to LB in this test. Round Robin (Round) is the worst regarding the total number of power-on PMs. The reason for MBFD, ONWM, and OFWM to perform better than ONWIM, OFWIM, and Round is that ONWM and OFWM are among better algorithms to reduce the total number of power-on PMs.

FIGURE 11 presents the total energy consumption comparisons as the maximum duration of VMs varies from 50 to 800 slots when all other parameters are kept the same.

TABLE 2. 3 Types of PMs for divisible configuration.

PM Type	CPU (units)	Memory (GB)	Storage (GB)	P_{min}	P_{max}
1	16 (4 cores x 4 units)	30	3380	210	300
2	52 (16 cores x 3.25 units)	136.8	3380	420	600
3	40 (16 cores x 2.5 units)	14	3380	350	500

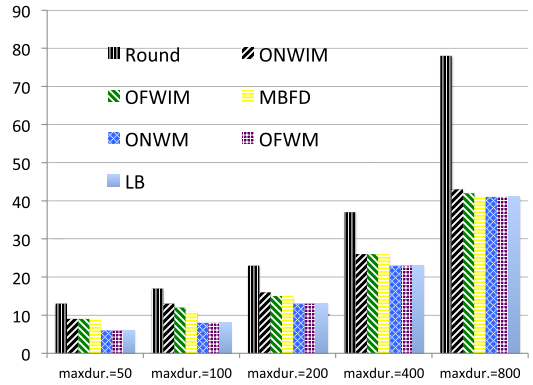

FIGURE 10. Total number of power-on PMs under policy A.

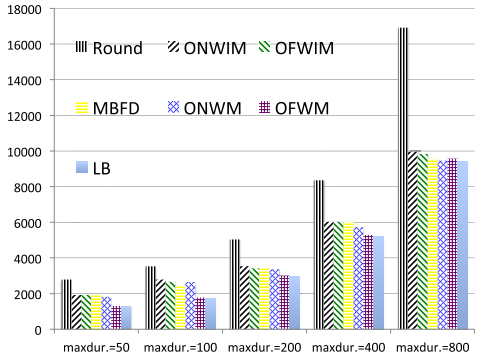

FIGURE 11. The total energy consumption (in KWh) under policy A.

With regarding to the total energy consumptions, Round> ONWIM ≥ OFWIM ≥ MBFD≥ ONWM≥ OFWM≥ *LB* in all cases. The reason for MBFD, ONWM, and OFWM to perform better than ONWIM, OFWIM, and Round is that ONWM and OFWM are among better algorithms to uses less total number of power-on PMs. When other parameters are the same, the total number of power-on PMs affects the total energy-consumption. The average energy efficiency (ratio) for (Round, ONWIM, OFWIM, MBFD,ONWM, OFWM) is respectively (1.861, 1.302,1.275,1.239, 1.236, 1.017) when maximum duration of VMs is 50, 100,200,400, 800 slots respectively. This is consistent with Theorem 2 and Observation 2.

2) POLICY B: TURING-OFF IDLE PMS WITH VM MIGRATIONS

FIGURE 12 gives the total number of power-on PMs of seven algorithms as the maximum duration of VMs varies from 50 to 800 slots when all other parameters are kept the

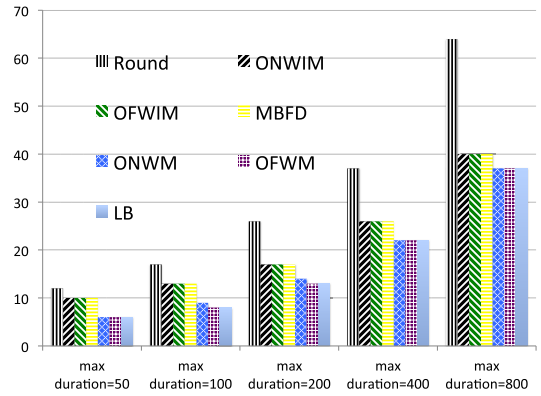

FIGURE 12. Total number of power-on PMs under policy B.

TABLE 3. The number of VM migrations.

maxdur.	OFWM	ONWM
50	2	3
100	4	5
200	5	7
400	16	21
800	30	37

same. In all cases, Round>ONWIM ≥ OFWIM ≥ MBFD≥ ONWM≥ OFWM≥ *LB*. The reason for MBFD, ONWM, and OFWM to perform better than ONWIM, OFWIM, and Round is that ONWM and OFWM are among better algorithms to use VM migrations to reduce the total number of power-on PMs.

The number of migrations for ONWM and OFWM are given in Table 3.

FIGURE 13 provides the total power-on time comparisons as the maximum duration of VMs varies from 50 to 800 slots when all other parameters are kept the same. With regarding to the total power-on time, Round>ONWIM ≥ OFWIM ≥ MBFD≥ ONWM≥ OFWM≥ *LB* for all cases. The reason for MBFD, ONWM, and OFWM to perform better than ONWIM, OFWIM, and Round is that ONWM and OFWM are among better algorithms to use VM migrations to reduce the total power-on time of power-on PMs. MBFD, ONWM, and OFWM perform similarly under SDC.

FIGURE 14 shows the total energy consumption as the maximum duration of VMs varies from 50 to 800 slots when all other parameters are kept the same. In all cases, Round>ONWIM ≥ OFWIM ≥ MBFD≥ ONWM≥ OFWM≥ *LB*. Combining results of FIGURE 12 and 13, we can see that the reason for MBFD, ONWM, and OFWM

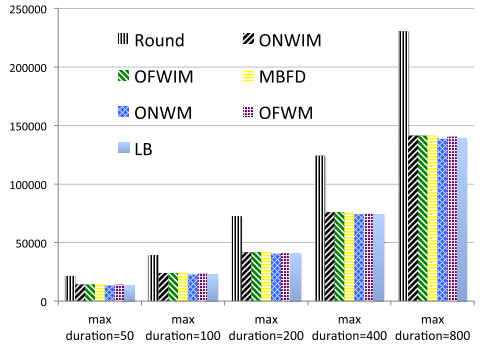

FIGURE 13. The total power-on time (milliseconds) of all PMs under policy B.

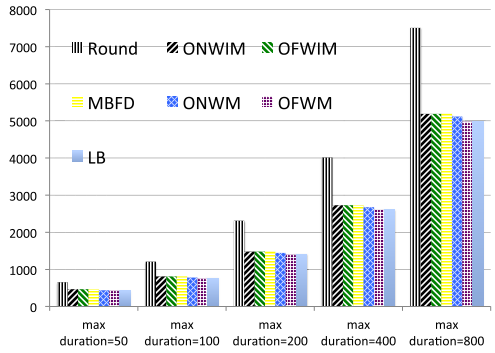

FIGURE 14. The total energy consumption (in KWh) under policy B.

to perform better than ONWIM, OFWIM, and Round is that ONWM and OFWM are among better algorithms to use VM migrations to reduce the combination of the total number of power-on PMs and their total power-on time. The average energy efficiency (ratio) for (Round, ONWIM, OFWIM, MBFD,ONWM, OFWM) is respectively (1.565, 1.062, 1.061,1.020,1.020,1.010) when max duration of VMs is 50, 100,200,400, 800 slots respectively. This is consistent with Observation 3.

VI. CONCLUSIONS AND FUTURE WORK

This paper proposed analytical methods for average power consumption and total energy consumption. Our theoretical results are validated by different scheduling scenarios in real tests and simulation. These results show that our proposed methods can be applied to design and evaluate energy-efficiency in cloud computing. There are a few research issues under further investigation:

- Considering other overheads during VM migrations: It is possible to reduce the total energy consumption by limiting number of VM migrations. As frequently migrating VMs can cause network congestion and vibration, the number of VM migrations should be minimized. Other overheads including delay, overloaded PMs and management costs than additional energy consumption during VM migrations should also be considered.
- Evaluating more power consumption models such as DVFS (Dynamic Voltage Frequency Scaling). Intensive research already show that combining DVFS can further improve energy-efficiency. We are analyzing the energy-efficiency combining DVFS and CPU utilization.
- Considering energy consumption of other applications. The power consumption model based on CPU utilization works fine for compute-intensive applications. Other applications such as RAM-intensive, IO-intensive and their combinations are also under investigation.

REFERENCES

[1] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generat. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[2] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, "Bin packing with divisible item sizes," *J. Complex.*, vol. 3, no. 4, pp. 406–428, 1987.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1978.

[4] M. Guazzone, C. Anglano, and C. Canonico, "Energy-efficient resource management for cloud computing infrastructures," in *Proc. Int. Conf. Cloud Comput. Technol. Sci.*, vol. 1, Athens, Greece, Nov./Dec. 2011, pp. 424–431.

[5] J. Kleinberg and E. Tardos, *Algorithm Design*, London, U.K.: Pearson, 2005.

[6] K. H. Kim and A. Beloglazov, and R. Buyya, "Power-aware provisioning of virtual machines for real-time cloud services," *Concurrency Comput., Pract. Exper.*, vol. 23, no. 13, pp. 1491–1505, 2011.

[7] M. Y. Kovalyov, C. T. Ng, and T. C. E. Cheng, "Fixed interval scheduling: Models, applications, computational complexity and algorithms," *Eur. J. Oper. Res.*, vol. 178, no. 2, pp. 331–342, 2007.

[8] A. W. J. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. R. Spieksma, *Interval Scheduling: A Survey*. Hoboken, NJ, USA: Wiley, 2007. [Online]. Available: http://www.interscience.wiley.com

[9] V. Mathew, R. K. Sitaraman, and P. Shenoy, "Energy-aware load balancing in content delivery networks," in *Proc. INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 954–962.

[10] D. Nurmi *et al.*, "The eucalyptus open-source cloud-computing system," in *Proc. 9th IEEE Int. Symp. Cluster Comput. Grid*, Shanghai, China, May 2009, pp. 124–131.

[11] L. Lefèvre and A.-C. Orgerie, "Designing and evaluating an energy efficient cloud," *J. Supercomput.*, vol. 51, no. 3, pp. 352–373, 2010, doi: 10.1007/s11227-013-0974-z.

[12] P. Winkler and L. Zhang, "Wavelength assignment and generalized interval graph coloring," in *Proc. 14th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2003, pp. 830–831.

[13] M. Flammini *et al.*, "Minimizing total busy time in parallel scheduling with application to optical networks," *Theor. Comput. Sci.*, vol. 411, nos. 40–42, pp. 3553–3562, 2010.

[14] H. Liu, C. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," in *Proc. HPDC*, San Jose, CA, USA, 2011, pp. 171–182.

[15] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. ACM Int. Symp. Comput. Archit.*, San Diego, CA, USA, 2007, pp. 13–23.

[16] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," in *Proc. Workshop Modeling Benchmarking Simulation*, 2006, pp. 1–8.

[17] C. Mastroianni, M. Meo, and G. Papuzzo, "Probabilistic consolidation of virtual machines in self-organizing cloud data centers," *IEEE Trans. Cloud Comput.*, vol. 1, no. 2, pp. 215–228, Jul./Dec. 2013.

[18] *Parallel Workloads Archive*. Accessed: Apr. 2013. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload

[19] L. Shi, Z. Zhang, and T. Robertazzi, "Energy-aware scheduling of embarrassingly parallel jobs and resource allocation in cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1607–1620, Jun. 2017.

[20] W. Tian and C. S. Yeo, "Minimizing total busy time in offline parallel scheduling with application to energy efficiency in cloud computing," *Concurrency Comput., Pract. Exper.*, vol. 27, no. 9, pp. 2470–2488, 2015.
[21] T. Li, Y. Ren, D. Yu, S. Jin, and T. Robertazzi, "Characterization of input/output bandwidth performance models in NUMA architecture for data intensive applications," in *Proc. 42nd Int. Conf. Parallel Process. (ICPP)*, Oct. 2013, pp. 369–378.
[22] T. Li, Y. Ren, D. Yu, and S. Jin, "Resources-conscious asynchronous high-speed data transfer in multicore systems: Design, optimizations, and evaluation," in *Proc. 29th IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2015, pp. 1097–1106.
[23] T. Li, Y. Ren, D. Yu, and S. Jin, "Analysis of NUMA effects in modern multicore systems for the design of high-performance data transfer applications," *Future Gener. Comput. Syst.*, vol. 74, pp. 41–50, Sep. 2017.
[24] W. Tian *et al.*, "On minimizing total energy consumption in the scheduling of virtual machine reservations," *J. Netw. Comput. Appl.*, vol. 113, pp. 64–74, Jul. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804518301267

PING KUANG is currently an Associate Professor with the University of Electronic Science and Technology of China. His research interests include scheduling algorithms for cloud computing and machine learning algorithms.

WENXIA GUO is currently pursuing the Ph.D. degree with the University of Electronic Science and Technology of China. Her research interests include approximation algorithm for NP-hard problems and scheduling algorithms for resource allocation in cloud computing and big data processing.

HONGJIAN LI is currently an Associate Professor with the Chongqing University of Post and Telecommunication. His research interests include scheduling algorithms for resource allocation in cloud computing and big data process platforms.

WENHONG TIAN (M'09) received the Ph.D. degree from the Computer Science Department, North Carolina State University. He is currently a Professor with the University of Electronic Science and Technology of China. His research interests include dynamic resource scheduling algorithms and management in cloud data centers, dynamic modeling, and performance analysis of communication networks. He authored about 40 journal and conference papers, and three English books in related areas. He is a member of the ACM and CCF.

RAJKUMAR BUYYA (F'15) is currently a Redmond Barry Distinguished Professor and the Director with the Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, Australia. He is also serving as the Founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in cloud computing. He has authored over 625 publications and seven text books, including *Mastering Cloud Computing* (McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese, and international markets, respectively). He also edited several books, including *Cloud Computing: Principles and Paradigms* (Wiley Press, USA, 2011). He served as a Future Fellow of the Australian Research Council from 2012 to 2016. He is a Scopus Researcher of the Year 2017 with the Excellence in Innovative Research Award by Elsevier for his outstanding contributions to cloud computing. He is one of the highly cited authors in computer science and software engineering worldwide (with h-index of 116 and g-index of 255, and over 69 800 citations). He is recognized as a Web of Science Highly Cited Researcher by Thomson Reuters in both 2016 and 2017.