International Journal of Scientific Research in Computer Science and Engineering
Vol.6, Issue.6, pp.16-26, December (2018)
Research Paper
E-ISSN: 2320-7639

Scheduling Reservations of Virtual Machines in Cloud Data Center for Energy Optimization

Sebagenzi Jason[1*], Suchithra. R[2]

[1]Jain University, Bangalore-560043, India
[2]Head of Department of MSc IT, Jain University, Bangalore-560043, India

**Corresponding Author: sebagenzij@yahoo.fr*

Available online at: www.isroset.org

Received: 07/Dec/2018, Accepted: 22/Dec/2018, Online: 31/Dec/2018

Abstract – The present paper examine the scheduling reservations of energy-efficient of virtual machine (VM) in a Cloud Data center. Focusing on CPU-intensive applications, the target of this paper is to schedule all reservations non-preemptively, subjecting to constraints to capacities of physical machine (PM) and running time interval spans, in order to minimize the consumption of the total energy of all physical machines. This problem is an NP-complete. The best solution for this problem is a 5-approximation algorithm by using First-Fit-Decreasing algorithm and 3-approximation algorithm for in case of offline parallel machine scheduling with unit demand. Combining the characteristics of workload and optimality in interval spans, a method to find the optimal solution with the minimum number of job migrations is proposed, and a 2-approximation algorithm called Longest Loaded Interval First algorithm (LLIF) for general cases. At the end, how the algorithms are applied to minimize the total energy consumption in a Cloud Data center will be shown.

Keywords **-** Virtual machine reservation, Energy efficiency, Cloud Data centers, Resource scheduling.

I. INTRODUCTION

Cloud computing has progressed from different recent advancements in Grid computing, virtualization, utility computing, web computing and other associated technologies. It provides three level of services, that is to say Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Software as a Service (SaaS). In this paper, we focus on CPU intensive computing at IaaS level in Cloud Data centers.

Virtual machine reservation services is provided by Cloud computing providers (such as Amazon) with specified computing units.

For that purpose, in advance, customers solicit some units of computing resources to use for a period of time in the future. Providers will have enough time to schedule. The resources in this paper comprise:

a) Physical Machines (PMs): Hardware-based devices which contain multiple virtual machines. Each Physical Machine can be composed by a CPU, hard drives, network cards, memory, and etc.
b) Virtual Machine (VMs): Virtual computing platforms on Physical Machines (PM) represented as an emulation of a computer system obtained by using softwares of virtualization. Each Virtual Machine has elements of virtual CPUs, storage, memory, network cards, and other components.

The process and architecture of Virtual Machine reservation scheduler are provided in Figure 1, by making reference to Amazon Elastic Computer Cloud (EC2).

As mentioned in the diagram, the important processes of resource scheduling are:

1. **User request reservation:** the user commence to do a reservation by using the Internet.
2. **Management Scheduling:** Scheduler Center take decisions by considering the operational characteristics of the request (quality and quantity requirements) and the user's identity. The request is given to a data center, then the data center management program submits it to the Scheduler Center, at the end, the Scheduler Center assign the request based on algorithms scheduling;
3. **Feedback to users:** Algorithms of scheduling provide resources which are available to the user.
4. **Scheduling execution:** Scheduling results (such as deploying steps) are sent to the next stage;
5. **Updating and optimization:** The scheduler updates information of a resource, optimizes resources in the data center according to the optimizing objective functions.

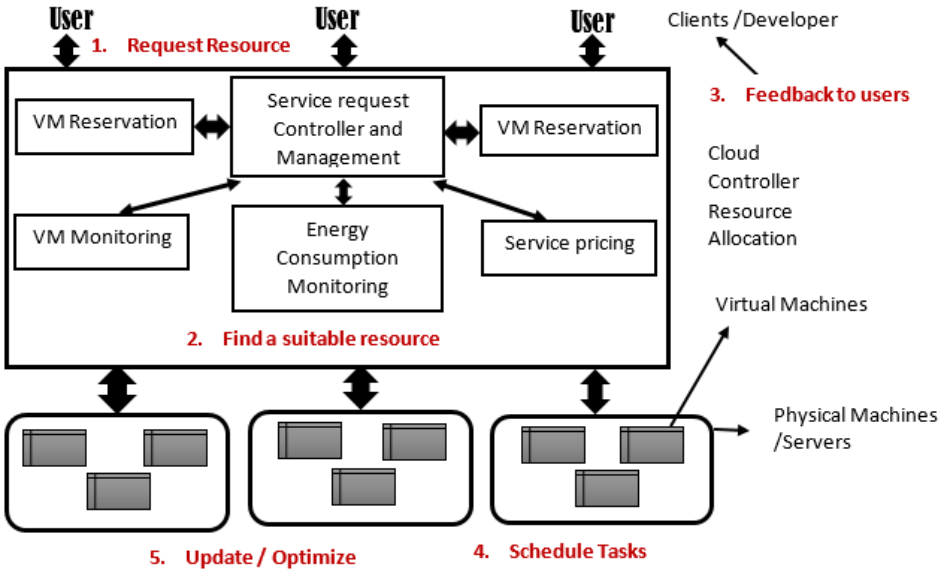

Figure 1. Architecture of Virtual Machine reservation in a Cloud data center

In the reservation services process, customers receive list of charges according to the energy of the computing resources as well as the total amount of time computing. The scheduler executes periodically for a fixed period of time depending on workloads in realistic scenarios.

From the provider's side, the total energy cost of computing resources is related to the total powered-on time of all computing resources. Because Cloud data centers utilize very large amounts of energy, its cost (electricity price) is regularly increasing.

So preference to reduce 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 like Round Robin and First-Fit are used by Elastic Computer Cloud (Amazon EC2) and VMWare (VMWare) [6] [8] [9].

To measure the performance 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 [18] [19].

Too many researchers are near of the present research and their papers discuss this issue under general parallel machine scheduling context, and provide a comprehensive review for the fixed interval scheduling problem [7] [12 [14].

The Virtual Machines problem of reservations can be formulated as follows: There are m deterministic reservations submitted to the scheduler in advance to be scheduled offline on multiple physical machines (PMs) with bounded capacities. Every Virtual Machine reservation is identified by a start-time, an end-time, and a capacity demand. The objective is to schedule all reservations non-preemptively, subjecting to constraints of Physical Machine capacities and running time interval spans, such that the total energy consumption of all PMs is reduced.

This problem is an NP-hard. The problem has been considered in optical networks and demonstrated that the problem is NP-hard already for $g = 2$, where g is the total capacity of a physical machine in terms of CPU [2] [4].

In this paper, the assumption that the total CPU capacity of a Physical Machine, g, is measured in abstract units such as EC2 Compute Unit (ECU).

Researchers was considering the same scheduling problem in optical network where jobs are given as interval spans with unit demand [3] [5].
For this kind 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 First Fit Decreasing algorithm sorts all jobs' process time in non-increasing order and allocates the job in that order to the first machine which can host [16] [17] [20].

Khandekar et al. (2010) was proposing a 5-approximation algorithm for this scheduling problem. He tries to separate all jobs into a wide and a narrow types by their demands when $\alpha = 0.25$, which is the parameter of demand of narrow jobs which are occupying the portion of the total capacity of a machine.

Tian et al. also was proposing 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. In this paper, the Virtual Machine requests and jobs are used interchangeably.

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

1. Proposing an approach to reduce total energy consumption of virtual machine reservations by minimizing total energy consumption of all Physical Machines.
2. Deducing a theoretical lower bound with limited number of Virtual Machine migrations.
3. Proposing an algorithm, which is good than the best-known 3-approximation algorithm.
4. Approving theoretical results by intensive simulation of trace-driven and synthetically generated data.

The organization of the paper is as follows: Background is provided in Section II. Section III presents problem statements. Section IV presents the longest loaded Interval First algorithm. Section V considers Performance evaluation. Conclusion is conducted in Section VI. Finally future work is discussed in section VII.

II. BACKGROUND

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 $k = \frac{T}{l_0}$ (always making it a positive integer). The starting-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] = $[s_i, e_i, d_i]$. Start-time s_i and end-time e_i are non-negative integers [10] [11].
2. There are no precedence constraints for all jobs other than those implied by the start-time and end-time. Preemption is not considered.

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

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

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

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 Physical Machines. For Virtual Machine reservations, the power-on time here means the power-on time of all Physical Machines, only including busy time, and the idle time is not counted. The Physical Machine will be turned off or put into sleep mode so that the energy consumption during idle time can be ignored.
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, $w(j) = djpj$. Then the total workload of all jobs *J* is $W(J) = \sum_{j=1}^{n} w(j)$.
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.
6. Strongly divisible capacity of jobs and machines: the capacity of all jobs form a divisible sequence, i.e., the sequence of distinct capacities $d_1 \geq d_2 \geq \ldots \geq d_i \geq d_i + 1 \geq \ldots$ taken on by jobs (the number of jobs of each capacity is arbitrary)

is such that for all $i > 1$, $d_i + 1$ exactly divides d_i .

7. Let us consider 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 di in L exactly divides the capacity g (Coffman et al., 1987).

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 Virtual Machine represents the total capacity of (CPU, memory, storage) in a Physical Machine. For Example, Virtual Machine type 1-1:

1. Shown in Table 1 has memory of 1.875 GB, CPU of 1 unit, $storage\ of$ 211.25 GB, and $type1\ Physical\ Machine$ as shown in Table 2 has memory of 30 GB, CPU of 16 $units$, storage of 3380 GB. However, VM type 1-1 (1) has CPU $\frac{1}{16}$, memory $\frac{1}{16}$ ($=\frac{1.875}{30}$, storage $\frac{1}{16}$ ($=\frac{211.25}{3380}$) 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 Virtual Machine to represent the total capacity of a Virtual Machine, especially the energy consumption model in Equation (5)–(11) is proportional to the CPU utilization [1] [13].

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 (13 cores x 2.5 units)	14	3380

Table 2: 3 types of physical machines for strongly divisible capacity configuration

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 Tables 1 and 2).

III. PROBLEM STATEMENT

The problem has the following formulation: the input is a set of n jobs (VM requests) $J = j_1, j_2, .. j_n$. Each job j_i is associated with an interval $[s_i, e_i]$ in which it should be processed, where si is the start-time and e_i the end-time, both in discrete time.

It is possible to set $p_i = e_i - s_i$ as the process time of $job\ j_i$. For 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 Physical Machine provides. Every job requests a capacity di, which is a natural number between 1 and g. The power-on time of PM_i is denoted by its working time interval length bi.

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 Physical Machines to be used is part of the output of the algorithm and takes integer value.

Imagine for any scheduler S, the PMs are numbered as $PM_1, PM_2, ...$we denote by J_i the set of jobs assigned to PMi with the scheduler S.

The total period of PMi is the length of its busy intervals, i.e., $bi = span\ (Ji)$ for all $i \geq 1$ where $span\ (Ji)$ is the span of the set of job intervals scheduled on PMi.

Formally, assuming there are m Physical Machines in a Cloud Data center, Ei is the energy consumption of PMi during test, the problem can be restated as an optimization problem:

Minimize $\sum_{i=1}^{m} Ei$ (1)

Subject to (a) $\forall$ slot s, $\sum_{VMj \epsilon PMi} dj \leq g$

(b) $\forall ji, 0 \leq s_i < e_i$

Where (a) means that the sum of the capacity of all Virtual Machines (VMj) on a Physical Machine (PMi) cannot be

more than the available capacity a PM can offer; (b) means that each request has a fixed start-time si and end-time e_i, i.e., the processing interval is fixed.

IV. THE LONGEST LOADED INTERVAL FIRST ALGORITHM

In this part, 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 same 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 every Physical Machine and continues this process until all jobs are allocated.

The case that $d_i = 1$as shown in (Flammini et al., 2010), called Unit Demand Case, is a special case of $1 \leq di \leq g$ (let us call it General Demand Case).

In order for minimizing total power-on time, Unit Demand Case represents the worst case scenario for LLIF.

Proof. The proof is demonstrate here to understanding better. Take a Consideration of the General Demand Case, i.e., $1 \leq di \leq g$. The adversary generates the following case: there are $g2$ 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 $(i \bmod g) \neq 0$, then
Set $j = (i \bmod g)$;
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 $\frac{d_{g-1}}{k}$, …., … and finally allocates all the shortest requests to machine (M_g) with a power-on time of $\frac{d_1 T}{k^{g-i}}$

The total power-on time of optimal solution therefore is formulated as follows:

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

We consider the worst case (the upper bound). For any offline algorithm, let us call ALG_X, the upper bound is to make $\frac{ALGX}{OPT}$ the largest while keeping other conditions unchanged. Cleary, if OPT has the smallest value, equation (2) will have the largest value. When g, k and T is given, equation (2) will have smallest value if di has the smallest value, i.e
$di = 1$. It has the meaning 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.

The approximation ratio of the proposed LLIF algorithm for $MinTEC$ problem has an upper bound 2.

Let assume that all the jobs in subset Ji are assigned to machine . For that a set, the total power-on time of the assignment is exactly its span. The consideration of the upper bound for the worst case is taken.

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 situation, this is not generally true.

The construction of an adversary2 for LLIF algorithm and provide Proof in the following: The adversary as shown in Fig. 3, can submit $(kg + 1)$ jobs forming a clique (this is the case that all job intervals intersect each other, 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 [15] [21].

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 (thinking 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 only job stay for a single machine), this is to allocate from the bottom to the top.

$$\frac{LLIF(I)}{OPT(I)} = \frac{T_1 + T_2 + T_{g+2} + \cdots}{T_1 + T_{g+1} + T_{2g+1} + \cdots}$$

$$= \frac{1 + \frac{T_2}{T_1} + \frac{T_M}{T_1}}{1 + \frac{T_{g+1} + T_{2g+1} + T_0}{T_1}} \qquad (3)$$

A. *Applications to energy efficiency of virtual machine reservations.*

In this section, we introduce how our results are applied to Virtual Machine reservations in a Cloud Data center. We consider that virtual machine reservation for CPU-intensive applications in Cloud Data centers where CPU in Physical Machines are major resources [1] [3].
Each Virtual Machine has a start-time s_i, end-time e_i, CPU capacity demand d_i. The CPU capacity demand (d_i) of a Virtual Machine is a natural number between 1 and the total CPU capacity (g) of a Physical Machine.
These features are also reflected in Amazon $EC2$. Our objective here is to minimize total energy consumption of all Physical Machines.
This is exactly the same as the $MinTEC$ problem. So we can apply the results of the $MinTEC$ problem to the energy-efficiency of Virtual Machine reservations. The metrics for energy consumption will be presented in the following [5] [7].

B. *The power consumption model of a server*

In the literature, there are many research which indicates that the overall load of the system is typically proportional to the utilization of the CPU. This is of course true for the computing of CPU-intensive where there is domination of utilization of CPU.
The following power model which is linear, of a server is much used in literature [20] [22].

$$P(U) = kP_{max} + (1-k)P_{max}U$$

$$= P_{min} + (P_{max} - P_{min})U \qquad (4)$$

Where P_{max} is the power consumed when the server is fully utilized, P_{min} is the power consumed when the server is idle; k is the power fraction consumed by the idle server (studies show that on average it is about 0.7); and U is the utilization of the CPU.

In a real situation, the CPU utilization may change over time due to the workload variability.

Algorithm 2.1 OPT-Min-Migration.

Input: A job instance $J = \{j1, j2, j3 \dots jn\}$ 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.
Sort all jobs in non-decreasing order of their start-time (s_i for job i), such that $s1 \leq s2 \leq s3 \dots \leq sn$, set H=1;
For all the slot under consideration **do**
Consider Division Capacity, represent the load of slot I by $[l_i]$ (the minimum number of machines needed for it, taking integral value by ceiling function)
End

For all the jobs under consideration **do**
Find the longest continuous interval with the same load first, denoted as $[z_1, z_2]$;
For all the jobs either started or ended in $[z_1, z_2]$ **do**
Separating jobs into end-time first (ETF) and start time first (STF) groups, consider the longest job first in the same group;
If l_i is not reached in all slots of this interval **then**
Allocate to the first machine, open a new machine and set $H = H + 1$ if needed;
Else
For all the slots that the minimum number of machines will be more than l_i, by new allocation **do**
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 $[z_1, z_2]$.
End
End
End
Update the load of M_H, remove allocated jobs;
End
$T_{on} = \sum T_i$
Return the set of machines used, and the total power-on time of all machines
End

Algorithm 1: OPT-Min Migration

Thus, the utilization of the CPU 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 $[t_0, t_1]$:

$$E_i = \int_{t0}^{t1} P\,(Ui(t))dt \qquad (5)$$

When the average utilization is adopted, we have $\boldsymbol{U_i(t) = U}$, then

$$E_i = P(U_i)(t_1 - t_0)$$

$$= P_{min}\,T_i + (P_{max} - P_{min})U_iT_i \qquad (6)$$

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, $(P_{max} - P_{min})U_iT_i$ 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:

$$E_{ij} = (P_{min} + (P_{max} - P_{min})U' - (P_{min} + (P_{max} - P_{\min})U))(t_1 - t_0)$$

$$= P_{max} - P_{min})(U' - U)(t_1 - t_0)$$

$$= (P_{max} - P_{min})U_{ij}(t_1 - t_0) \qquad (7)$$

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

$$E_i = E_{i_{on}} + \sum_{j=1}^{k} E_{ij}$$

$$= P_{min}T_i + (P_{max} - P_{min})\sum_{j=1}^{k} u_{ij}t_{ij} \qquad (8)$$

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.

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

The total energy consumption of a Cloud Data center (CDC) is calculated as follows:

$$E_{CDC} = \sum_{i=1}^{n} E_i \qquad (9)$$

Algorithm 3.1 *Longest Loaded Interval First (LLIF).*

Input: A job instance $J = \{j1, j2, j3 \ldots jn\}$ and g, the maximum capacity g of a machine

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

Sort all jobs in non-decreasing order of their start-time (s_i for job i), such that $s1 \le s2 \le s3 \ldots \le sn$, set $H = 1$;
For all the slot under consideration **do**
Consider Division Capacity, represent the load of slot I by $[l_i]$ (the minimum number of machines needed for it, taking integral value by ceiling function)
End

For all the jobs under consideration **do**
Find the longest continuous interval with the same load first, denoted as $[z_1, z_2]$;
For all the jobs either started or ended in $[z_1, z_2]$ **do**
Separating jobs into end-time first (ETF) and start time first (STF) groups, consider the longest job first in the same group; allocate to the first machine, open a new machine and set $H = H + 1$ if needed;
End
Update the load of M_H, remove allocated jobs;
End

Algorithm 2: Longest Loaded Interval First (LLIF)

It is the sum of energy consumed by all Physical Machines in a CDC. Note that the energy consumption of all Virtual Machines on all Physical Machines is included.
In this research, the objective 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.

For a given set of Virtual Machine reservations, the total energy consumption of all Physical Machines is determined by the total power-on time and the workload of all Physical Machines.

Set $\alpha = P_{min}, \beta = P_{max} - P_{min}$), *we have*

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

$$E_{CDC} = \sum_{i=1}^{m} E_i$$

$= \sum_{i=1}^{m} (\alpha T_i + \beta U_i T_i)$ **(From (7), (8))**

$$= \alpha \sum_{i=1}^{m} T_i + \beta \sum_{i=1}^{n} \sum_{VM_j \in PM_i} u_{ij} t_{ij}$$

$= \alpha T + \beta L$ **(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 Physical Machines is determined by the total power-on time of all Physical Machines and the total workload caused by hosting Virtual Machines on all Physical Machines.
This completes the Proof. From Theorem 1-5, we also can induce the following observations, which are applicable to energy efficiency of Virtual Machine reservations.

1. It is possible to have the minimum total energy consumption (i.e., the optimum result) for a given set of Virtual Machine reservations in a Cloud Data center by applying Algorithm 2.1, OPT-MIN-Migration,.
2. Applying LLIF algorithm for Virtual Machine 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.

3. The proposed algorithm LLIF obtains optimal results, for one-sided clique case where all jobs have the same start-time or end-time as discussed [17] [19].

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. The same as the optimum solution does.

V. PERFORMANCE EVALUATION

A. Settings

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

Table 3:3 types of Physical Machines with Energy Consumption Metrics.

PM	CPU (Units)	MEM (GB)	STORAGE (GB)	Pmin	Pmax
1	16	30	3380	210	300
2	52	136	3380	420	600
3	40	14	3380	350	500

Table 3: 3types of Physical Machine with Energy Consumption Metrics.

Eight types of Virtual Machines are shown in Table 1.

B. Algorithms

We considered four algorithms in this paper:

- First-Fit Decreasing (FFD): This algorithm introduced in (Beloglazov et al., 2012), sorts, in non-increasing order of the process time all requests and then allocates the request to the first available Physical Machine. It has computational complexity of $O(nlog\,n)$ where n is the total number of requests.
- Earliest Start-Time First (EST): This algorithm sorts all requests in non-increasing order of their start-time first and then allocates the request to the first available Physical Machine. It has computational complexity of $O\,(nlog\,n)$ where n is the total number of requests.
- Longest Load Interval First (LLIF): This is the proposed algorithm. The important idea is to repeatedly consider a group of the longest load interval span first in all slots. It has computational complexity of $O\,(nlog\,n)$ where n is the total number of requests.

- Optimal solution (OPT): This represents the lower bound, obtained by Algorithm 2.1. The computational complexity of finding this lower bound is $O(k)$ where k is the total number of slots considered, and can be ignored.

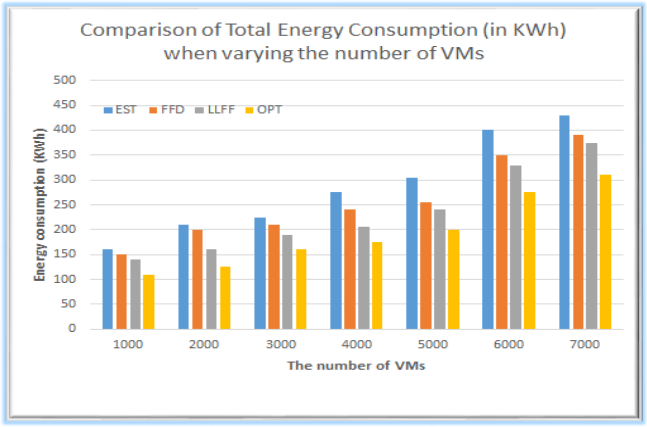

Figure 2: Total power-on time (minutes) of all PMs.

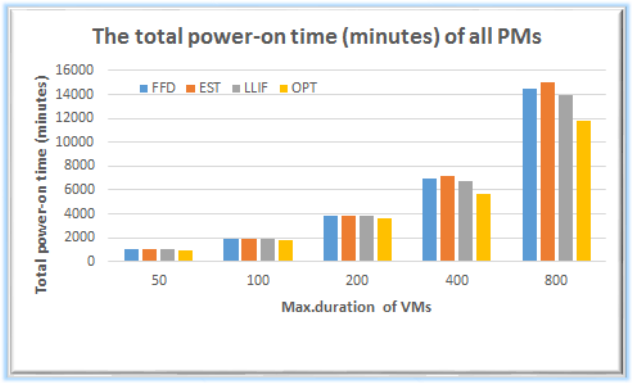

Figure 3: Comparison of total running time (in micro seconds) when varying the maximum duration of VMs.

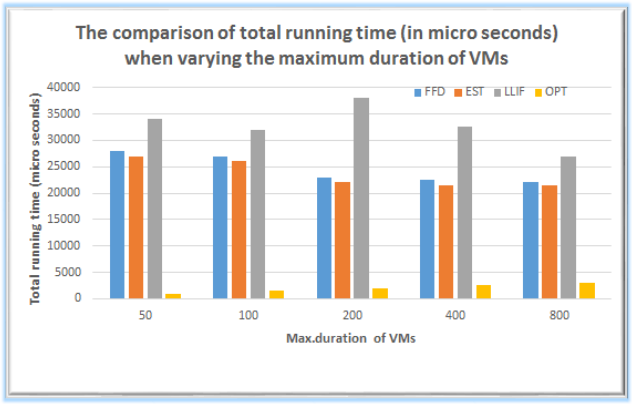

Figure 4: comparison of total energy consumption (in KWh) when varying the number of VMs.

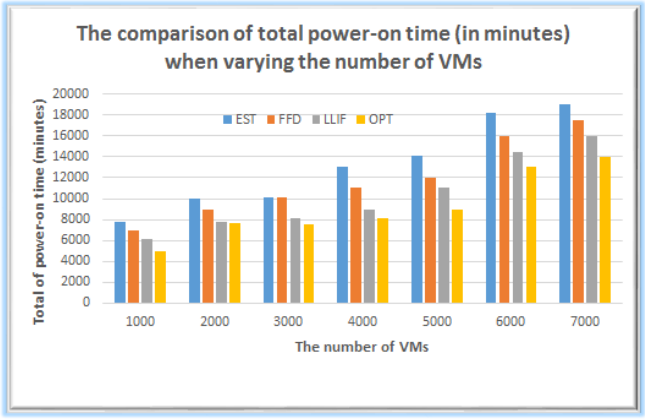

Figure 5: Comparison of total power-on time (in minutes) when varying the number of VMs.

VI. CONCLUSIONS

Eight types of Virtual Machines and three types of Physical Machines which are heterogeneous are considered. The simulation with enough Physical Machines has been performed so that all requests of Virtual Machine can be allocated without any rejection. Fig. 7 and Fig. 8 is presently showing the comparison of the total energy consumption and total power-on time (in minutes) respectively when varying the number of Virtual Machines using Parallel Workloads Archive data.

In this comparison, the total number of Virtual Machines is varying from 1000 to 7000 while other settings are the same.

The maximum is 20 and minimum number of processors is 1 in the requests, respectively. The average of processors is 12. It can be seen, regarding total energy consumption in all cases that EST ≥ FFD > LLIF > OPT. Results of LLIF are less optimal (OPT) solution.

It can be seen that the total 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 passes much time on finding, recursively, the longest load interval as shown in Algorithm 3.1.

Note that the number of Virtual Machine migrations in OPT is 12, 23, 34, 87, 158 when the number of total Virtual Machines varies from 1000 to 7000, respectively.

Results of LLIF is about 5%– 15% more energy-saving than FFD on the average and MFFDE are about 2%– 10% more energy-saving than FFD on the average, this means LLIF is a few percentages more energy-saving than MFFDE.

VII. FUTURE WORK

In this present paper, the efficiency of energy scheduling method for virtual machine reservations is suggested. The suggestion of a solution which is optimal with a number of job migrations which is minimal.

Then, the best-known bound 3-approximation has been ameliorated to 2-approximation by initiating Longest Loaded Interval First algorithm. Greatest of results are appropriate to only a single Cloud Data center as shown in Fig. 1. As for combined systems, results are willingly applicable by considering all machines in combined data centers.

Few more open research issues for the problem are there:

- Finding best above-optimal solution and providing proofs which are theoretical for the approximation algorithms. Although the problem is NP-complete in general, the presumption is above near-optimal solution for it. As for approximation algorithms, the approximation ratio comparing to optimal solution can be provided.
- Considering the energy consumption during migration transitions periods and Virtual Machine migration farther. Applying limited number of Virtual Machine migrations, it is possible to minimize the total energy consumption. However, often, migrating Virtual Machines can also cause vibration of network so that only limited number of Virtual Machine migrations can be taken. By considering offline scheduling, it is possible to take a number of migrations which is limited so that the total energy consumption can be minimized. Further, investigation will be done and also consideration of energy consumption during migration will be highlighted.
- Combining load-balancing and energy-efficiency together. Only considering energy-efficiency, for real application, might not be sufficient because it may cause problems like unbalance load for every Physical Machine. So the combination of load-balancing and energy efficiency can provide an integrated solution.

By considering these issues, the research can be conducted to further improvement of energy efficiency.

Acknowledgments

My acknowledgment is addressed to anyone who contributes and gave the input and constructive explanations on the improvements of the paper.

REFERENCES

[1] Baker, Thar, Asim, Muhammad, Tawfik, Hissam (2017). An energy-aware service composition algorithm for multiple cloud-based to applications. J.Netw. Comp. App. 89 (1), 96-108.

[2] Beloglizov, A., Abawajy, J., Buyya, R., 2012. Energy-aware resource allocation Heuristics for efficient management of data centers for cloud computing. Future Generat.Comput. System. 28 (5), 755 - 768.

[3] Beloglizov, A., Buyya, R., Lee, Y.C, Zomaya, A.Y., 2011. In:Zelkonwitz, M. (ed), A Taxonomy and survey of Energy-efficiency Data Centers and Cloud Computing Systems, Advances in computers, vol.82. Elsevier, Amsterdam, The Netherlands, PP 47 - 111.

[4] Bohrer, P., Elnozahy, E., Keller, T., Kistler, M., Lefurgy, C., McDowell, C., & Rajamony, R. (2014). The case for power management in Web servers. Norwell, MA, USA: Kluwer Academic Publishers.

[5] Bollen, J., & Heylighen, F. (2015). Algorithms for the Self-organization of Distributed, Multiuser Networks. Austrian: Cybernetics and Systems.

[6] Bonabeau, E., Dorigo, M., & Theraulaz, G. (2015). swarm Intelligence. USA: Oxford University Press.

[7] Coffman Jr., E.G., Garey, M.R., Johnson, D.S., 1987. Bin-Packing with divisible item sizes. J. Complex 3 (1987), 406 - 428.

[8] Elnozahy, E., Kistler, M., & Rajamony, R. (2015). Energy-Efficient server clusters. Power-Aware Computer Systems.

[9] Flammini, M., Monaco, G., Moscardelli, L., Shachnai, H., Shalom, M., Tamir, T., Zaks, S. 2010. Minimizing total power-

on time in parallel scheduling with application to optical networks. Theor. Comput. Sci. 411 (40 - 42), 3553-3562.

[10] Heylighen, F. (2015). The science of self-organization and Adaptivity. USA: Encyclopedia of life support. .

[11] Heylighen, F., & Gershenson, C. (2015). The meaning of self-organization in computing. IEE Intelligent Systems.

[12] Khargharia, B., Hariri, S., & Yousif, M. (2015). Autonomic power and performance management for computing systems. USA: Cluster Computing.

[13] Kim, K., Beloglazov, A., Buyya, R., 2011. Power- aware provisioning of virtual machines for real-time Cloud services. Concurrency Comput. Pract. Exp. 23 (13), 1491-1505.

[14] Lefurgy, C., Rajamani, K., Rawson, F., Felter, W., Kistler, M., & Keller, T. (2015). Energy Management for commercial servers. USA: Computer 36 (12).

[15]Mathew, V., Sitaraman, R.K., Shenoy, P., 2012. Energy-aware load balancing in content delivery networks. In: Proceedings of INFOCOM 2012, 25-30 March, PP. 954-962 Olrando, F1.

[16] Pinheiro, E., Bianchini, R., Carrera, E., & Heath, T. (2015). Load Balancing and unbalancing for power and performance in cluster-based systems. In proceedings of the Workshop on Compilers and Operating Systems for Low Power.

[17]White, R., & Abels, T. (2014). Energy Resorce management in the virtual data center. Washington, DC, USA: Proceedings of the International Symposium on Electronics and the Environment.

[18] Jun, C., Yunchuan , Q., Yu Ye, & Zhuo, T. (2015). A Live Migration Algorithm for Virtual Machine in a Cloud Computing Environment. Chine: UIC-ATC-ScalCom-CBDCom-IoP.

[19] Megha, Desai R.; Hiren, Patel B.;. (2015). Efficient Virtual Machine Migration in Cloud Computing. Fifth International Conference on Communication Systems and Networking Technologies.

[20] Mofijul, I. M. (2015). A Genetic Algorithm for Virtual Machine Migration in Heterogeneous Mobile Cloud Computing. Bangladesh.

[21] Pankajdeep , K., & Rani, A. (2015). Virtual Machine Migration in Cloud Computing. Internationale Journal of Grid Distribution Computing Vol.8.

[22] Rabiatul , A., Ruhani, R. A., Norliza, Z., & Mustaffa, S. (2015). Virtual Machine Migration Implementation in Load Balancing for cloud computing. Mara (UiTM).

Authors Profile

Mr. Sebagenzi Jason pursed Bachelor of Science from Adventist university of Central Africa (Rwanda), and Master of Science in Information Technology from Jain University (India) in year 2011. He is currently pursuing Ph.D. in the University of Jain in India since 2016. His main research work focuses on Cloud Management He has 9 years of teaching experience.

Dr.Suchithra R pursed Bachelor of Commerce, Master of Computer Application and Ph.D in Computer Science from Manonmaniam Sundaranar University , India in year 2009. She worked as Associate Professor and Head of MS (IT) Department in Jain university since 2010. She has published more than 20 research papers in reputed international journal.

