Multimedia Tools and Applications (2023) 82:21385–21413
https://doi.org/10.1007/s11042-022-14009-1

Task consolidation based power consumption minimization in cloud computing environment

Shaimaa Badr[1] · Ahmed El Mahalawy[2] · Gamal Attiya[2] · Aida A. Nasr[3]

Received: 30 November 2021 / Revised: 27 June 2022 / Accepted: 23 September 2022 /
Published online: 26 October 2022

Abstract

Cloud Computing is playing a huge role in future technology. Further, with the explosive growth of the Internet and cloud computing, several service providers, such as Amazon, Microsoft, IBM, and Google, have expanded their data centers and rapidly deployed data centers in different places around the world to deliver various cloud computing services. However, several challenges are raised with the wide spread use of cloud environment such as power consumption, load balance, reliability, scalability, and security. This paper tackles the power consumption problem and presents an efficient algorithm, called Task Consolidation based Power Minimization (TCPM), to efficiently schedule tasks onto available resources of the cloud environment so as to minimize power consumption. In proposed TCPM algorithm, several benefits of the existing algorithms are enhanced and incorporated into the TCPM algorithm, where the best-fit procedure is used to achieve the best possible resource utilization and avoid wasting energy. The results of the proposed TCPM algorithm are compared with other recent algorithms such as FCFS, WWO, and MCT algorithms using the CloudSim toolkit.

✉ Shaimaa Badr
Princess.basant@gmail.com

Ahmed El Mahalawy
Ahmed.elmahalawy@el-eng.menofia.edu.eg

Gamal Attiya
Gamal.mahrous@el-eng.menofia.edu.eg

Aida A. Nasr
Dr.aida_nasr@ics.tanta.edu.eg

[1] SCADA Engineer, Middle Egypt Regional Control Center, Egyptian Electricity Transmission Company, El-Minia, Egypt

[2] Computer Science and Engineering Dept., Faculty of Electronic Engineering, Menoufia University, Shibin el Kom, Egypt

[3] Information Technology Dept., Faculty of Computers and Informatics, Tanta University, Tanta, Egypt

Keywords Cloud computing · Power consumption · Scheduling · Task consolidation · Data centers · And CloudSim

1 Introduction

Cloud computing is a type of distributed system that relies on virtualized resources to manage applications. Such system is viewed as a form of Internet-based computing in which different services such as data centers, storage, and applications are handed via the Internet [27].The numerous components in terms of databases, software capabilities, applications, and so on that are created to use the power of cloud resources to handle business challenges are referred to as cloud architecture. The entire cloud architecture is designed to provide users with high bandwidth, allowing them to access data and applications without interruption, an on-demand dynamic network with the ability to move rapidly and effectively between servers or even platforms, and, most significantly, network security.

Different cloud-based services have their specific cloud architectures [25]:

- Software as a Service (SaaS) refers to software that is hosted and updated through the internet. Users do not need to install applications locally when using SaaS.
- Platform as a Service (PaaS) is a type of middleware service that provides users with application platforms and databases.
- Infrastructure as a Service (IaaS) makes cloud-based infrastructure and hardware including servers, networks, and storage devices available to users on a pay-per-use basis.

With the explosive growth of the internet, several network operators (such as Amazon, Microsoft, IBM, and Google) have expanded their network infrastructure and greater power data centers in different places around the world to produce cloud computing services. However, many issues concerning cloud computing need to be handled, including power consumption, scalability, reliability, cost, and security. This paper tackles the power consumption problem.

Cloud vendors are using a considerable number of servers in their network infrastructure. These in turn make data centers to consume an excessive amount of energy and increase carbon footprints inside the environment [2]. As to a study published in [13], if current technical and management growth continues, data center power consumption will explode, resulting in a massive increase in energy consumption. As a consequence, the focus of the present cloud research is on how to minimize energy consumption and carbon emissions through resource management and usage.

This paper presents an efficient algorithm; called Task Consolidation based Power Minimization (TCPM), to efficiently schedule tasks onto available resources of the cloud environment and apply task consolidation strategy to minimize power consumption in data centers. In proposed TCPM algorithm, several benefits of the existing algorithms are enhanced and incorporated into the TCPM algorithm, where the best-fit procedure is used to achieve the best possible resource utilization and avoid wasting energy. The TCPM algorithm is evaluated considering different condition and results are compared with the most recent algorithms such as FCFS, WWO, and MCT algorithms using the CloudSim toolkit.

The main contributions of our suggested TCPM algorithm, which combines various advantages of earlier algorithms, like the following: firstly, Use the best-fit strategy to maximize resource efficiency while reducing energy waste. Secondary combines Task Consolidation and Task Scheduling, both of which are incorporated into the TCPM algorithm to reduce power consumption, then schedule Tasks into virtual machines which Idle VM Empty. Subsequently, it works to achieve the best use of power consumption which improves reliability, throughput, and power consumption while lowering the failure rate.

The remainder of this paper is organized as follows. Section 2 presents power consumption scenarios and energy models in a cloud environment. Section 3 introduces the idea of task scheduling and task consolidation in cloud environment. Section 4 introduces a series of state-of-the-art of algorithms for minimizing energy consumption. Section 5 presents the proposed TCPM algorithm in details while Section 6 describes the proposed TCPM algorithm by illustration example. Section 7 presents the experimental results. Finally, the concluding remarks are listed in section 8.

2 Power consumption and energy models

Figure 1 shows the data center energy consumption scenarios 2016:2030 [13]. Data center electricity expectations will expand from 286 TWh in 2016 to around 321 TWh in 2030. The authors in [13] illustrate a forecasting model of datacenter energy

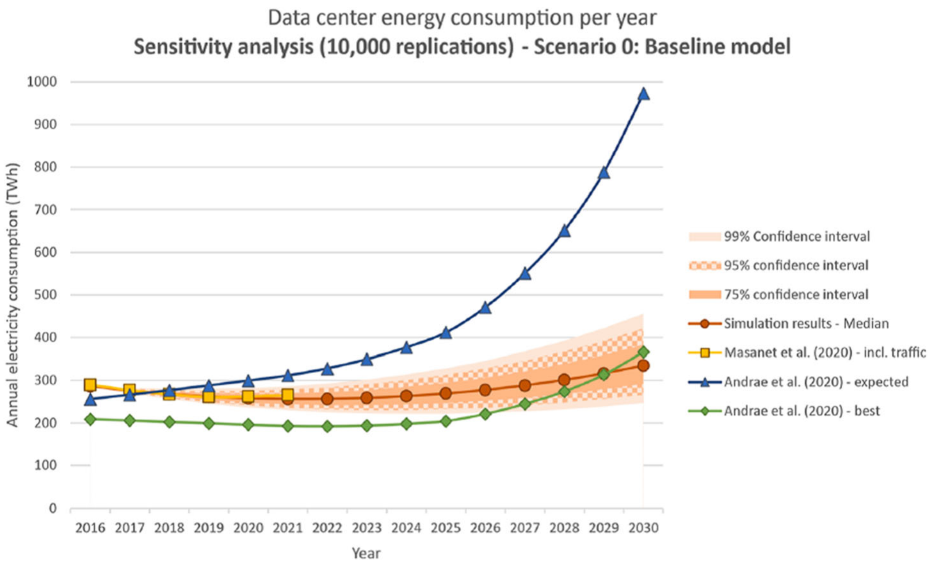


Fig. 1 Data center energy consumption scenario 2016:2030 [13]

requirements explaining and knowledge consumption growth and imply that this rapid expansion isn't fully compensated by cost savings of data center technological advancements.

To handle power consumption, two forms of energy models are applied in algorithms to enhance resource utilization simultaneously with minimizing power consumption. The energy models may be used to build mathematical models of resource utilization.

A. **Energy Model 1**

The first energy model computes resource utilization using the following formula [7, 11, 14]:

$$\boldsymbol{Ui}=\textstyle\sum_{j=1}^{n}\boldsymbol{u_{i,j}}$$

The total utilization of a virtual machine***i*** is the sum of utilization of all tasks scheduled onto that virtual machine.

The energy consumption may be calculated using the following formula:

$$\boldsymbol{Ei}=(\boldsymbol{Pmax}\ -\boldsymbol{Pmin})\times\boldsymbol{Ui}+\boldsymbol{Pmin}$$

Where, $\boldsymbol{P_{max}}$ is the power consumption at the peak load (or 100% utilization) and $\boldsymbol{P_{min}}$ is the minimum power consumption of the active mode.

B. **Energy Model 2**

The second energy model estimates the energy using the amount of energy consumed by Idle and Non-Idle Virtual machines [6, 21, 22], as follows.

$$\boldsymbol{E}=\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}}\boldsymbol{E}(\boldsymbol{IVM_i})\times\boldsymbol{f}+\boldsymbol{E}(\boldsymbol{NVM_i}(\boldsymbol{U}))$$

$$Where,\ f=\begin{cases}1, & if\ \ idle\\ 0, & otherwise\end{cases}$$

Where, U is the virtual machine utilization, $E(IVM_i)$ is the energy consumption that consumed by idle virtual machine resources, and $E(NVM_i(U))$ is the energy consumption that consumed by non-idle virtual machine resources.

Idle Virtual Machine Resources: $E(IVM_i)$ = P20 × Δv and $\Delta\boldsymbol{v}=\boldsymbol{vmax}-\boldsymbol{vmin}$

Where, *vmin* is start time and *vmax* is finish time.

Non-Idle Virtual Machine Resources: $E(NVM_i(U))$ may be computed as shown in Fig. 2**.**

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

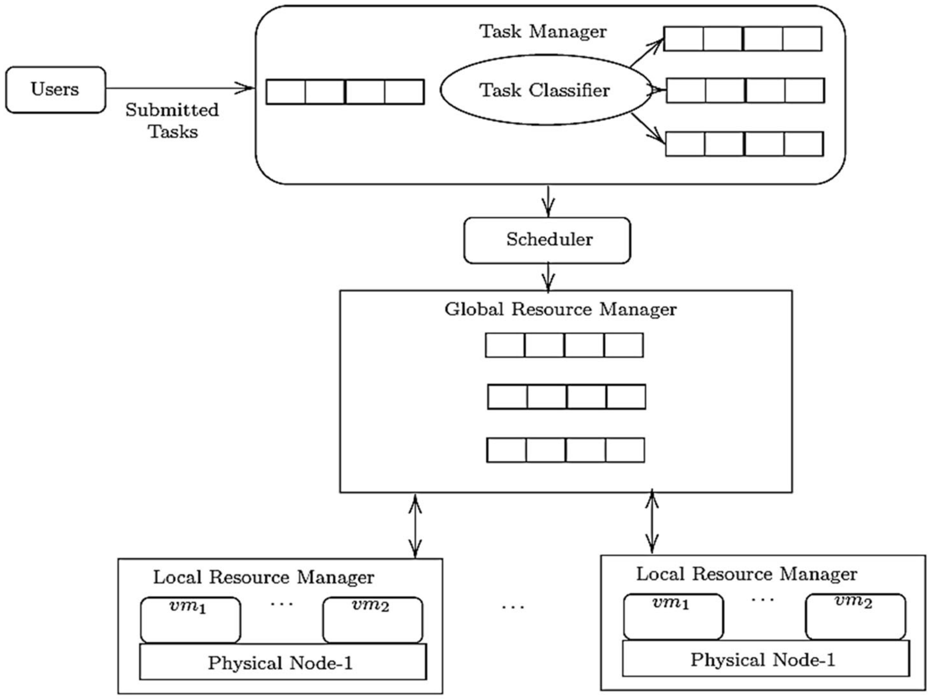
Fig. 2 Energy consumption of non-idle VM resources

3 Scheduling and task consolidation

Scheduling is the process of distributing a set T of n tasks to a set R of m cloud computing resources (virtual machines) [18]. There are two levels of scheduling in the cloud data centers: a series of rules for deploying VMs in a server at the *server level* and a series of rules for assigning tasks to VMs [12] at the *VM level*, as shown in Fig. 3. This study focuses on task scheduling algorithms at the VM level. The scheduling methodology is the strategy for identifying resources to implement tasks to reduce waiting and execution time [23]. Task consolidation is an efficient strategy for reducing power consumption. It employs virtualization technologies to consolidate tasks. The goal of the task consolidation technique is to maximize the utility of available resources in data centers. Simultaneously, eliminating resource dependency that is the main reason behind wasting a considerable amount of energy is inefficient resource utilization and overload of Communication [24].

4 Related work

Power consumption minimization has become a popular research issue throughout time. Many studies have focused on optimizing energy use through scheduling while others focused on combining task consolidation and scheduling to reduce power consumption. This section concentrates on the most successful strategies to minimize power consumption in cloud computing environments through using *task consolidation* and *task scheduling* techniques.



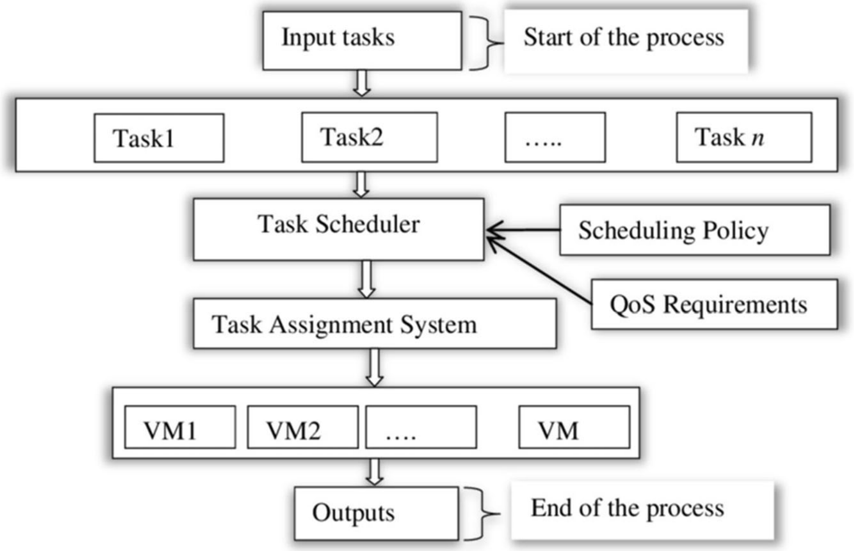


Fig. 3 Scheduling process

A. First Come First Serve

The authors introduced the FCFS (First Come First Serve) algorithm to scheduling and organizing processes Execution based on their arrival time [8]. The Primarily

Attributable of the FCFS is fulfillment first task or resource demand, followed by the following resource demand on the list when the previous one is finished. The Essential feature of FCFS Algorithm is employs the First-In, First-Out (FIFO) algorithm where automatically schedule tasks [8]. In a summary, it's an abstracted method of distributing resources to tasks throughout time as the current task is completed and the new task starts.

B. **Energy-Aware Task Consolidation**

Hsu et al. [9] proposed the ETC (Energy-Aware Task Consolidation) algorithm to minimize virtual machine power consumption. According to the authors, the system's resources limited to the CPU. To consolidate tasks on a virtual machine for a cloud system, they recommend a 70% utilization threshold [9] for CPU utilization resources on a virtual machine where it employs a best-fit migration approach for transferring jobs to whatever VM is available. When a cluster reaches its usage threshold (more than 70%) the task will be migrated to next cluster. If more than one virtual machine is suitable for consolidating the task, their technique will assess the power consumption for each one and compare them to see which one has the lowest power consumption before consolidating the task.

C. **Energy-Efficient Task Scheduling Algorithm**

Sanjaya K. Panda and Prasanta K. Jana [20] proposed an Energy-Efficient Task Scheduling Algorithm (ETSA) to minimize virtual machine power consumption. They combined task consolidation and scheduling to enhance the technique and make it stand out from the competition. The ETSA algorithm assumes heterogeneous resources and therefore does not enforce a fixed start or finish time of tasks [20]. Further, neither task has the same execution time nor resource utilization on virtual machines. That is both execution time and resource utilization for each task is distinct on a specified virtual machine as well as other virtual machines. Both the completion time and total utilization are estimated by using the ETSA algorithm. The ETSA algorithm employs the same energy model of the ECTC and MaxUtil algorithm.

D. **Minimum Completion Time**

R. F. Freund et al. [8] proposed a Minimum Completion Time (MCT) Algorithm to reduce virtual machine power consumption [8]. Tasks should be arranged in the MCT based on their minimal completion time [17] and projected execution time, with the task with the lowest ECT being assigned to the identified virtual machine. Some tasks are scheduled without consideration for a minimum execution time on virtual machines or resources. The MCT method assigns each task to the resource with the best chance of completing it in the shortest amount of time. The task's completion time is calculated as [19]:

$$\text{Completion time} = \text{Execution time} + \text{Ready time}$$

Where, the ready time of a resource is the time it takes to complete all of its tasks. As a result of this method, some tasks are assigned to resources with longer execution times. The MCT algorithm has an O(n) time complexity [1].

E. **Minimum Execution Time**

R. F. Freund et al. **[**8**]** introduced the Minimum Execution Time (MET) Algorithm to reduce virtual machine power consumption. The fundamental idea of MET is to assign a task to a virtual machine or resource based on the quickest delivery execution time, which could result in significant data redundancy **[**15**]**. The MET algorithm locates the task that takes the least amount of time to complete and assigns it to the most appropriate resource. The allocation technique is based on FCFS and determined by resource availability. IT could result in data redundancy among resources. The MET method takes O (n) time To finish a task [1]. The MET is not suitable for high computation environment due to data redundancy among machines. The Most Basic feature of this method is that it takes the shortest expected execution time and distributes it across the available resources.

F. **Water Wave Optimization**

Zheng Yu-Jun in [3] proposed a WWO (Water Wave Optimization Algorithm) to minimize virtual machine power consumption. The WWO algorithm's methodology is a modern nature-inspired optimal control algorithm that uses shallow water wave theory to simulate wave motion to improve the system performance [3]. The waves consider three basic procedures to achieve an efficient optimization solution: propagation, refraction, and breaking [5, 16, 26, 28–31]. The most important characteristic of the WWO algorithm is that it employed shallow water wave theory to handle optimization issues by simulating wave motion.

G. **Energy and Performance- Efficient Task Scheduling Algorithm**

The authors in [10] proposed an Energy and Performance-Efficient Task Scheduling (EPETS) algorithm to achieve better quality and decrease total energy consumption within the time constraints of the deadline. They managed the algorithm in a virtualized cloud as a heterogeneous. There are two stages to it. The first stage involves initial scheduling to minimize processing time and fulfill task deadlines while ignoring energy consumption. The second stage involves task reassignment scheduling. It finds the best execution destination and transfers tasks within the deadline threshold while consuming less energy. Furthermore, reach a good compromise between task scheduling and energy efficiency.

H. Elite learning Harris hawks optimizer

Dina A. Amer et al. presented an Elite Learning Harris Hawks Optimizer (ELHHO) algorithm to improve the quality of the exploration phase of the traditional HHO algorithm [4]. The modifications are employing a scientific intelligent method entitled elite opposition-based learning. ELHHO is an improved Harris Hawks optimizer (HHO) to prevent local optimality and fulfill the quality of service. In addition to minimizing schedule length, the minimal completion time method is employed as an initial phase to establish a specified initial solution rather than a random solution at each running time. ELHHO combines two scientifically intelligent methods: elite opposition-based learning (EOBL) and minimum completion time (MCT).

I. Energy-Saving Task Consolidation

Sanjaya K. Panda and Prasanta K. Jana in [21] proposed an ESTC (Energy-Saving Task Consolidation) algorithm to minimize power consumption on distributed cloud computing systems. The authors found that idle resource utilization consumes a significant amount of energy [21] compared to other resources. As a result, they developed the ESTC technique, which increases the employment of idle resources. It distributes tasks within all virtual machine resources first ‘no idle virtual machines empty’. then broadcasts the remaining others. The next stage in this technique is to estimate the power consumption using an energy model. The authors used the same energy model for all virtual machines in the ESTC algorithm as the STC method.

J. Dynamic Threshold-based Task Consolidation

Priyanka et al. proposed the DTTC (Dynamic Threshold-based Task Consolidation) algorithm to minimize power consumption by restricting virtual machines according to pre-defined usage levels [22]. Instead of using the SLA like in the SLA, the DTTC uses a threshold to assign tasks. Furthermore, unlike the STC algorithm, the authors employed distinct energy models for different virtual machines instead of utilizing the same energy model for all virtual machines. The first VM, for example, has the energy model. On the other hand, the energy model for the second VM was built by multiplying each value of the energy model 1 by 5. After then, the energy model is multiplied by 5, and so on (Table 1).

Table 1 Power consumption of VMs

Level	Utilization	Power Consumption (in Watts)
Idle	1~20	P20=78.5
1	21~30	P30=83
	31~40	P40=85
	41~50	P50=88
2	51~60	P60=93
	61~70	P70=102
3	71~80	P80=109
	81~90	P90=122
	91~100	P100=136

Table 2 Example for SLA

SLA LEVEL	UTILIZATION
1	40%
2	70%
3	100%

K. **SLA-based Task Consolidation**

Priyanka et al. [22] proposed a STC (SLA-based Task Consolidation) algorithm that employs the SLA (Service Level Agreements) to reduce power consumption. The user-specified SLA level and matching utilization are compared to the usage of available VMs when a task is entered into the system. According to Table 2, the STC technique uses the same energy model [22] for all virtual machines and assumes three different virtual machine use levels. Each VM has its own threshold, such as virtual machine1's threshold of 40%, virtual machine2's threshold of 70%, and virtual machine3's threshold of 100%. The SLA scheduling is the same as FCFS Scheduling. However, the FCFS has one identified Threshold of 100%, whereas the SLA Scheduling has several Thresholds 'SLA' as indicated in Table 2.

5 Proposed algorithm

This section presents a new efficient Task Consolidation based Power Minimization (TCPM) algorithm to reduce energy consumption. The proposed TCPM algorithm combines Task Consolidation with Task Scheduling to achieve maximum resource efficiency while decreasing energy waste. Tasks are distributed across all virtual machines using a best-fit technique to enhance resource efficiency while reducing energy waste. By focusing on tasks consolidated into a single task, parallel time and parallel resources have been integrated into the TCPM approach, resulting in lower power consumption and improve reliability.

In the proposed TCPM algorithm, the virtual machines are assumed to be homogeneous use the same form of the energy model for all the virtual machines, **Energy Model 2** shown in Table 1. The proposed TCPM methodology consists of three parts:

A. **TCPM Consolidation**
B. **TCPM Scheduling**
C. **Power Consumption Calculation**

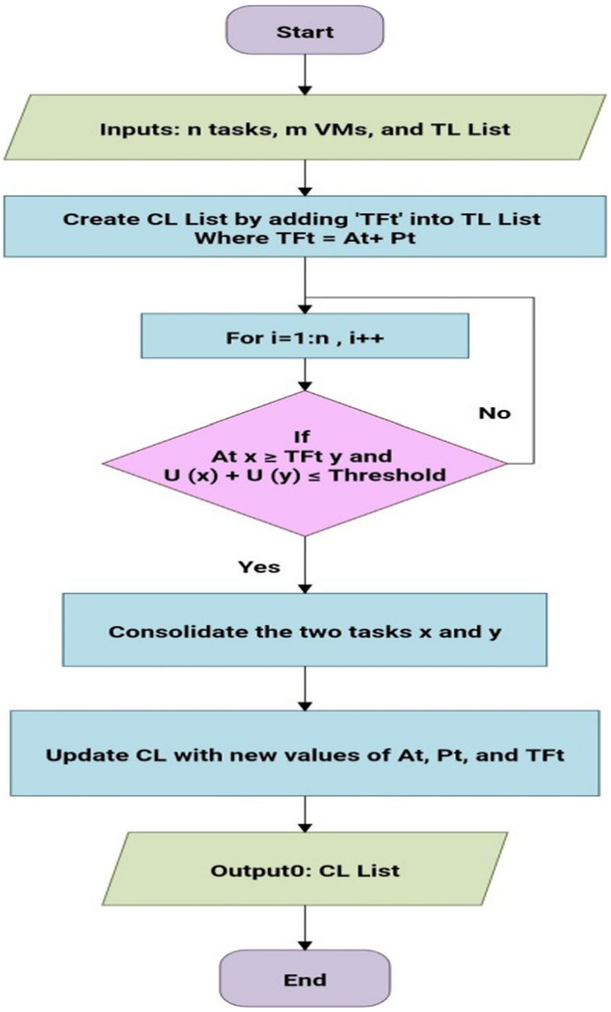

Start
Inputs: n tasks, m VMs, and TL List
Create CL List by adding 'TFt' into TL List
Where TFt = At+ Pt
For i=1:n , i++
No
If
At x ≥ TFt y and
U (x) + U (y) ≤ Threshold
Yes
Consolidate the two tasks x and y
Update CL with new values of At, Pt, and TFt
Output0: CL List
End

Fig. 4 Flowchart of TCPM consolidation

Algorithm: TCPM Framework.

```
1.  Start.
2.  Read Inputs: n tasks and m Virtual Machines.
3.  Algorithm Input: Task List "TL".
4.  Implement TCPM CONSOLIDATION.
5.  Output0: "CL" List.
6.  Implement TCPM SCHEDULING.
7.  Output1: "AL" List.
8.  Calculate Power Consumption.
9.  Output2: Power Consumption
10. End
```

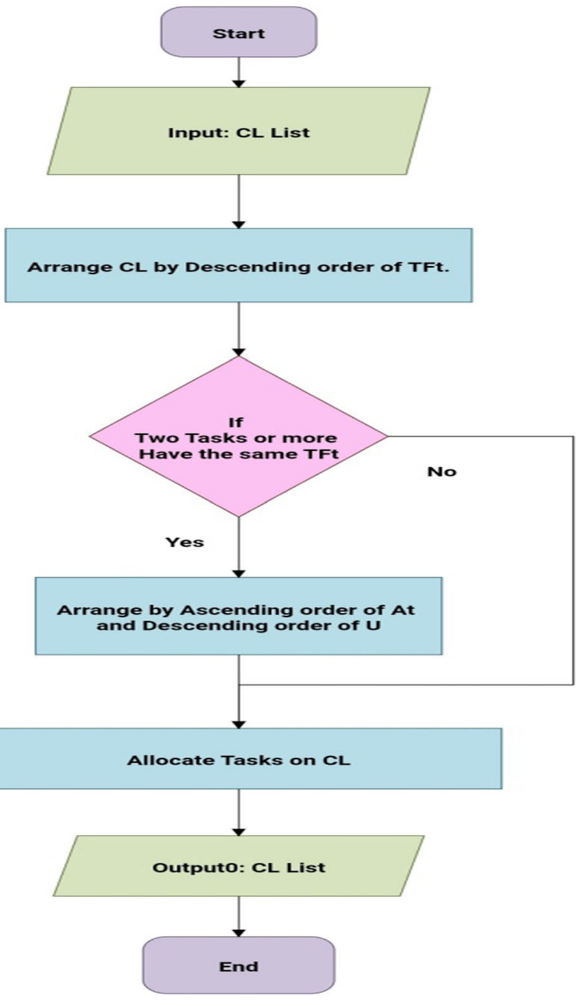

Start
Input: CL List
Arrange CL by Descending order of TFt.
If Two Tasks or more Have the same TFt
No
Yes
Arrange by Ascending order of At and Descending order of U
Allocate Tasks on CL
Output0: CL List
End

Fig. 5 Flowchart for sorting CL scheduling

A. TCPM Consolidation

The TCPM consolidation first consolidates tasks that have an arrival time greater than the 'TFt' of the second task and achieve VM utilization < Threshold (70%). Then, update the Consolidation List (CL) with new arrival time, processing time, and 'TFt' values of tasks. It finally, sorts the CL. Fig. 4 shows the flowchart of TCPM consolidation.

In the TCPM Consolidation, Sorting CL SCHEDULING sorts the consolidation list by descending order of "TFt", as shown in Fig. 5. If two tasks or more have the same TFt value, apply the priority for the tasks arrive first which is to have the minimum time and work to the task that has the higher utilization first following to the consolidated tasks.

$$U_{new}\ (VM) \leq \text{Threshold}$$

Where, $U_{new}\ (VM) = U_{old}\ (VM) + U\ (_{\text{Existing Task}})$

Distribute an existing task to specified VM; otherwise, Task is rejected (VM migration) as indicated in the following TCPM Scheduling algorithm.

Algorithm: TCPM Consolidation.

```
1.  Start.
2.  Read Inputs:"TL" task List
3.  Create Consolidation Task List "CL"
     By adding extra column 'TFt' into "TL" List
    Where TFt =At + Pt
4.  For each task i=0,1,…,n
5.  If   $At_x \geq TFt_y$ & $U(x) + U(y) \leq$ Threshold     do
6.  Consolidate Task (x) with Task (y) and Consider
    them as a Single task with New Arrival Time equal
    Minimum Arrival time of two Consolidated tasks
    "CT", 0and New Processing Time equal
    Summation of two Consolidated tasks "CT"
        $At_{new} = Min (At(x),At(y))$,
        $Pt_{new} = Sum (Pt(x), Pt(y))$.
7.  Back to For loop
8.  Update TFt with new At, &Pt
9.  Update CL with new At, Pt, &TFt
10. Else
11. No Change
12. End If
13. End For
14. Sorting "CL" Scheduling
15. Output0: "CL" List
16. End
```

Algorithm: Sorting CL SCHEDULING.

```
1.  Start.
2.  Read Inputs: "CL" List
3.  For each task i=1,2,.....,n
4.  Find the Maximum value of At first "Descending order
    of TFt"
5.  Allocated it on CL List
                        Only Tasks have the same TFt
6.  While only tasks have the same TFt value    do
    Find the Minimum value of  Arrival  time "At" to the
    first "Ascending order of "At" " and descending order
    of "U"
7.  End While
8.  Allocated it on CL List
9.  End For
10. Output0: "CL" List
11. End
```

B. TCPM Scheduling

In the proposed TCPM algorithm, the TCPM scheduling methodology, shown in Fig. 6, generally limits the number of idle resources by distributing a task to a virtual machine that is already idle. Unless No device is idle, go into the remaining tasks and implements the FCFS algorithm when utilization of a VM became after adding utilization for the existing task that need to schedule as in the following way.

Algorithm: TCPM Scheduling.

```
1.  Start.
2.  Read Inputs:“CL”
3.  Create Allocation List “AL” by adding an
    Extra column into “CL” List called “ Machine-
    Id”, And update Temporary Finish Time "TFt"
    With Actual Finish Time "AFt"
            First m Tasks
4.  For each task i=1,2,…,m
5.  Schedule by FIFO “No Idle VM Empty”
6.  End For
    Remaining Tasks
7.  while task n > m
8.  If $U_{new}(VM) \leq$ Threshold   do
    Where $U_{new}(VM) = U_{old}(VM) + U(\text{Existing Task})$
9.  Allocated Existing Task to indicated VM
10. Else
11. Reject The Task (VM Migration)
12. End If
13. If several virtual machines are suitable, the
    best-fit strategy is to determine the most
    optimal one
14. Update “AL” List
15. End while
16. Output1: The Allocation List “AL”
17. End
```

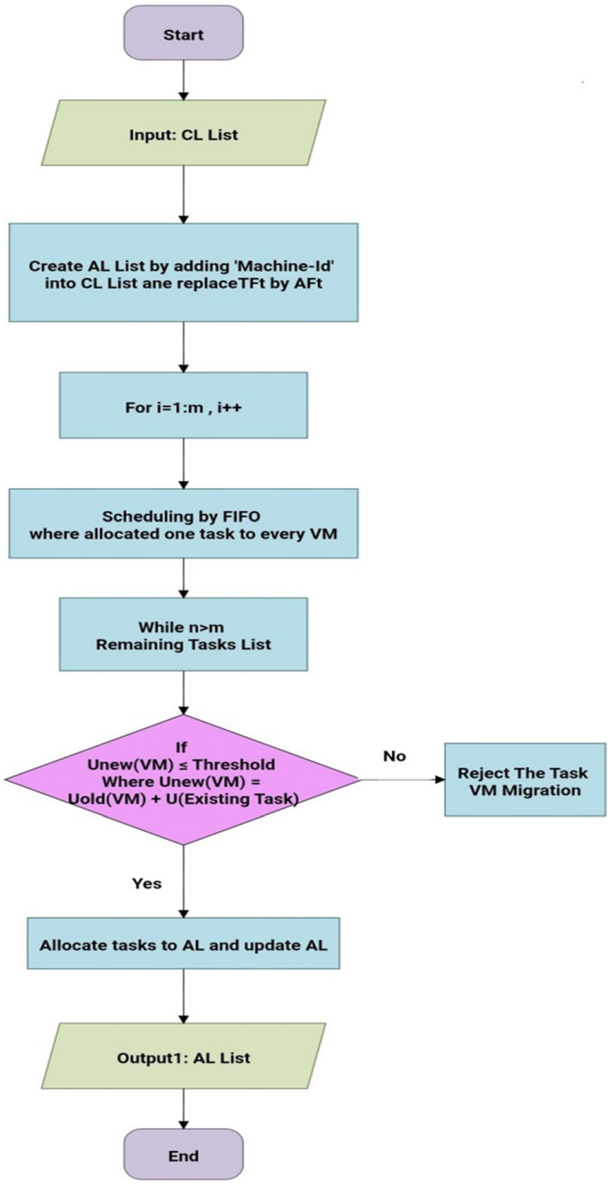

Start
Input: CL List
Create AL List by adding 'Machine-Id' into CL List ane replaceTFt by AFt
For i=1:m , i++
Scheduling by FIFO where allocated one task to every VM
While n>m Remaining Tasks List
If Unew(VM) ≤ Threshold Where Unew(VM) = Uold(VM) + U(Existing Task)
No
Reject The Task VM Migration
Yes
Allocate tasks to AL and update AL
Output1: AL List
End

Fig. 6 Flowchart for TCPM scheduling

C. **Power Consumption**

The power consumption procedure is employed to evaluate Power Consumption. We implemented Energy Model 2 and Output2 represents the power consumption. Fig. 7 shows the flowchart of power consumption.

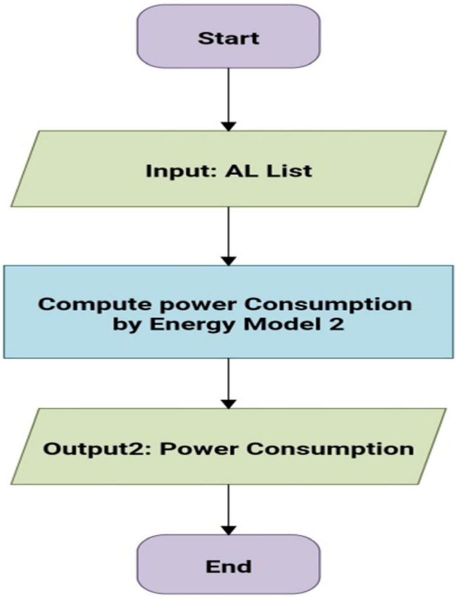

Start
Input: AL List
Compute power Consumption by Energy Model 2
Output2: Power Consumption
End

Fig. 7 Flowchart of Power Consumption

Algorithm: Power Consumption.

```
1. Start.
2. Input: Allocation List "AL"
3. Calculate Power Consumption
4. Print Power Consumption
5. Output2: Power Consumption
6. End
```

The proposed TCPM algorithm The overall procedure of the proposed TCPM algorithm is listed in the following table. Fig. 8 shows the flowchart of the proposed TCPM algorithm.
Algorithm: TCPM Algorithm.

```
1.  Start.
2.  Read Inputs: n tasks, m VMs, &"TL" Task List
    TL➔(Task-Id, Arrival Time "At", Processing Time "Pt", Utilization "U" %).
3.  Create Consolidation Task List "CL"
        By adding extra column 'TFt' into "TL" List
    Where TFt =At + Pt
    CL➔(Task-Id, Arrival Time "At", Processing Time "Pt", Temporary Finish Time "TFt",
    Utilization "U" %).
4.  For each task i=0,1,…,n
5.  If   At x ≥ TFt y & U(x) + U(y) ≤ Threshold     do
6.  Consolidate Task (x) with Task (y) and Consider them as a Single task with New Arrival Time
    equal Minimum Arrival time of two Consolidated tasks "CT", and New Processing Time equal
    Summation of two Consolidated tasks "CT"
    $At_{new}$ = Min (At (x),At (y) ),
    $Pt_{new}$= Sum (Pt (x),Pt (y) ).
7.  Back to loop
8.  Update TFt with new At, &Pt
9.  Update CL with new At, Pt, &TFt
10. Else
11. No Change
12. End If
13. End For

    CL List
14. For each task i=1,2,…..,n
15. Find the Maximum value of TFt first "Descending order of TFt"
16. Allocated it on CL List

    Only Tasks have the same TFt
17. While only tasks have the same TFt value    do
    Find the Minimum value of Arrival time "At" to the first "Ascending order of "At" " and
    descending order of "U"
18. End While
19. Allocated it on CL List
20. End For
21. Output0: CL
22. Create Allocation List "AL" by adding an Extra column into "CL" List called " Machine-Id",
    And update Temporary Finish Time "TFt" With Actual Finish Time "AFt"

    AL➔ (Task-Id, Arrival Time "At", Machine-Id, Processing Time "Pt", Actual Finish Time
    "Aft", Utilization "U" %).

    First m Tasks
23. For each task i=1,2,…,m
24. Schedule by FIFO "No Idle VM Empty"
25. End For
    Remaining Tasks
26. while task n > m
27. If   $U_{new}$(VM) ≤ Threshold     do
    Where $U_{new}$(VM) = $U_{old}$(VM) + U (Existing Task)
28. Allocated Existing Task to indicated VM
29. Else
30. Reject The Task (VM Migration)
31. End If
32. If several virtual machines are suitable, the best-fit strategy is to determine the most optimal one
33. Update "AL" List
34. End while
35. Output1: "AL" List

36. Calculate Power Consumption
37. Print Power Consumption
38. Output2: Power Consumption
39. End
```

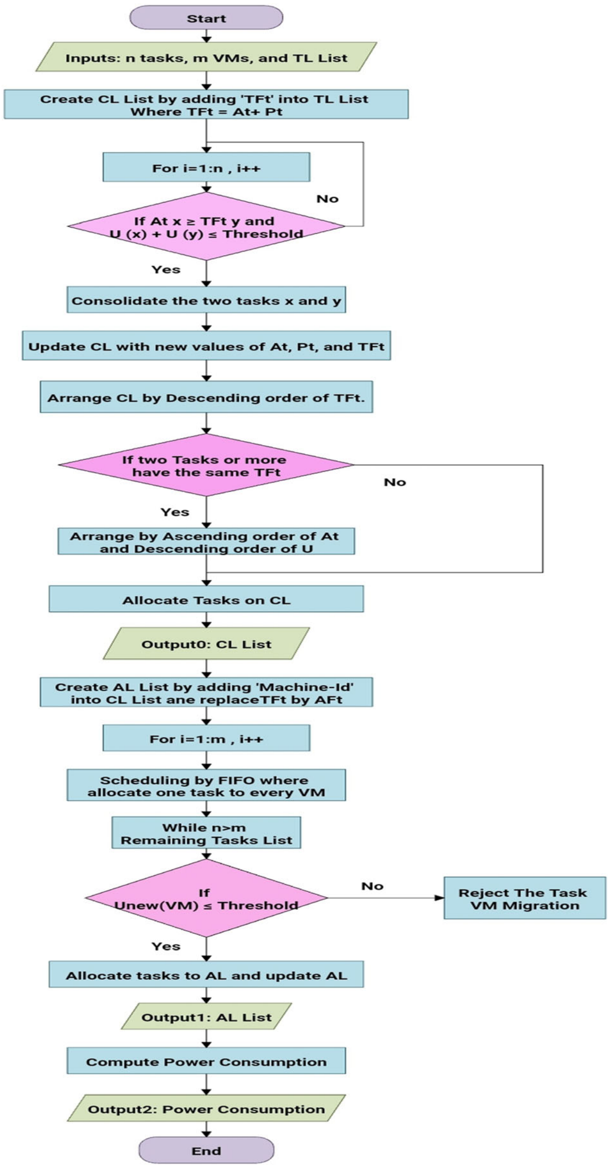

Start
Inputs: n tasks, m VMs, and TL List
Create CL List by adding 'TFt' into TL List Where TFt = At+ Pt
For i=1:n , i++
If At x ≥ TFt y and U (x) + U (y) ≤ Threshold
No
Yes
Consolidate the two tasks x and y
Update CL with new values of At, Pt, and TFt
Arrange CL by Descending order of TFt.
If two Tasks or more have the same TFt
No
Yes
Arrange by Ascending order of At and Descending order of U
Allocate Tasks on CL
Output0: CL List
Create AL List by adding 'Machine-Id' into CL List ane replaceTFt by AFt
For i=1:m , i++
Scheduling by FIFO where allocate one task to every VM
While n>m Remaining Tasks List
If Unew(VM) ≤ Threshold
No
Reject The Task VM Migration
Yes
Allocate tasks to AL and update AL
Output1: AL List
Compute Power Consumption
Output2: Power Consumption
End

Fig. 8 Flowchart of TCPM algorithm

Table 3 Task list

Task-Id	At	Pt	U
t_0	0 s	50 s	30%
t_1	10 s	20 s	30%
t_2	12 s	35 s	40%
t_3	15 s	15 s	30%
t_4	20 s	30 s	60%
t_5	30 s	25 s	30%

6 Illustration example

The following example illustrates the application of the proposed TCPM approach. The Task List (TL), shown in Table 3, has four tuples: *Task-Id* is the task identifying number, *At* is the arrival time, *Pt* is the processing time, and *U* is the Resources utilization in percentage. Further, there are three virtual machines in this example and we assume that an idle resource is often used 20% of the virtual machines utilization and requires 78.5 watts, as shown in Table 1 ‘energy model’.

A. **TCPM Consolidation**

Create Consolidation List (CL), shown in Table 4, by adding extra column ‘TFt’ into the TL. Where, TFt = At + Pt.

In the illustration example, only one task, t_5, has At equals 30 that less than other TFt tasks with the minimum TFt of 30. Thus, t_5 achieves the conditions requirements.

$$\mathrm{At}\ x \geq \mathrm{TFt}\ y \& \mathrm{U}(x) + \mathrm{U}(y) \leq \mathrm{Threshold},$$

Hence, $t_5$can be consolidated with t_1 or $t_3$becausethey have the same TFt‘30’ but t_1 has the minimum At time of them. To avoid wasting time, the best-fit strategy is used, and t_5 will be consolidated with t_1. As a result, Consolidated Task ‘CT_1’ for $t_1$and t_5 have At $_{new}$ = 10, and TFt $_{new}$ = 55 (as shown in Table 5).

The CL is sorted by descending order of TFt. If two tasks have the same TFt, ex. t_0 and $t_{4,}$ so first allocated the tasks that arrive first, here $t_{0.}$ Table 6 shows the sorting process.

B. **TCPM Scheduling**

Table 4 Consolidation list

Task-Id	At	Pt	TFt	U
t_0	0 s	50 s	**50 s**	30%
t_1	10 s	20 s	**30 s**	30%
t_2	12 s	35 s	**47 s**	40%
t_3	15 s	15 s	**30 s**	30%
t_4	20 s	30 s	**50 s**	60%
t_5	**30 s**	25 s	**55 s**	30%

Table 5 Consolidation list

Task-Id		At		Pt	TFt		U
CT_1	t_1	10 s	10 s	20 s	**30 s**	**55 s**	30%
	t_5	30 s		25 s	**55 s**		30%
t_0		0 s		50 s	**50 s**		30%
t_2		12 s		35 s	**47 s**		40%
t_3		15 s		15 s	**30 s**		30%
t_4		20 s		30 s	**50 s**		60%

Table 6 Consolidation list

Task-Id		At		Pt	TFt		U
CT_1	t_1	10 s	10 s	20 s	30 s	55 s	30%
	t_5	30 s		25 s	55 s		30%
t_0		0 s		50 s	50 s		30%
t_4		20 s		30 s	50 s		60%
t_2		12 s		35 s	47 s		40%
t_3		15 s		15 s	30 s		30%

The TCPM scheduling creates an Allocation List (AL) by first adding an extra column into the CL list called "Machine-Id" and update Temporary Finish Time "TFt" With Actual Finish Time "AFt", as shown in Table 7. Then,schedule tasks by FIFO "No Idle VM Empty". Where, every virtual machine must take task then scheduling remaining tasks to VMs by applying the condition:

$$U_{new}(VM) \leq \text{Threshold}$$

Where, $U_{new}(VM) = U_{old}(VM) + U_{Existing}(Task)$.

TCPM algorithm the tasks in the TCPM algorithm are allocated to virtual machines, as shown in Fig. 9.

C. **Power Consumption Calculation**

To evaluate virtual machine energy consumption, from the **Energy Model 2**, Table 1, and Fig. 2, the mathematical methodology of energy consumption is

$$E = \sum_{i=1}^{m} E(IVMi) \times f + E(NVMi(U))$$

Idle Resources Compute from

$$E(IVMi) = P20 \times \Delta v$$

$$\Delta v = vmax - vmin$$

vmax Finish Time = Arrival Time + Processing Time.

Table 7 Allocation list

Task-Id		Machine-ID	At		Pt	AFt		U
CT_1	t_1	**VM0**	10 s	10 s	20 s	30 s	55 s	30%
	t_5		30 s		25 s	55 s		30%
t_0		**VM1**	0 s		50 s	50 s		30%
t_4		**VM2**	20 s		30 s	50 s		60%
t_2		**VM0**	12 s		35 s	47 s		40%
t_3		**VM1**	15 s		15 s	30 s		30%

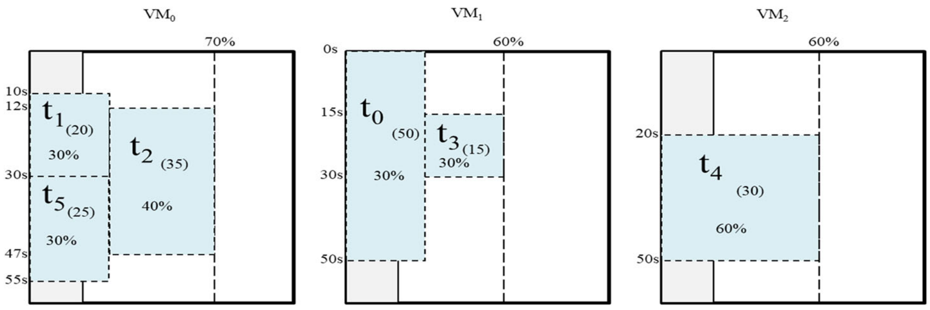


Fig. 9 Mapping tasks on virtual machines by TCPM algorithm

As a result, E = E (VM0) + E (VM1) + E (VM2).

E(VM0) = $P20 \times \Delta v + P30 \times \Delta v + P40 \times \Delta v + P50 \times \Delta v + P60 \times \Delta v + P70 \times \Delta v$ =(78.5×55) + (83 × 45) + (85 + 88 + 93 + 102) (35) = 20,932.5 W.
E(VM1) = $P20 \times \Delta v + P30 \times \Delta v + P40 \times \Delta v + P50 \times \Delta v + P60 \times \Delta v$= (78.5×55) + (83 × 50) + (85 + 88 + 93) (15) = 12,457.5 W.
E(VM0) = $P20 \times \Delta v + P30 \times \Delta v + P40 \times \Delta v + P50 \times \Delta v + P60 \times \Delta v$= (78.5×55) + (83 + 85 + 88 + 93) (30) = 14,787.5 W.

The total energy consumption by **the proposed TCPM algorithm** is E = 48,177.5 W.

FCFS algorithm The tasks in the FCFS algorithm are allocated to virtual machines, as shown in Fig. 10.

The total energy consumption by **the FCFS algorithm** is E = 60,127.5 W.

MCT algorithm The tasks in the MCT algorithm are allocated to virtual machines, as shown in Fig. 11.

The total energy consumption by **the MCT algorithm** is E = 59,212.5 W.

ETC algorithm The tasks in the ETC algorithm are allocated to virtual machines, as shown in Fig. 12.

The total energy consumption by the **ETC algorithm** is E = 50,092.5 W.

The final results from the above Illustration Example clearly show that our proposed TCPM algorithm reduced the energy consumption compared with recent algorithms FCFS, MCT, and ETC algorithms.

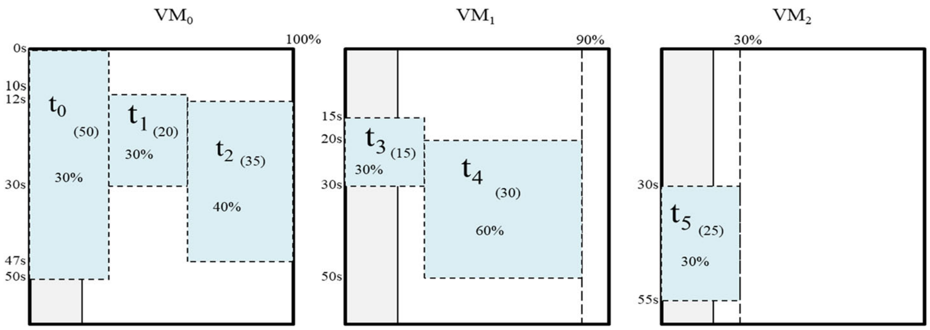


Fig. 10 Mapping tasks on virtual machines by FCFS algorithm

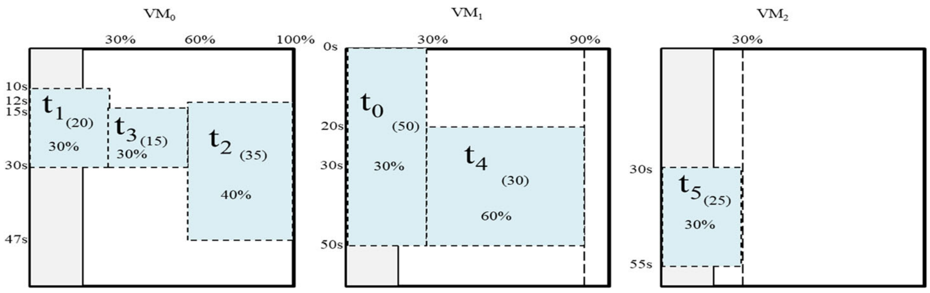


Fig. 11 Mapping the tasks on virtual machines by MCT Algorithm

7 Simulation experiments

In this section, five different datasets are created with the CloudSim and used to evaluate the proposed algorithm. The datasets include 50, 200, 800, 400 and 2000 tasks that may be scheduled onto the available VMs in the cloud environment.

A. Experimental Environment

The simulation was carried out by using the Cloudsim 3.0.3 ToolKit on a 64-bit Microsoft Windows 10 platform including an Intel Core i7-version processor, 3.0 GHz CPU, and 8 GB RAM.

B. Experimental Setup

The CloudSim3.0.3 simulator, which runs on Eclipse IDE for Java and DSL Developers - 2021-03, was used to implement the proposed algorithm. The experiments are carried out using sequential distribution, which is the standard in the CloudSim tool kit, while other existing algorithms such as FCFS, MCT, WWO, and ELHHO are implemented in the simulator.

C. Experimental Results

To evaluate the performance of the proposed TCPM algorithm, it first implemented in the CloudSim and used to schedule tasks of a given dataset onto the available VMs. Then, the results are compared with the most recent existing algorithms, such as FCFS, MCT, WWO, and ELHHO considering several simulation parameters as Power Consumption, Reliability, Throughput, and Failure rate.

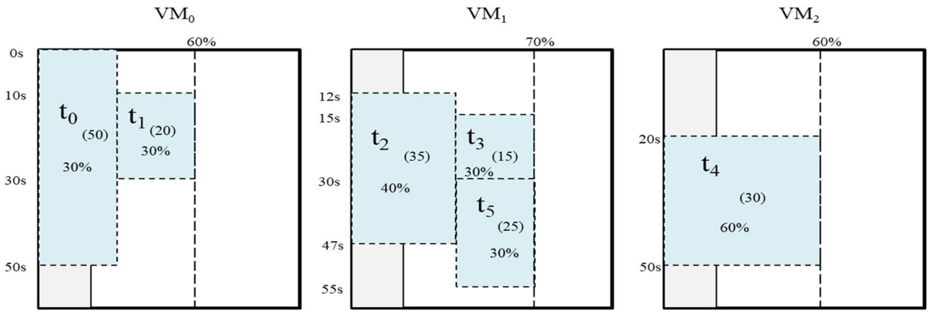


Fig. 12 Mapping the tasks on virtual machines by ETC algorithm

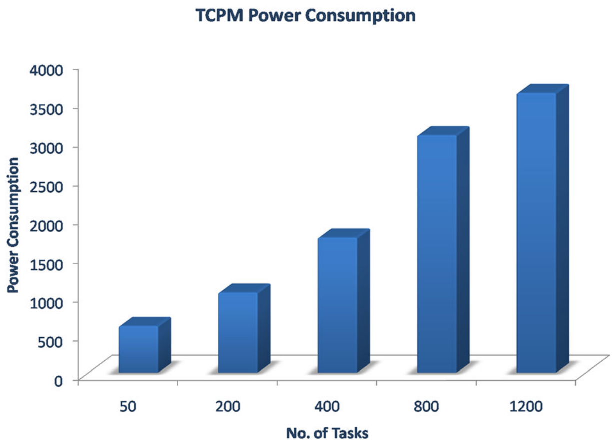


Fig. 13 Power consumption by TCPM

8 Power consumption

Power consumption is the amount of energy used by the cloud system. In this evaluation the power consumption is computed by **Energy Model 2** as illustrated previously. Fig. 13 shows the power consumption when applying the proposed TCPM algorithm to distribute different tasks onto VMs.

Comparative study between the power consumption by the proposed TCPM algorithm and the existing FCFS, MCT, WWO, and ELHHO algorithms considering five different datasets is shown in Fig. 14. From the figure, it is clear that the proposed TCPM successfully saved the power consumption. The TCPM algorithm achieves the minimum value compared with the others algorithms for the five datasets.

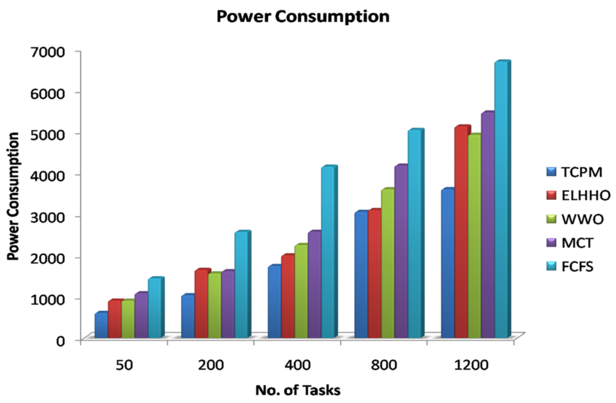


Fig. 14 Power consumptionby different algorithms

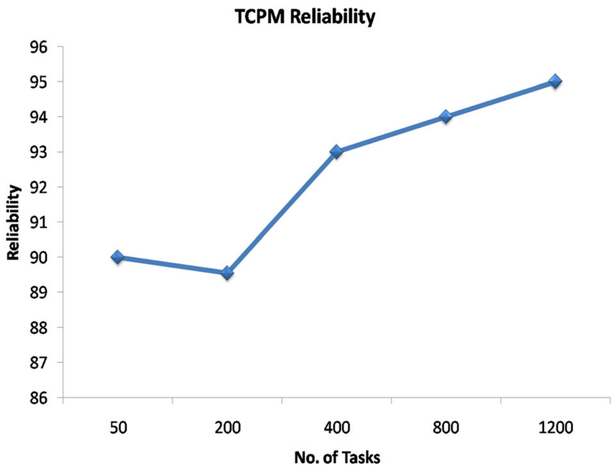


Fig. 15 TCPM Reliability

9 Reliability

Reliability is the quality of the algorithm in which experiments run without error. Figure 15 shows the Reliability when applying the proposed TCPM algorithm to distribute different tasks onto VMs.

Comparative study between Reliability by the proposed TCPM algorithm and the existing FCFS, MCT, WWO, and ELHHO algorithms considering five different datasets is shown in Fig. 16. From the figure, it is clear that the proposed TCPM successfully distribute different tasks onto VMs with more reliability. The TCPM algorithm achieves the maximum reliability value compared with the others algorithms for the five datasets.

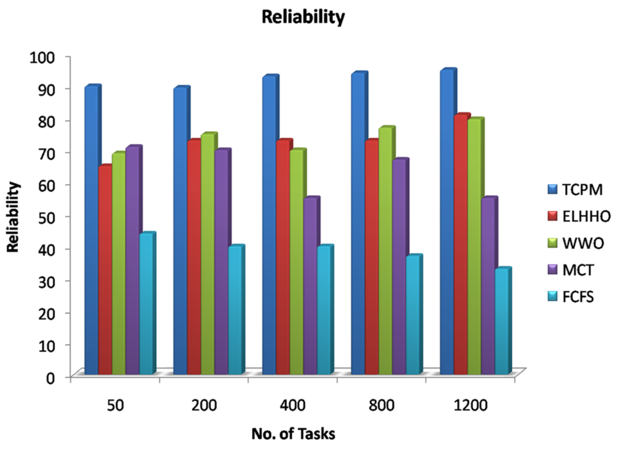


Fig. 16 Reliability for algorithms

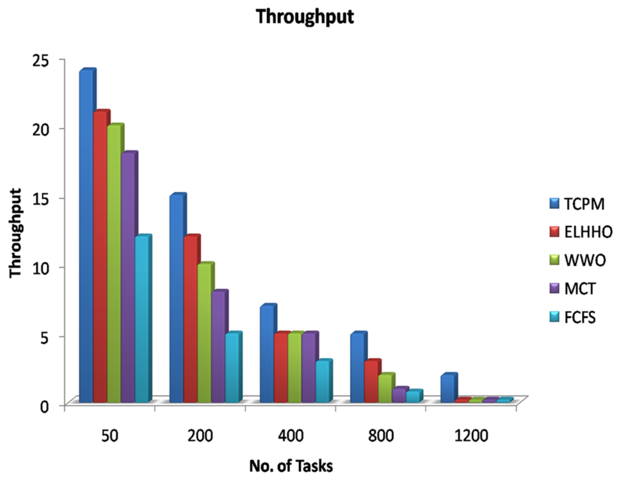


Fig. 17 Throughput for algorithms

10 Throughput

Throughput is the maximum number of tasks that complete its processing in a specific amount of time.

Comparative study between throughput of the proposed TCPM algorithm and the existing FCFS, MCT, WWO, and ELHHO algorithms considering five different datasets is shown in Fig. 17. From the figure, it is clear that the proposed TCPM successfully distribute different tasks onto VMs with more throughputs. The TCPM algorithm achieves the maximum throughput value compared with the others algorithms for the five datasets.

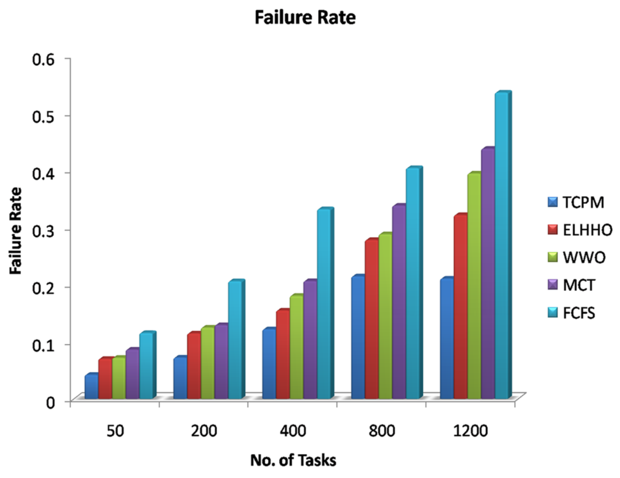


Fig. 18 Failure rate for algorithms

Table 8 Data center specifications

Specifications	Value
Data Centers Number	1
Hosts Number	3
User Number	1

Table 9 Host specifications

Specifications	Value
Storage	1 Million
RAM	2024 * 10 MB
BW	100,000
Shared Policy	Space Shared Policy

11 Failure rate

The failure rate is the estimated amount of times an algorithm will fail in a specific amount of time.

Comparative study between failure rate of the proposed TCPM algorithm and the existing FCFS, MCT, WWO, and ELHHO algorithms considering five different datasets is shown in Fig. 18. From the figure, it is clear that the proposed TCPM successfully distribute different tasks onto VMs with minimum failure rate. The TCPM algorithm achieves the minimum failure rate value compared with the others algorithms for the five dataset.

12 Conclusion

An efficient algorithm called Task Consolidation based Power Minimization (TCPM) is developed in this paper. The main purpose is to schedule tasks onto available resources of the cloud environment efficiently with minimize power consumption. The proposed TCPM algorithm combines many benefits of the existing algorithms. The best-fit procedure is used to achieve the best possible resource utilization and avoid wasting energy. Task consolidation is incorporated into the TCPM algorithm to minimize power consumption. Further, tasks are scheduled into all virtual machine first (No Idle VM Resources) and the remaining tasks are

Table 10 VMs specifications

Specifications	value
VMs Number	10, 25, 50 and 70
V_{mips}	500 to 2000
V_{mem}	500 MB
V_{CPUS}	2, 4, 8, 16, and 32
Virtual Machine Manager	Xen
BW	0.5 Gb/s
Size	100 MB

sorted by descending order of TFt to achieve the best use of power consumption for idle VM Resources until 20% of Resources consumes became take the highest processing time. The proposed TCPM algorithm is evaluated considering 5 different datasets. The results of the proposed TCPM algorithm are compared with more recent algorithms such as FCFS, MCT, WWO, and ELHHO algorithms using the CloudSim toolkit.

In the future work, the TCPM algorithm may enhance by identifying more than one thresholds (ex. 70%, 85%, either). Compute their power consumption. Then, compare the results and select the threshold that achieves the enhanced power consumption.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

The DA statement The Dataset are generated randomly to cloudsim simulation. The length of the tasks is from 1000 to 10,000 Million instructions. The data center Specifications are given in Table 8 and the host Specifications are given in Table 9 while the VMs Specifications are given in Table 10.

References

1. Afaf Abdelkader Abdelhafiz (2018) “Tuples: A New Scheduling Algorithm”, J Comput 13(11):1309–1315
2. Aishwarya, Anusha K, Gagana, Megha (n.d.) Survey on Energy Consumption in Cloud Computing. Int J Eng Res Technol 9(4): 2278–0181
3. Amer DA, Attiya G, Ziedan I, Nasr AA (May 2021) A new task scheduling algorithm based on water wave optimization for cloud computing. Int J Comput (0975–8887) 183(3):65–75
4. Amer DA, Attiya G, Zeidan I, Nasr AA (2021) Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. J Supercomput 78:2793–2818
5. Arulkumar V, Bhalaji N (n.d.) Load balancing in cloud computing using water wave algorithm. Article in Concurrency and Computation Practice and Experience, September 2019, © 2019 John Wiley & Sons, Ltd.
6. Badr S, El Mahalawy A, Attiya G, Nasr AA (n.d.) A Review on Task Consolidation for Cloud Computing Environment. ICEEM2021, ©2021 IEEE
7. Bharathi A, Mohana RS, Ushapriya A (January 2014) Reducing energy consumption and increasing profit with task consolidation in clouds. Int J Eng Sci Innov Technol (IJESIT) 3(1):200–207
8. Elzeki OM, Rashad MZ, Elsoud MA (July 2012) Overview of scheduling tasks in distributed computing systems. Int J Soft Comput Eng (IJSCE) ISSN: 2231–2307 2(3)
9. Hsu C, Chen S, Lee C, Chang H, Lai K, Li K, Rong C, Optimizing Energy Consumption with Task Consolidation in Clouds. Inf Sci 258:452-462
10. Hussain M, Wei L-F, Lakhan A, Wali S, Ali S, Hussain A (2021), Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. Sustain Comput-Infor 30:100517

11. Kaur A, Rupinderkaur, Jain P (August 2013) Algorithms for Task Consolidation Problem in a Cloud Computing Environment. Int J Comput Appl (0975–8887) 75(4):17–22
12. Khurma RA, Al Harahsheh H, Sharieh A (September 2018) Task scheduling algorithm in cloud computing based on modified round robin algorithm. J Theor Appl Inf Technol 96(17)
13. Koot M, Wijnhoven F (2021), Usage impact on data center electricity needs: a system dynamic forecasting model. Appl Energy 291:116798
14. Lee, Zomaya A (2012) Energy-efficient utilization of resources in cloud computing systems. J Supercomput 60:268–280
15. Madni SHH, Latiff MSA, Abdullahi M, Abdulhamid S'i M, Usman MJ (2017) Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. PLoS One 12(5): e0176321. https://doi.org/10.1371/journal.pone.0176321
16. Medara R, Singh RS, Selva Kumar U, Barfa S (n.d.) Energy Efficient Virtual Machine ConsolidationUsing Water Wave Optimization. ©2020 IEEE
17. Mehdi NA, Mamat A, Amer A, Abdul-Mehdi ZT (December 2011) Minimum Completion Time for Power-Aware Scheduling in Cloud Computing. Article
18. Mekala MS, Viswanathan P (July 2021) CTRV: resource based task consolidation approach in cloud for green computing. Distributed and Parallel Databases, Springer
19. Mishra SK et al (2020) Energy-aware task allocation for multi-cloud networks. IEEE Access 8:178825–178834
20. Panda SK, Jana PK (2019) An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems. Clust Comput 22:509–527
21. Panda SK, Jana PK (n.d.) An Efficient Energy Saving Task Consolidation Algorithm for Cloud Computing Systems. 2014 International Conference on Parallel, Distributed and Grid Computing, pp. 262–267
22. Panigrahi P, Panda SK, Tripathy CR (October 2015) Energy efficient task consolidation algorithms for cloud computing systems. J Inf Process 94:34–45
23. Reda NM, Tawfik A, Marzok MA, Khamis SM (2015) Sort-Mid tasks scheduling algorithm in grid computing", Cairo University. J Adv Res
24. Singh P, Sengupta J, Suri PK (2020) CPU and memory requirement based task consolidation for reducing energy consumption in cloud computing. J Crit Rev 7(09)
25. Singhn P, Jain EA (April 2014) Survey Paper on Cloud Computing. Int J Eng Technol Innov 3(4):2319
26. Siva M, Balamurugan R, Lakshminarasimman L (2016) Water Wave Optimization Algorithm for Solving Economic Dispatch Problems with Generator Constraints. Int J Intell Eng Syst 9(4):31–40
27. Carolan J, Gaede S (2009) Introduction to Cloud Computing Architecture, Sun Microsystems Inc. White Paper. Sun Microsystems Inc., Santa Clara, 2009.
28. Taherian Dehkordi S, Khatibi Bardsiri A, Zahedi MH (2019) Prediction and diagnosis of diabetes mellitus using a water wave optimization algorithm. J AI Data Mining 7(4):617–630
29. Wu X, Zhou Y, Lu Y (n.d.) Elite Opposition-Based Water Wave Optimization Algorithm for Global Optimization. Research Article, Hindawi, Mathematical Problems in Engineering, Volume 2017, Article ID 3498363, 25 pages
30. Yan Z, Zhang J, Tang J (2020) Modified water wave optimization algorithm for underwater multilevel thresholding image segmentation", Springer. Multimed Tools Appl 79:32415–32448
31. Zheng Y-J (March 2015) Water wave optimization: a new nature-inspired metaheuristic. Comput Oper Res 55:1–11