Energy-Saving Virtual Machine Scheduling in Cloud Computing with Fixed Interval Constraints

Nguyen Quang-Hung(✉), Nguyen Thanh Son, and Nam Thoai

Faculty of Computer Science and Engineering,
Ho Chi Minh City University of Technology,
VNU-HCM 268 Ly Thuong Kiet Street, Ho Chi Minh City, Vietnam
{hungnq2,sonsys,nam}@cse.hcmut.edu.vn

Abstract. Energy efficiency has become an important measurement of scheduling algorithms for Infrastructure-as-a-Service (IaaS) clouds. This paper investigates the energy-efficient virtual machine scheduling problems in IaaS clouds where users request multiple resources in fixed intervals and non-preemption for processing their virtual machines (VMs) and physical machines have bounded capacity resources. Many previous works are based on migration techniques to move on-line VMs from low utilization hosts and turn these hosts off to reduce energy consumption. However, the techniques for migration of VMs could not use in our case. The scheduling problem is NP-hard. Instead of minimizing the number used physical machines, we propose a scheduling algorithm EMinTRE-LDTF to minimize the sum of total busy time of all physical machines that is equivalent to minimize total energy consumption. In this paper, we present the proved approximation in general and special cases of the scheduling problem. Using Feitelson's and Lublin99's parallel workload models in the Parallel Workloads Archive, our simulation results show that algorithm EMinTRE-LDTF could reduce the total energy consumption compared with state-of-the-art algorithms including Tian's Modified First-Fit Decreasing Earliest, Beloglazov's Power-Aware Best-Fit Decreasing and Vector Bin-Packing Norm-based Greedy. Moreover, the EMinTRE-LDTF has less total energy consumption compared with our previous heuristic (e.g. MinDFT) in the simulations.

Keywords: Energy efficiency · Power-aware · Virtual machine · VM placement · VM allocation · IaaS · Scheduling · Cloud computing

1 Introduction

An Infrastructure-as-a-Service (IaaS) cloud system provides users with computing resources in terms of virtual machines (VMs) to run their applications [2,3,12,16,24]. These IaaS cloud systems are often built from virtualized data centers [2,3,24]. Power consumption in a large-scale data center requires multiple megawatts [8,16]. Le et al. [16] estimate the energy cost of a single data center

A. Hameurlain et al. (Eds.): TLDKS XXXI, LNCS 10140, pp. 124–145, 2017.
DOI: 10.1007/978-3-662-54173-9_6

is more than $15M per year. As these data centers have more physical servers, they will consume more energy. Therefore, advanced scheduling techniques for reducing energy consumption of these cloud systems are highly concerned for any cloud providers to reduce energy cost. Increasing energy cost and the need to environmental sustainability address energy efficiency is a hot research topic in cloud systems. Energy-aware scheduling of VMs in IaaS cloud is still challenging [12,16,23,25,27].

Many previous works [3,4,20] proved that the virtual machine allocation is NP-hard and proposed to address the problem of energy-efficient scheduling of VMs in cloud data centers. They [3,4,20] present techniques for consolidating virtual machines in cloud data centers by using bin-packing heuristics (such as First-Fit Decreasing [20], and/or Best-Fit Decreasing [4]). They attempt to minimize the number of running physical machines and to turn off as many idle physical machines as possible. Consider a d-dimensional resource allocation where each user requests a set of virtual machines (VMs). Each VM requires multiple resources (such as CPU, memory, and IO) and a fixed quantity of each resource at a certain time interval. Under this scenario, using a minimum number of physical machines may not be a good solution. In a homogeneous environment where all physical servers are identical, the power consumption of each physical server is linear to its CPU utilization, i.e., a schedule with longer working time will consume more energy than another schedule with shorter working time.

Table 1. Example showing that using a minimum number of physical servers is not optimal. (*: demand resources are normalized to the maximum capacity resources of physical machines).

VM ID	CPU*	RAM*	Network*	Starttime	Dur. (hour)
VM1	0.5	0.1	0.2	0	10
VM2	0.5	0.5	0.2	0	2
VM3	0.2	0.4	0.2	0	2
VM4	0.2	0.4	0.2	0	2
VM5	0.1	0.1	0.1	0	2
VM6	0.5	0.5	0.2	1	9

Our work studies increasing time and resource efficiency-based approach to allocate VMs onto physical machines in other that it minimizes total energy consumption of all physical machines. Each VM requests resource allocation in a fixed starting time and non-preemption for the duration time. We present here an example to demonstrate our ideas to minimize total energy consumption of all physical machines in the VM placement with fixed starting time and duration time. For example, given six virtual machines (VMs) with their resource demands described in Table 1. Note that the maximum capacity of each resource is 1. In the example, a bin-packing-based algorithm could result in a schedule S_1 in which

two physical servers are used: one for allocating VM1, VM3, VM4, and VM5; and another one for allocating VM2 and VM6. The resulted total completion time is (10 + 10) = 20 h. However, in another schedule S_2 in which where VMs are placed on three physical servers, VM1 and VM6 on the first physical server, VM3, VM4 and VM5 on the second physical server, and VM2 on the third physical server, then the total completion time of the five VMs is only (10 + 2 + 2) = 14 h.

This paper presents a proposed heuristic, denoted as EMinTRE-LDTF, to allocate VMs that request multiple resources in the fixed interval time and non-preemption into physical machines to minimize total energy consumption of physical machines while meeting all resource requirements. Using numerical simulations, we compare EMinTRE-LDTF with the state-of-the-art algorithms include Power-Aware Best-Fit Decreasing (PABFD) [4], vector bin-packing norm-based greedy (VBP-Norm-L2) [20], and Modified First-Fit-Decreasing-Earliest (Tian-MFFDE) [26]. Using two parallel workload models [9,17] in the Feitelson's Parallel Workloads Archive [10], our simulation results show that EMinTRE-LDTF could reduce the total energy consumption compared with PABFD [4], VBP-Norm-L2 [20], and Tian-MFFDE [26]. Moreover, EMinTRE-LDTF has less total energy consumption compared with MinDFT-LDTF [21] in the simulations.

The rest of this paper is structured as follows. Section 2 discusses related works. Section 3 describes the energy-aware VM allocation problem with multiple requested resources, fixed starting and duration time. We also formulate the objective of scheduling, and present our theorems. The proposed EMinTRE-LDTF algorithm present in Sect. 4. Section 5 discusses our performance evaluation using simulations. Section 6 concludes this paper and introduces future works.

2 Related Work

The interval scheduling problems have been studied for many years with objective to minimizing total busy time. In 2007, Kovalyov et al. [15] has presented work to describe characteristics of a fixed interval scheduling problem in which each job has fixed starting time, fixed processing time, and is only processed in the fixed duration time on a available machine. The scheduling problem can be applied in other domains. Angelelli et al. [1] considered interval scheduling with a resource constraint in parallel identical machines. The authors proved the decision problem is NP-complete if number of constraint resources in each parallel machine is a fixed number greater than two. Flammini et al. [11] studied using new approach of minimizing total busy time to optical networks application. Tian et al. [26] proposed a Modified First-Fit Decreasing Earliest algorithm, denoted as Tian-MFFDE, for placement of VMs energy efficiency. The Tian-MFFDE sorts list of VMs in queue order by longest their running times first) and places a VM (in the sorted list) to any first available physical machine that has enough VM's requested resources. Our VM placement problem differs from

these interval scheduling problems [1,15,26], where each VM requires for multiple resource (e.g. computing power, physical memory, network bandwidth, etc.) instead of all jobs in the interval scheduling problems are equally on demanded computing resource (i.e. each physical machine can process the maximum of g jobs in concurrently).

Energy-aware resource management in cloud virtualized data centers is critical. Many previous research [3,4,7,14,25] proposed algorithms that consolidate VMs onto a small set of physical machines (PMs) in virtualized datacenters to minimize energy/power consumption of PMs. A group in Microsoft Research [20] has studied first-fit decreasing (FFD) based heuristics for vector bin-packing to minimize number of physical servers in the VM allocation problem. Some other works also proposed meta-heuristic algorithms to minimize the number of physical machines. Beloglazov et al. [3,4] have presented a modified best-fit decreasing heuristic in bin-packing problem, denoted as PABFD, to place a new VM to a host. PABFD sorts all VMs in a decreasing order of CPU utilization and tends to allocate a VM to an active physical server that would take the minimum increase of power consumption. Knauth et al. [14] proposed the OptSched scheduling algorithm to reduce cumulative machine up-time (CMU) by 60.1% and 16.7% in comparison to a round-robin and First-fit. The OptSched uses an minimum of active servers to process a given workload. In a heterogeneous physical machines, the OptSched maps a VM to a first available and the most powerful machine that has enough VM's requested resources. Otherwise, the VM is allocated to a new unused machine. In the VM allocation problem, however, minimizing the number of used physical machines is not equal to minimizing total of total energy consumption of all physical machines. Previous works do not consider multiple resources, fixed starting time and non-preemptive duration time of these VMs. Therefore, it is unsuitable for the power-aware VM allocation considered in this paper, i.g. these previous solutions can not result in a minimized total energy consumption for VM placement problem with certain interval time while still fulfilling the quality-of-service.

Chen et al. [7] observed there exists VM resource utilization patterns. The authors presented an VM allocation algorithm to consolidate complementary VMs with spatial and temporal-awareness in physical machines. They introduce resource efficiency and use norm-based greedy algorithm, which is similar to in [20], to measure distance of each used resource's utilization and maximum capacity of the resource in a host. Their VM allocation algorithm selects a host that minimizes the value of this distance metric to allocate a new VM. Our proposed EMinTRE-LDTF uses a different metric that unifies both increasing time and resource efficiency. In our proposed metric, the increasing time is the difference between two completion time of a host after and before allocating a VM.

Our proposed EMinTRE-LDTF algorithm that differs from these previous works. Our EMinTRE-LDTF algorithm use the VM's fixed starting time and duration to minimize the total busy time on physical machines, and consequently minimize the total energy consumption in all physical servers. To the best of

our knowledge, no existing works that surveyed in [5,13,18,19] have thoroughly considered these aspects in addressing the problem of VM placement.

3 Problem Description

3.1 Notations

We use the following notations in this paper:

vm_i: The i^{th} virtual machine to be scheduled.

M_j: The j^{th} physical server.

S: A feasible schedule.

P^{idle}: The idle power consumption of a physical machine.

P^{max}: The maximum power consumption of a physical machine.

$P_j(t)$: The power consumption of M_j at a time point t.

ts_i: The fixed starting time of vm_i.

d_i: The duration time of vm_i.

T: The maximum schedule length, which is the time that the last virtual machine will be finished.

$\mathscr{J}_j$: The set of virtual machines that are allocated to M_j in the whole schedule.

T_j: The total busy time (working time) of M_j.

e_i: The energy consumption for running vm_i in the physical machine that vm_i is allocated.

g: The maximum number of virtual machines that can be assigned to any physical machine.

3.2 Problem Formulation

Consider the following scheduling problem. We are given a set of n virtual machines $\mathscr{V} = \{vm_1, \ldots, vm_n\}$ to be scheduled on a set of m identical physical servers $\mathscr{M} = \{M_1, \ldots, M_m\}$, each server can host a maximum number of g virtual machines. Each VM needs d-dimensional demand resources in a fixed interval with non-migration. Each vm_i is started at a fixed starting time (ts_i) and is non-preemptive during its duration time (d_i). Types of resource considered in the problem include computing power (i.e., the total Million Instruction Per Seconds (MIPS) of all cores in a physical machine), physical memory (i.e., the total MBytes of RAM in a physical machine), network bandwidth (i.e., the total Kb/s of network bandwidth in a physical machine), and storage (i.e., the total free GBytes of file system in a physical machine), etc.

The objective scheduling is to find out a feasible schedule S to minimize the total energy consumption of m physical servers. The objective scheduling is presented as:

$$\textbf{Minimize } (P^{idle} \times \sum_{j=1}^{m} T_j + \sum_{i=1}^{n} e_i) \tag{1}$$

where T_j is the total busy time of M_j. The $P^{idle} \times T_j$ is the minimum energy consumption of M_j, denoted as E_j^{min}, to keep it is on and active for during its total busy time (T_j), i.e., $E_j^{min} = P^{idle} \times T_j$. The $P^{idle} \times \sum_{j=1}^{m} T_j$ is sum of the minimum energy consumption of m used physical servers. The T_j is defined as the length of union of interval times of all VMs that are allocated to a physical machine M_j. Let $\mathscr{J}_j$ be set of virtual machines that are allocated to M_j in the whole schedule. T_j is defined as following:

$$T_j = len(\bigcup_{vm_i \in \mathscr{J}_j} [ts_i, ts_i + d_i]) \tag{2}$$

The scheduling problem has the following hard constraints, which are firstly described in our previous work [21], as following:

- Constraint 1: Each VM is only processed by a physical server at any time with non-migration and non-preemption.
- Constraint 2: Each VM does not request any resource larger than the maximum total capacity resource of any physical server.
- Constraint 3: The sum of total demand resources of these allocated VMs is less than or equal to the total capacity of the resources of M_j. Each VM is represented as a d-dimensional vector of demand resources, i.e. $vm_i = (x_{i,1}, x_{i,2}, ..., x_{i,d})$. Similarly, each physical machine is denoted as a d-dimensional vector of capacity resources, i.e. $M_j = (y_{j,1}, y_{j,2}, ..., y_{j,d})$. Thus we have $\forall r \in \{1, ..., d\}, i \in \{1, 2, ..., n\}, j \in \{1, 2, ..., m\}$:

$$\sum_{vm_i \in \mathscr{J}_j} x_{i,r} \leq y_{j,r} \tag{3}$$

where:

- $x_{i,r}$ is resource of type r (e.g. CPU core, computing power, memory, etc.) requested by the vm_i (i = 1, 2,..., n).

- $y_{j,r}$ is capacity resource of type r (e.g. CPU core, computing power, memory, etc.) of the physical machine M_j ($j = 1, 2, ..., m$).

With at least one type of resource (i.e., $d \geq 1$), the scheduling problem is NP-hard [20].

3.3 Power Consumption Model

In this paper, we use the following energy consumption model proposed in [8] for a physical machine. The power consumption of M_j, denoted as $P_j(.)$, is formulated as follow $\forall j \in \{1, 2, ..., m\}$:

$$P_j(t) = P^{idle} + (P^{max} - P^{idle})U_j(t) \tag{4}$$

in which $U_j(t)$ is the CPU utilization of M_j at time t, P^{idle} and P^{max} are the idle power and the maximum power consumed at 0% and 100% CPU utilization respectively of a physical machine (all physical machines are homogeneous).

We assume that all cores in CPU are homogeneous, i.e. $\forall c = 1, 2, ..., PE_j : MIPS_{j,c} = MIPS_{j,1}$. The $U_j(t)$ is formulated as follow:

$$U_j(t) = (\frac{1}{PE_j \times MIPS_{j,1}}) \sum_{c=1}^{PE_j} \sum_{vm_i \in \mathscr{J}_j} mips_{i,c} \quad (5)$$

The energy consumption of M_j in the time period $[t_1, t_2]$ is formulated as follow:

$$E_j = \int_{t_1}^{t_2} P_j(U_j(t))dt \quad (6)$$

where:
$U_j(t)$: The CPU utilization of M_j at time t and $0 \leq U_j(t) \leq 1$.
PE_j: The number of processing elements (i.e. cores) of M_j.
$MIPS_{j,c}$: The maximum total computing power (in MIPS) of c^{th} processing element on M_j.
$mips_{i,c}$: The allocated MIPS of the c^{th} processing element to vm_i by M_j.

3.4 Preliminaries

Definition 1 (Length of intervals). *Given a time interval $I = [s, f]$, the length of I is $len(I) = f - s$. Extensively, to a set $\mathscr{I}$ of intervals, length of $\mathscr{I}$ is $len(\mathscr{I}) = \sum_{I \in \mathscr{I}} len(I)$.*

Definition 2 (Span of intervals). *For a set $\mathscr{I}$ of intervals, we define the span of $\mathscr{I}$ as $span(\mathscr{I}) = len(\bigcup \mathscr{I})$.*

Definition 3 (Optimal schedule). *An optimal schedule is the schedule that minimizes the total busy time of physical machines. For any instance $\mathscr{J}$ and parameter $g \geqslant 1$, $OPT(\mathscr{J}, g)$ denotes the cost of an optimal schedule.*

In this paper, we denote $\mathscr{J}$ is set of time intervals that derived from given set of all requested VMs. In general, we use instance $\mathscr{J}$ is alternative meaning to a given set of all requested VMs in context of this paper.

Observations: Cost, capacity, span bounds. For any instance $\mathscr{J}$, which is set of time intervals derived from given set of all requested VMs, and capacity parameter $g \geqslant 1$, which is the maximum number of VMs that can be allocated on any physical machine, the following bounds are held:

- The optimal cost bound: $OPT(\mathscr{J}, g) \leq len(\mathscr{J})$.
- The capacity bound: $OPT(\mathscr{J}, g) \geqslant \dfrac{len(\mathscr{J})}{g}$.
- The span bound: $OPT(\mathscr{J}, g) \geqslant span(\mathscr{J})$.

For any feasible schedule s on a given set of virtual machines, the total busy time of all physical machines that are used in the schedule s is bounded by the maximum total length of all time intervals in a given instance $\mathscr{J}$. Therefore,

the optimal cost bound holds because $OPT(\mathscr{J}, g) = len(\mathscr{J})$ iff all intervals are non-overlapping, i.e., $\forall I_1, I_2 \in \mathscr{J}$ then $I_1 \cap I_2 = \emptyset$.

Intuitively, the capacity bound holds because $OPT(\mathscr{J}, g) = \dfrac{len(\mathscr{J})}{g}$ iff, for each physical server, exactly g VMs are neatly scheduled in that physical server. The span bound holds because at any time $t \in \bigcup \mathscr{J}$ at least one machine is working.

3.5 Theorems

Theorem 1. *Given a cloud system with a set of identical physical machines, assume that the power consumption of a physical machine is* $P(u) = P^{idle} + (P^{max} - P^{idle})u$, *where* P^{idle} *is the idle power consumption,* P^{max} *is the maximum power consumption, and* u *is the CPU utilization in percentage* ($0 \leq u \leq 1$). *We denote* e_{ij} *is energy consumption of the virtual machine* i^{th} *that is scheduled or mapped on the physical machine* j^{th}. *If the utilization* u *of the mapped virtual machine is a constant, then the energy consumption of each virtual machine,* e_{ij}, *is independent of any mapping (i.e. any schedule). We have* $\forall i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\} : e_{ij} = e_i$.

Proof. Recall that the energy consumption is formulated in Eq. (6), and power consumption, $P(u)$, is a linear function of CPU utilization, u. Therefore $\forall i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}$, we see that e_{ij} is the integral of the $P(u)$ over any time interval $[t_1, t_2]$, and is the same value, denoted as e_i.

From Theorem 1, we can imply the following theorem.

Theorem 2. *Minimizing total energy consumption in (1) is equivalent to minimizing the sum of total busy time of all physical machines* ($\sum_{j=1}^{m} T_j$).

$$\textbf{\textit{Minimize}}\ (P^{idle} \times \sum_{j=1}^{m} T_j + \sum_{i=1}^{n} e_i) \ \sim\ \textbf{\textit{Minimize}}\ (\sum_{j=1}^{m} T_j) \tag{7}$$

Proof. According to the objective function described in (1), P^{idle} is constant while e_i is independent of any mapping (i.e. any schedule).

Based on the above observations, we propose our energy-aware algorithms denoted as EMinTRE-LDTF which is presented in the next section.

Definition 4. *For any schedule we denote by* $\mathscr{J}_j$ *the set of virtual machines allocated to the physical machine* M_j *by the schedule. Let* T_j *denote the total busy time of* M_j *is the span of* $\mathscr{J}_j$, *i.e.,* $T_j = span(\mathscr{J}_j)$.

Definition 5. *For any instance* $\mathscr{J}$, *the total busy time of the entire schedule of* $\mathscr{J}$ *computed by the algorithm* H, *which denoted as* $cost^H(\mathscr{J})$, *is defined as:*

$$cost^H(\mathscr{J}) = \int_0^{span(\mathscr{J})} N^H(t)dt \tag{8}$$

in which $N^H(t)$ is the number of physical machines used at the time t by the algorithm H.

Definition 6. *For any instance $\mathscr{J}$ and parameter $g \geqslant 1$, $E^{OPT}(\mathscr{J}, g)$, which is denoted as the minimized total energy consumption of all physical machines in an optimal schedule for the $\mathscr{J}$, is formulated as: $E^{OPT}(\mathscr{J}, g) = P^{idle} \cdot OPT(\mathscr{J}, g) + \sum_{i=1}^{n} e_i$.*

Theorem 3. *For any instance $\mathscr{J}$, the lower and upper of the total energy consumption in an optimal schedule are bounded by: $P^{idle} \cdot \frac{len(\mathscr{J})}{g} \leq E^{OPT}(\mathscr{J}, g) \leq P^{max} \cdot len(\mathscr{J})$.*

Proof. For any instance $\mathscr{J}$, let $OPT(\mathscr{J}, g)$ be the total busy time of the optimal schedule for the $\mathscr{J}$, and let E^* be the total energy consumption for the optimal schedule for the $\mathscr{J}$.

The total energy consumption of an optimal schedule needs to account for all physical machines running during $OPT(\mathscr{J}, g)$. We have: $E^* = P^{idle} \cdot OPT(\mathscr{J}, g) + \sum_{i=1}^{n} e_i$.

From Definition 6, we have $E^{OPT}(\mathscr{J}, g) = E^*$.

Apply the capacity bound in Theorem 3.4, we have $OPT(\mathscr{J}, g) \geq \frac{len(\mathscr{J})}{g}$. Thus, $E^* \geq P^{idle} \cdot \frac{len(\mathscr{J})}{g} + \sum_{i=1}^{n} e_i$.

Recall that the energy consumption of each virtual machine is non-negative, thus $e_i > 0$. Therefore, $E^* \geq P^{idle} \cdot \frac{len(\mathscr{J})}{g}$. Thus

$$E^{OPT}(\mathscr{J}, g) \geq P^{idle} \cdot \frac{len(\mathscr{J})}{g} \tag{9}$$

We prove the upper bound of the minimized total energy consumption as following. Apply the optimal cost bound in the Observations, we have $OPT(\mathscr{J}, g) \leq len(\mathscr{J})$.

Thus

$$E^* \leq P^{idle} \cdot len(\mathscr{J}) + \sum_{i=1}^{n} e_i. \tag{10}$$

Apply the linear power consumption as in the Eqs. (4) and (6), the energy consumption of each i-th virtual machine in period time of $[ts_i, ts_i + d_i]$ that denotes as e_i is:

$$e_i = \int_{ts_i}^{ts_i + d_i} P_j(U_{vm_i})dt = (P_j^{max} - P_j^{idle}) \cdot U_{vm_i} \cdot d_i$$

where U_{vm_i} is the percentage of CPU usage of the i-th virtual machine on a j-th physical machine.

Because any virtual machine always requests CPU usage lesser than or equal to the maximum total capacity CPU of every physical machine, i.e., $U_{vm_i} \leq 1$.

$$\Rightarrow e_i \leq (P_j^{max} - P_j^{idle}) \cdot d_i$$

Note that in this proof, all physical machines are identical with same power consumption model thus P^{max} and P^{idle} are the maximum power consumption and the idle power consumption of each physical machine. Thus:

$$e_i \leq (P^{max} - P^{idle}) \cdot d_i$$

Let I_i is interval of each i-th virtual machine, $I_i = [ts_i, ts_i + d_i]$. By the definition the length of interval is $len(I_i) = d_i$ that is duration time of each i-th virtual machine. Thus:

$$e_i \leq (P^{max} - P^{idle}) \cdot len(I_i)$$

The total energy consumption of n virtual machines is formulated as:

$$\sum_{i=1}^{n} e_i \leq \sum_{i=1}^{n} [(P^{max} - P^{idle}) \cdot len(I_i)] \Leftrightarrow \sum_{i=1}^{n} e_i \leq (P^{max} - P^{idle}) \cdot \sum_{i=1}^{n} len(I_i)$$

$$\Leftrightarrow \sum_{i=1}^{n} e_i \leq (P^{max} - P^{idle}) \cdot len(\mathscr{J}). \tag{11}$$

From Equation (10), we have:

$$E^* \leq P^{idle} \cdot len(\mathscr{J}) + \sum_{i=1}^{n} e_i E^* \leq P^{idle} \cdot len(\mathscr{J}) + (P^{max} - P^{idle}) \cdot len(\mathscr{J})$$

$$E^* \leq (P^{idle} + (P^{max} - P^{idle})) \cdot len(\mathscr{J}) \tag{12}$$

From the Equation (12):

$$E^* \leq P^{max} \cdot len(\mathscr{J}) \tag{13}$$

$$\Leftrightarrow E^{OPT}(\mathscr{J}, g) \leq P^{max} \cdot len(\mathscr{J}) \tag{14}$$

From both of two Eqs. (9) and (14), we have:

$$P^{idle} \cdot \frac{len(\mathscr{J})}{g} \leq E^{OPT}(\mathscr{J}, g) \leq P^{max} \cdot len(\mathscr{J}) \tag{15}$$

We prove the theorem.

4 Scheduling Algorithm

4.1 EMinTRE-LDTF Scheduling Algorithm

In this section, we present our energy-aware scheduling algorithm, namely, EMinTRE-LDTF. EMinTRE-LDTF presents a metric to unify the increasing time and estimated resource efficiency when mapping a VM onto a physical machine. Then, EMinTRE-LDTF will choose a host that minimizes the metric. Our previous MinDFT-LDTF and MinDFT-LFT, which use core algorithm MinDFT in [21], only focused on minimizing the increasing time when mapping a VM onto a physical machine. MinDFT-LDTF sorts the list of VM oder by longest duration time first, and MinDFT-LFT sorts the list of VM oder by latest finishing time first. Algorithm EMinTRE-LDTF additionally considers resource efficiency during an execution period of a physical machine in order to fully utilize resources in a physical machine. Algorithm EMinTRE-LDTF differs from EMinRET [22] in the equation of TRE and EMinTRE-LDTF does not have swapping step as in EMinRET.

Based on Eq. 5, the utilization of a resource r (resource r can be CPU, physical memory, network bandwidth, storage, etc.) of the M_j, denoted as $U_{j,r}$, is formulated as:

$$U_{j,r} = \sum_{s \in n_j} \frac{V_{s,r}}{H_{j,r}}. \tag{16}$$

where n_j is the list of virtual machines that are assigned to the M_j, $V_{s,r}$ is the amount of requested resource r of the virtual machine s (note that in our study the value of $V_{s,r}$ is fixed for each user request), and $H_{j,r}$ is the maximum capacity of the resource r in M_j.

Inspired by the work from Microsoft research team [7,20], the resource efficiency of a physical machine j^{th}, denoted by RE_j, is Norm-based distant [20] of two vectors: normalized resource utilization vector and unit vector **1**. The resource efficiency is formulated as:

$$RE_j = \sum_{r \in \mathscr{R}} ((1 - U_{j,r}) \times w_r)^2 \tag{17}$$

where R is the set of resource types in a host ($\mathscr{R} = \{\text{cpu, ram, netbw, io, storage}\}$) and w_r is weight of resource r in a physical machine.

In this paper, we propose a unified metric for increasing time and resource efficiency of the host j-th that is calculated as:

$$TRE_j = \begin{cases} \sqrt{RE_j}, & \text{if } t^{diff} = 0. \\ (\frac{t^{diff}}{3600} \times w_{r=time})\sqrt{RE_j}, & \text{if } t^{diff} \neq 0. \end{cases} \tag{18}$$

EMinTRE-LDTF chooses a physical host that has a minimum value of the TRE metric to allocate for a VM. We present the pseudo-code of EMinTRE-LDTF in Algorithm 1. EMinTRE-LDTF has two (2) steps: firstly, EMinTRE-LDTF sorts the list of VMs by longest duration time first, and secondly,

Algorithm 1. EMinTRE-LDTF: Energy-Aware Minimizing Resource Efficiency - Time

```
1: function EMinTRE-LDTF
2:     Input: vmList - a list of virtual machines to be scheduled, hostList - a list of physical servers
3:     Output: a feasible schedule or null
4:     vmList = sortVmListByOrderLongestDurationTimeFirst( vmList )                    ▷ 1
5:     m = hostList.size(); n = vmList.size();
6:     T[j] = 0, ∀j ∈ [1, m]
7:     for i = 1 to n do                                                               ▷ on the VMs list
8:         vm = vmList.get(i)
9:         allocatedHost = null
10:        T1 = sumTotalHostCompletionTime( T )
11:        minTRE = +∞
12:        for j = 1 to m do                                                           ▷ on the hosts list
13:            host = hostList.get( j )
14:            hostVMList = sortVmListByOrder( host.getVms(), order=[starttime, finishtime])
15:            if host.checkAvailableResource( vm ) then
16:
17:                preTime = T[ host.id ]
18:                T[ host.id ] = host.estimateHostTotalCompletionTime( vm )
19:                T2 = sumTotalHostCompletionTime( T )
20:                diffTime = Math.max( T2 - T1, 0)
21:                TRE = EstimateMetricTimeResEff( diffTime, host )
22:                if (minTRE > TRE ) then
23:                    minTRE = TRE
24:                    allocatedHost = host
25:                end if
26:                T[ host.id ] = preTime      ▷ Next iterate over the hostList and choose the host that minimize the value of different time and resource efficiency
27:            end if
28:        end for
29:        if (allocatedHost != null) then
30:            allocate the vm to the host
31:            add the pair of vm (key) and host to the mapping
32:        end if
33:    end for
34:    return mapping
35: end function
36: sumTotalHostCompletionTime(T[]) = ∑_{j=1}^{m} T_j      ▷ T[1...m]: Array of total completion times of m physical servers
```

EMinTRE-LDTF schedule the first VM in the sorted list of VMs to a host that has the minimum of the TRE. The EMinTRE-LDTF solves the scheduling problem in time complexity of $\mathcal{O}(n \times m \times q)$ where n is the number of VMs to be scheduled, m is the number of physical machines, and q is the maximum number of allocated VMs in the physical machines $M_j, \forall j = 1, 2, ..., m$.

4.2 Approximation Algorithm for General Case

In this section, we claim that algorithm EMinTRE-LDTF for general instance yields its approximation ratio are g, where g is the maximal number of virtual

Algorithm 2. Estimating the metric for increasing time and resource efficiency

```
1: function EstimateMetricTimeResEff
2:     Input: (t^diff, host) - t^diff is a different time, host is a candidate physical machine
3:     Output: TRE - a value of metric time and resource efficiency
4:     Set ℛ={cpu, ram, netbw, io, storage, time}
5:     j = host.getId(); n_j = host.getVMList();
6:     for r ∈ ℛ do
7:         Calculate the resource utilization, U_{j,r} as in the Equaltion (16).
8:     end for
9:     weights[] ← Read resource weights from configuration file
10:    Calculate the different time and resource efficiency metric for host j denoted as TRE_j
       as in the Equaltion (18)
11:    return TRE
12: end function
```

machines can be assigned to each physical machine. EMinTRE-LDTF sorts the list of virtual machines in order of their longest duration time first.

Theorem 4. *For any instance $\mathscr{J}$, EMinTRE-LDTF is a g-approximation algorithm where g is the maximal number of virtual machines can be assigned to each physical machine, i.e. the total busy time of schedule for $\mathscr{J}$ outputted by EMinTRE-LDTF is the maximum g times the total busy time of optimal schedule. We denote $EMinTRE-LDTF(\mathscr{J})$ is cost of algorithm EMinTRE-LDTF for a given instance $\mathscr{J}$ that is defined in the Definition 5. Formally,*

$$EMinTRE - LDTF(\mathscr{J}) \leq g \cdot OPT(\mathscr{J}). \tag{19}$$

Proof. Let $N(t)$ denote the number of virtual machines that could be placed at time t. Let $N^H(t)$ denote the number of used physical machines at time t in a schedule that is resulted by algorithm EMinTRE-LDTF, and $N^{OPT}(t)$ denotes the number of machines used at time t in an optimal schedule.

For any time $t \geq 0$, each of using $N^H(t)$ physical machines has at least one allocated virtual machine, i.e., $N(t) \geq N^H(t)$. Clearly, at any time $t \geq 0$ and $g \geq 1$, $N^{OPT}(t) \geq \frac{N(t)}{g} = \frac{N^H(t)}{g}$.

The total busy time of the entire schedule of an instance $\mathscr{J}$ denoted as $EMinTRE-LDTF(\mathscr{J})$ is calculated by taking the integral of $N^H(t)$ with all values of $t \in [0, span(\mathscr{J})]$. Thus, we have:

$$N^{OPT}(t) \geq \frac{N^H(t)}{g} \Longleftrightarrow N^H(t) \leq g \cdot N^{OPT}(t)$$

$$\Rightarrow \int_0^{span(\mathscr{J})} N^H(t)dt \leq g \cdot \int_0^{span(\mathscr{J})} N^{OPT}(t)dt$$

$$\Longleftrightarrow EMinTRE - LDTF(\mathscr{J}) \leq g \cdot OPT(\mathscr{J})$$

Thus, by applying the same reasoning to other algorithms such as EMinTRE-LDTF, we have:

$$EMinTRE - LDTF(\mathscr{J}) \leq g \cdot OPT(\mathscr{J})$$

4.3 Approximations for Special Cases

Proper Interval Graphs. In this section we consider instances in which no virtual machine's time interval is properly contained in another. The intersection graphs for such instances are known as *proper interval graphs*. Algorithm EMinTRE-LDTF for proper interval graphs includes two steps. In the first step, the list of virtual machines is sorted by their earliest starting time first. In the second step, each virtual machine is placed at the currently filled physical machine so that TRE metric of the physical machine is minimized, unless the placement violates the hard constraint on capacity of the physical machine, in which case a new physical machine is opened.

Theorem 5. *Algorithm EMinTRE-LDTF yields a* $(3 - \frac{2}{g})$ *for proper interval graphs, where* g *is the maximum number of virtual machines that could be placed on a physical machine in satisfying all their resource requirement*

Proof. Let $N(t)$ denote the number of virtual machines that could be placed at time t. Let $N^H(t)$ denote the number of used physical machines at time t in a schedule that is resulted by algorithm EMinTRE-LDTF (H), and $N^{OPT}(t)$ denotes the number of machines used at time t in an optimal schedule (OPT).

Theorem 6. (Proposition). *For any* t, $N(t) \geq (N^H(t) - 2)g + 2$.

Proof. For a given $t > 0$, let $m = N^H(t)$ denote number of used physical machines at time t. There are $m - 2$ additional used machines denote as $M_2, ..., M_{m-1}$. The first machine M_1 has at least one virtual machine that is placed on the M_1, and other machines $M_2, ..., M_{m-1}$ are assigned fully g virtual machines on each, and the m-th machine M_m is assigned at least one virtual machine. Since the graph is proper, suppose that the first machine processes at time t one virtual machine u, any virtual machine v is placed to another machine starts after the u and ends after u, thus v is running at time t. With m used physical machines at time t, the number of running virtual machines at time t is at least $(m - 2)g + 2$. Therefore $N(t) \geq (m - 2)g + 2$.

For any t, $N^{OPT}(t) \geq \frac{N(t)}{g}$. Applying the proposition, we have at any t, $N(t) \geq (N^H(t) - 2)g + 2$

$$N^{OPT}(t) \geq \frac{(N^H(t) - 2)g + 2}{g}$$

Recall $g > 0$, thus

$$N^H(t) \leq N^{OPT}(t) + 2 - \frac{2}{g}$$

The total busy time of entire schedule of $\mathscr{J}$ is:

$$EMinTRE - LDTF(\mathscr{J}) = \int_0^{span(\mathscr{J})} N^H(t)dt$$

$$EMinTRE - LDTF(\mathscr{J}) \leq \int_0^{span(\mathscr{J})} (N^{OPT}(t) + 2 - \frac{2}{g})dt$$

$$EMinTRE - LDTF(\mathscr{J}) \leq OPT(\mathscr{J}) + (2 - \frac{2}{g}) \cdot span(\mathscr{J}) \quad (20)$$

With related to the span bound in Definition 5, we have $OPT(\mathscr{J}) \geq span(\mathscr{J})$ thus inequality (20) equivalent to:

$$EMinTRE - LDTF(\mathscr{J}) \leq OPT(\mathscr{J}) + (2 - \frac{2}{g}) \cdot OPT(\mathscr{J})$$

$$\frac{EMinTRE - LDTF(\mathscr{J})}{OPT(\mathscr{J})} \leq 3 - \frac{2}{g} \quad (21)$$

This gives the statement of the Theorem 5.

Theorem 7. *If all virtual machines are homogeneous and each physical machine processes only one virtual machine then the EMinTRE-LDTF algorithm yields an optimal solution for proper interval graphs.*

Proof. In case all virtual machines are homogeneous and each physical machine processes only one virtual machine, thus $k = 1$ and $g = 1$. We have $EMinTRE-LDTF(\mathscr{J}) \geq OPT(\mathscr{J})$ and apply Theorem 5, thus $EMinTRE-LDTF(\mathscr{J}) = OPT(\mathscr{J})$.

5 Performance Evaluation

5.1 Algorithms

In this section, we study the following VM allocation algorithms:

- PABFD, a power-aware and modified best-fit decreasing heuristic [3,4]. The PABFD sorts the list of VM_i (i=1, 2,..., n) by their total requested CPU utilization, and assigns new VM to any host that has a minimum increase in power consumption.
- VBP-Norm-L2, a vector packing heuristics that is presented as Norm-based Greedy with degree 2 [20]. Weights of these Norm-based Greedy heuristics use FFDAvgSum which are $exp(x)$, which is the value of the exponential function at the point x, where x is average of sum of demand resources (e.g. CPU, memory, storage, network bandwidth, etc.). VBP-Norm-L2 assigns new VM to any host that has minimum of these norm values.
- MinDFT-LDTF and MinDFT-LFT: core of both MinDFT-LDTF and MinDFT-LFT algorithms is MinDFT [21]. MinDFT-LDTF sorts the list of VM_i (i = 1, 2,..., n) by their starting time (ts_i) and respectively by their finished time ($ts_i + dur_i$), then MinDFT-LDTF allocates each VM (in a given sorted list of VMs) to a host that has a minimum increase in total completion times of hosts. MinDFT-LFT differs MinDFT-LDTF in sorting the list of VMs, MinDFT-LFT sorts the list of VMs by their respectively finished time in latest finishing time first.

– EMinTRE-LDTF, the algorithm is proposed in the Sect. 4. EMinTRE-LDTF sorts the list of VMs (input) by VM's longest duration time first and host's allocated VMs by its finishing time and place a VM to any physical machine that minimizes time-resource efficiency (TRE) metric.

5.2 Methodology

We evaluate these algorithms by simulation using the CloudSim [6] to create a simulated cloud data center system that has identical physical machines, heterogeneous VMs, and with thousands of CloudSim's cloudlets [6] (we assume that each HPC job's task is modeled as a cloudlet that is run on a single VM). The information of VMs (and also cloudlets) in these simulated workloads is extracted from two parallel job models are Feitelson's parallel workload model [9] and Lublin99's parallel workload model [17] in Feitelson's Parallel Workloads Archive (PWA) [10]. When converting from the generated log-trace files, each cloudlet's length is a product of the system's processing time and CPU rating (we set the CPU rating is equal to included VM's MIPS). We convert job's submission time, job's start time (if the start time is missing, then the start time is equal to sum of job's submission time and job's waiting time), job's request run-time, and job's number of processors in job data from the log-trace in the PWA to VM's submission time, starting time and duration time, and number of VMs (each VM is created in round-robin in the four types of VMs in Table 2 on the number of VMs). Eight (08) types of VMs as presented in the Table 2 are used in the [26] that are similar to categories in Amazon EC2's VM instances: high-CPU VM, high-memory VM, small VM, and micro VM, etc. All physical machines are identical and each physical machine is a typical physical machine (Hosts) with 16 cores CPU (3250 MIPS/core), 136.8 GBytes of available physical memory, 10 Gb/s of network bandwidth, 10 TBytes of available storage. Power model of each physical machine is 175 W at idle power and 250 W at maximum power consumption (the idle power is 70% of the maximum power consumption as in [3,4,8]). In the simulations, we use weights as following: (i) weight of increasing time of mapping a VM to a host: {0.001, 0.01, 1, 100, 3600}; (ii) all weights of computing resources (e.g. number of MIPS per CPU core, physical

Table 2. Eight (08) VM types in simulations.

VM type	MIPS	Cores	Memory (Unit: MBytes)	Network (Unit: Mbits/s)	Storage (Unit: GBytes)
Type 1	2500	8	6800	100	1000
Type 2	2500	2	1700	100	422.5
Type 3	3250	8	68400	100	1000
Type 4	3250	4	34200	100	845
Type 5	3250	2	17100	100	422.5
Type 6	2000	4	15000	100	1690
Type 7	2000	2	7500	100	845
Type 8	1000	1	1875	100	211.25

Table 3. Information of a typical physical machine (host) with 16 cores CPU (3250 MIPS/core), 136.8 GBytes of available physical memory, 10 Gb/s of network bandwidth, 10 TBytes of storage and idle, maximum power consumption is 175, 250 (W).

Type	MIPS	Cores	Memory (Unit: MBytes)	Network (Unit: Mbits/s)	Storage (Unit: GBytes)	P^{idle} (Unit: Watts)	P^{max} (Unit: Watts)
M1	3250	16	140084	10000	10000	175	250

Table 4. The normalized total energy consumption. Simulation results of scheduling algorithms solving scheduling problems with 12681 VMs and 5000 physical machines (hosts) using Feiltelson's parallel workload model [9].

Algorithms	#Hosts	#VMs	Energy (KWh)	Norm. energy
PABFD	5000	12681	1,055.42	1.598
VBP-Norm-L2	5000	12681	1,054.69	1.597
Tian-MFFDE	5000	12681	660.30	1.000
MinDFT-LDTF	5000	12681	603.90	0.915
MinDFT-LFT	5000	12681	503.43	0.762
EMinTRE-LDTF	5000	12681	496.55	0.752

Table 5. The normalized total energy consumption. Simulation results of scheduling algorithms solving scheduling problems with 29,177 VMs and 10,000 physical machines (hosts) using Feiltelson's parallel workload model [9].

Algorithms	#Hosts	#VMs	Energy (KWh)	Norm. energy
PABFD	10000	29177	2261.878	1.540
VBP-Norm-L2	10000	29177	2260.615	1.539
Tian-MFFDE	10000	29177	1468.409	1.000
MinDFT-LDTF	10000	29177	1373.764	0.936
MinDFT-LFT	10000	29177	1109.852	0.756
EMinTRE-LDTF	10000	29177	1092.365	0.744

memory (RAM), network bandwidth, and storage) are equally to 1. We simulate on combination of these weights. The total energy consumption of each EMinTRE-LDTF is the average of five times simulation with various weights of increasing time (e.g. 0.001, 0.01, 1, 100, or 3600) (Tables 4, 5, 6 and 7).

We choose Modified First-Fit Decreasing Earliest (denoted as Tian-MFFDE) [26] as the baseline because Tian-MFFDE is the best algorithm in the energy-aware scheduling algorithm to time interval scheduling. We also compare our proposed VM allocation algorithms with PABFD [4] because the PABFD is a famous power-aware best-fit decreasing in the energy-aware scheduling research community, and two vector bin-packing algorithms (VBP-Norm-L1/L2) to show the importance of with/without considering VM's starting time and finish time in reducing the total energy consumption of VM placement problem.

Table 6. The normalized total energy consumption. Simulation results of scheduling algorithms solving scheduling problems with 8847 VMs and 5000 physical machines (hosts) using Lublin99's parallel workload model [17].

Algorithms	#Hosts	#VMs	Energy (KWh)	Norm. energy
PABFD	5000	8847	460.664	1.601
VBP-Norm-L2	5000	8847	453.229	1.575
Tian-MFFDE	5000	8847	287.779	1.000
MinDFT-LDTF	5000	8847	263.860	0.917
MinDFT-LFT	5000	8847	232.286	0.807
EMinTRE-LDTF	5000	8847	220.675	0.767

Table 7. The normalized total energy consumption. Simulation results of scheduling algorithms solving scheduling problems with 19853 VMs and 5000 physical machines (hosts) using Lublin99's parallel workload model [17].

Algorithms	#Hosts	#VMs	Energy (KWh)	Norm. energy
PABFD	5000	19853	3107.78	1.424
VBP-Norm-L2	5000	19853	3106.56	1.423
Tian-MFFDE	5000	19853	2182.54	1.000
MinDFT-LDTF	5000	19853	1927.52	0.883
MinDFT-LFT	5000	19853	1746.12	0.800
EMinTRE-LDTF	5000	19853	1485.13	0.680

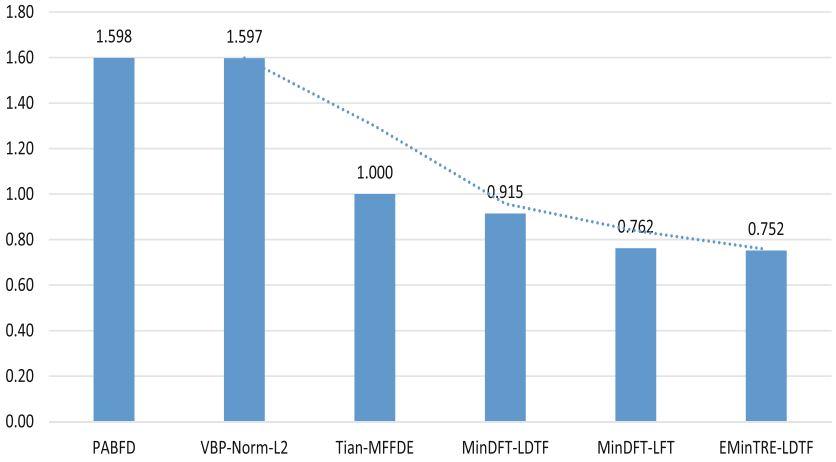

Fig. 1. The normalized total energy consumption compare to Tian-MFFDE. Result of simulations with Feitelson's Parallel Workload Archive Model [9] that includes 1,000 jobs have total of 12,681 VMs.

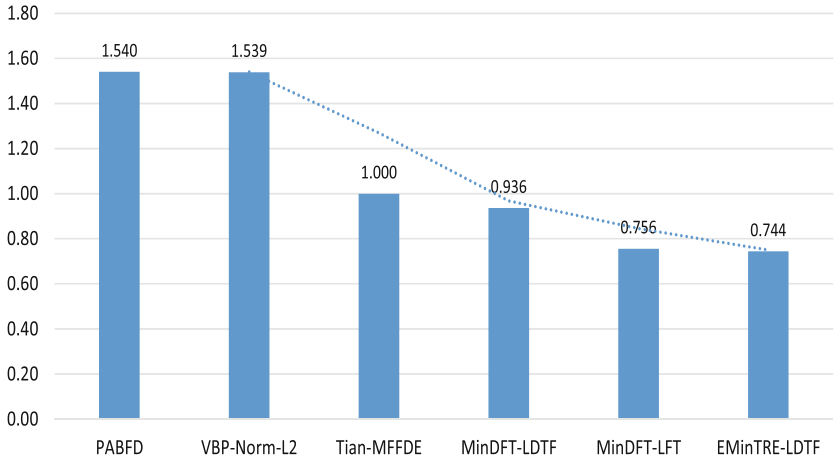

Fig. 2. The normalized total energy consumption compare to Tian-MFFDE. Result of simulations with Feitelson's parallel workload model [9] that includes 2,000 jobs have total of 29,177 VMs.

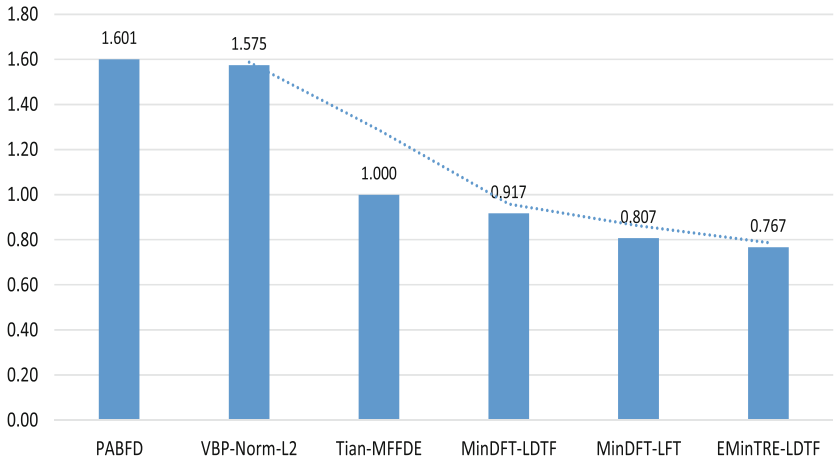

Fig. 3. The normalized total energy consumption compare to Tian-MFFDE. Result of simulations with Lublin99's parallel workload model [17] that includes 1,000 jobs have total of 8,847 VMs.

5.3 Results and Discussions

The Tables 4 and 5 show simulation results of scheduling algorithms solving scheduling problems with 12681 VMs - 5000 physical machines (hosts) and 29,177 VMs - 10,000 physical machines (hosts), in which VM's data is converted from the Feiltelson's parallel workload model [9] with 1000 jobs and 2000 jobs. The Tables 6 and 7 show simulation results of scheduling algorithms solving schedul-

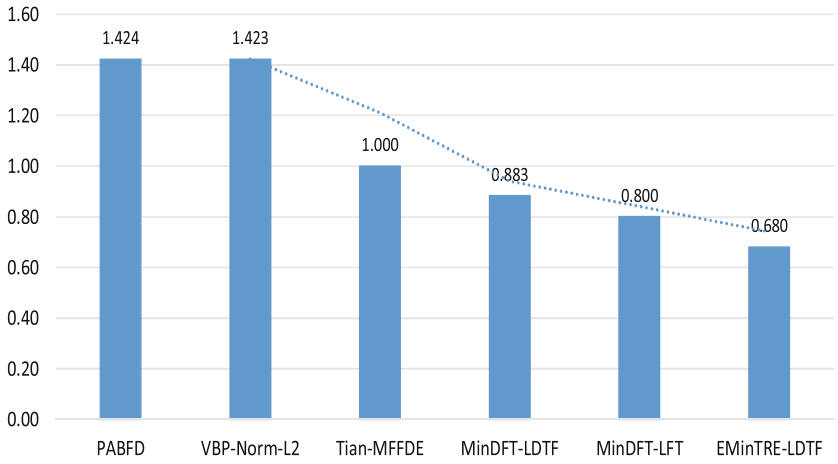

Fig. 4. The normalized total energy consumption compare to Tian-MFFDE. Result of simulations with Lublin99's parallel workload model [17] that includes 2,000 jobs have total of 19,853 VMs.

ing problems with 8847 VMs - 5000 physical machines (hosts) and 19853 VMs - 5000 physical machines (hosts), in which VM's data is converted from the Lublin99's parallel workload model [17].

Four (04) figures include Figs. 1, 2, 3, and 4 show bar charts comparing energy consumption of VM allocation algorithms that are normalized with the Tian-MFFDE. None of the algorithms use VM migration techniques, and all of them satisfy the Quality of Service (e.g. the scheduling algorithm provisions maximum of user VM's requested resources). We use total energy consumption as the performance metric for evaluating these VM allocation algorithms.

Simulated results show that, compared with Tian-MFFDE [26] EMinTRE-LDTF can reduce the total energy consumption by average 26.5%. EMinTRE-LDTF also can reduce the total energy consumption in compared with PABFD [4], VBP-Norm-L2 [20] and MinDFT-LDTF, MinDFT-LFT.

6 Conclusions and Future Work

In this paper, we formulated an energy-aware VM allocation problem with multiple resource, fixed interval and non-preemption constraints. We also discussed our key observation in the VM allocation problem, i.e., minimizing total energy consumption is equivalent to minimize the sum of total completion time of all physical machines (PMs). Our proposed algorithm EMinTRE-LDTF can reduce the total energy consumption of the physical servers compared with the state-of-the-art algorithms in simulation results using two (02) parallel workload models [9,17]. Algorithm EMinTRE-LDTF is proved g approximations in general case and $(3 - 2/g)$ in proper interval graphs.

As future work, we are developing EMinTRE-LDTF into a cloud resource management software (e.g. OpenStack Nova Scheduler). Additionally, we are working on IaaS cloud systems with heterogeneous physical servers and job requests consisting of multiple VMs using EPOBF [23]. We are studying the use of Machine Learning techniques to choose the right weights of time and resources (e.g. computing power, physical memory, and network bandwidth).

Acknowledgment. A preliminary version of this work that has been published in the Proceedings of the Future Data and Security Engineering Second International Conference (FDSE 2015). This work was partially supported by the Erasmus Mundus Gate project at the Johannes Kepler University (JKU) Linz, Austria. I am thankful to a.Univ.-Prof. Dr. Josef Küng, JKU Linz for his help.

References

1. Angelelli, E., Filippi, C.: On the complexity of interval scheduling with a resource constraint. Theor. Comput. Sci. **412**(29), 3650–3657 (2011)
2. Barroso, L.A., Clidaras, J., Hölzle, U.: The datacenter as a computer: an introduction to the design of warehouse-scale machines. Synth. Lect. Comput. Archit. **8**(3), 1–154 (2013)
3. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comp. Syst. **28**(5), 755–768 (2012)
4. Beloglazov, A., Buyya, R.: Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency Comput. Pract. Experience **24**(13), 1397–1420 (2012)
5. Beloglazov, A., Buyya, R., Lee, Y.C., Zomaya, A.: A taxonomy and survey of energy-efficient data centers and cloud computing systems. Adv. Comput. **82**, 1–51 (2011)
6. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exper. **41**(1), 23–50 (2011)
7. Chen, L., Shen, H.: Consolidating complementary VMs with spatial/temporal-awareness in cloud datacenters. In: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, pp. 1033–1041. IEEE, April 2014
8. Fan, X., Weber, W.D., Barroso, L.: Power provisioning for a warehouse-sized computer. In: ISCA, pp. 13–23 (2007)
9. Feitelson, D.G.: Packing schemes for gang scheduling. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 1996. LNCS, vol. 1162, pp. 89–110. Springer, Heidelberg (1996). doi:10.1007/BFb0022289
10. Feitelson, D.G.: Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/. Accessed 31 Jan 2014
11. Flammini, M., Monaco, G., Moscardelli, L., Shachnai, H., Shalom, M., Tamir, T., Zaks, S.: Minimizing total busy time in parallel scheduling with application to optical networks. Theor. Comput. Sci. **411**(40–42), 3553–3562 (2010)

12. Garg, S.K., Yeo, C.S., Anandasivam, A., Buyya, R.: Energy-efficient Scheduling of HPC Applications in Cloud Computing Environments. CoRR abs/0909.1146 (2009)
13. Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P.P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q.M., Tziritas, N., Vishnu, A., Khan, S.U., Zomaya, A.: A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Computing **98**(7), 751–774 (2014)
14. Knauth, T., Fetzer, C.: Energy-aware scheduling for infrastructure clouds. In: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, pp. 58–65. IEEE, December 2012
15. Kovalyov, M.Y., Ng, C., Cheng, T.E.: Fixed interval scheduling: models, applications, computational complexity and algorithms. Eur. J. Oper. Res. **178**(2), 331–342 (2007)
16. Le, K., Bianchini, R., Zhang, J., Jaluria, Y., Meng, J., Nguyen, T.D.: Reducing electricity cost through virtual machine placement in high performance computing clouds. In: SC, p. 22 (2011)
17. Lublin, U., Feitelson, D.G.: The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J. Parallel Distrib. Comput. **63**(11), 1105–1122 (2003)
18. Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J.M., Vasilakos, A.V.: Cloud computing: survey on energy efficiency. ACM Comput. Surv. **47**(2), 33:1–33:36 (2014)
19. Orgerie, A.C., de Assuncao, M.D., Lefevre, L.: A survey on techniques for improving the energy efficiency of large-scale distributed systems. ACM Comput. Surv. **46**(4), 1–31 (2014)
20. Panigrahy, R., Talwar, K., Uyeda, L., Wieder, U.: Heuristics for Vector Bin Packing. Technical report, Microsoft Research (2011)
21. Quang-Hung, N., Le, D.-K., Thoai, N., Son, N.T.: Heuristics for energy-aware VM allocation in HPC clouds. In: Dang, T.K., Wagner, R., Neuhold, E., Takizawa, M., Küng, J., Thoai, N. (eds.) FDSE 2014. LNCS, vol. 8860, pp. 248–261. Springer, Heidelberg (2014). doi:10.1007/978-3-319-12778-1_19
22. Quang-Hung, N., Thoai, N.: EMinRET: heuristic for energy-aware VM placement with fixed intervals and non-preemption. In: 2015 International Conference on Advanced Computing and Applications (ACOMP), pp. 98–105. IEEE, November 2015
23. Quang-Hung, N., Thoai, N., Son, N.T.: EPOBF: energy efficient allocation of virtual machines in high performance computing cloud. In: Hameurlain, A., Küng, J., Wagner, R., Dang, T.K., Thoai, N. (eds.) TLDKS XVI. LNCS, vol. 8960, pp. 71–86. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45947-8_6
24. Sotomayor, B.: Provisioning Computational Resources Using Virtual Machines and Leases. Ph.D. thesis. University of Chicago (2010)
25. Takouna, I., Dawoud, W., Meinel, C.: Energy efficient scheduling of HPC-jobs on virtualize clusters using host and VM dynamic configuration. Operating Syst. Rev. **46**(2), 19–27 (2012)
26. Tian, W., Yeo, C.S.: Minimizing total busy time in offline parallel scheduling with application to energy efficiency in cloud computing. Concurrency Comput. Pract. Experience **27**(9), 2470–2488 (2013)
27. Viswanathan, H., Lee, E.K., Rodero, I., Pompili, D., Parashar, M., Gamell, M.: Energy-aware application-centric VM allocation for HPC workloads. In: IPDPS Workshops, pp. 890–897 (2011)