An Efficient Energy Saving Task Consolidation Algorithm for Cloud Computing Systems

Sanjaya K. Panda, *IEEE Member*, Prasanta K. Jana, *IEEE Senior Member*
Department of Computer Science and Engineering
Indian School of Mines, Dhanbad, India
sanjayauce@gmail.com, prasantajana@yahoo.co.in

***Abstract*—Task consolidation is a process of maximizing resource utilization in a cloud system. However, maximum usage of resources does not necessarily imply that there will be proper use of energy as some resources which are sitting idle, also consume considerable amount of energy. Recent studies show that energy consumption due to idle resources is approximately 1 to 20%. So, the idle resources are assigned with some tasks to utilize the idle period, which in turn reduces the overall energy consumption of the resources. Note that higher resource utilization merely leads to high energy consumption. So, the tasks are likely to be assigned to all the resources for the proper use of energy. In this paper, we propose an energy saving task consolidation (ESTC) which minimizes the energy consumption by utilizing the idle period of the resources in a cloud environment. ESTC achieves it by assigning few tasks to all available resources to overcome the idleness of the resources. In addition to this, it calculates the energy consumption on arrival of a task to make the scheduling assessment. We perform extensive experiments to measure the performance of ESTC and we compare it with the recent energy-aware task consolidation (ETC) algorithm. The results show that the proposed algorithm outperforms ETC in terms of energy consumption and the total number of task completion.**

Keywords-Task Consolidation*; *Energy Consumption*; *Resource Utilization*; *Data Center*; *Virtual Machine

I. INTRODUCTION

The popularity and interest of ***Cloud Computing*** is exponentially growing due to recent advances in virtualization and software application technologies [1-3]. These technologies enable both consumers and providers to utilize the cloud resource fruitfully and efficiently. The resources are distributed throughout the globe that ranges from a small server (or PCs) to data center. These resources are combined together to solve various science, business and engineering applications. So, numerous attention have been paid by the researchers for efficient mapping of applications into cloud resources [4–11]. The literature aim to maximize the resource utilization. However, efficient resource utilization does not always imply better energy consumption [12].

To fully utilize the cloud resources (especially data center), the cloud service providers are deploying several customer applications without any major concern of energy consumption. Essentially, high performance is achieved in the data center. But the energy consumed in a data center is same as the energy consumed by 25,000 householders. Moreover, the amount of carbon dioxide emits from a data center in 2008, will be quadruple by 2020 [13-14]. So, energy consumption is a crucial issue in ***Cloud Computing***. Therefore several attempts [12], [15] have been made to develop energy efficient algorithms. However, the algorithms have not drawn notable efforts to the energy consumption of idle resources.

We propose here, an energy saving task consolidation (ESTC) algorithm to minimize the energy consumption by utilizing the idle period of the resources in a cloud environment. The algorithm has two steps. In the first step, it utilizes the idle period of the available resources and in the second step, it dispatches tasks to a resource that consumes less power. This step is applicable if and only if the first step fails to assign the tasks. The rationale behind the first step is that when we dispatch tasks to resources in a chronological order or using some technique such as best-fit strategy as used by [12], some of the resources remain idle. However, the unused resources draw significant amount of power. This phenomenon motivates the idea of utilizing the idle period in order to save energy. Note that, there may be a scenario for which the idle period is fully occupied by the arrived tasks. So, tasks are not scheduled as per the above strategy. We use hereafter, resource and virtual machine (VM) interchangeably.

The rest of this paper is organized as follows. Section II discusses the related work with their pros and cons. We present the cloud model and the problem statement in Section III. We describe the proposed algorithm with an illustrative example in Section IV followed by the experimental results in Section V. Finally, we conclude the paper in Section VI.

II. RELATED WORK

In the recent years, many task scheduling algorithms have been designed to solve various issues of ***Cloud Computing***. However, quite a few researchers have studied the problem of energy consumption for ***Cloud Computing***. Lee et al. [15] have proposed two energy conscious task consolidation algorithms, namely ECTC and MaxUtil. However, the algorithms do not consider the idle power draw of available resources. Tian et al. [16] have presented an online real-time scheduling for energy-efficient cloud computing. But, arbitrary-size jobs are not considered for simplicity. Kim et al. [17] have presented a virtual machine scheduling algorithm which provides resource based on the energy budget. However, the estimated energy consumption gives an error rate of less than 5% of the total energy consumption. Srikantaiah et al. [18] have formed task consolidation problem into bin packing problem. Wen et al.

[19] have proposed a hierarchical scheduling algorithm for applications to reduce the energy consumption. However, the application selects the nodes with lower temperature for scheduling assessment. Hsu et al. [12, 20] have proposed an energy-aware task consolidation (ETC) technique to minimize the energy consumption by restricting the CPU use up to certain limits. The algorithm has addressed the idle power drawn by the resources without utilizing the idle period (i.e., 1% to 20%). The algorithm presented in this paper is completely different from that of [12], [15] with respect to following aspects. 1) Our algorithm aims to utilize the idle period of the available resource without compromising the scheduling performance. 2) The algorithm also minimizes the energy consumption in contrast to [12] by choosing the best resource among the available resources. 3) Due to limitation of the resources, the task properties may not be satisfied. Hence, we show the total number of task completion with respect to the tasks submission which is not done by [12], [15]. Experimental results show the superiority of the proposed algorithm over [12] with respect to energy consumption and total number of task completion metrics.

III. Cloud Model and Problem Statement

A. A Motivational Example

Assume a typical example in which approximate power consumption of VM with respect to the utilization is shown in Table I. This is the same table as used in [21]. The utilization value is divided into four levels based on the power consumption in contrast to six levels as addressed by [12]. The utilization values from 1 to 20 are called idle level, from 21 to 50 is level 1, from 51 to 70 is level 2 and 71 to 100 is level 3 [12]. Here, we assume that an idle resource draws 20% utilization. Therefore, the idle resource consumes 78.5 watts as per Table I. Note that the idle resource consumes 78.5 watts even if the resource does not execute a single task. In contrary, a resource with 50% utilization consumes 334.5 watts (i.e., a total of 78.5 + 83 + 85 + 88 watts). Now suppose, the basic information of a set of tasks with their identity (*ID*), arrival time (*AT*), start time (*ST*), period (*P*), finish time (*FT*) and utilization (*U*) is given as shown in Table II. The power consumption of the tasks with respect to utilization is given in Table I, e.g., T_1 has power consumption of 83 watts as it has 25% utilization. There are three different tasks with arrival time 0, 4 and 10 respectively. These tasks are needed 25% utilization for 50 time units, 35% for 40 time units and 30% utilization for 54 time units respectively. If we dispatch task T_1 to an idle VMs, then it consumes 6000 watts (i.e., (78.5 + (8.3 × 5)) × 50 as per Table I) as task T_1 requires an utilization of 25%. A single VM can accommodate these tasks as sum of the utilization value (i.e., 90%) does not exceed 100%. But, the energy consumption is drastically increased. One of the solutions is to start another idle VM that can share some workloads. So, the energy consumption is reduced up to some extent. This is the main motivation behind the proposed work.

B. Cloud Model

Consider a cloud system that consists of several physical machines (or data center). The cloud provider is able to create VMs instantly and deploy these VM in the physical machines. We assume that the VMs are homogeneous in terms of their computational capacity, memory and platform.

TABLE I. Power Consumption of VMs with respect to Utilization [21]

Level	Utilization	Power Consumption (in Watts (approx.)) (p_i)
Idle	1~20	p_{20} = 78.5
1	21~30	p_{30} = 83
	31~40	p_{40} = 85
	41~50	p_{50} = 88
2	51~60	p_{60} = 93
	61~70	p_{70} = 102
3	71~80	p_{80} = 109
	81~90	p_{90} = 122
	91~100	p_{100} =136

TABLE II. A Set of Tasks with Their 6-Tuple

ID	*AT*	*ST*	*P*	*FT*	*U*
T_1	0	2	50	52	25%
T_2	4	7	40	47	35%
T_3	10	15	54	69	30%

A VM may have following states: execute, sleep/awake and idle. However, we use only two states i.e., execute and idle for simplicity. When a VM is computing tasks then that VM is present in the execute state. In contrary, when a VM is not computing any task or waiting for a task to arrive, the VM will be in the idle state. On the other hand, when a VM receives a task, it makes a transition from idle-to-execute state. It returns back to idle state by making execute-to-idle state transition, once the task is successfully executed.

C. Problem Statement

Consider a set of *m* virtual machines $V = \{V_1, V_2, V_3,\ldots, V_m\}$ and a set of *n* independent tasks $T = \{T_1, T_2, T_3,\ldots, T_n\}$ in which each task T_i is a 6-tuple {*ID*, *AT*, *ST*, *P*, *FT*, *U*} where *ID* denotes the unique identification number, *AT* is the arrival time, *ST* is the start time, *P* is the period, *FT* is the finish time and *U* is the utilization. The finish time of a task T_i is the sum of *ST* and *P* and the utilization is the maximum percentage of time, a task can make the CPU busy. The problem is to map the tasks and the VMs such that the energy consumption is minimized. However, the mapping has some restrictions as follows. 1) The order of tasks execution is determined as per the *AT* of tasks. Alternatively, the task with earliest *AT* is scheduled before the task with latest *AT*. Unlike task scheduling, the sequence of task execution is predetermined. 2) A task is not migrated and split at any cost. 3) A task T_i is not assigned to VM V_j if the utilization of a task for the requested period is not satisfied by that VM V_j.

IV. Proposed Algorithm

We now present our proposed energy saving task consolidation (ESTC) algorithm as follows. As stated earlier that it is a two-step procedure. In the first step, it aims to minimize energy by properly utilizing the idle period of the VMs. The energy consumption of an idle VM *i* (denoted as $E(IVM_i)$) is mathematically defined as follows

$$E(IVM_i) = p_{20} \times \Delta\upsilon \quad (1)$$

where $\Delta\upsilon = \upsilon_{max} - \upsilon_{min}$, p_{20} is the power consumption at the resource utilization rate of [1% to 20%], υ_{min} and υ_{max} are the start-time and end-time of a resource respectively. For example, an idle resource starts at 10 time unit and ends at 20 time unit. Then the energy consumption is 78.5 × 10 = 785

watts. In second step, if the idle period is occupied by the already arrived task, then the task is dispatched to a VM that consumes less power. Note that, this step is not applicable if it is possible to assign the task to an idle resource. The energy consumption of a non-idle virtual machine i for x% utilization (denoted as $(E(NVM_i(x)))$ is shown in Equation 2. For example, a non-idle VM 2 starts at 10 time unit and ends at 20 time unit for 55% resource utilization. Then the energy consumption is calculated as follows.

$E(NVM_2(55))=$

$$(p_{20}\times\Delta\upsilon)+(p_{30}\times\Delta\upsilon)+(p_{40}\times\Delta\upsilon)+(p_{50}\times\Delta\upsilon)+(\frac{p_{60}}{10}\times(x\%50)\times\Delta\upsilon)$$

$= (78.5 \times 10) + (83 \times 10) + (85 \times 10) + (88 \times 10) +$
$(9.3 \times 5 \times 10) = 3810$ watts.

We use the following terminologies for the proposed method. The pseudo code of the algorithm is shown in Figure 1.

Notation	Description
Q	Queue of all the tasks
T_i	i^{th} task
V_j	j^{th} virtual machine
$U_k(V_j)$	Uthilization of j^{th} virtual machine at k time unit
$U_k(T_i)$	Uthilization of i^{th} task at k time unit
τ_1	A threshold value (i.e. idle period consumption)
$EE(V_j)$	Estimated energy of j^{th} virtual machine
$EU(V_j)$	Estimated utlization of j^{th} virtual machine

A. An Illustration

Let us consider the set of tasks of Table II (refer Section III). The task-VM mapping according to ETC algorithm is shown in Figure 2. The energy consumption for ETC algorithm is calculated as follows.

$$E=\sum_{i=1}^{m}E\left(IVM_i\right)\times f+E\left(NVM_i\left(x\right)\right)$$

where $f=\begin{cases}1 & \text{if } f \text{ is idle}\\ 0 & \text{Otherwise}\end{cases}$

Therefore, $E = (78.5 \times 69 + (8.3 \times 5) \times 50 + ((8.3 \times 5) + 85 + 88 + 93) \times 40) + (78.5 \times 69 + 83 \times 54) + (78.5 \times 69) =$ 35106.5 watts.

The task-VM mapping for ESTC algorithm is shown in Figure 3. The energy consumption for ESTC algorithm is calculated as follows.

$E = (78.5 \times 69 + (8.3 \times 5) \times 50) + (78.5 \times 69 + (83 + (8.5 \times 5)) \times 40) + 78.5 \times 69 + 83 \times 54 = 27826.5$ watts.

The gray color indicates the idle period of VMs. The illustration clearly shows the superiority of the proposed algorithm over the recent ETC algorithm.

B. Time Complexity

Let the total number of tasks be l and the total number of VMs be m. First of all, the ESTC picks the first task T_1 and call SCHEDULE-BY-IDLE-PERIOD-FILLING (T_1, ST, FT) (Procedure 1). To find whether an idle period is available or not, the Procedure 1 takes $O(m(f–s))$ time (in worst case for m VMs) where f and s are the finish time and start time respectively (Lines 1 to 7 of Procedure 1). To assign the task to a VM, it takes $O(f - s)$ time (Lines 8 to 14). Note that, we assume the threshold value (i.e., τ_1) is 20. If the task properties are not satisfied by Procedure 1 or the idle period is occupied by already arrived task, then the algorithm calls SCHEDULE–BY–ENERGY–SAVING (T_1, ST, FT) (Procedure 2). In order to find whether the task is accommodated or not as well as the estimated energy consumption the Procedure 2 takes $O(m(f–s))$ time (in worst case for m VMs) (Lines 1 to 13). To choose a VM which consumes less energy consumption takes $O(m)$ time (Lines 15 to 19). To assign the task T_1 to VM V_j, it takes $O(f–s)$ time. So, the overall time complexity of Procedure 1 and Procedure 2 is $O(m(f–s))$. If task T_1 is not schedulable by both procedures then it rejects T_1. However, the overall time complexity of ECTC is $O(lm(f–s))$ time as ECTC invoke Procedure 1 and Procedure 2 l times.

V. Experimental Results

We evaluate the proposed algorithm through simulation run with some randomly generated datasets. The experiments were carried out using MATLAB R2012a version 7.14.0.739 on an Intel Core 2 Duo processor, 2.20 GHz CPU and 4 GB RAM running on Microsoft Windows 7 platform.

A. Datasets

In this simulation, we considered five different datasets generated using MATLAB random function. Each dataset has five instances. The first dataset contains 100 tasks to be scheduled to the available virtual machines and we denote it by 100_*ix_yy*. Here, *ix* denotes the instance number (i.e., instance 1 (*i*1), instance 2 (*i*2) and so on) and *yy* shows the number of virtual machine that is 10, 15 or 20 as used by [12]. Therefore, this structure makes 15 different instances (100_*i*1_10, 100_*i*1_15, 100_*i*1_20, 100_*i*2_10, 100_*i*2_15, 100_*i*2_20, 100_*i*3_10, 100_*i*3_15, 100_*i*3_20, 100_*i*4_10, 100_*i*4_15, 100_*i*4_20, 100_*i*5_10, 100_*i*5_15 and 100_*i*5_20). Similarly, the second dataset contains 200 tasks to be scheduled to the available virtual machines and we denote it by 200_*ix_yy*. Accordingly, we denote the third, fourth and fifth dataset as 300_*ix_yy*, 400_*ix_yy* and 500_*ix_yy* respectively. Therefore, five different dataset makes 75 different instances. For the sake of clarity, the range of experimental parameter is shown in Table III. We denote the range by [a~b] which indicate the experimental parameter lies between a and b (both inclusive).

TABLE III. Range of Experimental Parameter

Parameter	100_*ix_yy*	200_*ix_yy*	300_*ix_yy*	400_*ix_yy*	500_*ix_yy*
Arrival time	[1~200]	[1~400]	[1~600]	[1~800]	[1~1000]
Start time	[1~200]	[1~400]	[1~600]	[1~800]	[1~1000]
Finish time	[100~500]	[200~1000]	[300~1500]	[400~2000]	[500~2500]
Utilization	[1~100]	[1~100]	[1~100]	[1~100]	[1~100]

B. Experiments

In these experiments, we claim that few tasks are incomplete as no data center contains unlimited resources [2]. We consider the task incomplete (TI) in both ETC and ESTC algorithm and rename the algorithms as TI-ETC and TI-ESTC respectively. We first experiment 100_*ix_yy* instances. The energy consumption and the number of task incomplete of ETC algorithm with the proposed ESTC are jointly shown in Table IV. The comparison results clearly show that 75 out of 75 instances (i.e., 100%) give better results for the proposed algorithm than the ETC with respect to following cases. 1) Both energy consumption and incomplete tasks are minimized

(for example, 100_*i*1_10, 100_*i*1_15, 100_*i*1_15 and so on). 2) Energy consumption is minimized without any variation in incomplete task (for example, 100_*i*2_15, 100_*i*4_10 and so on). 3) Incomplete task is minimized with a little variation in energy consumption (for example, 100_*i*4_20, 100_*i*5_10, 100_*i*5_20 and so on).

$$E\left(NVM_i(x)\right) = \begin{cases} (p_{20}\times\Delta u)+(\frac{p_{30}}{10}\times(x\%20)\times\Delta u) & \text{if } 21\% \le x \le 30\% \\ (p_{20}\times\Delta u)+(p_{30}\times\Delta u)+(\frac{p_{40}}{10}\times(x\%30)\times\Delta u) & \text{if } 31\% \le x \le 40\% \\ (p_{20}\times\Delta u)+(p_{30}\times\Delta u)+(p_{40}\times\Delta u)+(\frac{p_{50}}{10}\times(x\%40)\times\Delta u) & \text{if } 41\% \le x \le 50\% \\ (p_{20}\times\Delta u)+(p_{30}\times\Delta u)+(p_{40}\times\Delta u)+(p_{50}\times\Delta u)+(\frac{p_{60}}{10}\times(x\%50)\times\Delta u) & \text{if } 51\% \le x \le 60\% \\ (p_{20}\times\Delta u)+(p_{30}\times\Delta u)+(p_{40}\times\Delta u)+(p_{50}\times\Delta u)+(p_{60}\times\Delta u)+(\frac{p_{70}}{10}\times(x\%60)\times\Delta u) & \text{if } 61\% \le x \le 70\% \\ (p_{20}\times\Delta u)+(p_{30}\times\Delta u)+(p_{40}\times\Delta u)+(p_{50}\times\Delta u)+(p_{60}\times\Delta u)+(p_{70}\times\Delta u)+(\frac{p_{80}}{10}\times(x\%70)\times\Delta u) & \text{if } 71\% \le x \le 80\% \\ (p_{20}\times\Delta u)+(p_{30}\times\Delta u)+(p_{40}\times\Delta u)+(p_{50}\times\Delta u)+(p_{60}\times\Delta u)+(p_{70}\times\Delta u)+(p_{80}\times\Delta u)+(\frac{p_{90}}{10}\times(x\%80)\times\Delta u) & \text{if } 81\% \le x \le 90\% \\ (p_{20}\times\Delta u)+(p_{30}\times\Delta u)+(p_{40}\times\Delta u)+(p_{50}\times\Delta u)+(p_{60}\times\Delta u)+(p_{70}\times\Delta u)+(p_{80}\times\Delta u)+(p_{90}\times\Delta u)+(\frac{p_{100}}{10}\times(x\%90)\times\Delta u) & \text{if } 91\% \le x \le 100\% \end{cases} \quad (2)$$

Algorithm 1: Energy Saving Task Consolidation

```
1. while Q ≠ NULL do
2.     Call SCHEDULE–BY–IDLE–PERIOD–FILLING (T_i, ST, FT)
3.     if (task properties are not satisfied as above)
4.         Call SCHEDULE–BY–ENERGY–SAVING (T_i, ST, FT)
5.     else
6.         reject task T_i
7.     endif
8. endwhile
```

Procedure 1: SCHEDULE–BY–IDLE–PERIOD–FILLING (T_i, *ST*, *FT*)

```
1. for V_j ∈ V
2.     Set Count = 0
3.     for k = ST to FT
4.         if U_k(V_j) + U_k(T_i) < τ_1
5.             Count = Count + 1
6.         endif
7.     endfor
8.     if Count == P_i
9.         Assign task T_i to VM V_j
10.        for k = ST to FT
11.            U_k(V_j) = U_k(V_j) + U_k(T_i)
12.        endfor
13.        Return
14.    endif
15. endfor
```

The rationale behind the improvement is that ESTC assigns the tasks as per the idle period followed by minimum energy consumption. Thus, the energy consumption is reduced in greater extent.

Procedure 2: SCHEDULE–BY–ENERGY–SAVING (T_i, *ST*, *FT*)

```
1. for V_j ∈ V
2.     Set EE(V_j) = 0
3.     for k = ST to FT
4.         if U_k(V_j) + U_k(T_i) > 100
5.             EE(V_j) = -1 and Break
6.         else
7.             x = U_k(V_j) + U_k(T_i)
8.             Calculate E(NVM_j(x))
9.             EE(V_j) = EE(V_j) + E(NVM_j(x))
10.        endif
11.    endfor
12.    EE(V_j) = EE(V_j) / P_i
13. endfor
14. Set temp = EE(V_1)
15. for V_j ∈ V
16.    if temp > EE(V_j)
17.        temp = EE(V_j)
18.    end if
19. endfor
20. Assign task T_i to VM V_j that holds temp
21. for k = ST to FT
22.     U_k(V_j) = U_k(V_j) + U_k(T_i)
23. endfor
```

Figure 1. Pseudo code for ESTC algorithm

Subsequently, we experiment 200_*ix_yy* instances. Energy consumption and total number of incomplete task of the proposed algorithm ESTC are compared with that of ETC and shown in Table V. Similarly, the comparison of 300_*ix_yy*, 400_*ix_yy* and 500_*ix_yy* instances is shown in Table VI, Table VII and Table VIII respectively. This can be noted that the proposed ESTC improves 10% energy consumption over the recent ETC task consolidation algorithm.

1~10
11~20
21~30
31~40
41~50
51~60
61~70

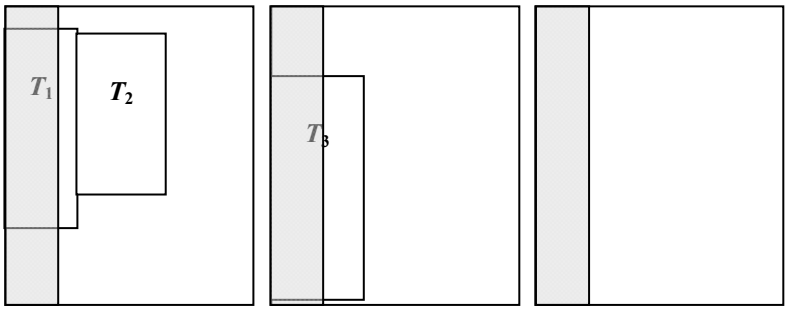

Figure 2. Task-VM Mapping for ETC algorithm

1~10
11~20
21~30
31~40
41~50
51~60
61~70

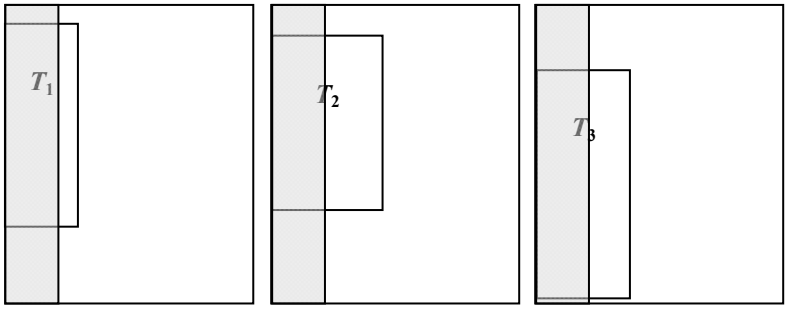

Figure 3. Task-VM Mapping for ESTC algorithm

TABLE IV. COMPARISON OF ENERGY CONSUMPTION FOR ETC AND ESTC ALGORITHM USING 100_IX_YY DATASET

	Algorithm	**10**	**15**	**20**	**Algorithm**	**10**	**15**	**20**
100_*i*1_yy	ETC	1.9419e+006	3.1670e+006	3.7033e+006	TI-ETC	46	35	30
	ESTC	1.9252e+006	2.9642e+006	3.2931e+006	TI-ESTC	45	34	29
100_*i*2_yy	ETC	2.2723e+006	3.5669e+006	4.1244e+006	TI-ETC	52	40	34
	ESTC	2.2991e+006	3.3189e+006	3.8848e+006	TI-ESTC	51	40	33
100_*i*3_yy	ETC	2.5041e+006	3.6461e+006	4.7132e+006	TI-ETC	51	38	29
	ESTC	2.2151e+006	3.5766e+006	4.5743e+006	TI-ESTC	49	37	27
100_*i*4_yy	ETC	2.2167e+006	2.9254e+006	3.7395e+006	TI-ETC	52	45	38
	ESTC	2.1413e+006	2.7586e+006	3.8954e+006	TI-ESTC	52	44	35
100_*i*5_yy	ETC	2.4556e+006	3.5355e+006	4.1115e+006	TI-ETC	54	40	30
	ESTC	2.4690e+006	3.5168e+006	4.4948e+006	TI-ESTC	53	39	27

TABLE V. COMPARISON OF ENERGY CONSUMPTION FOR ETC AND ESTC ALGORITHM USING 200_IX_YY DATASET

	Algorithm	**10**	**15**	**20**	**Algorithm**	**10**	**15**	**20**
200_*i*1_yy	ETC	5.2635e+006	8.0105e+006	9.7707e+006	TI-ETC	133	112	94
	ESTC	4.9906e+006	7.2500e+006	9.4878e+006	TI-ESTC	130	110	92
200_*i*2_yy	ETC	5.5409e+006	7.7255e+006	1.0179e+007	TI-ETC	134	112	99
	ESTC	5.2677e+006	7.3099e+006	9.7130e+006	TI-ESTC	130	112	97
200_*i*3_yy	ETC	5.8739e+006	7.8620e+006	9.7827e+006	TI-ETC	135	114	98
	ESTC	5.2499e+006	7.6839e+006	1.0246e+007	TI-ESTC	134	113	97
200_*i*4_yy	ETC	5.8334e+006	8.2769e+006	9.9849e+006	TI-ETC	139	118	104
	ESTC	5.6315e+006	7.9368e+006	9.6357e+006	TI-ESTC	138	118	103
200_*i*5_yy	ETC	5.0232e+006	6.7355e+006	9.4177e+006	TI-ETC	137	113	96
	ESTC	4.6718e+006	6.9156e+006	8.9989e+006	TI-ESTC	136	111	96

TABLE VI. COMPARISON OF ENERGY CONSUMPTION FOR ETC AND ESTC ALGORITHM USING 300_IX_YY DATASET

	Algorithm	**10**	**15**	**20**	**Algorithm**	**10**	**15**	**20**
300_*i*1_yy	ETC	8.4118e+006	1.2574e+007	1.5303e+007	TI-ETC	226	202	186
	ESTC	7.9396e+006	1.2192e+007	1.5766e+007	TI-ESTC	220	200	179
300_*i*2_yy	ETC	8.6432e+006	1.2571e+007	1.7043e+007	TI-ETC	230	208	187
	ESTC	7.9564e+006	1.2274e+007	1.6609e+007	TI-ESTC	229	202	183
300_*i*3_yy	ETC	8.4450e+006	1.2356e+007	1.6313e+007	TI-ETC	233	205	179
	ESTC	8.1523e+006	1.2262e+007	1.5113e+007	TI-ESTC	226	201	179
300_*i*4_yy	ETC	8.0138e+006	1.1808e+007	1.5923e+007	TI-ETC	220	196	173
	ESTC	7.6379e+006	1.2037e+007	1.4913e+007	TI-ESTC	215	188	168
300_*i*5_yy	ETC	8.5490e+006	1.2129e+007	1.6351e+007	TI-ETC	227	210	187
	ESTC	8.2090e+006	1.1858e+007	1.5945e+007	TI-ESTC	224	204	182

TABLE VII. COMPARISON OF ENERGY CONSUMPTION FOR ETC AND ESTC ALGORITHM USING 400_IX_YY DATASET

	Algorithm	10	15	20	Algorithm	10	15	20
400_*i*1_yy	ETC	1.1112e+007	1.6494e+007	2.1960e+007	TI-ETC	317	285	266
	ESTC	1.0746e+007	1.6323e+007	2.1600e+007	TI-ESTC	314	281	260
400_*i*2_yy	ETC	1.1807e+007	1.8108e+007	2.3679e+007	TI-ETC	329	303	278
	ESTC	1.1426e+007	1.7048e+007	2.2588e+007	TI-ESTC	323	300	274
400_*i*3_yy	ETC	1.1429e+007	1.7278e+007	2.3136e+007	TI-ETC	316	285	264
	ESTC	1.1105e+007	1.6691e+007	2.2167e+007	TI-ESTC	312	282	257
400_*i*4_yy	ETC	1.1745e+007	1.7508e+007	2.2585e+007	TI-ETC	314	285	259
	ESTC	1.1608e+007	1.6994e+007	2.1872e+007	TI-ESTC	308	284	258
400_*i*5_yy	ETC	1.1314e+007	1.6977e+007	2.2130e+007	TI-ETC	318	293	261
	ESTC	1.0979e+007	1.5988e+007	2.2236e+007	TI-ESTC	316	282	259

TABLE VIII. COMPARISON OF ENERGY CONSUMPTION FOR ETC AND ESTC ALGORITHM USING 500_IX_YY DATASET

	Algorithm	10	15	20	Algorithm	10	15	20
500_*i*1_yy	ETC	1.4683e+007	2.1986e+007	2.9023e+007	TI-ETC	416	394	362
	ESTC	1.4224e+007	2.0812e+007	2.6857e+007	TI-ESTC	414	381	357
500_*i*2_yy	ETC	1.3442e+007	2.0667e+007	2.7323e+007	TI-ETC	413	385	360
	ESTC	1.3259e+007	2.0018e+007	2.6296e+007	TI-ESTC	411	379	350
500_*i*3_yy	ETC	1.3703e+007	2.0566e+007	2.7434e+007	TI-ETC	417	387	359
	ESTC	1.3449e+007	1.9775e+007	2.6693e+007	TI-ESTC	414	375	350
500_*i*4_yy	ETC	1.4962e+007	2.2384e+007	2.9495e+007	TI-ETC	397	360	333
	ESTC	1.4471e+007	2.1260e+007	2.7603e+007	TI-ESTC	385	359	328
500_*i*5_yy	ETC	1.3800e+007	2.0997e+007	2.7926e+007	TI-ETC	418	384	353
	ESTC	1.3512e+007	1.9942e+007	2.5958e+007	TI-ESTC	411	376	350

VI. CONCLUSION

We have presented a task consolidation algorithm for cloud computing systems. The algorithm has been shown to require $O(lm(f–s))$ time for l iterations. It was experimented extensively on several data sets. However, due to space limitation we have shown the experimental results only for few data sets. The experimental results show that the proposed ESTC reduces 10% energy consumption over the most recent task consolidation algorithm ETC in the experimented dataset. The comparison results show that the proposed algorithm outperforms ETC in terms of two performance metrics namely, energy consumption and the total number of task completion.

REFERENCES

[1] Chandrasekaran, A. and Kapoor, M. 2011. State of Cloud Computing in the Public Sector – A Strategic Analysis of the Business Case and Overview of Initiatives Across Asia Pacific. *Frost & Sullivan*, 1-17.

[2] Buyya, R., Yeo, C. S., Venugopal, S., Broberg J. and Brandic I. 2009. Cloud Computing and Emerging IT Platforms: Vision, Hype and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, *Elsevier*. 25, 599-616.

[3] Durao, F., Carvalho, J. F. S., Fonseka, A. and Garcia, V. C. 2014. A Systematic Review on Cloud Computing. *The Journal of Supercomputing*. 68, 3, 1321-1346.

[4] Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X. and Gu, Z. 2012. Online Optimization for Scheduling Preemptable Tasks on IaaS Cloud System. *Journal of Parallel Distributed Computing*, *Elsevier*. 72, 666-677.

[5] Tsai, J. , Fang, J. and Chou, J. 2013. Optimized Task Scheduling and Resource Allocation on Cloud Computing Environment using Improved Differential Evolution Algorithm. *Computers & Operations Research*, *Elsevier*. 40, 12, 3045-3055.

[6] Panda, S. K. and Jana, P. K. 2014. A Multi-Objective Task Scheduling Algorithm for Heterogeneous Multi-cloud Environment, *IEEE International Conference on Electronic Design, Computer Networks and Automated Verification*.

[7] Nathani, A., Chaudhary, S. and Somani, G. 2012. Policy Based Resource Allocation in IaaS Cloud. *Future Generation Computer Systems*, *Elsevier*. 28, 94-103.

[8] Huang, C., Guan, C., Chen, H., Wang, Y., Chang, S., Li, C. and Weng, C. 2013. An Adaptive Resource Management Scheme in Cloud Computing. *Engineering Applications of Artificial Intelligence*, *Elsevier*. 26, 382-389.

[9] Ming, G. and Li, H. 2012. An Improved Algorithm Based on Max-Min for Cloud Task Scheduling. *Recent Advances in Computer Science and Information Engineering*, *Lecture Notes in Electrical Engineering*, *Springer*. 125, 217-223.

[10] Chen, H., Wang, F., Helian, N. and Akanmu, G. 2013. User-Priority Guided Min-Min Scheduling Algorithm for Load Balancing in Cloud Computing. *National Conference on Parallel Computing Technologies*, 1-8.

[11] Panda, S. K. and Jana, P. K. 2014. An Efficient Task Scheduling Algorithm for Heterogeneous Multi-Cloud Environment. 3rd *IEEE International Conference on Advances in Computing, Communications & Informatics*, pp. 1204-1209.

[12] Hsu, C., Slagter, K. D., Chen, S. and Chung, Y. Optimizing Energy Consumption with Task Consolidation in Clouds. 2014. *Information Sciences*, *Elsevier*. 258, 452-462.

[13] Kaplan, J. M., Forrest W. and Kindler, N. 2008. Revolutionizing Data Center Energy Efficiency. *McKinsey & Company*.

[14] Beloglazov, A., Abawajy, J. and Buyya, R. 2012. Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing. *Future Generation Computer Systems*, *Elsevier*. 28, 755-768.

[15] Lee, Y. C. and Zomaya, A. Y. 2012. Energy Efficient Utilization of Resources in Cloud Computing Systems. *Journal of Supercomputing*, *Springer*. 60, 268-280.

[16] Tian, W., Xiong, Q. and Cao, J. 2013. An Online Parallel Scheduling Method with Application to Energy-Efficiency in Cloud Computing. *Journal of Supercomputing*, *Springer*. 66, 1773-1790.

[17] Kim, N., Cho, J. and Seo, E. 2014. Energy-credit Scheduler: An Energy-Aware Virtual Machine Scheduler for Cloud Systems. *Future Generation Computer Systems*, *Elsevier*. 32, 126-137.

[18] Srikantaiah, S., Kansal, A. and Zhao F. 2008. Energy Aware Consolidation for Cloud Computing. *International Conference on Power Aware Computing and Systems*.

[19] Wen, G., Hong, J., Xu, C., Balaji, P., Feng, S. and Jiang, P. 2011. Energy-aware Hierarchical Scheduling of Applications in Large Scale Data Centers. *International Conference on Cloud and Service Computing*.

[20] Hsu, C., Chen, S., Lee, C., Chang, H., Lai, K., Li, K. and Rong, C. 2011. Energy-Aware Task Consolidation Technique for Cloud Computing. *Third IEEE International Conference on Cloud Computing Technology and Science*.

[21] Lien, C., Liu, M. F., Bai, Y., Lin, C. H. and Lin, M. 2006. Measurement by the Software Design for the Power Consumption of Streaming Media Servers. *Instrumentation and Measurement Technology Conference*, *IEEE*. 1597-1602.