https://doi.org/10.1109/TIE.2019.2902786

IEEE Transactions on Industrial Electronics Published online on 8 Mar 2019. DOI: 10.1109/TIE.2019.2902786

Profile-guided Three-phase Virtual Resource Management for Energy Efficiency of Data Centers

Zhe Ding, Yu-Chu Tian, *Member, IEEE,* Maolin Tang, *Senior Member, IEEE,*
Yuefeng Li, You-Gan Wang and Chunjie Zhou

***Abstract*—Energy efficiency is a critical issue in the management of data centers, which form the backbone of cloud computing. Virtual resource management has a significant impact on improving the energy efficiency of data centers. Despite the progress in this area, virtual resource management has been considered mainly at two separate levels: application assignment and virtual machine placement. It has not been well investigated in a unified framework for both levels, limiting further improvement in the energy efficiency of data centers. To address this issue, this paper formulates the virtual resource management problem for energy efficiency as a constrained optimization problem. Then, the paper simplifies the problem through profile-guided task classification and problem decomposition for complexity reduction and improved energy efficiency. After that, a three-phase framework and algorithms are presented for profiling and profile updating, task classification and application assignment, and successive virtual machine placement. Experimental studies show energy savings of 8% to 12% by the three-phase framework compared to the existing technique.**

***Index Terms*—Data center, energy efficiency, virtual resource management, profile, task classification**

I. Introduction

Modern industries have become data-based, demanding big data techniques and cloud support on a massive scale [1]. This demand for data leads to a significant demand for electricity to power the increasing number of data centers. From 2000 to 2005, the amount of energy consumed in data centers soared by about 115% globally. This energy consumption is 0.97% of the overall energy consumption worldwide [2]. It is predicted that by 2020 the energy consumption of data centers will increase at a scale of 140 billion kWh per year, requiring that 50 additional large power plants be built annually [3]. As a significant portion of the energy consumption in data centers is used to power physical machines (PMs) [4], energy efficiency has become a critical issue in the management of data centers.

Among many factors, virtual resource management has a significant impact on the improvement of the energy efficiency of data centers [5]. This has motivated recent research on power-aware virtual resource management strategies from various perspectives. For example, strategies and algorithms have been developed for energy-efficient resource scheduling [6]. Profiles are built for applications or computing tasks, virtual machines (VMs), and PMs for more energy-efficient application assignment [7]. First fit decreasing (FFD) and other strategies are employed for VM placement [8].

Despite the progress in improving the energy efficiency of data centers, virtual resource management remains challenging. While it has been considered at two separate levels: application assignment and VM placement, it has not been well investigated in a unified framework that considers both levels. In addition, obtaining a good virtual resource management plan from a huge search space is time-consuming, demanding a well-designed heuristic system with reduced complexity.

This paper presents a unified framework for energy-efficient virtual resource management in data centers. It formulates the virtual resource management problem as a constrained optimization problem. Then, it simplifies the problem through profile-guided task classification and problem decomposition for complexity reduction and improved energy efficiency. Furthermore, it presents a three-phase framework for profiling and profile updating, task classification and application assignment to VMs, and successive VM placement to PMs.

The paper is organized as follows: Section II reviews related work. Section III formulates the problem as a constrained optimization problem, and reduces the problem complexity. This guides our problem solving via a three-phase framework in Section IV. Simulation studies are conducted in Section V. Finally, Section VI concludes the paper.

II. Related Work and Motivations

Efforts have been made to develop energy-aware virtual resource management frameworks. An example is the work in [9]. In this example, heuristics are used to help optimize the input of energy-aware algorithms, and then generate a plan for allocating virtual resources. Quality-of-Service (QoS) performance and energy efficiency are addressed in the example. A similar work is presented in [10] that addresses VM placement and potential VM migration.

The framework in [9] faces some challenges. The optimization in the framework is based on multiple decision variables,

Manuscript received ????????; revised ????????; accepted ????????. Date of publication ????????; date of current version ????????. This work was supported in part by the Australian Research Council (ARC) through the Discovery Project Scheme under Grant DP170103305, and the National Science Foundation of China (NSFC) under Grant 6161873103. (Corresponding author: Yu-Chu Tian.)

Z. Ding, Y.-C. Tian, M. Tang and Y. Li are with the School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, QLD 4001, Australia. (e-mail: z6.ding@qut.edu.au; y.tian@qut.edu.au; m.tang@qut.edu.au; y2.li@qut.edu.au).

Y. Wang is with the School of Mathematical Sciences, Queensland University of Technology, Brisbane, QLD 4001, Australia. (e-mail: yougan.wang@qut.edu.au).

C. Zhou is with the School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: cjiezhou@ hust.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier ????????

making the optimization difficult to solve. In addition, the input in the framework is a series of VMs. However, in real-world data centers, the input is a stream of tasks as shown in the logs of a Google data center [11]. The tasks need to be packed to VMs before VM placement to PMs.

A. Profiling and Profile-based Task Assignment

In general, data centers, such as the Google example [11], record their running status in data logs. These logs involve rich information about the PM workload, job submission, task runtime, and resource requests and actual usage. By analyzing these logs, comprehensive knowledge can be extracted about tasks, jobs, and system resources. Collections of such knowledge form various profiles, such as applications, VM profiles, and PM profiles [12], [7]. Through profiles, the features and patterns of applications, VMs, and PMs can be captured.

The knowledge from profiles is useful for improving virtual resource management under various criteria, such as energy efficiency. For example, some long-running jobs execute continuously for a long time, or run frequently. They are better packed into a group of highly energy-efficient VMs separated from other VMs for short jobs. This knowledge has motivated profile-based application assignment [12], [7], in which specific algorithms are developed to achieve energy savings at the application assignment level.

Different from previous work in [12], [7], our work in this paper uses the knowledge extracted from profiles across the application assignment level and the VM placement level. In particular, tasks are classified into multiple classes for task assignment to VMs, as well as VM placement to PMs.

B. Task Characteristics and Classification

The concept of task classification has been used in practical management of data centers. Typical examples are Google [11] and VMware for improvement of Service Level Agreements (SLAs). For instance, Google separates its incoming tasks and jobs into multiple priorities. To guarantee major QoS, tasks or jobs with higher priority are allocated first before other tasks or jobs with lower priority. As the main focus here is on QoS and SLAs, energy efficiency has not been considered in the deployment of task classification.

The characteristics of tasks and jobs have an impact on energy-efficient task classification and virtual resource management. For example, some cloud services, i.e., email and web services, keep running 24 hours a day 7 days a week without a pause [13]. Mixing such services with sporadic tasks and jobs may lead to less energy-efficiency application assignment. However, stacking up all types of tasks in this way has been commonly accepted for energy optimization. A typical strategy for energy-efficient VM placement to PMs is FFD [14].

A poor VM placement may have to be adjusted at runtime through VM migration and consolidation. The live migration of a VM is not only energy consuming [15] but also challenging [16]. Thus, avoiding VM migrations is important for energy efficiency and QoS. However, this has not been directly considered in VM placement through investigations into the task classification. In comparison, our work in the present paper considers long-running tasks and jobs before any other tasks and jobs are processed, leading to improved energy efficiency and very reduced VM migrations.

C. VM Placement Strategies

The energy optimization problem of VM placement is NP-hard. It demands a significant computing effort for a solution. Consider placing 500 VMs to 100 PMs in a small-scale data center. The total number of combinations is 100^{500}. Assume 10 floating point operations are required for checking each of these combinations. Then, a total number of $10^{1,001}$ floating point operations will be executed for an exhaustive search. If the computation is conducted on the fastest supercomputer in the world, Sunway TaihuLight, at its Linpack Performance (Rmax) of $93,014.6$ Tflops, we will have to wait for more than 3.4×10^{976} years for an optimal solution! Thus, the exhaustive search technique is not practically viable for solving the problem, demanding heuristic strategies.

FFD is commonly used for heuristic VM placement. It is effective in dealing with general bin-packing problems like virtual resource management [17]. Recently, an advanced FFD algorithm was implemented for VM placement [14]. However, the algorithm does not attempt to improve the energy efficiency issue that our work aims to in the present paper. Nevertheless, when sorting VMs in terms of energy efficiency, FFD can be easily adopted for energy-efficient VM placement.

Similar to FFD, best fit decreasing (BFD) is an effective algorithm in dealing with bin-packing problems. It has been implemented in VM placement for energy optimization [18]. The implementation is based on a resource utilization ratio rather than real utilization measures.

A recent development in heuristic VM placement is a greedy algorithm [8]. It considers energy-efficient VM placement for multi-tenant data centers. An almost 50% energy savings is claimed over the original greedy algorithm. However, task assignment to VMs is not integrated in this algorithm.

Generic algorithm (GA) has also been investigated for virtualized data centers [19]. Searching in a bigger space than FFD and BFD in every step, GA optimizes the objective function better than FFD and BFD at a cost of increased execution time [20]. To improve the computational performance, GA has been trimmed with much reduced computational demand while still maintaining a good quality of solution [21]. However, existing GA has studied energy efficiency at either the VM placement level or the application assignment level [12]. There is still a lack of integration of both levels in GA. In addition, GA executes more slowly than FFD [22], [21]. For a scenario with thousands of pending tasks or VMs, it is difficult for GA to complete execution within 5 minutes.

D. Recent Practice and Investigations

Recently, the concept of energy-aware VM management was introduced in industrial practice. For example, Xen and VMware implemented subsystems or functions in their hypervisors for VM power control [6]. Xen's hypervisor allows switching between P-states (power-performance state) and

C-states (CPU sleeping states). This is beneficial for the energy efficiency of object virtualized systems [23]. VMware's vSphere (VMware ESXi) supports dynamic voltage and frequency scaling for energy efficiency or other performance requirements. Moreover, vSphere has a subsystem called VMware Distributed Power Management (DPM). VMware DPM has a mechanism to switch off idle servers according to the current monitoring result.

In spite of the practice by Xen and VMware, global task assignment (allocation) units are still lacking in general distributed systems [24], [5]. Thus, resource optimization for workload or energy efficiency becomes questionable. Meanwhile, a recent survey on task assignment in cloud computing [25] clearly identified that energy efficiency (power optimization) has been a focus of research. However, most of the current efforts focused on one-layer schemes, rather than a two-layer framework [12], [7].

Moreover, researchers have also recently examined task assignment [26], [27], [28], [29], [30]. This includes task assignment on mobile networks for load balancing [26]. Dynamic cloud task assignment is investigated in [27] through a propagation-based method. Time efficiency [28] and energy efficiency [29] are also studied for resource management. Furthermore, an energy-aware industrial-independent task assignment framework is developed in [30]. It conducts task consolidation for energy optimization as a third-party, global control unit. However, none of the methods mentioned above consider task assignment (to VMs) and VM placement (to PMs) simultaneously in a unified framework. As future intelligent energy systems should follow standardization for scalability [31], such a unified framework helps achieve standardized energy control and optimization in different scales of data centers.

E. Technical Gaps and Motivations

Two major technical gaps are identified. First, although various algorithms and schemes have been developed at the application assignment level or VM placement level, there is a lack of a unified framework that considers both levels simultaneously for energy-efficient virtual resource management. Second, profiles have been used to help improve energy efficiency at the application assignment level. However, the characteristics of computing tasks and jobs have not been well captured and employed for energy-efficiency virtual resource management. These two major gaps motivate our work in this paper for energy-efficient virtual resource management.

We do not aim to develop a resource allocation algorithm at either the application assignment level or the VM placement level. Instead, we present a profile-guided three-phase framework for unified virtual resource management across the application assignment and VM placement levels. This is achieved through task classification and problem reduction, both of which are driven by the extraction of rich information from profiles. Any specific resource allocation algorithms, such as FFD, can used as plugins for task assignment, or VM placement, or both in the proposed three-phase framework.

III. Problem Formulation and Reduction

This section starts with power modeling and energy modelling of PMs, i.e., CPUs. Then, the virtual resource management problem for energy efficiency is formulated as a constrained optimization problem. After our formulation, the problem is reduced through VM and PM perspectives, according to our insights into the Google data set. Symbols and notations used in this paper are listed in Table I.

TABLE I: Symbols and notations.

α	a constraint indicating the shape of P_{CPU}
E_j	energy consumption of the jth PM
$\mathbb{E}$	total power consumption of a data center
i, j	indices to indicate the ith VM and jth PM
k	index to indicate the kth time interval
N_c	the number of combinations in an exhaustive search
n_V, n_P	the numbers of VMs and active PMs, respectively
n_T	the number of time intervals
n_t	the number of tasks
n_{lvm}	the number of VMs that host long-running (persistent) tasks
n_{nvm}	the number of VMs that host normal tasks
n_{lpm}	the number of PMs that host long-running VMs
n_{rpm}	the number of the rest PMs other than n_{lpm}
n_{tt}	the number of tiny tasks
n_{tvm}	the number of VMs that host tiny tasks
P	power
t	time
t_{lp}, t_{np}	the times of the last and next placements, respectively
t_{pe}, t_{ps}	the end time and start time of a long-running task, respectively
T_m	duration between two planned placements
T_p	lifetime of a long-running task
u	CPU utilization
u_{tt}	requested computing resource of a tiny task
u_{vm}	computing resource allocated to a VM
V, V_i	the set of all VMs $V = \bigcup_{i=1}^{n_V} V_i$, and the ith VM, respectively

A. Problem Formulation

Power Modeling of CPUs. A CPU power model is related to CPU utilization, which is also known as CPU load [32]. For virtual resource management in data centers, a CPU power model aims to characterize the relationship between CPU utilization and its power consumption. A typical CPU power model reported in [32] is described as:

$$P_{CPU} = P^{(max)} - (P^{(max)} - P^{(min)}) \cdot e^{-\alpha \cdot u}, \quad (1)$$

where u stands for CPU utilization, $P^{(max)}$ is close to the maximum power when CPU utilization is 100%, $P^{(min)}$ is the minimum power when CPU utilization is 0%, and constant α defines the shape of the power model curve.

Energy Modeling of CPUs over Time. Generally, each PM hosts multiple VMs. However, these multiple VMs often run in different time slots. Consequently, the workload of the PM varies over the planned time intervals. To reduce the complexity of the problem, it is assumed that each VM has the same workload over time. Eventually, the utilization of a CPU over any time interval, divided by the start time and the end time of each VM, is stable. Thus, if the jth PM is divided into n_T time intervals during the lifetimes of the hosted VMs, the energy consumption E_j of the PM is calculated as:

$$E_j = \sum_{k=1}^{n_T} P_{jk} \cdot t_{jk}, \quad (2)$$

where t_{jk} represents the duration of the kth time interval and, P_{jk} stands for the instantaneous power of the jth PM in the time interval. P_{jk} is calculated from the power model (1).

Total Energy Consumption Model. Considering all PMs that host VMs, a unified total energy consumption model for a data center integrates the CPU power model and the CPU energy model. For a total number of n_P active PMs, with E_j calculated from Eq. (2), the total energy consumption $\mathbb{E}$ is:

$$\mathbb{E} = \sum_{j=1}^{n_P} E_j. \tag{3}$$

Task Assignment Model. When a task is submitted, it is packed into a VM, and then the VM is placed in a PM. As the size of a VM is not arbitrary in a data center, i.e., $u_v \in \{u_{v1}, u_{v2}, ..., u_{vm}\}$, a task with resource requirement u_{tk} has to be assigned to the fittest VM to satisfy:

$$u_{vn} < u_{tk} \leq u_{v(n+1)},\ n = 0, 1, ..., m-1. \tag{4}$$

Let n_{jk} denote the number of VMs placed to the jth PM during time slot k. Then, VM placement in PMs satisfies:

$$u_{jk} = \sum_{i=1}^{n_{jk}} u_{ijk},\ u_{ijk} \in \{u_{v1}, u_{v2}, ..., u_{vm}\}. \tag{5}$$

Constrained Optimization. Virtual resource management is responsible for packing computing tasks into VMs (application assignment) and placing VMs on PMs (VM placement). For energy-efficient data centers, it aims to minimize the energy consumption over a certain period of time subject to CPU and memory constraints. Therefore, virtual resource management is formulated as the following constrained optimization problem with respect to the set of virtual machine set $V = \bigcup_{i=1}^{n_V} V_i$:

$$\begin{cases} \min\limits_{V} \mathbb{E} = \sum\limits_{j=1}^{n_P} \sum\limits_{k=1}^{n_T} \left[P_j^{(max)} - \dfrac{P_j^{(max)} - P_j^{(min)}}{e^{\alpha \cdot u_{jk}}} \right] t_{jk} \\ \text{s.t.}\ \ u_{jk} = \sum_{i=1}^{n_{jk}} u_{ijk},\ u_{ijk} \in \{u_{v1}, u_{v2}, ..., u_{vm}\}, \\ \quad\ \ 0 \leq u^{(min)} \leq \forall u_{jk} \leq u^{(max)} \leq 100\%, \\ \quad\ \ 0 \leq \forall u_{jk\_memory} \leq 100\%, \end{cases} \tag{6}$$

where $j = 1, \cdots, n_P; k = 1, \cdots, n_T$. Now, we focus on solving this constrained optimization problem.

B. Insights into Problem Reduction

The optimization in Eq. (6) is a combinatorial optimization, which is NP-hard. The complexity of the optimization problem is characterized by the total number, N_c, of all possible combinations to be examined in an exhaustive search. For a data center with n_P PMs that host n_V VMs, we have

$$N_c = (n_P)^{(n_V)}. \tag{7}$$

If each task is packed into a separate VM, we have $n_V = n_t$ and thus, $N_c = (n_P)^{(n_t)}$. For a small-scale data center with $N_p = 100$ and $n_V = 500$, we have $N_c = 10^{1,000}$, implying that the search space for Eq. (6) is too big. Some simple heuristic VM placement strategies, e.g., FFD, can give a solution quickly but sacrifice the quality of solution.

Insights from Google's Data Set. Our analysis of the Cluster-usage Traces of a Google data center [11] shows that jobs or tasks differ in their behaviors and resource requirements. Firstly, long-running jobs keep running continuously for hours or days, while other jobs are executed for only minutes. To help reduce the search space for Eq. (6), we may place long-running jobs separately in a few stacks of PMs. Secondly, among all tasks other than long-running ones, some require only a small amount of CPU resources, e.g., 0.01% CPU resources. We call these tasks tiny tasks. Tiny tasks are too small to be allocated into a single VM. If we pack multiple tiny tasks into fewer VMs, the number of VMs will be reduced. A reduced number of VMs helps reduce the search space as well for Eq. (6). Thus, our problem reduction is from the VM and PM perspectives.

Problem Reduction from the VM Perspective. If the whole set of VMs in Eq. (6) is decomposed into several smaller sets, and these decomposed smaller sets of VMs are placed independently, the optimization are split into multiple smaller and independent ones. As the complexity of solving such bin-packing problems is often not less than $O(n \log n)$ [22], solving multiple smaller problems is easier than solving the original problem if the results are close to each other. As a result, the complexity of the optimization in Eq. (6) will be significantly reduced. This requires the knowledge of the profiles of the tasks or applications, VMs, and PMs [12], [7].

Our analysis of data center profiles including Google cloud traces [11] indicates that a significant percentage, e.g., 20%, of jobs and tasks are persistent and long-running. There are also a huge number of tiny jobs and tasks that are excuted for a short period of time. Other jobs and tasks are relatively big but run for minutes or hours. Thus, we consider classifying VMs into three groups: n_{pvm} VMs for long-running tasks, n_{nvm} VMs for normal tasks, and n_{tvm} VMs for tiny tasks, $n_V = n_{pvm} + n_{nvm} + n_{tvm}$. Then, Eq. (7) becomes:

$$N_c = (n_P)^{(n_{lvm})} \cdot (n_P)^{(n_{tvm}+n_{nvm})}. \tag{8}$$

With this task classification, if long-running and other VMs are placed separately, they are decoupled from each other. Thus, the problem complexity characterized by N_C is reduced to

$$N_c \to (n_P)^{(n_{lvm})} \text{ and } (n_P)^{(n_{tvm}+n_{nvm})}. \tag{9}$$

Furthermore, multiple tiny tasks are packed in a single VM[12], [7]. Thus, the number of VMs hosting tiny tasks is much small than that hosting tiny tasks, i.e., $n_{tvm} < n_{tt}$. Consequently, the total number of VMs is lower than the total number of tasks; i.e., $n_V < n_t$. As a results, the problem size N_c is also reduced:

$$N_c < (n_p)^{(n_V)} < (n_p)^{(n_t)}. \tag{10}$$

Thus, the input scale of the problem is reduced significantly.

Problem Reduction from the PM Perspective. As the VMs for long-running tasks are treated independently, they are placed on n_{lpm} PMs rather than on n_P PMs, $n_{lpm} < n_P$. Then, the remaining VMs hosting other tasks are placed on n_{rpm} PMs, where $n_{rpm} \leq n_P$. Therefore, Eq. (9) is further reduced to:

$$N_c \to (n_{lpm})^{(n_{lvm})} \text{ and } (n_{rpm})^{(n_{tvm}+n_{nvm})}. \tag{11}$$

This is a significant reduction from the original Eq. (7). Although the problem is still NP-hard, its search space is greatly reduced. Therefore, task classification followed by successive VM placement plays an important role in our constrained optimization of energy-efficient virtual resource management of data centers.

IV. A Three-Phase Framework

Making use of the knowledge of profiles, a three-phase virtual resource management framework is presented in this section for energy-efficient virtual resource management of data centers. The framework is graphically depicted in Fig. 1.

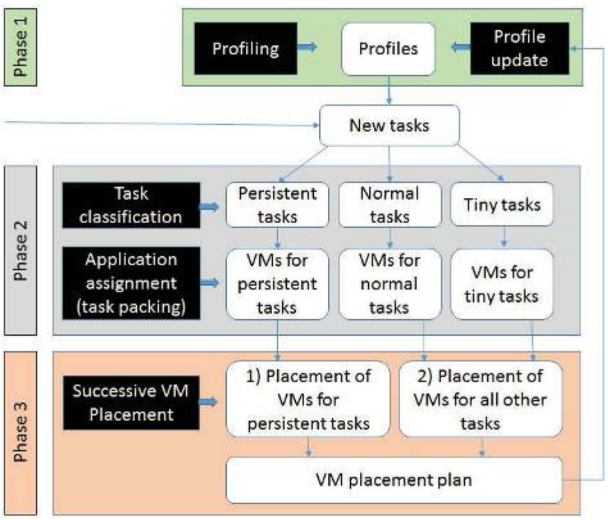


Fig. 1: The architecture of our three-phase framework.

Phase 1: Profiling and Profile Update. Profiling is conducted in two stages: profile building and analysis. Building profiles is a process of recording, extracting, and forming machine-readable data sets of computing tasks, VMs, and PMs. Analyzing the resulting data sets means exacting useful information about constraints and utilization of resources, as well as the characteristics of the tasks, VMs and PMs. Although profiling has been discussed previously [7], [12], the objective and applications of profiles are different in the present paper. We focus on virtual resource management through task classification derived from profiles.

Table II shows examples of a job profile and a task profile. A job may be recorded in multiple measurement periods. A long-running job is identified if it runs in a large or infinite number of consecutive periods. For tasks, we observed from data center logs that for the CPU usage and the CPU request, the peak CPU Usage is occasionally higher than the CPU request. This implies that the corresponding task occasionally consumes more CPU resources than the requested ones.

After the profiles are built, further analysis is conducted to extract useful information for managing the virtual resources. Then, the total CPU usage and request, as well as other information, become known. Thus, a rough estimate of the required number of PMs for hosting all VMs packed with all tasks is revealed. For instance, if the CPU request is $300-500$ normalized CPU cores, and each PM in the data center has 0.5 normalized CPU cores, then approximately $600-1000$ PMs are required for the requested CPU resources.

TABLE II: Overview of job and task profiles.

Job profile	Task Profile
Job ID	Job ID
Number of tasks	Task serial No.
Start time of the job	Start time of the task
End time of the job	End time of the task
Task CPU request	CPU request, peak CPU usage of this task, ID of related local PM

Phase 2: Task Classification and Assignment. From the insights developed previously for problem reduction, all tasks are classified into three classes: long-running, normal, and tiny tasks. These classes are quantified later in the simulations.

With three separate classes for all tasks, long-running tasks are packed into VMs that are primarily for long-running tasks. Then, normal tasks are assigned to VMs that are primarily for normal tasks. After that, tiny tasks are placed into the VMs that are used for long-running and normal tasks. If these VMs are not sufficient for all tiny tasks, new VMs are created to host all remaining tiny tasks that have not yet been assigned.

Phase 3: Successive VM Placement. For energy-efficient VM placement, all PMs are sorted in descending order in terms of energy efficiency. Then, the PMs with higher energy efficiency are filled up first. This leads to a solution with satisfactory energy efficiency for virtual resource management.

In the successive VM placement, the first step is to place all VMs with long-running tasks on PMs. The number of PMs required to host these VMs is much lower than the PMs to host all VMs. The corresponding search space is $(n_{ppm})^{(n_{pvm})}$.

The second step is to place the VMs with normal tasks on PMs successively. Some, but not all, PMs that have already been assigned VMs with long-running tasks may still have space to host more VMs. Thus, the search space in this step is less than $(n_P)^{(n_{nvm}+n_{tvm})}$.

The third step is to place all remaining VMs with only tiny tasks on PMs. Then, a plan for virtual resource management is obtained for implementation.

Algorithm Design. Algorithms are designed for the profile-guided three-phase framework shown in Fig. 1 for energy-efficient virtual resource management of data centers. Algorithms 1 and 2 are for phases 2 and 3, respectively.

The first part of Algorithm 1 shown in lines 1 to 7 is for classification of all tasks or applications. The information extracted from the task profiles includes lifetimes and CPU requests of all tasks. If the lifetime of a task is quite long (line 2), this task is classified into the long-running task class (line 3). Otherwise, if the CPU request of a task is far below the smallest VM size (line 4), the task is a tiny task (line 5). All other remaining tasks are normal tasks (lines 6 to 7).

The second part of Algorithm 1 is application assignment in lines 8 to 21. It assigns long-running tasks to VMs first (lines 8 to 10). Then, normal tasks are assigned to VMs of proper sizes (lines 11 to 13). After that, for each of the tiny tasks (line 14), the algorithm tries to assign the task to existing VMs that are filled with tasks (lines 15 to 17). If no such VM is found (line 18), this task is assigned to a new VM of proper size

Algorithm 1: Task classification and assignment.

Input: All tasks (applications).
Output: A plan for application assignment to VMs.
Initialize: an empty long-running task set, an empty normal task set, and an empty tiny task set; an empty long-running VM set, an empty normal VM set, and an empty tiny VM set.

1 **foreach** *task in all given tasks* **do**
2   **if** $T_p \gg T_m$ **then**
3   Append this task to the long-running task set;
4   **else if** $u_{tt} \ll \forall VMcapacity$ **then**
5   Append this task to the tiny task set;
6   **else**
7   Append this task to the normal task set;
8 **foreach** *task in the long-running task set* **do**
9   Assign this task to a VM of proper size;
10   Append this VM to the long-running VM set;
11 **foreach** *task in the normal task set* **do**
12   Assign this task to a VM of proper size;
13   Append this VM to the normal VM set;
14 **foreach** *task in the tiny task set* **do**
15   **foreach** *VM in the long-running, normal, and tiny VM sets* **do**
16   **if** $u_{vm} + u_{tt} < Capacity$ **then**
17   Assign this task to this VM; **break**;
18   **else**
19   Assign this task to a new VM of proper size;
20   Append this VM to the tiny VM set;
21 **return** *with an application assignment plan*;

Algorithm 2: Successive VM placement.

Input: All VMs filled with tasks (applications).
Output: A plan for VM placement on PMs

1 **forall** *Long-running VMs* **do**
2   Place them on PMs via a placement policy, e.g., FFD;
3 **forall** *Normal VMs and Tiny VMs* **do**
4   Place them on PMs via a placement policy, e.g., FFD;
5 **foreach** *PM of all PMs* **do**
6   **if** *this PM is not placed on any VM* **then**
7   Switch off this PM to save energy;
8 **return** *with a VM placement plan*;

(line 19), and this new VM is appended to the tiny VM set (line 20). After all tasks are assigned to VMs, the algorithm is terminated with an application assignment plan (line 21).

Successive VM placement in Algorithm 2 starts with placement of the VMs packed with long-running tasks (lines 1 and 2). This is followed by placement of the VMs that host normal and tiny tasks (lines 3 and 4). After all VMs are placed, the algorithm looks through all PMs (line 5). If a PM is not filled with a VM (line 6), it is marked to switch off to save energy (line 7). After all PMs are checked, the algorithm is terminated in line 8 with a VM placement plan.

TABLE III: Settings of data centers, PMs, and VMs.

	PART 1: Three scales of data centers					
# PMs	Large-scale	Medium-Scale	Small-scale			
High-performing PMs	10	6	0			
Medium-performing PMs	15	9	2			
Low-performing PMs	600	300	150			
	PART 2: Three types of PMs					
	High-, medium- and low-performing PMs					
For each PM	High	Medium	Low			
CPU capacity	1.5	1.0	0.5			
# CPUs	48	32	16			
Max Power/CPU (Watt)	220	250	280			
Min Power/CPU (Watt)	160	160	160			
Energy efficiency	High	Medium	Low			
	PART 3: Six types of VMs					
For each VM	Huge	Large	Medium	Normal	Small	Tiny
Normalized CPU capacity	0.45	0.3	0.15	0.10	0.045	0.015

V. Simulation Experiments

Our experiments aim to evaluate the effectiveness of the proposed three-phase framework. The underlying strategy for application assignment and VM placement uses the popular FFD. We evaluate the energy consumption savings achieved with the integration of this three-phase framework. The simulation results include an application assignment plan and a VM placement plan, the number of active PMs required to host all VMs packed with all computing tasks, the total energy consumption of a data center over the evaluation period, and the computing times of the virtual resource allocation using FFD with and without using the proposed framework.

The experiments test small-, medium-, and large-scale data centers. The Google Cluster-Usage Traces are used to simulate computing tasks with normalized resources (including CPU) measurement [11]. The data center records for a month are huge. For the demonstration in this paper, we sample data records only for a period of about 24 hours. More specifically, we sample a record from every 200 records of the Google Cluster-Usage Traces to form a data set from a large-scale data center. Similarly, we sample a record from every 400 records to form a data set for a medium-scale data center. For a small-scale data center, we sample a record from every $1,000$ records to form a data set.

The settings of the three types of data centers equipped with different types of PMs are summarized in Table III. The distribution of high-, medium-, and low-performing PMs is based on the proportion extracted from the Google Cluster-Usage Traces. For each CPU, the power model in Eq. (1) [32] is used. It is seen from Part 2 of Table III that a high-performing PM has high energy efficiency. This information is used in sorting all PMs in descending order. Moreover, the VMs in these experiments are configured with a few fixed types as in commercial Amazon EC2.

From our design, jobs and tasks are classified into long-running, normal, and tiny ones. For a long-running task, let T_p, t_{ps}, and t_{pe} denote its lifetime, start time, and end time, respectively. In addition, let T_m denote the time duration, i.e., the measurement period, between two planned placements. Moreover, use t_{lp} and t_{np} to denote the times of the last and next placements, respectively. Then, a long-running task behaves with

$$T_m \ll T_p,\ t_{ps} \ll t_{lp},\ t_{np} \ll t_{pe}. \tag{12}$$

After the long-running tasks are separated, the next focus is on the separation of the large number of tiny tasks. These tiny tasks typically have a lifetime of less than a normalized measurement period, a CPU request less than a given threshold, and a tiny system utilization, e.g., 0.01% normalized CPU. Let u_{tt} be the CPU resource request of such a tiny task, and u_{vm} be the allocated size of a VM. A tiny task implies that

$$u_{tt} \ll u_{vm}. \tag{13}$$

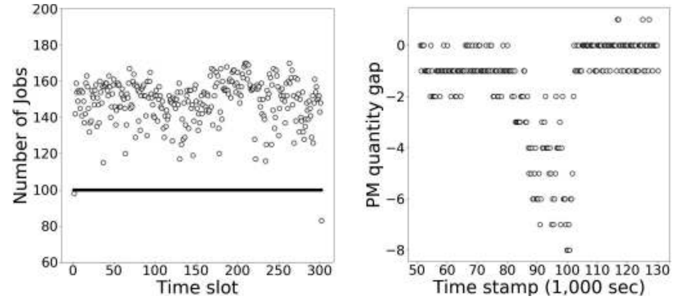
In our design, multiple tiny tasks are packed in a VM, implying that $n_{tvm} \ll n_{tt}$. After long-running and tiny tasks are separated, the tasks left over are normal tasks. Each normal or long-running task is assigned to a single VM.

The simulation uses the Google data set as input tasks. It also integrates the three-phase framework incorporating with FFD as the underlying VM placement algorithm. The output of each run of the simulations is a placement plan with estimated energy consumption. The simulations are conducted on a desktop computer. The computer is equipped with an Intel Core 2 Q6700 3.4 GHz CPU and 32 GB DDR4 2666 MHz RAM. It runs the Windows 10 Professional operating system.

Results for a Small-scale Data Center. The profiling and task classification results for a small-scale data center are depicted in Fig. 2. A quantitative analysis of Fig. 2a shows that the total number of jobs at one time is in the range of 120 to 160 over the observed period. Among these jobs, there are 100 long-running jobs. All tiny tasks are assigned to a few VMs. By using our three-phase framework with underlying FFD for VM placement, savings are achieved in the number of PMs to host all VMs as shown in Fig. 2b. The savings are in comparison with standard FFD without integration of the three-phase framework. It reaches 8 at peak time. This translates into energy savings from 110 kWh to 101 kWh, implying an 8% drop.

Results for a Medium-scale Data Center. For the medium-scale data center, there are 200 to 300 jobs at a time over the observed period. Among these jobs, 100 jobs are long-running ones, and the others are tiny and normal jobs. Similar to the small-scale data center scenario above, experiments are conducted to evaluate the proposed three-phase framework with underlying FFD for VM placement. Fig. 3 shows the number of PMs saved from the framework in comparison with the standard FFD without the use of the framework. The highest number of PMs that the framework can save at peak time is recorded as 19. This translates into significant energy savings. A quantitative analysis reveals a reduction in energy consumption from 190 kWh to 167 kwh, indicating 12% savings.

Results for a Large-scale Data Center. For the large-scale data center, there are 350 to 450 jobs at a time over the observed period. Among these jobs, 100 jobs are long-running ones. For tiny tasks, Fig. 4a shows how many VMs are created



(a) Long-running job classification (b) The number of saved PMs

Fig. 2: Results for a small-scale data center.

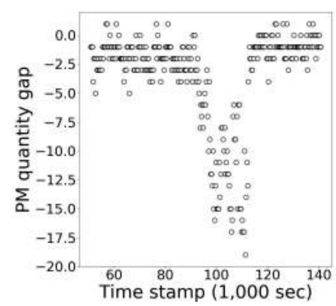
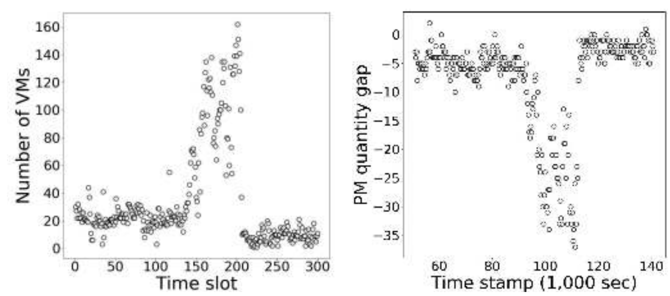


Fig. 3: Medium-scale data center: the number of saved PMs.



(a) VMs hosting tiny tasks (b) The number of saved PMs

Fig. 4: Results for a large-scale data center.

to host the jobs. The three-phase framework is implemented with FFD as the underlying strategy for task assignment and VM placement. In comparison with the standard FFD without using the framework, we achieved significant savings of the number of PMs for the data center, as shown in Fig. 4b. At peak time, we use 37 fewer PMs. This leads to a decrease in the energy consumption from 440 kWh to 386 kWh over the observed period, i.e., a 12% reduction.

Energy Savings. A quantitative evaluation is carried out for energy savings for the small-, medium-, and large-scale data centers under consideration. The energy savings are derived from the framework embedded with FFD in comparison with the standard FFD without using the framework. The power model in Eq. (1) is used to compute the energy consumption. The comparisons of the total energy savings over the observed period are summarized in Table IV. The energy savings achieves 8% for the small-scale data center, and reaches 12% for medium- and large-scale data centers. According to a

TABLE IV: Energy savings.

Data center ⟹	Small-scale	Medium-scale	Large-scale
Standard FFD	110 kWh	190 kWh	440 kWh
The framework with FFD	101 kWh	167 kWh	386 kWh
Energy saving	8%	12%	12%

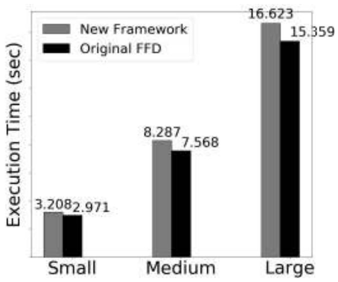


Fig. 5: Execution time versus the input scale measured in the number of data records.

Huawei report, a data center with a power usage effectiveness (PUE) of 2 costs about $15 million annually for energy consumption in servers. This implies that a 12% energy savings means an annual decrease of about $1.8 million for electricity.

Scalability. The computing times of FFD with integration of the framework are recorded for small-, medium-, and large-scale data centers. They are compared with the computing times of standard FFD without the use of this framework. The results are shown in Fig. 5. The computing times of the standard FFD grow linearly with the increase in the input scale. FFD integrated with this framework also behaves with a linear increase in the computing time with the increase in the input scale, indicating good scalability of the proposed framework. Although FFD integrated with this framework consumes a little more time to execute than the standard FFD, the difference in the computing times of these two methods is small, e.g., 16.6225 sec versus 15.359 sec for $95,331$ input records of data traces. It is negligible in practical applications.

VI. Conclusion

Improving energy efficiency through virtual resource management was investigated for data centers. It was mathematically described as a constrained optimization problem. Due to the significant complexity, the optimization problem was simplified for reducing the complexity and improving the energy efficiency. By extracting useful information from the profiles of the computing tasks, VMs, and PMs, a profile-guided three-phase framework has been presented with algorithm implementation for a solution to the optimization problem. This simulation experiments showed energy savings of 8% to 12% from the proposed framework over the existing technique for various scales of data centers. This result implies a significant decrease in operational costs for data centers.

References

[1] S. Yin, X. Li, H. Gao, and O. Kaynak, “Data-based techniques focused on modern industry: An overview,” *IEEE Trans. Ind. Electron.*, vol. 62, no. 1, pp. 657–667, 2015.

[2] C. Möbius, W. Dargie, and A. Schill, “Power consumption estimation models for processors, virtual machines, and servers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1600–1614, 2014.

[3] S. Kumar and M. Pandey, “Energy aware resource management for cloud data centers,” *Int. Journal of Computer Science and Information Security*, vol. 14, no. 7, p. 844, 2016.

[4] J. Barr. (2015) Cloud computing, server utilization, & the environment. [Online]. Available: https://aws.amazon.com/es/blogs/aws/cloud-computing-server-utilization-the-environment/

[5] W.-Z. Zhang, H.-C. Xie, and C.-H. Hsu, “Automatic memory control of multiple virtual machines on a consolidated server,” *IEEE Transactions on Cloud Computing*, no. 1, pp. 2–14, 2017.

[6] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya *et al.*, “A taxonomy and survey of energy-efficient data centers and cloud computing systems,” *Advances in Computers*, vol. 82, no. 2, pp. 47–111, 2011.

[7] M. Vasudevan, Y.-C. Tian, M. Tang, and E. Kozan, “Profile-based application assignment for greener and more energy-efficient data centers,” *Future Generation Computer Systems*, vol. 67, pp. 94–108, 2017.

[8] X. Dai, J. M. Wang, and B. Bensaou, “Energy-efficient virtual machines scheduling in multi-tenant data centers,” *IEEE Trans. Cloud Computing*, vol. 4, no. 2, pp. 210–221, Apr-Jun 2016.

[9] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[10] 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 and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[11] C. Reiss, J. Wilkes, and J. L. Hellerstein, “Google cluster-usage traces: format+ schema,” *Google Inc., White Paper*, pp. 1–14, 2011.

[12] M. Vasudevan, Y.-C. Tian, M. Tang, E. Kozan, and W. Zhang, “Profile-based dynamic application assignment with a repairing genetic algorithm for greener data centers,” *Journal of Supercomputing*, vol. 73, no. 9, pp. 3977–3998, Sep 2017.

[13] C. J. Kraenzel, J. D. Immerman, W. A. Mills, M. A. Nowacki, and J. J. Lu, “System and method for managing concurrent scheduled or on-demand replication of subscriptions,” 18 Jan 2005, US Patent 6,845,383.

[14] C. Zhao and J. Liu, “A virtual machine dynamic consolidation algorithm based dynamic complementation and ffd algorithm,” in *Fifth Int. Conf. on Commun. Syst. and Netwk. Techn. (CSNT)*, 2015, pp. 333–338.

[15] T. N. Malleswari, G. Vadivu, and D. Malathi, “Live virtual machine migration techniquesa technical survey,” in *Intelligent Computing, Communication and Devices*. Springer, 2015, pp. 303–319.

[16] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, F. Xia, and S. A. Madani, “Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues,” *The Journal of Supercomputing*, vol. 71, no. 7, pp. 2473–2515, 2015.

[17] C. Thraves and L. Wang, “Power-efficient assignment of virtual machines to physical machines,” in *First Int. Workshop on Adaptive Resource Management and Scheduling for Cloud Computing (ARMS-CC)*, vol. 8907, Paris, France, 15 July 2014, p. 71.

[18] Z. Liu, Y. Xiang, and X. Qu, “Towards optimal CPU frequency and different workload for multi-objective vm allocation,” in *12th Ann. IEEE Consumer Commun. and Networking Conf. (CCNC)*, 2015, pp. 367–372.

[19] P. Lama, “Autonomic performance and power control in virtualized datacenters,” Ph.D. dissertation, University of Colorado, Colorado Springs, CO USA, 2007.

[20] W. Zhang, S. Han, H. He, and H. Chen, “Network-aware virtual machine migration in an overcommitted cloud,” *Future Generation Computer Systems*, vol. 76, pp. 428–442, 2017.

[21] C. Sonkiln, M. Tang, , and Y.-C. Tian, “A decrease-and-conquer genetic algorithm for energy efficient virtual machine placement in data centers,” in *IEEE 15th Int. Conf. Ind. Inform. (INDIN2017)*, Eden, Germany, 24-26 July 2017.

[22] W. F. De La Vega and G. S. Lueker, “Bin packing can be solved within 1+ ε in linear time,” *Combinatorica*, vol. 1, no. 4, pp. 349–355, 1981.

[23] J. N. Matthews, E. M. Dow, T. Deshane, W. Hu, J. Bongio, P. F. Wilbur, and B. Johnson, *Running Xen: a hands-on guide to the art of virtualization*. Prentice Hall PTR, 2008.

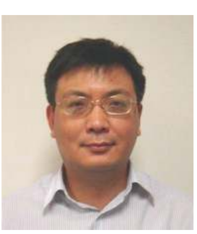
[24] Y. Jiang, “A survey of task allocation and load balancing in distributed systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 585–599, 2016.

[25] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu *et al.*, “A survey and taxonomy on energy efficient resource allocation

techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, 2016.
[26] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, 2015.
[27] Y. Kong, M. Zhang, and D. Ye, "A belief propagation-based method for task allocation in open and dynamic cloud environments," *Knowledge-Based Systems*, vol. 115, pp. 123–132, 2017.
[28] S. K. Mishra, M. A. Khan, B. Sahoo, and S. K. Jena, "Time efficient task allocation in cloud computing environment," in *2nd Int. Conf. for Convergence in Technology (I2CT)*, Mumbai, India, 7-9 April 2017, pp. 715–720.
[29] S. K. Mishra, D. Puthal, B. Sahoo, P. P. Jayaraman, S. Jun, A. Y. Zomaya, and R. Ranjan, "Energy-efficient VM-placement in cloud data center," *Sustainable Computing: Informatics and Systems*, 2018, in press, DOI: 10.1016/j.suscom.2018.01.002.
[30] B. Primas, P. Garraghan, D. McKee, J. Summers, and J. Xu, "A framework and task allocation analysis for infrastructure independent energy-efficient scheduling in cloud data centers," in *IEEE Int. Conf. on Cloud Computing Tech. & Sci. (CloudCom)*, 2017, pp. 178–185.
[31] T. Strasser, F. Andrén, J. Kathan, C. Cecati, C. Buccella, P. Siano, P. Leitao, G. Zhabelova, V. Vyatkin, P. Vrba *et al.*, "A review of architectures and concepts for intelligence in future electric energy systems," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2424–2438, 2015.
[32] D. Versick, I. Waßmann, and D. Tavangarian, "Power consumption estimation of CPU and peripheral components in virtual machines," *ACM SIGAPP Applied Computing Rev.*, vol. 13, no. 3, pp. 17–25, 2013.

Zhe Ding received the Master by Research degree in information technology from Queensland University of Technology, Brisbane QLD, Australia, in 2017, where he is currently a Ph.D. candidate with the School of Electrical Engineering and Computer Science. His research interests include Big-Data processing, virtualization, and cloud computing.

Yu-Chu Tian (M'00) received the Ph.D. degree in computer and software engineering from the University of Sydney, Sydney NSW, Australia, in 2009 and the Ph.D. degree in industrial automation from Zhejiang University, Hangzhou, China, in 1993. He is currently a Professor with the School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane QLD, Australia. His research interests include big data computing, distributed computing, cloud computing, real-time computing, computer networks, and control systems.



Maolin Tang (M'04—SM'10) received the Ph.D. degree in computer systems engineering from Edith Cowan University, Perth, Australia. He is currently a Senior Lecturer with the School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane QLD, Australia. His current research interests include evolutionary computation and cloud computing, and the applications of evolutionary computation in cloud computing.

Yuefeng Li received the Ph.D. degree in computer science from Deakin University, Melbourne VIC, Australia, in 2001. He is currently a Professor with the School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane QLD, Australia. His research interests include data mining, text mining, topic modelling, ontology Learning and Web Intelligence, and knowledge and data engineering.

You-Gan Wang received the Ph.D. degree in statistics from Oxford University, Oxford, United Kingdom, in 1991. He is currently a Professor with the School of Mathematical Sciences, Queensland University of Technology. Brisbane QLD, Australia/ His research interests include statistical modelling, data science, machine learning, and their applications in medical sciences and natural resource management.

Chunjie Zhou received the Ph.D. degree in control science and engineering from Huazhong University of Science and Technology, Wuhan, China, in 2001, where he is currently a Professor with the School of Automation. His research interests include industrial communications, artificial intelligence, networked control systems, and security of industrial control systems.