On Minimizing Total Energy Consumption in the Scheduling of Virtual Machine Reservations

Wenhong Tian[1,2],Majun He[1] ,Wenxia Guo[1] , Wenqiang Huang[1],
Xiaoyu Shi[2] ,Mingsheng Shang[2] , Adel Nadjaran Toosi[3],Rajkumar Buyya[3]

[1] *School of Information and Software Engineering, University of Electronic Science and Technology of China (UESTC)*
[2] *Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences,Chongqing,China*
[3]*CLOUDS Lab., Dept. of Information and Computing Systems, The University of Melbourne, Australia.*
E-mail: tianwenhong@cigit.ac.cn,
tian_wenhong@uestc.edu.cn,{adel.nadjaran,rbuyya}@unimelb.edu.au

Abstract

This paper considers the energy-efficient scheduling of virtual machine (VM) reservations in a Cloud Data center. Concentrating on CPU-intensive applications, the objective is to schedule all reservations non-preemptively, subjecting to constraints of physical machine (PM) capacities and running time interval spans, such that the total energy consumption of all PMs is minimized (called MinTEC for abbreviation). The MinTEC problem is NP-complete in general. The best known results for this problem is a 5-approximation algorithm for special instances using First-Fit-Decreasing algorithm and 3-approximation algorithm for general offline parallel machine scheduling with unit demand. By combining the features of optimality and workload in interval spans, we propose a method to find the optimal solution with the minimum number of job migrations, and a 2-approximation algorithm called LLIF for general cases. We then show how our algorithms are applied to minimize the total energy consumption in a Cloud Data center. Our theoretical results are validated by intensive simulation using trace-driven and synthetically generated data.

Keywords:
Energy Efficiency, Cloud Data centers, resource scheduling, virtual machine reservation

1. Introduction

Cloud computing has evolved from various recent advancements in virtualization, Grid computing, Web computing, utility computing and other related technologies. It offers three level of services, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In this paper, we concentrate on CPU-intensive computing at IaaS level in Cloud Data centers. Cloud computing providers (such as Amazon) offer virtual machine reservation services with specified computing units. For reservation services, customers request certain units of computing resources in advance to use for a period of time in the future, so providers can have enough time to do scheduling. The resources in this paper include:

1. Physical Machines (PMs): physical computing devices which can host multiple virtual machines; each PM can be a composition of CPU, memory, hard drives, network cards, and etc..
2. Virtual Machine (VMs): virtual computing platforms on PMs using virtualization software; each VM has a number of virtual CPUs, memory, storage, network cards, and related components.

The architecture and process of VM reservation scheduler are provided in Figure 1.1, referring to Amazon EC2 [21]. As noted in the diagram, the major processes of resource scheduling are:

1. User reservation requesting: the user initiates a reservation through the Internet (such as a Cloud service provider's Web portal);
2. Scheduling management: Scheduler Center makes decisions based on the user's identity (such as geographic location, etc.) and the operational characteristics of the request (quantity and quality requirements). The request is submitted to a data center, then the data center management program submits it to the Scheduler Center, finally the Scheduler Center allocates the request based on scheduling algorithms;
3. Feedback: Scheduling algorithms provide available resources to the user;
4. Executing scheduling: Scheduling results (such as deploying steps) are sent to the next stage;
5. Updating and optimization: The scheduler updates resource information, optimizes resources in the data center according to the optimizing objective functions.

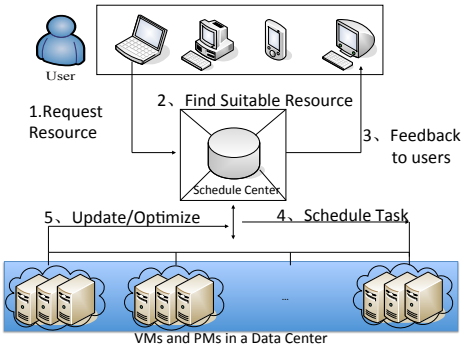

Figure 1.1: Referred architecture of VM reservation in a Cloud data center

In the reservation services, customers are billed in a way proportional to the total amount of computing time as well as energy of the computing resources. The scheduler executes periodically for a fixed period of time, for instance, every one hour, depending on workloads in realistic scenarios. From the providers' point of view, the total energy cost of computing resources is closely related to the total powered-on time of all computing resources. Since Cloud data centers consume very large amounts of energy, the energy cost (electricity price) is increasing regularly. So they like to minimize total power-on time to save energy costs. How to model this problem and solve it efficiently is not well studied in the literature. In practice, some simple algorithms (such as Round Robin and First-Fit) are used by EC2 [21] and VMWare [26]. To measure the performance (such as energy-efficiency) of different approximate algorithms, the approximation ratio, defined as the ratio of the result obtained by proposed algorithm over the optimal result, is widely used. Winkler et al. [18], Flammini et al. [4] and Khandekar et al. [6] are closely related to our research and are earlier papers that discuss this issue under general parallel machine scheduling context, and Kovalyov et al. [9] provide a comprehensive review for the fixed interval scheduling problem. The problem of VM reservations can be stated as follows. There are n deterministic reservations submitted to the scheduler in advance to be scheduled offline on multiple physical machines (PMs) with bounded capacities. Each VM reservation (job) is associated with a start-time, an end-time, and a capacity demand. The objective is to schedule all reservations non-

preemptively, subjecting to constraints of PM capacities and running time interval spans, such that the total energy consumption of all PMs is minimized (called MinTEC for abbreviation).

The MinTEC problem is NP-hard in a general case [18]. Winkler et al. [18] consider the problem in optical networks and show that the problem is NP-hard already for g=2, where g is the total capacity of a machine in terms of CPU. In this study, we assume that the total CPU capacity of a PM, g, is measured in abstract units such as EC2 Compute Unit (ECU)[1]. Flammini et al. [4] consider the same scheduling problem in optical network where jobs are given as interval spans with unit demand (one unit from total capacity), for this version of the problem a 4-approximation algorithm called FFD (First Fit Decreasing) for general inputs and better bounds for some subclasses of inputs are provided. The FFD algorithm basically sorts all jobs' process time in non-increasing order and allocates the job in that order to the first machine which can host. Khandekar et al. [6] propose a 5-approximation algorithm for this scheduling problem by separating all jobs into wide and narrow types by their demands when $\alpha = 0.25$, which is the demand parameter of narrow jobs occupying the portion of the total capacity of a machine. Tian et al. [32] propose a 3-approximation algorithm called MFFDE for general offline parallel machine scheduling with unit demand and the MFFDE algorithm applies FFD with earliest start-time first. In this work, we aim to propose better methods for the optimal energy-efficient scheduling with concentration on VM reservations. The jobs and VM requests are used interchangeably in this paper.

The major **contributions** of this paper include:

1. Proposing an approach to minimize total energy consumption of virtual machine reservations by minimizing total energy consumption (MinTEC) of all PMs.
2. Deducing a theoretical lower bound for the MinTEC problem with limited number of VM migrations.
3. Proposing a 2-approximation algorithm called LLIF, which is better than the best-known 3-approximation algorithm.
4. Validating theoretical results by intensive simulation of trace-driven

[1]The EC2 Compute Unit (ECU) provides the relative measure of the integer processing power of an Amazon EC2 instance and provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor

and synthetically generated data.

The rest of the paper is organized as follows. Formal problem statement is provided in Section 2. Section 3 presents our proposed algorithm LLIF with theoretical analysis. Section 4 considers how our results are applied to the energy efficiency of VM reservations. Performance evaluation is conducted in Section 5. Related work is discussed in Section 6. Finally we conclude in section 7.

2. Problem Formulation

2.1. Preliminaries

For energy-efficient scheduling, the objective is to meet all reservation requirements with the minimum total energy consumption based on the following assumptions and definitions.

1. All data is given to the scheduler since we consider offline scheduling unless otherwise specified, the time is discrete in slotted window format. We partition the total time period [0, T] into slots of equal length (l_0) in discrete time, thus the total number of slots is k=T/l_0 (always making it a positive integer). The start-time of the system is set as s_0=0. Then the interval of a reservation request i can be represented in slot format as a tuple with the following parameters: [StartTime, EndTime, RequestedCapacity]=$[s_i, e_i, d_i]$. With both start-time s_i and end-time e_i are non-negative integers.
2. For all jobs, there are no precedence constraints other than those implied by the start-time and end-time. Preemption is not considered.

Definition 1. The Interval Length: given a time interval I_i= [s_i, t_i] where s_i and t_i are the start slot and end slot, the length of I_i is $|I_i|$=t_i-s_i. The length of a set of intervals I=$\bigcup_{i=1}^{k} I_i$, is defined as $len(I)$=$|I|$=$\sum_{i=1}^{k} |I_i|$, i.e., the length of a set of interval is the sum of the length of each individual interval.
Definition 2. The Interval Span: the span of a set of intervals, $span(I)$, is defined as the length of the union of all intervals considered.
Example#1: if I=$\{[1,4],[2,4],[5,6]\}$, then $span(I)$=|[1,4]|+|[5,6]|=(4-1)+(6-5)=4, and $len(I)$ =|[1,4]|+|[2,4]|+|[5,6]|=6. Note that $span(I) \leq len(I)$ and equality holds if and only if I is a set of non-overlapping intervals.
Definition 3. The Total Power-on Time: For any instance I and capacity

parameter $g \geq 1$, let $OPT(I)$ denote the minimum total power-on time of all PMs. For VM reservations, the power-on time here means the power-on time of all PMs, only including busy time, and the idle time is not counted. The PM will be turned off or put into sleep mode so that the energy consumption during idle time can be ignored.

Example#2: Note that the total power-on time of a machine is the sum of all intervals during which the machine is powered on. As in Example#1, a machine is busy (powered-on) during intervals [1, 4] and [5, 6], based on our definition of interval span for each job, the total power-on time of this machine is (4-1)+(6-5)=4 time units (or slots). The interval [4, 5] (idle period) is not counted into the total power-on time of the machine.

Definition 4. The Workload: for any job j, denote its process time as p_i=e_i-s_i, its workload is denoted by $w(j)$, which is its capacity demand d_j multiplies its process time p_j, i.e, $w(j)$=$d_j p_j$. Then the total workload of all jobs J is $W(J)$=$\sum_{j=1}^{n} w(j)$.

Definition 5. The Approximation Ratio: an offline deterministic algorithm is said to be C-approximation for the objective of minimizing the total energy consumption if its total energy consumption is at most C times that of an optimum solution.

Definition 6. Strongly divisible capacity of jobs and machines: the capacity of all jobs form a divisible sequence, i.e., the sequence of distinct capacities $d_1 \geq d_2 \geq ... \geq d_i \geq d_{i+1} \geq ...$ taken on by jobs (the number of jobs of each capacity is arbitrary) is such that for all $i > 1$, d_{i+1} exactly divides d_i. Let us say that a list L of items has divisible item capacity if the capacities of the items in L form a divisible sequence. Also, if L is the list of items and g is the total capacity of a machine, we say that the pair (L, g) is weakly divisible if L has divisible item capacities and strongly divisible if in addition the largest item capacity d_1 in L exactly divides the capacity g [3].

Example#3: If the total capacity of a PM is g=8, and the requested capacity of each VM is one of $\{1, 2, 4, 8\}$, then the sequence forms a strongly divisible capacity. Obviously, if all jobs have unit demand (eg. request only 1 CPU from the total capacity of 8 CPUs in a PM), then the sequence of requested capacities also forms a strongly divisible capacity.

In the following sections, unless otherwise specified, the strongly divisible capacity case is considered. Actually, in strongly divisible capacity configuration the CPU capacity of a VM represents the total capacity of (CPU, memory, storage) in a PM. For example, VM type 1-1(1) shown in Table 1 has

memory of 1.875GB, CPU of 1 unit, storage of 211.25GB, and type-1 PM as shown in Table 2 has memory of 30GB, CPU of 16 units, storage of 3380GB. Therefore, VM type 1-1(1) has CPU 1/16, memory 1/16 (=1.875/30), storage 1/16(=211.25/3380) of the total CPU,memory and storage capacity of type-1 PM, respectively. In this strongly divisible capacity case we can use the CPU capacity of a VM to represent the total capacity of a VM, especially the energy consumption model in Equ (5-11) is proportional to the CPU utilization.

Note that the assumption of strongly divisible capacity is a valid assumption and is used by commercial cloud service providers such as Amazon where the CPU capacity of different VM instances are often evenly divisible (see Table 1 and Table 2).

2.2. Problem Statement

The problem has the following formulation: the input is a set of n jobs (VM requests) $J= j_1, ..., j_n$. Each job j_i is associated with an interval $[s_i, e_i]$ in which it should be processed, where s_i is the start-time and e_i the end-time, both in discrete time. Set p_i=e_i-s_i as the process time of job j_i. For the sake of simplicity, we concentrate on CPU-intensive applications and consider CPU-related energy-consumption only. The capacity parameter $g \geq 1$ is the maximal CPU capacity a single PM provides. Each job requests a capacity d_i, which is a natural number between 1 and g. The power-on time of PM_i is denoted by its working time interval length b_i. The optimizing objective is to assign the jobs to PMs such that the total energy consumption of all PMs is minimized. Note that the number ($m \geq 1$) of PMs to be used is part of the output of the algorithm and takes integer value. This problem is called MinTEC problem for abbreviation. The following Observation 1 is given in [6]:

Observation 1 . For any instance J and capacity parameter $g \geq 1$, the following bounds hold:
The capacity bound: $OPT(J) \geq \frac{W(J)}{g}$
The span bound: $OPT(J) \geq span(J)$.
The capacity bound holds since g is the maximum capacity that can be achieved in any solution. The span bound holds since only one machine is enough when $g = \infty$.

Observation 2. The upper bound for the optimal total power-on time is: $OPT(J) \leq len(J)$. The equality holds when g=1, or all intervals are not

overlapped when $g \geq 1$.
Suppose for any scheduler S, the PMs are numbered as $PM_1, PM_2, \ldots$. We denote by J_i the set of jobs assigned to PM_i with the scheduler S. The total busy period of PM_i is the length of its busy intervals, i.e., b_i=$span(J_i)$ for all $i \geq 1$ where $span(J_i)$ is the span of the set of job intervals scheduled on PM_i.

Formally, assuming there are m PMs in a Cloud Data center, E_i is the energy consumption of PM_i during test, the problem (MinTEC) can be restated as an optimization problem:

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{m} E_i && (1)\\ &\text{subject to} && \text{(a)} \quad \forall \text{ slot } s, \sum_{VM_j \in PM_i} d_j \leq g \\ & && \text{(b)} \quad \forall j_i, 0 \leq s_i < e_i \end{aligned}$$

where (a) means that the sum of the capacity of all VMs (VM_j) on a PM (PM_i) cannot be more than the available capacity a PM can offer; (b) means that each request has a fixed start-time s_i and end-time e_i, i.e., the processing interval is fixed.

THEOREM 1: The lower bound of the total power-on time for MinTEC problem is the sum of the minimum number of machines used in each slot, i.e., the lower bound is to allocate exactly minimum number of machines needed to each time slot.
Proof: The main problem MinTEC aiming to address is offline scheduling, for a given set of jobs J, we can find the minimum number of machines needed for each time slot, denoted as $l_1, l_2, ...l_k$ for total k time slots under consideration, where l_i is the minimum number of machines needed for time slot i. By the definition of the interval span and power-on time of each machine, $OPT(I) = \sum_{i=1}^{k} \lceil \frac{L_i}{g} \rceil = \sum_{i=1}^{k} l_i$, here L_i is the sum of load for time slot i. The total power-on time of all machines is the sum of minimum number of machines in all time slots in this way, i.e., the lower bound is the sum of the minimum number of machines used in each slot. This is the minimum total power-on time of all machines. This completes the proof.
Remark#1: The theoretical lower bound given in THEOREM 1 is not easy to achieve if each request has to be processed on a single PM without migra-

tion. Finding a subset of jobs for each machine to minimize total power-on time is known to be NP-complete [10].

Example #4: As shown in Figure 2.2, considering there are 4 job requests

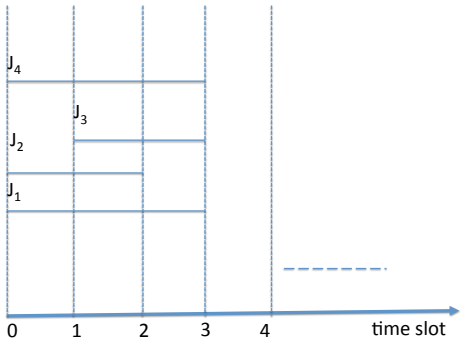

Figure 2.2: Referred architecture of VM reservation in Cloud Data centers

and g=3, Jobs J_1, J_2, J_3, J_4 have start-time, end-time and capacity demand [0, 3, 1], [0, 2, 1], [1, 3, 1], [0, 3, 1] respectively. The minimum number of PMs needed is 1, 2, 1 respectively in three time slots and total power-on time is 4 by theoretical lower bound. Without job migration, one solution is to allocate J_1, J_2 and J_4 to one PM and J_3 to another PM; or allocate J_1, J_3, J_4 to one PM and allocate J_2 to another PM; in either case, the actual total number of PMs needed is 2 and the total power-on time is 5. With job migration, one can allocate J_1, J_2 and J_4 to one PM (m_1) during interval [0, 3], allocate J_3 to another PM (m_2) during interval [1, 2] and migrate J_3 to m_1 during interval [2,3]; in this way, the total power-on time is 4, equals to the lower bound.

To see the hardness of the MinTEC problem, its NP-completeness is proved as follows:

THEOREM 2. MinTEC problem is a NP-complete problem in the general case.

Proof: For completeness, we sketch the proof as follows by reduction a known NP-complete problem to MinTEC problem. We know that K-PARTITION problem is NP-complete [13]: for a given arrangement S of positive numbers and an integer K, partition S into K ranges so as the sums of all the ranges are close to each other. K-PARTITION problem can be reduced to our

MinTEC problem as follows. For a set of jobs J, each has capacity demand d_i (set as positive number), partitioning J by their capacities into K ranges, is the same to allocate K ranges of jobs with capacity constraint g (i.e. the sum of each range is at most g). On the other hand, if there is a solution to K-PARTITION for a given set of intervals, there exists a schedule to MinTEC problem for the given set of intervals. Since K-PARTITION is NP-hard in the strong sense, our problem is also NP-hard. In this way, we have found that the MinTEC problem is NP-complete problem.

As proved in [6], it is NP-hard to approximate our problem already in the special case where all jobs have the same (unit) processing time and can be scheduled in one fixed time interval, by a simple reduction from the subset sum problem.
Remark#2: This can also be proved by reducing a well-know NP complete problem, the set partitioning problem to our (MinTEC) problem in polynomial time (see for example [10] for a proof).

THEOREM 3. MinTEC problem obtains optimum result if job migration is allowed.
Proof: From THEOREM 1, we know that there is a theoretical lower bound for MinTEC problem. The MinTEC as proved in THEOREM 2, is NP-complete in general case without job migration. However with job migration, a job can be migrated from one PM to another PM to be continuously proceeded, it is possible to obtain the lower bound. The method is introduced in Algorithm 2.1 OPT-Min-Migration. Algorithm 2.1 firstly sorts all jobs in non-decreasing order of jobs' start-time (line 1) and represents load of each slot by the minimum number of machines needed (line 3-4); then it finds the longest continuous interval $[z_1, z_2]$ with the same load and separates jobs into two groups (line 5-9); it allocates jobs in each group by First Fit Decreasing (FFD); and migrates the job to an existing PM when the minimum number of machines will be more than the slot load (line 12-15); it updates load of each PM and repeats the major steps until all jobs are allocated (line 17-21). Basically, if a new allocation passes through an interval that already has the minimum number of machines used (by the lower bound calculation), then during this interval, the new allocation will be migrated to an existing machine that still can host in that interval, so that no more than the minimum number of machines is needed for any slot (or interval). Because the minimum number of machines needed in each slot can be found exactly and

the number of migrations (i.e., the minimum number of migrations) can be found by Algorithm 2.1. In this way, the algorithm obtains the theoretical lower bound (denoted as OPT in this paper) with the cost of the minimum number of total migrations. This completes the proof.

The OPT-Min-Migration finds the lower bound with the cost of minimum number of job migrations. Without job migration, only approximation is possible. In the following, a 2-approximation algorithm is proposed.

3. The Longest Loaded Interval First Algorithm

In this section, a 2-approximation algorithm called Longest Loaded Interval First (LLIF) is introduced. The LLIF algorithm schedules the requests from the longest loaded slots first. The LLIF algorithm is described in Algorithm 3.1:

LLIF algorithm is similar to Algorithm 2.1 except that there is no job migration in LLIF algorithm. LLIF firstly finds the longest continuous interval with the same load, denoted as $[z_1, z_2]$, and separates jobs in $[z_1, z_2]$ as end-time first and start-time first groups, considers the longest job firstly in the same group; then it decides if the theoretical maximum load (number of PMs) is reached in $[z_1, z_2]$, if not, it allocates the job to the first available PM or opens a new PM when needs, else the allocation is migrated to an existing PM which still can host in $[z_1, z_2]$. LLIF updates the load of each PM and continues this process until all jobs are allocated.

Observation 3. The case that d_i=1 as shown in [4], called Unit Demand Case, is a special case of $1 \leq d_i \leq g$ (let us call it General Demand Case). As for minimizing total power-on time, Unit Demand Case represents the worst case scenario for LLIF.

Proof: The proof is sketched here for better understanding. Consider the General Demand Case, i.e.,$1 \leq d_i \leq g$. The adversary generates the following case: there are g^2 jobs in g groups, each group of jobs have the same start-time at s_i=0, demand d_i (for $1 \leq i \leq h$, and $\sum_{i=1}^{h} d_i = g$), each has end-time at e_i=$\frac{T}{k^{g-j}}$ where T is the time length of consideration, k is natural number, and if $(\ i \ mod\ g) \neq 0$, then set j=$(\ i \ mod\ g)$; else j=g. In this case, for the optimal solution, one can allocate all the longest requests to a machine (M_1) for a power-on time of d_gT, then allocates all the second longest requests to another machine (M_2) for a power-on time of $\frac{d_{g-1}T}{k}$, ... , and finally allocates

all the shortest requests to machine (M_g) with a power-on time of $\frac{d_1 T}{k^{g-1}}$. The total power-on time of optimal solution therefore is :

$$OPT(I) = \sum_{i=1}^{g} \frac{d_i T}{k^{g-i}} = T \sum_{i=1}^{g} \frac{d_i}{k^{g-i}} \tag{2}$$

We consider the worst case (the upper bound). For any offline algorithm, let us call ALG_X, the upper bound is to make $\frac{ALG_X}{OPT}$ the largest while keeping other conditions unchanged. Obviously, if OPT has the smallest value, the equation (2) will have the largest value. When k, g and T is given, the equation (2) will have smallest value if d_i has the smallest value, i.e., d_i=1. This means that Unit Demand Case represents the worst-case scenario and the proof is completed.
In the following section, the worst case (unit demand case) is considered.
THEOREM 4. The approximation ratio of our proposed LLIF algorithm for MinTEC problem has an upper bound 2.
Proof: Let us assume that all the jobs in subset J_i are assigned to machine M_i. For such a set, the total power-on time of the assignment is exactly its span. We just consider the upper bound for the worst case.

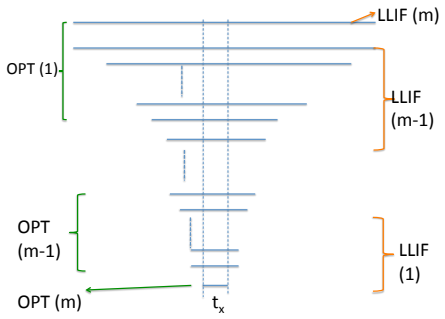

Figure 3.3: The upper bound for LLIF algorithm

Ideally LLIF(J) equals to the optimal solution by the definition of interval span since it behaves as THEOREM 1 suggests, allocating the minimum number of machines to each time slot. But in some cases, this is not generally true. We further construct an adversary[2] for LLIF algorithm and provide

[2]According to the knowledge of the algorithm, the adversary generates the worst pos-

proof in the following: The adversary as shown in Figure 3.3, submits $(kg+1)$ jobs forming a clique (this is the case that all job intervals intersect each other, see [4, 6] for a formal definition), k is a positive integer, all started and ended at different time with different span lengths, and sorted in non-decreasing order of their start-time (similarly, span lengths in this case). The total power-on time of the optimal solution is determined by the span length of the longest job with span T_1, (g+1)-th job with span T_{g+1}, (2g+1)-th job,..., and the shortest job (assuming that the shortest job has the longest loaded interval comparing to all jobs in this case), this is to consider allocation from the top to the bottom. LLIF treats the longest loaded interval first, its total power-on time is determined by the (kg-g+1)-th job, (kg-2g+1)-th job,..., the 2-nd longest job with span T_2, and the longest job with span T_1 (one job left for a single machine), this is to allocate from the bottom to the top. In this case

$$\begin{aligned} \frac{LLIF(I)}{OPT(I)} &= \frac{T_1 + T_2 + T_{g+2} + ...}{T_1 + T_{g+1} + T_{2g+1} + ...} \\ &= \frac{1 + \frac{T_2}{T_1} + \frac{T_M}{T_1}}{1 + \frac{T_{g+1} + T_{2g+1} + T_O}{T_1}} \end{aligned} \tag{3}$$

where T_M, T_O are the remaining time span for other jobs in LLIF and OPT, respectively. Equation (3) will have upper bound 2 when T_1=T_2 and other span lengths are negligible comparing to T_1; for other cases, LLIF(I) equals to OPT(I). One can also easily check that LLIF(I)=OPT(I) for clique, proper intervals and other special cases discussed in [4][6]. This completes the proof. Our extensive simulation results validate THOREM 4 in performance evaluation section.

4. Applications to Energy Efficiency of Virtual Machine Reservations

In this section, we introduce how our results are applied to VM reservations in a Cloud Data center. We consider that virtual machine reservation for CPU-intensive applications in Cloud Data centers where CPU in PMs are major resources [2][6]. Each VM has a start-time s_i, end-time e_i, CPU

sible input for the algorithm.

capacity demand d_i. The CPU capacity demand (d_i) of a VM is a natural number between 1 and the total CPU capacity (g) of a PM. These features are also reflected in Amazon EC2. Our objective here is to minimize total energy consumption of all PMs. This is exactly the same as the MinTEC problem. So we can apply the results of the MinTEC problem to the energy-efficiency of VM reservations. The metrics for energy consumption will be presented in the following.

4.1. Metrics for energy-efficiency scheduling

4.1.1. The power consumption model of a server

There are many research works in the literature indicating that the overall system load is typically proportional to CPU utilization (see Beloglazov et al. [1], Matthew et al. [31]). This is especially true for CPU-intensive computing where CPU utilization dominates. The following linear power model of a server is widely used in literature (see for example [1][31] and references therein).

$$\begin{aligned} P(U) &= kP_{max} + (1-k)P_{max}U \\ &= P_{min} + (P_{max} - P_{min})U \end{aligned} \tag{4}$$

where P_{max} is the maximum power consumed when the server is fully utilized, P_{min} is the power consumption when the server is idle; k is the fraction of power consumed by the idle server (studies show that on average it is about 0.7); and U is the CPU utilization. In a real environment, the utilization of the CPU may change over time due to the workload variability. Thus, the CPU utilization is a function of time and is represented as $U_i(t)$. Therefore, the total energy consumption (E_i) by a physical machine can be defined as an integral of the power consumption function during [t_0, t_1]:

$$E_i = \int_{t_0}^{t_1} P(U_i(t))dt \tag{5}$$

When the average utilization is adopted, we have $U_i(t) = U_i$, then

$$\begin{aligned} E_i &= P(U_i)(t_1 - t_0) = P(U_i)T_i \\ &= P_{min}T_i + (P_{max} - P_{min})U_iT_i \end{aligned} \tag{6}$$

where T_i is the power-on time of machine PM_i, the first term $P_{min}T_i$, is the energy consumed by power-on time of PM_i, denoted as $P_{min}T_i$=$E_{i_{on}}$; the

second term, $(P_{max} - P_{min})U_i T_i$ is the energy increase by hosting VMs on it. Assuming that a VM_j increases the total utilization of PM_i from U to U' and set $U' - U = u_{ij}$, and VM_j works in full utilization in the worst case. Defining E_{ij} as the energy increase after running VM_j on PM_i from time t_0 to t_1, we obtain that:

$$\begin{aligned} E_{ij} &= (P_{min} + (P_{max} - P_{min})U' - \\ &\quad (P_{min} + (P_{max} - P_{min})U))(t_1 - t_0) \\ &= (P_{max} - P_{min})(U' - U)(t_1 - t_0) \\ &= (P_{max} - P_{min})u_{ij}(t_1 - t_0) \end{aligned} \tag{7}$$

For VM reservations, we can further obtain that the total energy consumption of PM_i, the sum of its idle energy consumption ($E_{i_{on}}$) and the total energy increase by hosting all VMs allocated to it.

$$\begin{aligned} E_i &= E_{i_{on}} + \sum_{j=1}^{k} E_{ij} \\ &= P_{min} T_i + (P_{max} - P_{min}) \sum_{j=1}^{k} u_{ij} t_{ij} \end{aligned} \tag{8}$$

where u_{ij} is the utilization increase of PM_i with the allocation of VM_j, and t_{ij} is the time length (duration) of VM_j running on PM_i.

4.1.2. The total energy consumption of a Cloud Data center (CDC)

The total energy consumption of a Cloud Data center (CDC) is computed as

$$E_{CDC} = \sum_{i=1}^{n} E_i \tag{9}$$

It is the sum of energy consumed by all PMs in a CDC. Note that the energy consumption of all VMs on all PMs is included. The objective of our research is to minimize total energy consumption by considering time and capacity constraints. The following theorem establishes the relationship between total energy consumption, the total power-on time and the total workload of all

PMs in a CDC.

THEOREM 5. For a given set of VM reservations, the total energy consumption of all PMs is determined by the total power-on time and the workload of all PMs.

Proof: Set α=P_{min}, $\beta = (P_{max} - P_{min})$, we have

$$E_i = E_{i_{on}} + \sum_{i=1}^{k} E_{ij} \quad (From\ (6-7)) \tag{10}$$

$$\begin{aligned} E_{CDC} &= \sum_{i=1}^{m} E_i \\ &= \sum_{i=1}^{m} (\alpha T_i + \beta U_i T_i) \quad (From (7), (8)) \\ &= \alpha \sum_{i=1}^{m} T_i + \beta \sum_{i=1}^{n} \sum_{VM_j \in PM_i} u_{ij} t_{ij} \\ &= \alpha T + \beta L \end{aligned} \tag{11}$$

where $T = \sum_{i=1}^{m} T_i$ is the total busy (power-on) time of all PMs, L is total workload of all VMs (which is fixed once the set of VM requests is given). From equation (11), we can see that the total energy consumption of all PMs is determined by the total power-on time of all PMs and the total workload caused by hosting VMs on all PMs. This completes the proof.

From THEOREM 1-5, we also can induce the following observations, which are applicable to energy efficiency of VM reservations.

Observation 4. Applying Algorithm 2.1, OPT-MIN-Migration, we can have the minimum total energy consumption (i.e., the optimum result) for a given set of VM reservations in a Cloud Data center.

Observation 5. Applying LLIF algorithm for VM reservations, the approximation ratio has upper bound 2 regarding the total energy consumption comparing with the optimum solution.

Notice that the upper bound 2 is obtained for the worst case. As for average cases, we did intensive tests under different scenarios and find that LLIF

algorithm is near optimal.

Observation 6. For one-sided clique case where all jobs have the same start-time or end-time as discussed in [4, 6], our proposed Algorithm LLIF obtains optimal results.
Proof: For one-sided clique case, where all jobs have same start-time or end-time. Since LLIF considers the longest loaded interval first, in this case it is to allocate the longest group of jobs to the first PM, and the second longest group jobs to the second PM, and so on. This is exactly the same as the optimum solution does. This completes the proof.

5. Performance Evaluation

5.1. Settings

Table 1 shows eight types of VMs from Amazon EC2 online information, where one CPU unit equals to 1Ghz CPU of Intel 2007 processors, MEM is abbreviation for memory. Amazon EC2 does not provide information on its hardware configuration. However, we can therefore form three types of different PMs based on compute units. In a real Cloud Data center, for example, a PM with 2×68.4GB memory, 16 cores×3.25 units, 2×1690GB storage can be provided. The configuration of VMs and PMs are shown in Table 1 and 2. Table 3 also provides different P_{min} and P_{max} for different type of PMs, which are obtained from real power tests. For comparison, we assume that all VMs occupy all their requested capacity (the worst case). In this case, eight types of VMs are considered as shown in Table 1.

5.2. Algorithms

We considered four algorithms in this paper:

- First-Fit Decreasing (FFD): This algorithm introduced in [2], firstly sorts all requests in non-increasing order of their process time and then allocates the request to the first available PM. It has computational complexity of $O(nlogn)$ where n is the total number of requests.
- Earliest Start-Time First (EST): This algorithm firstly sorts all requests in non-increasing order of their start-time and then allocates the request to the first available PM. It has computational complexity of $O(nlogn)$ where n is the total number of requests.

- Longest Load Interval First (LLIF): This is our proposed algorithm in Section IV, the major idea is to repeatedly consider a group of the longest load interval span first in all slots. It has computational complexity of $O(nlogn)$ where n is the total number of requests.
- Optimal solution (OPT): This represents the theoretical lower bound, obtained by Algorithm 2.1. The computational complexity of finding this theoretical lower bound is $O(k)$ where k is the total number of slots considered, and can be ignored.

5.3. Simulation by Synthetically Generated Data

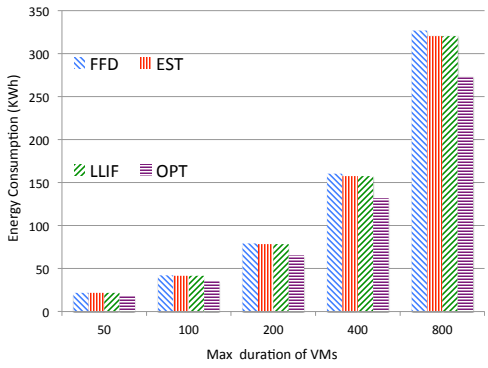

Figure 5.4: The comparison of total energy consumption when varying maximum duration of VM requests

A Java discrete simulator is used for performance evaluation (see [30] for the detailed introduction of the tool). All requests follow Poisson arrival process and have exponential service time, the mean inter-arrival period is set as 5 slots, the maximum duration of requests is set as 50, 100, 200, 400, 800 slots respectively. Each slot is 5 minutes. For example, if the requested duration (service time) of a VM is 20 slots, actually its duration is 20×5=100 minutes. For each set of inputs (requests), simulations are run 10 times and all the results shown in this paper are the average of the 10 runs. The total number of VMs is 1000 in all simulation.

Figure 5.4 provides the comparison of total energy consumption when varying maximum duration of VM requests. In this comparison, the maximum duration of VM requests is varying from 50 to 800 slots. It can be

seen that EST≥FFD>LLIF>OPT regarding total energy consumption in all cases. As we expected, LLIF in our experiments performs better than the theoretical worst case analysis of 2-approximation while it achieves results close to OPT. This validates our theoretical analysis which implies that LLIF incurs at most twice of the optimal in the worst case.

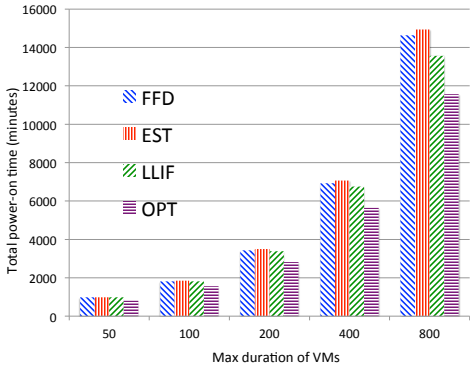

Figure 5.5: The total power-on time (minutes) of all PMs

Figure 5.5 shows the comparison of total power-on time when varying maximum duration from 50 to 800 slots of VM requests. It can be seen that EST≥FFD>LLIF>OPT regarding total energy consumption in all cases. Again results of LLIF are less than two times of results of optimal (OPT) solution.

Since the total energy consumption is strongly related to the total power-on time of all PMs, the similarity between Figure 5-4 and 5-5 are observed. Figure 5.6 also shows the comparison of the total running time of three algorithms EST, FFD and LLIF when the maximum duration of VMs is varying from 50 to 800. It can be observed that the total simulation running time of LLIF is slightly larger than both EST and FFD, while EST and FFD have running time close to each other. This is because LLIF spends more time on finding the longest load interval recursively as described in Algorithm 3.1. Also finding (theoretical) OPT results costs linear time with the total loads on all slots and can be computed in much shorter time than EST, FFD and LLIF. Note that the number of VM migrations in OPT is 2, 4, 6, 15, 27 when maximum duration varies from 50 to 800, respectively.

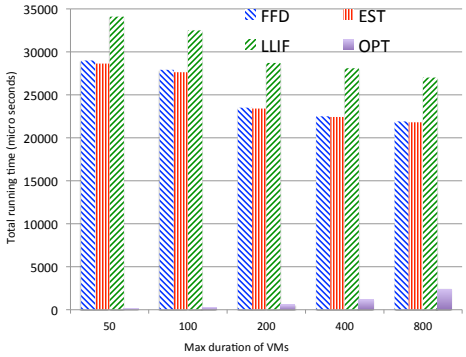

Figure 5.6: The comparison of the total running time (in micro seconds) when varying the maximum duration of VMs

5.4. Replaying with Real Traces

To be realistic, we utilized the log data of Parallel Workloads Archive (PWA) [28]. Because of lack of real data sets, the PWA data after conversion (stated in the text) can be representative of MinTEC model where jobs are represented in tuples [start-time, end-time, demand capacity]. The log contains months of records collected by a large Linux cluster. Each row of data in the log file contains 18 elements; we only need the requestID, start time, duration, and the number of processors in our simulation since these features are consistent with our problem model. To enable those data to be fit with our simulation, we convert the units from seconds in PWA log file into minutes, because we set a minute as a time slot length. Another conversion is that the different number of processors in PWA log file are corresponding to 8 types of VM requests. To simplify the simulation, three types of heterogeneous PMs and eight types of VMs are considered (can be dynamically configured and extended). We perform the simulation with enough PMs so that all VM requests can be allocated without rejection. Figure 5.7 and Figure 5.8 show the comparison of the total energy consumption (in Kilo Watts hours, KWh for abbreviation) and total power-on time (in minutes) respectively when varying the number of VMs using PWA [28] data. In this comparison, the total number of VMs is varying from 1000 to 7000 while other settings are the same. The minimum and maximum number of processors in the requests is 1 and 20, respectively. The average number

of processors is 12. It can be seen that EST≥FFD>LLIF>OPT regarding total energy consumption in all cases. And results of LLIF are less than two times of results of optimal (OPT) solution. Similar results as those by synthetically generating data for four algorithms are observed.

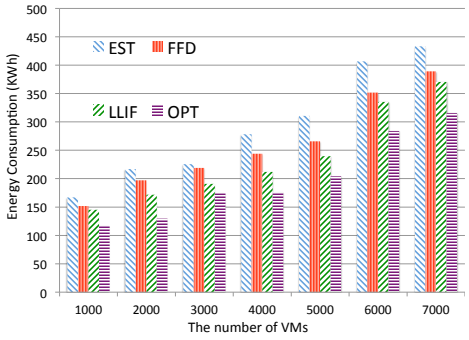

Figure 5.7: The comparison of total energy consumption (in KWh) when varying the number of VMs

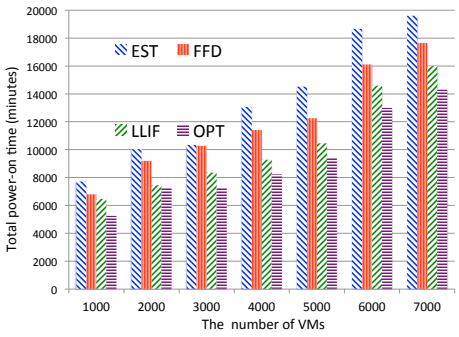

Figure 5.8: The comparison of total power-on time (in minutes) when varying the number of VMs

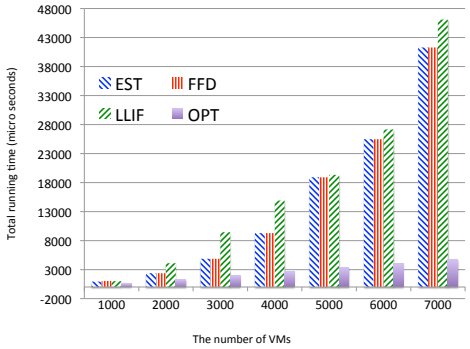

Figure 5.9: The comparison of the total running time (in micro seconds) when varying the number of VMs

Figure 5.9 shows the comparison of the total running time of three algorithms EST, FFD and LLIF when the total number of VMs is varying from 1000 to 7000. It can be observed that the total simulation running time of LLIF is slightly larger than both EST and FFD, while EST and FFD have running time close to each other. This is because LLIF spends more time on finding the longest load interval recursively as described in Algorithm 3.1. Finding (theoretical) OPT results costs linear time with the total loads on all slots and is negligible. Note that the number of VM migrations in OPT is 12, 23, 34, 87, 158 when the number of total VMs varies from 1000 to 7000, respectively.

In the simulation and trace-driven tests by the same configurations, we found that results of MFFDE are about 2%-10% more energy-saving than FFD on the average and LLIF is about 5%-15% more energy-saving than FFD on the average, this means LLIF is a few percentages more energy-saving than MFFDE.

6. Related Work

For the background and general introduction of cloud computing and energy-efficient scheduling, Beloglazov et al. [1] propose a taxonomy and survey of energy-efficient data centers and Cloud computing, especially the models for power consumption and energy consumption can be applied. Liu

et al. [12] present the GreenCloud architecture, which aims to reduce data center power consumption while guaranteeing the performance from users' perspective. Rings et al. [14] consider the opportunities for integrating Grid and Cloud computing with the next generation networks and suggest related standards. Rimal et al. [15] discuss architectural requirements for Cloud computing systems from an enterprise approach. Nunez et al. [13] introduce a simulator for Cloud infrastructure. Srikantaiah et al. [15] study the interrelationships between energy consumption, resource utilization, and performance of consolidated workloads. Lee et al. [10] introduce two online heuristic algorithms for energy-efficient utilization of resources in Cloud computing systems by consolidating active tasks. Feller et al. [34] propose a novel fully decentralized dynamic VM consolidation schema based on an unstructured peer-to-peer (P2P) network of PMs. Guazzone et al. [5] consider a two-level control model to automatically allocate resources to reduce the energy consumption of web-service applications. You et al. [19] investigate QoS-aware service redeployment problem (SRP) with objective to minimize the redeployment cost and propose a novel heuristic algorithm. Saovapakhiran et al. [16] design an algorithm for admission control and resource allocation in order to deal with unreliably excessive computing resources. Manvi et al. [35] bring out an exhaustive survey of resource scheduling techniques for IaaS in cloud computing and also put forth the open challenges for further research. Sharma et al. [36] present a thorough review of existing techniques for reliability and energy efficiency and their trade-off in cloud computing. In [37], Zhang et al. survey more than 150 articles in the latest years and review the state art of the algorithms to realize these objectives. Baker et al. [38] present a network-based routing algorithm to find the most energy efficient path to the cloud data centre for processing and storing big data. Baker et al. [39] develop a novel multi-cloud IoT service composition algorithm called (E2C2) that aims at creating an energy-aware composition plan by searching for and integrating the least possible number of IoT services, in order to fulfil user requirements.

For online energy-efficient scheduling, Kim et al. [7] model a real-time service as a real-time VM request, and use dynamic voltage frequency scaling schemes for provisioning VMs in Cloud Data centers. Tian et al. [27] propose an online scheduling algorithm for the problem of immediate (on-spot) requests.
As for offline energy-efficient scheduling, Beloglazov et al. [2] consider the

off-line VM allocation based on modified best-fit bin packing heuristics without considering VM life cycles where the problem formulation is different from our proposed one. Winkler et al. [18], Flammini et al. [4] and Khandekar et al. [6] are closely related to our research and are earlier papers to discuss this issue under general parallel machine scheduling context, and Kovalyov et al. [9] provide a comprehensive review for fixed interval scheduling problem. The MinTEC problem is NP-hard in general case [18]. Winkler et al. [18] show that the problem is NP-hard already for g=2, where g is the total capacity of a machine in term of CPU. Flammini et al. [4] consider the MinTEC scheduling problem where jobs are given as interval spans with unit demand, for this version of the problem a 4-approximation algorithm for general inputs and better bounds for some subclasses of inputs are provided. Khandekar et al. [6] propose a 5-approximation algorithm for this scheduling problem by separating all jobs into wide and narrow types by their demands when $\alpha = 0.25$, which is the demand parameter of narrow jobs occupying the portion of the total capacity of a machine. A 3-approximation algorithm is introduced in [32] for general offline parallel machine scheduling. Orgerie et al [33] discuss energy-efficient reservation framework for distributed systems with consideration of switching off unused resources for energy saving purposes and prediction algorithms employed to avoid useless off-on cycles.

Through extensive analysis of open literatures and references therein, we found that there is still lack of research on VM reservations considering both capacity and interval span constraints. Specifically, there is a need to consider the allocation of VMs with full life cycle constraints, which is often neglected [6, 12]. Since reservation services in Infrastructure as a Service (IaaS) is one of the key services widely provided by many operators, it is very important to develop energy-efficient resource scheduling [21].

7. Conclusions and future work

In this paper, an energy-efficient scheduling method for virtual machine reservations is proposed. We proposed an optimal solution with the minimum number of job migrations. Then we improved the best-known bound 3-approximation to 2-approximation by introducing LLIF algorithm. Most of our results are applicable to a single Cloud Data center as shown in Figure 1.1. As for federated systems, our results are readily applicable by considering

all machines in federated data centers. There are a few more open research issues for the problem:

- Finding better near-optimal solution and providing theoretical proofs for the approximation algorithms. Although the problem is NP-complete in general, we conjecture there is near-optimal solution for it. As for approximation algorithms, the theoretical approximation ratio comparing to optimal solution can be provided.

- Considering VM migration further and the energy consumption during migration transitions periods. Applying limited number of VM migrations, it is possible to reduce total energy consumption. However, frequently migrating VMs can also cause network vibration so that only limited number of VM migrations should be taken. For offline scheduling, it is also possible to take a limited number of migrations when allocation so that the total energy consumption can be reduced. We will investigate this further and consider energy consumption during migration.

- Combing energy-efficiency and load-balancing together. Just considering energy-efficiency may not be enough for real application because it may cause problems such as unbalance load for each PM. So we will combine load-balancing and energy efficiency together to provide an integrated solution.

  We are conducting research to further improve energy efficiency by considering these issues.

Acknowledgments

This research is sponsored by the National Natural Science Foundation of China (NSFC) (Grand Number:61672136, 61650110513, 61602434), Sicience and Technology Plan of Sichuan Provice (2016GZ0322), Xi Bu Zhi Guang Plan of Chinese Academy of Science (R51A150Z10). The problem statement as presented in Section 2.2 is also discussed in our earlier paper [32] although solutions provided in this paper are new.

References

[1] A. Beloglazov, R.Buyya, Y.C. Lee, and A.Y. Zomaya, *A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems*, Advances in Computers, vol. 82, pp. 47-111, M. Zelkowitz (editor), Elsevier, Amsterdam, The Netherlands, 2011.

[2] A. Beloglazov, J. Abawajy, 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.

[3] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson, 1987. *Bin-Packing with Divisible Item Sizes*, J. Complexity 3(1987), 406-428.

[4] M. Flammini, G. Monaco, L. Moscardelli, H. Shachnai, M. Shalom, T. Tamir and S. Zaks, S., *Minimizing Total power-on time in Parallel Scheduling with Application to Optical Networks*, Theoretical Computer Science. vol. 411 (40-42), pages 3553-3562, 2010.

[5] M. Guazzone, C. Anglano, M. Canonico, *Energy-Efficient Resource Management for Cloud Computing Infrastructures*, In Proceedings of 3rd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2011, pp.424-431, Nov. 29 2011-Dec. 1 2011, Athens.

[6] R. Khandekar, B. Schieber, H. Shachnai, and T. Tamir, *Minimizing power-on time in Multiple Machine Real-time Scheduling*, IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010), Pages.169-180.

[7] K. Kim, A. Beloglazov, R. Buyya, *Power-aware provisioning of virtual machines for real-time Cloud services*, Concurrency and Computation: Practice and Experience, vol. 23, no. 13, pp. 1491-1505, 2011.

[8] J. G. Koomey, GROWTH IN DATA CENTER ELECTRICITY USE 2005 TO 2010, technical report, August 1, 2011.

[9] M. Y. Kovalyov, C.T. Ng, E. Cheng, *Fixed interval scheduling: Models, applications, computational complexity and algorithms*, European Journal Of Operational Research, vol. 178, no. 2, pp. 331-342, 2007.

[10] Y.C. Lee, A.Y. Zomaya, *Energy Efficient Utilization of Resources in Cloud Computing Systems*, Journal of Supercomputing, vol. 60, no. 2, pp. 268-280, 2012.

[11] K. Li, Optimal power allocation among multiple heterogeneous servers in a data center, Sustainable Computing: Informatics and Systems 2 (2012) 13-22.

[12] L. Liu, H. Wang, X. Liu, X. Jin, W.B. He, Q.B. Wang, Y. Chen, *Greencloud: a new architecture for green data center*, Proceedings of 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session, ICAC-INDST'09, pages 29-38, New York, NY, USA, 2009. ACM.

[13] A.Nunez, J.L. Vzquez-Poletti, A.C. Caminero, G. G. Casta, J. Carretero, I.M. Llorente, J. Carretero, I. M. Llorente, iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator, Journal of Grid Computing (2012) 10:185-209.

[14] T. Rings, G. Caryer, J. Gallop, J. Grabowski, T. Kovacikova, S. Schulz, I. Stokes-Rees, Grid and Cloud Computing: Opportunities for Integration with the Next Generation Network, Journal of Grid Computing (2009) 7:375-393.

[15] B. P. Rimal , A. Jukan , D. Katsaros , Y. Goeleven, Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach, Journal of Grid Computing March 2011 , Volume 9 , Issue 1 , pp 3-26.

[16] B. Saovapakhiran, M. Devetsikiotis, *Enhancing Computing Power by Exploiting Underutilized Resources in the Community Cloud*, In Proceedings of IEEE International Conference on Communications (ICC 2011), pp. 1-6, 5-9 June 2011, Kyoto.

[17] S. Srikantaiah, A. Kansal, F. Zhao, *Energy Aware Consolidation for Cloud Computing*, In Proceedings of the 2008 conference on Power aware computing and systems, pp. 1-10, USENIX Association Berkeley, CA, USA.

[18] P. Winkler, L. Zhang, *Wavelength assignment and generalized interval graph coloring*. In SODA, pages 830-831, 2003.

[19] K. You, Z. Qian, S. Guo, S. Lu, D. Chen, *QoS-aware Service Redeployment in Cloud*, In Proceedings of In Proceedings of IEEE International Conference on Communications,ICC 2011, pp. 1-5, 5-9 June 2011, Kyoto.

[20] L. Youseff, et al., *Toward A Unified Ontology Of Cloud Computing*, In Proceedings of Grid Computing Environments Workshop, GCE'08, pp.1-10, 12-16 Nov. 2008, Austin, TX.

[21] Amazon EC2, http://aws.amazon.com/ec2/.

[22] DMTF Cloud Management, http://www.dmtf.org/standards/cloud

[23] Google App Engine, https://cloud.google.com/

[24] IBM blue cloud, http://www.ibm.com/grid/

[25] Microsoft Windows Azure, http://www.microsoft.com/windowsazure

[26] VMWare, http://www.vmware.com/

[27] WH.Tian, Q. Xiong, J. Cao, An Online Parallel Scheduling Method With Application to Energy-Efficiency in Cloud Computing, Journal of Supercomputing, December 2013, Volume 66, Issue 3, pp 1773-1790.

[28] Parallel Workloads Archive, www.cs.huji.ac.il/labs/parallel/workload, Last Access, April 2013.

[29] T. Knauth, C. Fetzer, Energy-aware Scheduling for Infrastructure Clouds, In Proceedings of CloudCom 2012, pp. 58-65, IEEE Computer Society Washington, DC, USA.

[30] WH. Tian, Y. Zhao, MX. Xu, YL. Zhong, XS. Sun, A Toolkit For Modeling and Simulation of Real-time Virtual Machine Allocation in a Cloud Data Center, IEEE Transactions on Automation Science and Engineering,(Online, July 2013), pp.153-161, Volume 12, Number 1, January 2015.

[31] V. Mathew, R. K. Sitaraman and P.Shenoy, Energy-Aware Load Balancing in Content Delivery Networks, In Proceedings of INFOCOM 2012, 25-30 March 2012, pp. 954-962, Orlando, FL.

[32] WH. Tian, and CS. Yeo, Minimizing total busy-time in offline parallel scheduling with application to energy efficiency in cloud computing, Concurrency and Computation: Practice and Experience, online first, Nov., 2013.

[33] A.C.Orgerie. An Energy-Efficient Reservation Framework for Large-Scale Distributed Systems, PhD Thesis, Feb 20, 2012.

[34] E.Feller, C. Morin, A. Esnault, A Case for Fully Decentralized Dynamic VM Consolidation in Clouds, Research Report n8032, August 2012.

[35] S. S. Manvi, G.K. Shyam, Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey, Journal of Network and Computer Applications, Volume 41, May 2014, Pages 424-440.

[36] Y. Sharma, B. Javadi, W. Si, D.Sun,Reliability and energy efficiency in cloud computing systems: Survey and taxonomy, Journal of Network and Computer Applications, Volume 74, October 2016, Pages 66-85.

[37] Jiangtao Zhang, Hejiao Huang, Xuan Wang, Resource provision algorithms in cloud computing: A survey, Journal of Network and Computer Applications, Volume 64, April 2016, Pages 23-42.

[38] T.Baker,B.Al-Dawsari, H.Tawfik,D.Reid, Y.Ngoko,GreeDi: An energy efficient routing algorithm for big data on cloud. Ad Hoc Networks,Volume 35, December 2015, Pages 83-96.

[39] Thar Baker, Muhammad Asim, Hissam Tawfik,Bandar Aldawsari, Rajkumar Buyya An energy-aware service composition algorithm for multiple cloud-based IoT applications, Journal of Network and Computer Applications, Volume 89, 1 July 2017, Pages 96-108.

Figure 7.10:

Figure 7.11:

Dr. Wenhong Tian has a PhD from Computer Science Department of North Carolina State University. He is a professor at University of Electronic Science and Technology of China. His research interests include dynamic resource scheduling algorithms and management in Cloud Data Centers, dynamic modeling and performance analysis of communication networks. He published about 40 journal and conference papers, and 3 English books in related areas. He is a member of ACM, IEEE and CCF.

Mr. Majun He is a master student at University of Electronic Science and Technology of China. His research interests include approximation algorithm for NP-hard problems, and scheduling algorithms for BigData processing platforms such as Spark.

Ms.Wenxia Guo is a PhD candidate at University of Electronic Science and Technology of China. Her research interests include approximation algorithm for NP-hard problems, and scheduling algorithms for resource allocation in Cloud Computing and BigData processing.

Mr. Wenqiang Huang is a master student at University of Electronic Science and Technology of China. His research interests include approximation algorithm for NP-hard problems, and scheduling algorithms for resource allocation in Cloud Computing and deep learning platforms such as Tensorflows.

Figure 7.12:

Figure 7.13:

Figure 7.14:

Figure 7.15:

Figure 7.16:

Dr. Xiaoyu Shi is an associate researcher at Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences,Chongqing,China . His research interests include cheduling algorithms for energy efficiency in Cloud Computing and deep learning platforms.

Dr. Mingsheng Shang is a researcher at Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences,Chongqing,China . His research interests include recommendation systems, Bigdata processing and deep learning.

Dr. Adel Nadjaran Toosi is a Research Fellow/Lecturer at the dept. of Computing and Information Systems of the University of Melbourne, Australia. He received his PhD degree in 2014 from the dept. of Computing and Information Systems of the University of Melbourne. His research interests include Distributed Systems, Cloud Computing, Cloud Federation and Inter-Cloud. His main focus is on pricing strategies, market and financial solutions for Cloud computing. Currently, he is working on economic aspects of the Inter-Cloud project, a framework for federated Cloud Computing.

Prof. Rajkumar Buyya is Professor of Computer Science and Software Engineering, Future Fellow of the Australian Research Council, and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He has authored over 450 publi-

Figure 7.17:

cations and four text books. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=87, g-index=176, 37500+ citations). Microsoft Academic Search Index ranked Pr. Buyya as the world's top author in distributed and parallel computing between 2007 and 2012.

Input: A Job instance $J=\{j_1, j_2, \ldots, j_n\}$, and g, the maximum capacity g of a machine

Output: The scheduled jobs (j_i), total power-on time (T_{on}) of all machines, the number of total PMs (H) used, and the number of migrations V_i in each slot i.

```
1  Sort all jobs in non-decreasing order of their start-time (s_i for job i),
   such that s_1 ≤ s_2... ≤ s_n, set H=1;
2  forall the slots under consideration do
3      Consider Divisible Capacity, represent the load of slot i by ⌈l_i⌉
       (the minimum number of machines needed for it, taking integral
       value by ceiling function)
4  end
5  forall the jobs under consideration do
6      Find the longest continuous interval with the same load first,
       denoted as [z_1, z_2];
7      forall the jobs either started or ended in [z_1, z_2] do
8          separating jobs into end-time first (ETF) and start-time first
           (STF) groups, consider the longest job first in the same group;
9          if l_i is not reached in all slots of this interval then
10             allocate to the first machine, open a new machine and set
               H=H+1 if needed;
11             else
12                 forall the slots that the minimum number of machines
                   will be more than l_i by new allocation do
13                     the allocation is migrated to an existing machine
                       which still have available capacities (g is not fully
                       used) in those slots, updates the number of
                       migrations (V_i) in each slot in [z_1, z_2].
14                 end
15             end
16         end
17         update the load of M_H, remove allocated jobs;
18     end
19     T_on=∑(T_i);
20     Return the set of machines used, and the total power-on time of all
       machines
21 end
```

Algorithm 2.1: OPT-Min-Migration

Input: A Job instance $J=\{j_1, j_2, \ldots, j_n\}$, the maximum capacity g of a machine

Output: The scheduled jobs (j_i), the number of total PMs (H) used and total power-on time (T_{on}) of all machines

1 Sort all jobs in non-decreasing order of their start-time (s_i for job i), such that $s_1 \leq s_2 ... \leq s_n$, set H=1;
2 **forall the** slots under consideration **do**
3 Consider Divisible Capacity, represent the load of slot i by $\lceil l_i \rceil$ (the minimum number of machines needed for it, taking integral value by ceiling function)
4 **end**
5 **forall the** jobs under consideration **do**
6 Find the longest continuous interval with the same load first, denoted as $[z_1, z_2]$;
7 **forall the** jobs either started or ended in $[z_1, z_2]$ **do**
8 separating jobs into end-time first (ETF) and start-time first (STF) groups, consider the longest job first in the same group;
9 allocate to the first machine, open a new machine and set H=H+1 if needed;
10 **end**
11 update the load of M_H, remove allocated jobs;
12 **end**
13 Count the workload and power-on times of all machines ;
14 Return the set of machines used, and the total power-on time of all machines

Algorithm 3.1: Longest Loaded Interval First (LLIF)

Table 1: 8 types of virtual machines (VMs) in Amazon EC2

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

Table 2: 3 types of PMs for strongly divisible capacity configuration

PM	CPU (units)	MEM (GB)	Storage(GB)
1	16 (4 cores x 4 units)	30	3380
2	52 (16 cores x 3.25 units)	136.8	3380
3	40 (16 cores x 2.5 units)	14	3380

Table 3: 3 types of PMs With Energy Consumption Metrics

PM	CPU (units)	MEM (GB)	Storage (GB)	P_{min}	P_{max}
1	16	30	3380	210	300
2	52	136	3380	420	600
3	40	14	3380	350	500