Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 7040276, 10 pages
http://dx.doi.org/10.1155/2016/7040276

Research Article

A Heuristic Task Scheduling Algorithm for Heterogeneous Virtual Clusters

Weiwei Lin,[1] Wentai Wu,[1] and James Z. Wang[2]

[1]*School of Computer Science and Engineering, South China University of Technology, Guangdong, China*
[2]*School of Computing, Clemson University, P.O. Box 340974, Clemson, SC 29634-0974, USA*

Correspondence should be addressed to Wentai Wu; cswuwt@mail.scut.edu.cn

Received 27 January 2016; Accepted 20 April 2016

Academic Editor: Ligang He

Cloud computing provides on-demand computing and storage services with high performance and high scalability. However, the rising energy consumption of cloud data centers has become a prominent problem. In this paper, we first introduce an energy-aware framework for task scheduling in virtual clusters. The framework consists of a task resource requirements prediction module, an energy estimate module, and a scheduler with a task buffer. Secondly, based on this framework, we propose a virtual machine power efficiency-aware greedy scheduling algorithm (VPEGS). As a heuristic algorithm, VPEGS estimates task energy by considering factors including task resource demands, VM power efficiency, and server workload before scheduling tasks in a greedy manner. We simulated a heterogeneous VM cluster and conducted experiment to evaluate the effectiveness of VPEGS. Simulation results show that VPEGS effectively reduced total energy consumption by more than 20% without producing large scheduling overheads. With the similar heuristic ideology, it outperformed Min-Min and RASA with respect to energy saving by about 29% and 28%, respectively.

1. Introduction

Cloud computing gains its popularity since it satisfies the elastic demands of computing capability from both individual and enterprise users. Cloud platforms not only support a diversity of applications, but also provide a virtualized environment for the applications to run in an efficient and low-cost manner [1]. As cloud computing is getting prevailing in IT industry, the huge amount of electricity consumed by cloud data centers also becomes a rising concern. According to the previous statistics, globally there are over 5 million data centers [2], which account for about 1.5% of the global energy consumption [3]. The figure may continuously go up as our demands for computing are still growing. Hence, in order to minimize the negative impact brought by energy wasting and overconsumption, it is of great necessity to improve resource utilization and to reduce energy consumption for cloud data centers.

Applying energy-aware resource scheduling is an effective way to save energy. Cloud data centers are usually virtualized. Thus in an IaaS (Infrastructure-as-a-Service) cloud, virtual machine (VM) is the basic unit for resource provisioning. After a user-defined job is submitted, it is first "sliced" into a number of tasks and generally each task will be assigned to one VM for execution. During the execution, the virtual resources allocated to the VM can be thought of being occupied by the task. The mapping from tasks to VMs is one-to-one. On the one hand, we do not consider a many-to-one mapping because resource competition often causes SLA (Service-Level Agreement) violations. On the other hand, one-to-many mapping can be avoided by a fine-grained job decomposition. Although the jobs or tasks may not contain any attributes initially, we can exploit available techniques to estimate their resources demands including total instructions, amount of disk I/O, and the data throughput on network. Besides, to attain the goal of saving task execution energy, it is of great necessity to consider servers' power efficiency. Assigning tasks to high-performance servers may enhance the data centers' overall performance but at the same time can cause extra energy

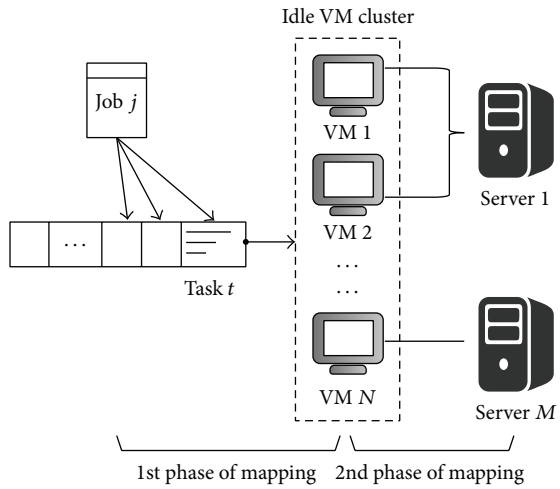

Figure 1: Resource scheduling in IaaS cloud.

consumption (i.e., operational cost). This is because some servers of high processing speed may not be power-efficient. Hence, we argue that the power efficiency of servers and VMs should be regarded as an important metric in today's energy-aware resource management.

Resource scheduling can be separated into two phases: task scheduling and VM scheduling. The first phase of mapping shown in Figure 1 represents task scheduling, which is the focus of this paper. Previous task scheduling algorithms (e.g., [4–6]) allocated tasks directly to physical servers. However, these algorithms are not rather feasible and effective since currently virtualization has been widely deployed on physical servers. The running environment for tasks is the virtual cluster. Besides, the majority of task scheduling algorithms use the strategy that VM is dynamically created on a selected server only when new tasks arrive. This kind of strategy is useful to aggregate workload in order to avoid too many idle servers. But it lowers the system's response ability as powering on a new VM takes time. An ideal target for task scheduling is to reduce system energy consumption with acceptable efficiency. Thus, in this paper, we propose to build a virtual cluster maintenance mechanism which combines VM "precreating" and "delayed shutdown." To be detailed, "precreating" means virtual machine can be started up on servers under relatively light workload before tasks arrive. "Delayed shutdown" allows a VM to stay alive for a certain period after it finishes its task. In cloud environment, this mechanism can maintain a large-scale idle VM cluster and thus allows a shorter task response time without bringing big overhead cost. At the same time, this mechanism is helpful to reduce migration operations, so it can be used to simplify VM consolidation (the 2nd mapping phase in Figure 1) strategies such as [7, 8].

Supported by VM "precreating" and "delayed shutdown" mechanism, we in this paper propose an energy-aware task scheduling framework for virtualized cloud environment. The framework consists of a task resource requirements prediction module, an energy estimate module, and a scheduler with a task buffer. The buffer works as an improvement on simple FIFO queue of arriving tasks. The size of the buffer is designed to be adaptive to the arrival rate of tasks. Receiving the output from the task resource requirements prediction module, task energy estimate module is responsible for estimating the energy consumption of executing. As the key part, the scheduler adopts a VM power efficiency-aware greedy scheduling algorithm (VPEGS) to schedule the tasks in the buffer heuristically. Experiments were conducted to evaluate the performance of VPEGS in a simulated heterogeneous virtual cluster. The results show that VPEGS averagely reduced more than 20% energy consumption and outperformed Min-Min [9], RASA [10], and Random-Mapping [11].

2. Related Work

Task scheduling has been proved to be a NP-problem [12]. Even with the mechanism of VM "precreating" and "delayed shutdown," task scheduling in a heterogeneous cloud is still a nontrivial problem. Heuristic scheduling algorithms such as Min-Min [9] and ant colony optimization [13, 14] are widely used in cloud task scheduling because they are quite efficient and sometimes able to approach optimal solutions [15]. Min-Min is a typical task scheduling algorithm oriented to heterogeneous infrastructures. Gutierrez-Garcia and Sim [11] compare 14 heuristic scheduling algorithms with respect to average task makespan. The results show that Min-Min and Max-Min [9] are the most effective among the algorithms using batch mode. Besides, Etminani and Naghibzadeh [16] proved that dynamically selecting Min-Min or Max-Min as the scheduler according to the standard deviation of expected task execution time can improve system performance. Priya and Subramani [10] propose a heuristic scheduling named RASA that consists of 3 phases. In initialization phase the execution efficiency matrix is initialized, while the scheduler finds the best-fit VM and returns its ID in the second and third phase. The idea of RASA is using Min-Min and Max-Min alternatively to schedule the tasks that arrived. Uddin et al. [17] tested and analyzed the performance of RASA, TPPC, and PALB in CloudSim considering power efficiency and cost as well as CO_2 emissions. They concluded that TPPC is most effective but neglected the detailed parameter settings of these algorithms.

Cloud servers are usually virtualized. Thus it is of great necessity to perform task scheduling in virtual clusters. Sampaio and Barbosa propose POFARE [4], considering both VM reliability and efficiency. This heuristic algorithm promotes the energy utilization (MFLOPS/Joules) but pays no attention to server virtualization. Lakra and Yadav [18] conducted task scheduling by solving a multiobjective optimization via nondominated sorting after quantifying the QoS values of tasks and VMs. However, it has the drawbacks of not being energy-aware and evaluating VM performance merely by MIPS (Million Instructions per Second). VM consolidation is another effective way to save energy with the basic ideology that powering off idle servers can reduce energy consumption. For example, HHCS [19], an energy-saving scheduling strategy, makes use of the advantages of two open-source schedulers (Condor and Haizea) in order

to further increase CPU utilization of physical servers. In addition, there are also implements (e.g., [20–22]) based on setting thresholds and constrains. Ding et al. [23] adopt this method to perform resource provisioning at the VM-level. Current technology allows dynamic VM migration, which is helpful to balance workload between servers. However, VM migration causes extra time and energy overheads. Hence, it is a better scheme to precreate and maintain a number of VMs on servers under light workload. Then these idle VMs can respond quickly when a new batch of tasks arrives.

3. Energy-Aware Task Scheduling Framework

3.1. Energy Estimate Module. In an IaaS cloud, virtualization makes physical resources "transparent" as the applications are run in VMs. To some extent, virtual machine provides independent runtime environment and it is also the basic unit allocated to user applications. In the proposed framework, the energy estimate module predicts the expected task energy consumption on each available VM and sends the data to the scheduler. For energy estimation, the required information includes task resource demands and the power efficiency of each VM.

Job submitted to the cloud will first be decomposed into several tasks. The decomposition principle can be data-based or function-based. Practically, total number of instructions and I/O data size can be estimated by analyzing the submitted code or exploiting other existing techniques. Actually there are many ways to estimate the resource demands of a task. The methods mentioned in [24] can be applied to process I/O-intensive tasks while, according to [25], the required amount of resources by the tasks belonging to the same job are usually similar. In this paper, we use four "static" attributes to profile a task: number of instructions, the size of data through disk input/output, the size of data through network transmission, and job_id indicating the job it is generated from. The values of these attributes remain unchanged despite the decisions of the scheduler. On the contrary, "dynamic" attributes, including the execution time and energy consumption of a task, are dependent on the features of the VM that executes it.

VM's power features are directly related to the features of its host. According to the definition of power efficiency, the power efficiency of a server can be defined in three aspects:

$$\begin{aligned} \mathrm{PE}_{\mathrm{proc}} &= \frac{\mathrm{proc\_perf}}{P_{\mathrm{proc}}}, \\ \mathrm{PE}_{\mathrm{io}} &= \frac{\mathrm{io\_rate}}{P_{\mathrm{io}}}, \\ \mathrm{PE}_{\mathrm{trans}} &= \frac{\mathrm{trans\_rate}}{P_{\mathrm{trans}}}, \end{aligned} \tag{1}$$

where proc_perf denotes the processor performance, which can be quantified using MIPS (Million Instructions per Second). io_rate and trans_rate represent the max disk I/O rate and max network transmission rate, respectively. Their metric is MB/s. P_{proc}, P_{io}, and P_{trans} are the power consumption of the corresponding functional components. All these data can be sampled on physical servers (e.g., we can obtain P_{proc} by measuring CPU power). It is worth noting that $\mathrm{PE}_{\mathrm{trans}}$ denotes the power efficiency in transport data between servers and it is stored in a matrix. $\mathrm{PE}_{\mathrm{trans}}$ are exploited to calculate the energy cost in multitask communications. In order to shield the complexity of network, we use a simple star network topology in designing the way that tasks communicate with each other, assuming that only tasks decomposed from the same job will conduct data transport between each other. We select one of them to be a "designated task" and other tasks follow the principle that they only send data to or receive data from the "designated task" (Figure 2).

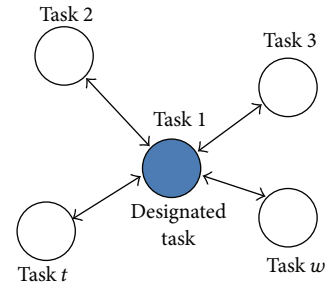

Figure 2: Simplified data exchanging by appointing designated task.

There exists a difference between the power efficiency of a VM and its host server because of virtualization. For example, different types of hypervisors suffer different degrees of degradation in VM performance. We use d_{proc}, d_{io}, and d_{trans} to represent the degradation in VM power efficiency of processing, disk I/O, and data transmission, respectively. Thus the power efficiency of a VM can be expressed as below:

$$\begin{aligned} \mathrm{PE}'_{\mathrm{proc}} &= \mathrm{PE}_{\mathrm{proc}} \cdot \left(1 - d_{\mathrm{proc}}\right), \\ \mathrm{PE}'_{\mathrm{io}} &= \mathrm{PE}_{\mathrm{io}} \cdot \left(1 - d_{\mathrm{io}}\right), \\ \mathrm{PE}'_{\mathrm{trans}} &= \mathrm{PE}_{\mathrm{trans}} \cdot \left(1 - d_{\mathrm{trans}}\right). \end{aligned} \tag{2}$$

As a summary, Table 1 lists the power features of VMs.

The dynamic power consumption of cloud data centers is mainly produced from the workload on each running server, while the resource demands of tasks are the major sources that drive server workloads. In cloud environment, the demands of tasks can be generally modeled by the task attributes mentioned above. However, it is very difficult to precisely predict the workload as a whole because actually a server has several components (e.g., CPU, memory, disk, and NIC) that keep producing static (idle) and dynamic power. Thus a possible way is to consider the workload of each component separately. We adopted this ideology and propose to calculate separately the power of computing, storage accessing, and communicating. Particularly in this paper we take the load of the whole server into account and use it to model performance loss.

Let P'_{proc}, P'_{io}, and P'_{trans} denote the VM's power consumption in processing, disk I/O, and network data transfer, respectively. We assume that VMs stay busy when executing the tasks assigned. So we regard P'_{proc}, P'_{io}, and P'_{trans}

Table 1: VM power features.

VM power features	Description
host_id	Indicates the VM's host server
proc_perf′	Max processing speed of the VM
io_rate′	Max disk I/O rate of the VM
trans_rate′	Max data transfer rate of the VM
PE'_{proc}	VM power efficiency (processor)
PE'_{io}	VM power efficiency (disk I/O)
PE'_{trans}	VM power efficiency (data transfer)
d_{proc}	PE_{cal} degradation
d_{io}	PE_{io} degradation
d_{trans}	PE_{trans} degradation

as unchanged values during execution. Considering task resource demands, VM power features, and the workload on host servers, we can estimate the energy consumption of a task run on a VM via

$$\begin{aligned} \text{task\_energy} &= E_{\text{proc}} + E_{\text{io}} + E_{\text{trans}} \\ &= P'_{\text{proc}} \cdot T_{\text{proc}} + P'_{\text{io}} \cdot T_{\text{io}} + P'_{\text{trans}} \cdot T_{\text{trans}}, \end{aligned} \tag{3}$$

where

$$\begin{aligned} T_{\text{proc}} &= \frac{N_{\text{proc}}}{\text{proc\_perf}' \cdot (1 - L)}, \\ T_{\text{io}} &= \frac{N_{\text{io}}}{\text{io\_rate}' \cdot (1 - L)}, \\ T_{\text{trans}} &= \frac{N_{\text{trans}}}{\text{trans\_rate}' \cdot (1 - L)}, \end{aligned} \tag{4}$$

where N_{proc} denotes the number of instructions, while N_{io} and N_{trans} represent the amount of disk data throughput and the amount of data transferred through network, respectively. These task attributes can be estimated by existing techniques. L is the performance loss caused by high workload on the server. It is intuitive that the higher the load a server works under, the greater value L has. The correlation between the workload of CPU and other components is quite complex, but there is a basic knowledge that the performance of the whole system probably degrades when CPU is working under high load. So as a simplification, we model L as follows:

$$L = \begin{cases} u^{1/\beta}, & 0 \le u \le 0.95, \\ 1, & 0.95 < u < 1, \end{cases} \tag{5}$$

where u represents the current CPU utilization of the host server. β is the high-load penalty factor and $\beta \in (0, 1]$. With (3) and (4), we finally have

$$\text{task\_energy} = \left(\frac{N_{\text{proc}}}{PE'_{\text{proc}}} + \frac{N_{\text{io}}}{PE'_{\text{io}}} + \frac{N_{\text{trans}}}{PE'_{\text{trans}}} \right) \cdot \frac{1}{1 - L}, \tag{6}$$

where PE'_{proc}, PE'_{io}, and PE'_{trans} represent the power efficiency of VM regarding instructions processing, disk I/O, and data transmission. From (6) we can see that assigning tasks to virtual machines with high power efficiency is of great significance to reduce energy consumption. Meanwhile, the workload on servers should also be considered because high load leads to great performance degradation, which increases the energy required to finish a task.

3.2. Task Buffer. There are two methods to determine the scheduling order: FIFO mode and buffer mode (or batch mode). In completely FIFO mode, all tasks are organized and scheduled sequentially according to the arrival time. Thus FIFO mode provides best fairness but may fail to satisfy the QoS (Quality of Service) of some specific tasks. As an improvement, buffer mode allows buffering a certain number of tasks and schedules them by some principles. Buffer mode is similar to priority queue but it is not global, which guarantees the scheduler's efficiency and enhances its effectiveness at the same time. Algorithms that adopt buffer or batch mode include Min-Min, Max-Min [9], and RASA [10]. Practically, it is not easy to determine the buffer's size because oversized buffer causes low efficiency while making it too small may reduce the chance to find better scheduling solutions.

In this paper, a variable-sized task buffer is adopted on the basis of a global FIFO queue. To be more detailed, tasks at the head of the FIFO queue are put in the buffer then their minimum energy consumption (relevant to currently available VMs) will be estimated. Tasks with lower predicted energy consumption will be scheduled with higher priorities. Assume that the arriving of cloud tasks is a Poisson process with its intensity equal to λ; then the expectation of task arrival interval is $1/\lambda$. Hence, it is a feasible way to set the buffer size to a multiple of λ (and round it):

$$\text{buf\_size} = \lceil \alpha \cdot \lambda \rceil, \tag{7}$$

where α is a system parameter that can be set empirically. Increasing the size of buffer is helpful to find better (more energy-saving) scheduling solutions when tasks arrive intensively. Meanwhile a smaller buffer can make the scheduler more efficient in the condition that the arrival rate is relatively low.

3.3. Task Scheduling Framework. Now we briefly depict the entire energy-aware task scheduling framework. After being submitted to the cloud, users' jobs are first decomposed into several tasks. These tasks are put in a FIFO queue and then those at the head are transferred to the task buffer. The energy estimate module is in charge of estimating the energy consumption of each task in the buffer. After receiving the output from energy estimate module, the scheduler finishes the scheduling of this batch of tasks. Then the next batch is pushed into the buffer and the above process is repeated. Figure 3 illustrates the whole energy-aware task scheduling framework.

The algorithm inside the scheduler is the key part for making energy-saving task allocations. Thus we propose an energy-saving heuristic task scheduling algorithm.

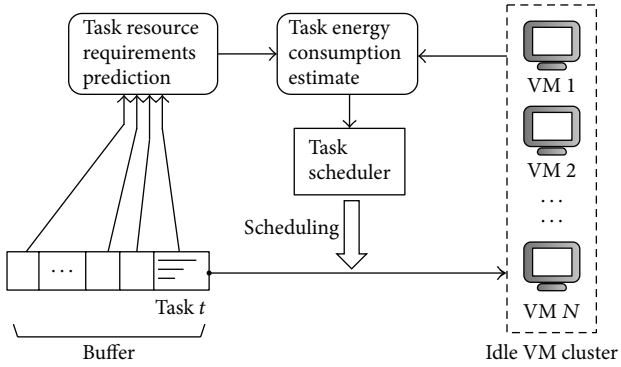

Figure 3: Energy-aware task scheduling framework.

Table 2: List of parameters used in the algorithm's pseudocode.

Parameter	Description
V	The set of currently available VMs
M	The set of physical servers
Buffer	Task buffer
Q	Global task queue (FIFO)
buf_size	The size of task buffer

4. VM Power Efficiency-Aware Greedy Scheduling Algorithm (VPEGS)

With the expansion of cloud data centers and the increase of computing demands from users, it is of great significance to consider the heterogeneity of both infrastructures and task demands. Currently, many researches (e.g., [23, 26]) only cast their sight on VM consolidation because it is an effective way to reduce wasted energy by controlling the workload on servers. However, if much load is imposed on servers with low power efficiency, it will cause higher energy cost to warrant the QoS of tasks, which is the situation that service providers are unwilling to face.

A feasible and effective solution is to consider power efficiency in task scheduling. In virtualized environment, colocated VMs can be regarded to have equal power efficiency, which can be calculated by applying (2). Thus, assuming that the infrastructure supports VM precreating and delayed shutdown, we propose a virtual machine power efficiency-aware greedy scheduling algorithm (VPEGS). The algorithm takes VM power efficiency and task demands into account and provides a sort of energy-saving task scheduling. We first list the parameters used in the algorithm and give brief descriptions (Table 2).

VPEGS is heuristic and takes the estimated task execution energy as the evaluation function. We exploit (6) to estimate the execution energy consumption ($\text{task\_energy}_{t,k}$) of task t on VM k, considering VM efficiency, efficiency loss caused by virtualization, and the performance loss caused by high server workload. Since we adopt task buffer, the process of scheduling is similar to Min-Min and RASA. In other words, the program attempt to search the buffer for a (t^*, k^*) satisfies

$$\text{task\_energy}_{t^*,k^*} = \min\left\{\text{task\_energy}_{t,k}\right\}, \quad (8)$$

```
Input: V, M, Q
Output: task-to-VM Mapping
(1)  Initialize Buffer
(2)  Initialize min_energy = MAX_FLOAT
(3)  while Q is not empty do
(4)    for i = 1 to min{size(Q), buf_size} do
(5)      t = dequeue(Q)
(6)      add t into Buffer
(7)    end
(8)    while Buffer is not empty do
(9)      for each task t in Buffer do
(10)       for each VM k in V do
(11)         calculate task_energy_{t,k}
(12)         if task_energy_{t,k} < min_energy then
(13)           min_energy = task_energy_{t,k}
(14)           selected_task = t
(15)           selected_VM = k
(16)         end if
(17)       end for
(18)     end for
(19)     assign selected_task to selected_VM
(20)     remove task t from Buffer
(21)     update the states of V and M
(22)   end while
(23) end while
(24) return task-to-VM Mapping
```

Algorithm 1: Virtual machine power efficiency-aware greedy scheduling algorithm (VPEGS).

where $t = 0, 1, \ldots, (\text{buf\_size} - 1)$ and $k = 0, 1, \ldots, n$. n is the number of VMs currently available. Then in this round, the scheduler assigns task t to VM k. The pseudocode of VPEGS is shown in Algorithm 1.

The task buffer is initialized first and then the global FIFO queue which dequeues the tasks at the head. After the buffer is filled (or FIFO queue becomes empty), the scheduler computes $\text{task\_energy}_{t,k}$ for each task t on every available VM k (line (11)). Its time complexity equals inspecting a $\text{buf\_size} * n$ sized matrix. The minimum element is found and the corresponding VM ID and task ID are recorded (lines (14)~(15)). Then the selected task is assigned to the selected VM. This process repeats until the buffer is clear. Then the next batch of tasks will be sent into it.

We analyze the complexity of VPEGS as below: each decision of assignment has to check the whole matrix whose size is $\text{buf\_size} * n$, so the complexity of assigning one task is $O(\text{buf\_size} * n)$. Suppose the total number of tasks that arrived is u. Thus the overall time complexity of finishing the scheduling is $O(u * \text{buf\_size} * n)$.

5. Algorithm Evaluation

5.1. Experimental Setup. We implemented VPEGS and evaluated it in a simulated environment. We also implemented Min-Min [9], RASA [10], and Random-Mapping [11] in order to compare their effectiveness. The algorithms and test programs were written in Java (JDK version 1.8.0_65).

Table 3: The setting of task attributes in the experiment.

Task attribute	Value
N_{proc}	$\sim U(170, 510)$
N_{io}	$\sim U(100, 3000)$
N_{trans}	$\sim U(100, 2000)$
Threads	{1, 2, 3, 4}

Table 4: The setting of server configurations in the experiment.

Type	PE_{proc}	PE_{io}	d_{proc}	d_{io}	d_{trans}	Number
Server 0	0.33	10.50	0.2	0.2	0.1	20
Server 1	0.15	9.00	0.1	0.3	0.1	8
Server 2	0.17	9.50	0.1	0.2	0.1	20
Server 3	0.28	21.50	0.1	0.1	0.1	12
Server 4	0.21	15.50	0.2	0.2	0.1	40

The simulation was run on a PC equipped with a dual-core Pentium CPU (2.10 GHz) and 4.0 GB memory.

For every task decomposed from a job, the experimental setting of its attributes is listed in Table 3. Where the metric of N_{proc} is Million instructions, while the unit of N_{io} and N_{trans} is MB. In order to simulate the difference of power efficiency between heterogeneous physical servers, we set 5 types of servers and the corresponding configurations are listed in Table 4. In the experiment, we suppose the power efficiency of data transfer is infinity (i.e., zero overhead) if two VMs are server-local. Otherwise, it equals 50. Meanwhile, the task with the smallest task_id is always appointed to be the "designated task" when multiple tasks that belong to the same job are active in the virtual cluster. Thus the elements in PE_{trans} matrix are defined as

$$\text{PE}_{\text{trans}}(i, j) = \begin{cases} +\infty, & i = j, \\ 50, & i \neq j. \end{cases} \tag{9}$$

5.2. Experimental Results. In the experiment we set the number of servers (m) to 100 according to Table 4 fixedly. The program randomly generated 250 to 300 VMs in the initialization phase. High-load penalty factor β was set to 0.15. The intervals of task arrivals followed exponential distribution with $\lambda = 3$ and the buffer size was set to 15 initially. After initialization, the test program utilized VPEGS, Min-Min, RASA, and Random-Mapping (RM) separately as the scheduling strategy to run the simulation. Each test repeated 30 times and we took the average as our results. The comparison regarding total system energy consumption is shown in Figure 4.

The result illustrates that VPEGS performed the best among the four scheduling algorithms with respect to energy saving (Figure 4). Min-Min and RASA had similar performance since their heuristic principles behind are similar. VPEGS saved 29.1% and 28.6% energy when compared to them on average. As for the reason, we argue that Min-Min and RASA in some way can be energy-saving because shortening overall execution time reduces the consumption

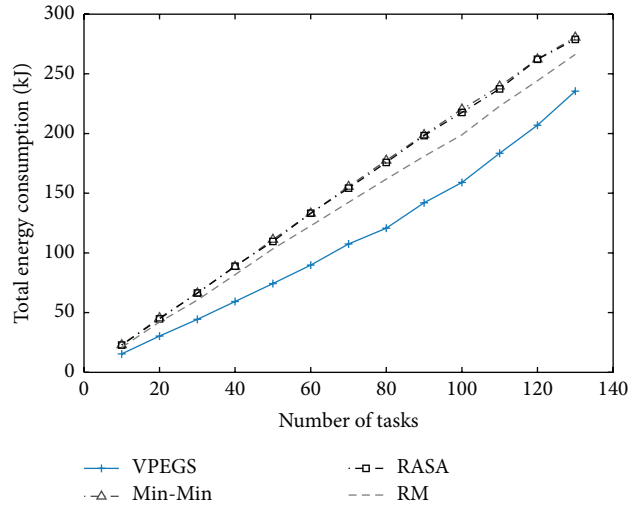

Figure 4: Comparing the performances of VPEGS, Min-Min, RASA, and RM on total energy consumption (buffer size = 15).

brought by server idle power. However, as power efficiency is not taken into account, assigning tasks to those high-performance nodes may cause extra energy consumption. On the contrast, VPEGS considers both the performance and power features of VMs and exploits power efficiency as the prime metric. Specifically speaking, Min-Min and RASA are more likely to utilize the servers with great processing speed or throughput rate, whereas VPEGS prefers those with high power efficiency. In our experiment, actually a big number of high-performance servers were of comparatively low power efficiency. As a result, VPEGS showed its advantage in energy saving. It is also noticed that Random-Mapping (RM) seemed to be slightly more energy-saving than Min-Min and RASA. Essentially this is because RM assigns tasks evenly so usually high workload would not be imposed on servers with low power efficiency. Averagely, VPEGS outperforms Random-Mapping by about 23.0%.

We also see that when the number of servers is fixed, it seems to be tougher to maintain energy-saving performance as the number of tasks increases (Figure 5). When virtual resources are sufficient to satisfying tasks' demands, using VPEGS can reduce total energy consumption by more than 20%. However, as the task arrival rate remained unchanged ($\lambda = 3$), the workload of the whole cluster went high as the total number of tasks increased. In other words, comparatively energy-efficient VMs were gradually used up. Onto the fixed-scale simulated data centers with 100 heterogeneous servers, the performance of VPEGS degraded in our experiment when the number of tasks is more than 90 (Figure 5).

We mentioned that the size of task buffer may influence the performances of scheduling algorithms. To verify it, we changed the task arrival rate, namely, λ, to 6. Correspondingly, the size of buffer was adjusted to 30. We reinitialized the clusters and conducted the experiment again. Figures 6 and 7 show the results. In this case, compared with Min-Min, RASA, and RM, VPEGS averagely saved 29.8%, 29.0%, and

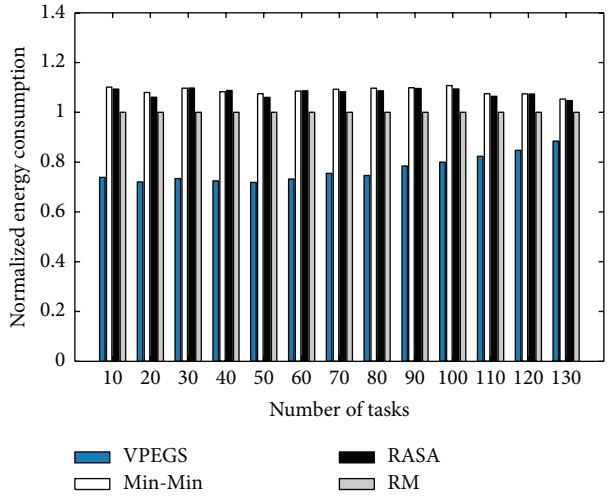

Figure 5: Normalized energy consumptions with different total number of tasks (buffer size = 15).

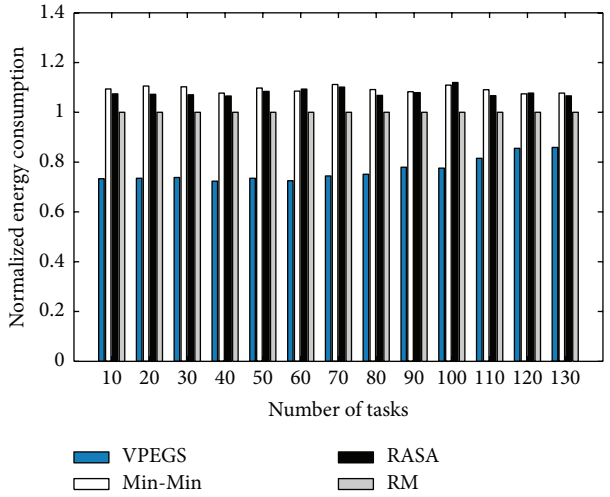

Figure 7: Normalized energy consumptions with different total number of tasks (buffer size = 30).

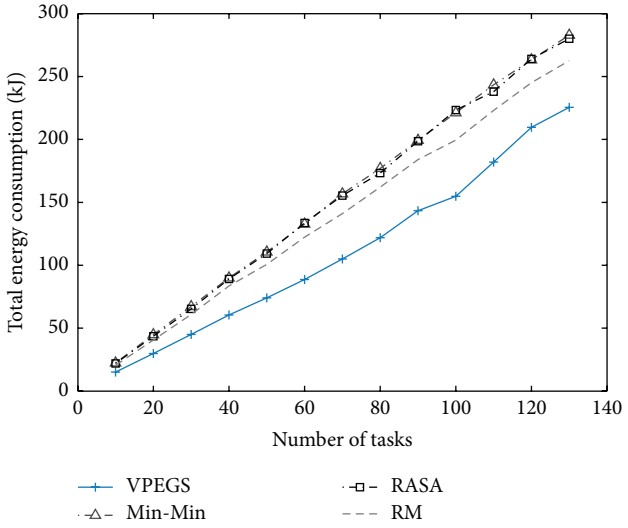

Figure 6: Comparing the performances of VPEGS, Min-Min, RASA, and RM on total energy consumption (buffer size = 30).

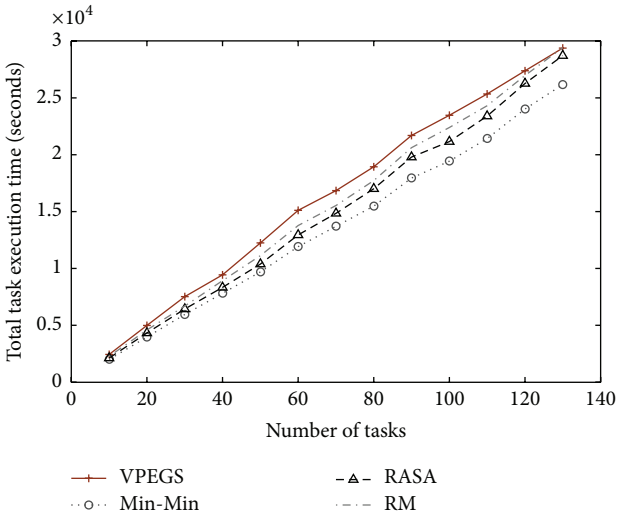

Figure 8: Comparing different task scheduling algorithms on the total task execution time (buffer size = 15).

23.3% energy, respectively. It is a little surprising to find that enlarging the task buffer does not have big impact. But we also see that the performance of VPEGS in this case was slightly improved when the number of tasks exceeded 100.

Tasks may not gain their earliest completion time in VPEGS since energy consumption is considered primarily. There is a kind of conflict, as mentioned in [27], between optimizing execution time and energy consumption. VPEGS, to some degree, sacrifices the efficiency of task execution to attain the goal of saving more energy. However, Min-Min and RASA pay more attention to reducing total makespan and task execution time. Figure 8 shows the total task execution time of Min-Min, RASA, Random-Mapping, and VPEGS with the buffer size equal to 15. As a result, Min-Min and RASA are effective in shortening the overall execution time of all the tasks. The reason is simple: Min-Min and RASA take predicted task completion time as the heuristic. Besides, short tasks outnumbered long tasks in our experiment; thus RASA yielded no better performance in reducing total execution time than pure Min-Min. We ran the test again after changing the task arrival rate and the buffer's size (Figure 9). Combining Figure 8 and Figure 9, we can see that the change of buffer size did not affect the total task execution time for VPEGS. But the time for RASA was shortened when we reduced the number of tasks per batch (Figure 9). This is because when the number of tasks in the buffer was reduced, RASA was more likely to make the same decisions with Min-Min.

We also carried out experiments to test the impact of the buffer's size on scheduling overheads (Figure 10). Scheduling overhead represents the average time that the scheduler takes to make a task assignment decision. We use the average time

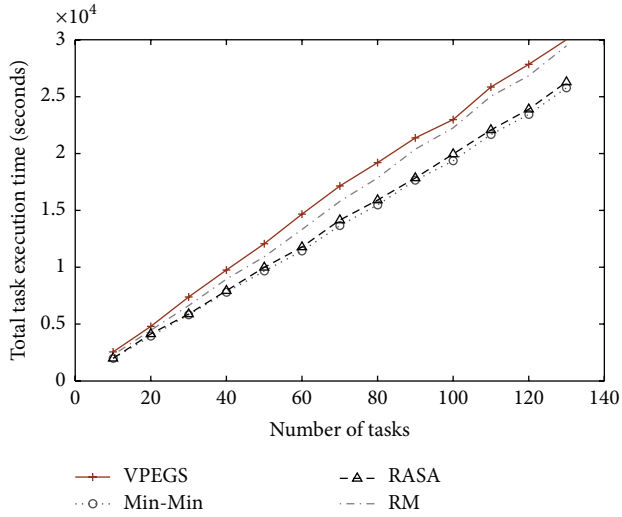

Figure 9: Comparing different task scheduling algorithms on the total task execution time (buffer size = 5).

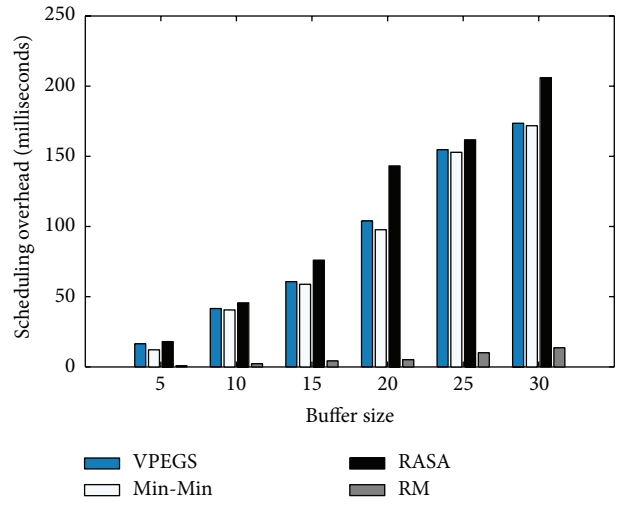

Figure 10: The scheduling overheads of VPEGS, Min-Min, RASA, and RM with different buffer sizes.

that a task stays in the buffer to evaluate the scheduler's efficiency in the experiment.

Though theoretically Min-Min, RASA as well as VPEGS have the same time complexity; it can be pointed out from Figure 10 that Min-Min suffers the least scheduling overheads among these three heuristic algorithms. The reason is that VPEGS spends extra time on estimating task energy. RASA checks whether the number of available VMs is odd before assigning tasks. VPEGS is slightly more efficient than RASA since it does not check the odd-even property and only considers current availability of VMs. In other words, tasks will not wait for occupied VMs in VPEGS even though sometimes waiting helps to shorten the makespan and execution time. On this point, we conclude that VPEGS, as a heuristic algorithm, only suffers small scheduling overheads when adopting an appropriate size of task buffer.

As a summary, the experimental results illustrate that as the scheduler of our proposed energy-aware scheduling framework, VPEGS is effective to schedule tasks in an energy-saving manner. Compared with traditional scheduling algorithms focusing on optimizing overall makespan and task execution time, VPEGS takes into account the task resource demands, VM power efficiency, and server workload. The target of algorithms like Min-Min and RASA are to shorten the makespan and total execution time of a batch of tasks. This is in some way helpful to save system energy when the differences between servers' power efficiencies are small and server idle power makes great impact on the total energy consumption. However, with the fast expanding on data centers' scale, cloud infrastructures probably consist of hundreds of different types of servers. This heterogeneity makes it necessary to consider more factors including server performance, power efficiency, and server workloads. Aiming at reducing the energy consumption of heterogeneous clusters, VPEGS provides a highly feasible way to conduct energy-aware task scheduling. We list the main advantages of VPEGS as follows:

(i) Multiple factors that influence system energy consumption are considered. VPEGS conducts scheduling according to the estimation on task energy, which takes into account the information about task resource demands, VM power efficiency, server workload, and performance loss.

(ii) VPEGS realizes a fine-grained resource provisioning and task scheduling at the level of virtual machine clusters supporting "precreating" and "delayed shutdown."

(iii) VPEGS has high feasibility since it works without any training. Besides, we set the size of the task buffer to an adaptive value to balance the scheduler's performance and efficiency.

(iv) Greedy strategy is made use of to realize low-overhead task scheduling.

6. Conclusion and Future Work

Cloud computing is believed to have great potential in satisfying diverse computing demand from both individuals and enterprises. But at the same time the overconsumption of electricity by cloud data centers becomes a big worry. Considering the virtualized environment in cloud data centers, in this paper, we propose an energy-aware task scheduling framework consisting of a task resource requirements prediction module, an energy estimate module, and a scheduler. Based on this framework, we propose a heuristic task scheduling algorithm named VPEGS. VPEGS takes into account task resource demands, server/VM power efficiency as well as server workload. Oriented to heterogeneous cloud environment, the proposed algorithm does not need training and is able to schedule tasks in an energy-saving manner. VPEGS shares the similar heuristic ideology with Min-Min and RASA, but it prominently saves system energy by sacrificing some efficiency in task execution. Experiment

based on simulation was carried out to evaluate VPEGS. The results illustrate its novelty that VPEGS reduced system energy consumption by over 20% when compared to the strategy of Random-Mapping. It also outperformed Min-Min and RASA in saving energy by approximately 29% and 28%, respectively, without producing large scheduling overheads.

Future research will focus on how to effectively combine task scheduling and VM consolidation strategies in order to further enhance the effectiveness of energy saving. Besides, we plan to make a deeper investigation into the factors or technologies (e.g., Dynamic Voltage Frequency Scaling) that influence server's power efficiency.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

Acknowledgments

Thanks are due to the helpful comments and suggestions from the anonymous reviewers. This work is partially supported by the National Natural Science Foundation of China (Grant no. 61402183), Guangdong Natural Science Foundation (Grant no. S2012030006242), Guangdong Provincial Scientific and Technological Projects (Grants nos. 2016A010101007, 2016B090918021, 2014B010117001, 2014A010103022, 2014A010103008, 2013B090200021, and 2013B010202001), Guangzhou Science and Technology Projects (Grants nos. 201601010047 and 20150504050525159), and Fundamental Research Funds for the Central Universities, SCUT (no. 2015ZZ0098).

References

[1] M. Pedram, "Energy-efficient datacenters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 10, pp. 1465–1484, 2012.

[2] How many data centers?, March 2013, http://www.datacenterknowledge.com/archives/2011/12/14/how-many-data-centers-emerson-says-500000/.

[3] J. Koomey, "Growth in data center electricity use 2005 to 2010," A Report by Analytical Press, 2011.

[4] A. M. Sampaio and J. G. Barbosa, "Towards high-available and energy-efficient virtual computing environments in the cloud," *Future Generation Computer Systems*, vol. 40, pp. 30–43, 2014.

[5] Z. Deng, G. Zeng, Q. He, Y. Zhong, and W. Wang, "Using priced timed automaton to analyse the energy consumption in cloud computing environment," *Cluster Computing*, vol. 17, no. 4, pp. 1295–1307, 2014.

[6] Y. Tian, C. Lin, and K. Li, "Managing performance and power consumption tradeoff for multiple heterogeneous servers in cloud computing," *Cluster Computing*, vol. 17, no. 3, pp. 943–955, 2014.

[7] H. M. Lee, Y.-S. Jeong, and H. J. Jang, "Performance analysis based resource allocation for green cloud computing," *The Journal of Supercomputing*, vol. 69, no. 3, pp. 1013–1026, 2014.

[8] A. Horri, M. S. Mozafari, and G. Dastghaibyfard, "Novel resource allocation algorithms to performance and energy efficiency in cloud computing," *Journal of Supercomputing*, vol. 69, no. 3, pp. 1445–1461, 2014.

[9] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of Parallel & Distributed Computing*, vol. 59, no. 2, pp. 107–131, 1999.

[10] S. M. Priya and B. Subramani, "A new approach for load balancing cloud computing," *International Journal of Engineering and Computer Science*, vol. 2, no. 5, pp. 1636–1640, 2013.

[11] J. O. Gutierrez-Garcia and K. M. Sim, "A family of heuristics for agent-based elastic Cloud bag-of-tasks concurrent scheduling," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1682–1699, 2013.

[12] D. Fernández-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427–1436, 1989.

[13] L. Zhu, Q. Li, and L. He, "Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm," *International Journal of Computer Science Issues*, vol. 9, no. 5, pp. 54–58, 2012.

[14] E. Pacini, C. Mateos, and C. G. Garino, "Balancing throughput and response time in online scientific clouds via ant colony optimization (sp2013/2013/00006)," *Advances in Engineering Software*, vol. 84, pp. 31–47, 2015.

[15] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 275–295, 2015.

[16] K. Etminani and M. Naghibzadeh, "A min-min max-min selective algorithm for grid task scheduling," in *Proceedings of the 3rd IEEE/IFIP International Conference in Central Asia on Internet (ICI '07)*, pp. 1–7, Tashkent, Uzbekistan, September 2007.

[17] M. Uddin, Y. Darabidarabkhani, A. Shah, and J. Memon, "Evaluating power efficient algorithms for efficiency and carbon emissions in cloud data centers: a review," *Renewable & Sustainable Energy Reviews*, vol. 51, pp. 1553–1563, 2015.

[18] A. V. Lakra and D. K. Yadav, "Multi-objective tasks scheduling algorithm for cloud computing throughput optimization," *Procedia Computer Science*, vol. 48, pp. 107–113, 2015.

[19] H. Kurdi and E. T. Alotaibi, "A hybrid approach for scheduling virtual machines in private clouds," *Procedia Computer Science*, vol. 34, pp. 249–256, 2014.

[20] W. Lin, J. Z. Wang, C. Liang, and D. Qi, "A threshold-based dynamic resource allocation scheme for cloud computing," *Procedia Engineering*, vol. 23, pp. 695–703, 2011.

[21] W. Lin, B. Liu, L. Zhu, and D. Qi, "CSP-based resource allocation model and algorithms for energy-efficient cloud computing," *Journal on Communications*, vol. 34, no. 12, pp. 33–41, 2013 (Chinese).

[22] W. Lin, C. Liang, J. Z. Wang, and R. Buyya, "Bandwidth-aware divisible task scheduling for cloud computing," *Software: Practice and Experience*, vol. 44, no. 2, pp. 163–174, 2014.

[23] Y. Ding, X. Qin, L. Liu, and T. Wang, "Energy efficient scheduling of virtual machines in cloud with deadline constraint," *Future Generation Computer Systems*, vol. 50, pp. 62–74, 2015.

[24] Q. Zhao, C. Xiong, C. Yu, C. Zhang, and X. Zhao, "A new energy-aware task scheduling method for data-intensive applications in the cloud," *Journal of Network and Computer Applications*, vol. 59, pp. 14–27, 2016.

[25] X. Bu, J. Rao, and C. Z. Xu, "Interference and locality-aware task scheduling for MapReduce applications in virtual clusters,"

in *Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing (HPDC '13)*, pp. 227–238, ACM, New York, NY, USA, June 2013.

[26] 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.

[27] Y. Wang and X. Wang, "Performance-controlled server consolidation for virtualized data centers with multi-tier applications," *Sustainable Computing: Informatics and Systems*, vol. 4, no. 1, pp. 52–65, 2014.