http://dx.doi.org/10.5455/sf.100672

Science Forum (Journal of Pure and Applied Sciences)

journal homepage: www.atbuscienceforum.com

A multi-objective task scheduling and resource allocation for energy efficiency in cloud computing using a heuristic approach

Fatima Umar Zambuk*, Mohammed Jiya, Mohammed Kabir Dauda, Ismail Aliyu, Maryam Maishanu, Maryam Abdullahi Musa

Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi, Nigeria

ABSTRACT

Cloud computing is one of the modern and trending technologies that touches lives in today's world. Resource allocation and task scheduling are the key aspects of today's cloud. A hybridized heuristic approach that commingles the Modified Ant Colony Optimization (MACO) and deterministic divide-and-conquer approach to perform resource allocation and task scheduling is purported in this paper. In the proposed scheme, before cloud resource allocation takes effect, each task unit is processed by the ACO. MACO allocates the resources considering cloud resources load as constraints and bandwidth. In addition, the divide-and-conquer preempts resource-intensive tasks thereby improving the solution. The proposed technique is found to be efficient with waiting time, response time, and energy consumption as compared with bat and ACO algorithms. Based on the extensive simulations conducted, the proposed method consumed energy of 700 joules while 1,200 and 1,400 joules were consumed by a bat and ACO, respectively.

ARTICLE INFO

Article history:
Received 22 July 2021
Received in revised form
22 July 2021
Accepted 01 August 2021
Published 12 October 2021
Available online 12 October 2021

KEYWORDS

Resource management
Task scheduling
Heuristic
Cloud computing
Energy consumption

1. Introduction

Cloud computing is an accelerating and far-flunk technology in today's world. It is applied almost in all forms of computing ranging from big data, IoT applications, data analytics, smart cities to mention but a few (Usman et al., 2019). Cloud computing has changed how enterprises or individuals compute with the traditional ways of deploying services. It has provided several different types of services to registered users as web services so that the users do not need to invest in computing infrastructure. For all the computing services offered, clients are expected to submit their job requests to the service provider through the Internet medium. Cloud service providers (CSPs) manage the resources to fulfill requests generated by clients. They are responsible for managing the computing resources used to cater to trillions of jobs requests in the Internet network. Quite a number of challenges are being faced for the management of such services that include effective policies for the management of such resources, excessive energy production that is harmful to the ecosystem, security, etc. CSP employs scheduling algorithms to schedule the incoming request (tasks) and to manage their computing resources efficiently. Task scheduling and resource management permit CSP to utilize resources up to their limits and maximize their revenue. In practice, effective task scheduling and resources allocation determine the performance of cloud computing processes. Based on this reason, several research studies have been conducted and many are still ongoing on task scheduling in cloud computing. Task scheduling

* **Corresponding author** Fatima Umar Zambuk ✉ fzambuk2001@yahoo.com ✉ Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi, Nigeria.

is the process of arranging incoming requests (tasks) in a certain manner so that the available resources will be properly utilized. It forms a foundation building block for the efficiency of cloud computing processes in terms of energy consumption and effective resource allocation. This research will focus on effective task scheduling so as to minimize the amount of energy that is being consumed in Cloud Data Centers (DCs).

According to research published in (Muhammad et al., 2019), there has been an estimated rise in power consumption of about 76% from 2017 to 2030 where DC contribution forms a major portion of this increase. As such, this research emphasizes the importance of reducing energy consumption in Clouds. Avgerinou et al. (2017) reported that an average DC consumes as much energy as 25,000 households having an energy bill of about 11.5 billion U.S. dollars which has been forecasted to double for every 5 years. Face to this electronic waste and this huge amount of energy used to power DC's, energy-efficient DC solutions have become one of the greatest challenges. Based on these challenges, providing solutions that scale multiple dimensions and CSP must also deal with the users' requirements which are becoming more and more complex.

Quite a number of heuristic approaches where task scheduling techniques have been reassessed based on deterministic and metaheuristic to address energy-efficient disputes were conducted. Deterministic techniques follow a stringent procedure of defined sequence of operation in solving task solution problems which are oftentimes reportedly inaccurate/ inconclusive, hence gets trapped in a local optimum. Metaheuristics, on the other hand, are sets of algorithmic concepts applicable to a broad set of various problems. They are a set of algorithm framework for general purpose applications in various optimization problems with comparatively little alterations permitting approximate good quality solutions to be found in many hard optimization problems within a relatively short time period. The aim of this research work is to design an energy-efficient framework for task scheduling using Modified Ant Colony Optimization (MACO) principle with divide-and-conquer technique for preempting intensive resources so as to improve the solution for the allocation of resources. Pre-empting intensive task is targeted to be achieved in determining better system responsiveness and scalability.The remaining sections of this paper are structured as follows: Section 2 presents energy-efficient research studies conducted by several other authors as Section 3 presents the proposed method. Section 4 presents results and discussion; finally, conclusions and future directions are presented in Section 5.

2. Review of Related Literature

Most of the existing literature works available on energy efficiency focused on threshold-based VM consolidation utilization strategy mainly on single CPU utilization. However, in Sajitha and Subhajini (2018), power consumption by a server in DC has to do with the available connected components (processor, RAM, hard disk, and bandwidth). Furthermore, most research studies considered only the current resource requirements available neglecting the future utilization during the virtual machine (VM) allocation. As such, a tremendous increase in the rate of service level agreement (SLA) violations occurs in the DC leading to unnecessary VM migrations. As a result, more energy consumption in the DC occurs. This section reviews some of the existing research studies on energy efficiency conducted by several other authors.

A SLA-aware algorithm was proposed in Pagare and Koli (2015) which makes decisions upon detecting overloaded hosts with SLA violations. Their technique also proposed alternatives for finding hosts that are underutilized. Two algorithms were combined together to achieve an energy performance trade-off. The overload detection algorithm looks for hosts that are overloaded and finds the status of the overloaded hosts whether it results in SLA violation or not. The study however does not consider when a host is in the normal state or whether a more priority task arrives that needs not to be delayed. A framework for host-load-categorization in VM consolidation in cloud-based DC was proposed in Alboaneen et al. (2014) to reduce energy consumption while meeting the quality of service requirements. The scheme adopted the underload detection algorithm and classified the underloaded hosts into three states, i.e., underloaded, normal, and critical. The technique balanced load distribution and failed to cater to critical mission tasks. Sharma and Saini (2016) purported a VM consolidation scheme to address energy-performance trade-off and SLA violations. They used a threshold based approach for addressing allocation and reallocation of virtual resources depending upon their load. The median method was also used to find upper and lower threshold values. Motwani and Pangal (2016)proposed the "Adaptive Load Detection Technique" to detect underloaded and overloaded hosts. The famous statistical Inter-Quartile Deviation method was used to adaptively detect load in the DC.

The proposed method evenly distributes task for processing between the hosts in the DC not considering the priority of the tasks to be processed.A resource-aware utility model and VM selection policy to guide VM migration process was proposed in Zhang et al. (2017). Also, Kaur et al. (2017) presented some alternative robust techniques for overload detection and determines an adaptive threshold for CPU utilization. Their technique was compared with other existing techniques in CloudSim simulator. The authors tried to optimize host overload detection. However, they use only CPU utilization, neglecting other critical resources parameters such as memory and network bandwidth. Shaw et al. (2017) proposed an approach for adding a constraint to existing VM consolidation technique to mitigate unnecessary VM migration. A heuristic algorithm was also proposed for VM selection. Sajitha and Subhajini (2018), also proposed a dynamic method which considers multiple factors such as memory, CPU and bandwidth utilization on hosts in a DC. The framework targets to empower VM consolidation by using regression analysis model. The scheme auto adjusts user tasks for processing into three threshold values; lower threshold, middle (prone-to-upper), and upper threshold by adopting the Energy Conscious Dynamic VM Consolidation. However, pre-emption was not considered on the scheme, as such results to wastage of resources leading to energy leakage. Lastly, Wang and Tianfield (2018) proposed a new VM migration selection policy termed as high CPU utilization-based migration host selection and novel VM placement policy, termed as Space Aware Best Fit Decreasing (SABFD) for energy utilization in a DC. The scheme actively categorizes tasks for processing before allocation. However, SABFD makes categorization and allocation but failed to address critical mission tasks that need not to be delayed for allocating VM for their processing or pre-empt tasks during processing.

3. Methodology

The constraint satisfaction rule for energy minimization adopted in Gawali and Shinde (2018) failed to address the issue of task pre-emption which causes excessive energy leakage as a result of not addressing critical tasks which need not to be delayed. The proposed research will focus on improving the rule by injecting the mechanism of divide-and-conquer approach to improve on waiting time, response time and energy efficiency in multi-dimensional spaces.

For sufficient allocation of resources where tasks can be executed by cloud users, each task requires sufficient hardware components (RAM, processor, and bandwidth). For n heterogeneous VMs; vm_1, vm_2, vm_3, …, v_n running in the DC, here, scheduler tries to map T_i task to vm_j VM. The value of decision variable $DV_{i,j}$ if task T_i is matched with VM, vm_j = 1, otherwise = 0:

$$DC_{i,j} \begin{cases} 1, \textit{ if } T_i \textit{ is assigned to } VM_i \\ 0, \text{otherwise} \end{cases} \quad (1)$$

Task request is being initiated by the scheduler when cloud user's demand for cloud services in the DC and assign it to the VM for necessary actions. Decision Controller as can be seen from Equation (1) is 1 for successful assignment, otherwise is 0.

VM receives user job for processing in millions length of instruction. Tasks execution capability Processing retain a DC depends on the quality of hardware resource configurations. The mechanism for space shared here; execution of task will be carried out one after the other implying CPU executes one task per CPU/core. $DC_{i,j}$ will be applicable in waiting queue for remaining in coming tasks for VM assignment or reassignment of tasks that were pre-empted. Remaining time is the overhead (time interval between) when a user job is submitted for VM assignment and the delay experienced before execution starts or time to a pre-empted task to go back to ready queue for VM assignment. When a user job is submitted, it has to be put in ready queue for it to be scheduled for VM assignment or pre-empted task pool.

$$RT_i = WT_i - ET_i \quad (2)$$

Equation (2) defines the resource provisioning remaining time. Here, WT_i is the workload submission time for jobs to be assigned to a VM or the pre-empted task and ET_i is the execution start time for submitted jobs after leaving the ready queue or the continuous time for a pre-empted task. The ET_i as shown in Equation (3) must be less than or equal to the WT_i for task execution to avoid resource contention:

$$ET_i <= WT_i \quad (3)$$

VM execution time is given in Equation (4) for task to be in execution without delay

$$ET_i = \text{Decision Variable } DV_{i,j} * \max T_i \quad (4)$$

To enhance system performance through pipelining, execution time of tasks needs to be speed up. Adequate pipelined system also leads to better system throughput (million number of task request completed per unit

time). As pointed out in Simao et al. (2019), achieving a scalable system also results in better energy performance in a DC by incorporating the smallest amount of extra resources as possible through efficient technique in cloud platform. Scalabl techniques in Cloud also burst traffic and heavy workloads to cope with the voluminous services demanded. Optimization metric used in this research is based on response time and waiting time minimization. Waiting time is the most populous optimization metric for energy efficiency used in cloud computing (Kalra and Singh, 2015) as clients look for the fastest application while CSPs tries to service more users to earn more profit (Sujata, 2017). Waiting time covers the earliest start time on the first task to be assigned to a VM for processing of the last finish time of the last task in the readyº queue.

4. Simulation Environment

Our proposed algorithm was coded and simulated with the java (jdk 1.7) and cloudsim simulator kit where its performance was tested with the bat algorithm (Ullah et al., 2019) and ACO algorithm (Pang et al., 2017). The simulator conducts the performance assessments and the comparison. The objective of the numerical evaluation is to quantify the percentage of three performance matrices; i.e., waiting time, response time, and energy savings or power consumption savings that can be expected when combining the exact allocation algorithm and the consolidation process using our proposed pre-emption algorithm. The answers provided by the numerical analysis concern with the complexity in scalability of the proposed algorithm in terms of size of the DC on the arrival rate of task requests is also synonymous to load on the system. Note, however, that the simulation are conducted for an arrival rate strictly lower than the rate of VM job departures from the system; thus simulations correspond to cases where the likelihood of finding an optimal or a good solution is high.

Table 1 displays the simulation configuration for the scenario used for the proposed research. The simulation scenario include a DC having 10 PMs, 40 VMs where a range of 10–200 tasks length ranging from 20,000–500,000 Million instruction Length (MLi) was issued. Four Linovo PMs used run at 3.0 GHz [CO (TM) i7- 3540MI (R)], 8 GB primary memory and 1 TB secondary storage. The remaining six are HP with 2.7 GHz [CO (TM) i5-2540MIntel (R)], 8 GB primary memory, and 1 TB secondary storage. Each VM runs at 512 MB with a storage capacity of 4 GB that randomly selects PM processor. The Cloudsim version 3. 0 tool kit used combined various features such as containers, modeling of Web applications on multi clouds and VM extensions with performance monitoring. Based on different clouds and VM extensions with performance monitoring. Based on different number of tasks with random length cloudlet (tasks), the simulation was run for more than 3 hours (150 times approximately) and the result was calculated using the policy of space shared in the cloudsim.

5. Analysis and Comparison of Experimental Results

After an in-depth analysis of computation experimental results, 10 VMs executed a set of tasks ranging from 10–30 MLi, 20 VMs executed 20–50 MLi, 30 VMs for 40–100 MLi, and 40 VMs for 80–120.

5.1. Time analysis

Table 2 records the task range and VMs used to obtain waiting time and response time by each algorithm. MACO records the least time as compared to the benchmarked algorithms as can be seen from the table. From Figures 1 to 8, waiting time and response times are recorded in seconds (due to cloudsim relative time unit) from the *y*-axis with the total number of VMs used to process those tasks on the *x*-axis. With 10 VMs running 10 MLi tasks, MACO takes a response time of 620 seconds and waiting time of 480 seconds as compared to ACO that takes 656 seconds response time and 480 seconds response time and bat with 680 seconds response time and 480 seconds waiting

Table 1. Simulation configuration.

	Secondary memory (GB)	Primary memory	Model	Operating system	Bandwidth (bps)	Max power (watts)
PM	1,024	8 GB	Linovo [Intel ®, core(TM) i7-3540 M @3.0 GHz]	Windows 8 64 bit, × 64 based	70,000	190
			HP [AMD Athlon(tm) 64 X2 @ 2.6 GHz]	Windows 10 pro 64 bit, × 64 based	100,000	268
VM	4	512 MB	Red Hat 2.6.24.3		512	

Table 2. Time computation.

No. of VMs	No. of tasks	Bat (seconds)		ACO (seconds)		MACO (seconds)	
		Resp	Wait	Resp	Wait	Resp	Wait
10	10	680	480	656	480	620	480
10	20	699	480	766	480	640	480
10	30	699	480	768	480	601	480
10	40	630	577	802	506	610	480
20	50	739	576	809	1,109	630	480
20	60	744	800	830	1,101	630	**499**
20	70	767	801	920	1,102	630	**499**
20	80	769	810	930	1,111	620	**498**
30	90	777	902	980	1,111	620	**501**
30	100	819	902	750	1,111	630	**501**
30	110	810	1,122	715	1,123	660	**512**
30	120	809	1,122	1,110	1,132	600	**511**
40	130	920	1,132	1,214	1,130	680	**512**
40	150	1,130	1,133	91,422	1,144	590	**532**
40	170	1,232	1,133	1,530	1,145	590	**522**
40	200	1,550	1,143	1,650	1,143	516	532

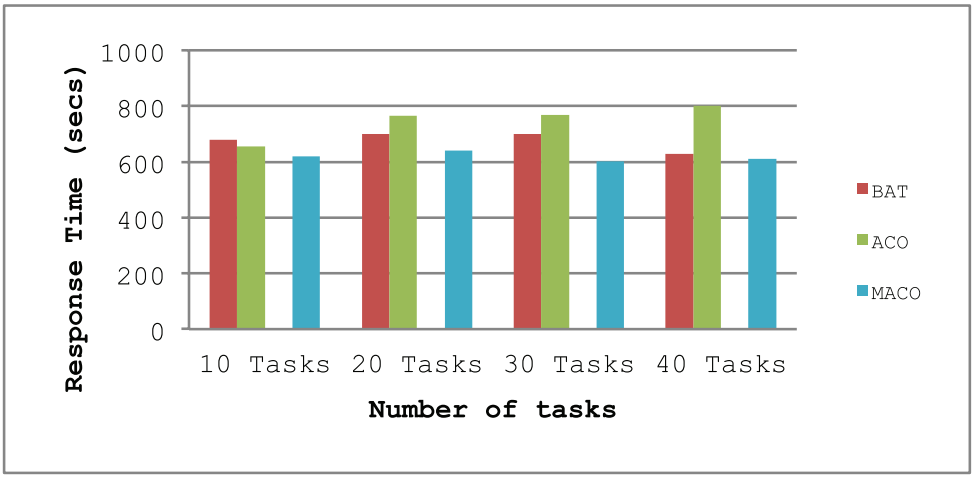

Figure 1. Response time for 10 VMs.

time. With increase in number of tasks, response time and waiting time also increases except for MACO as it normalizes due to its ability to process important task of high priority through the pre-emption on least important tasks. At the last lap of the simulation with the highest number of tasks (200) with 40 VMs, MACO also takes the least time of 516 seconds response time and 532 waiting time as compared to the benchmarked ACO with 1,650 and bat with 1,550 seconds response time and 1,143 seconds waiting time. This clearly shows that our proposed method of task pre-emption ability reduces time (waiting and response) and normalization of task processing.

5.2. Energy analysis

Power consumption results are depicted from Figures 9 to 12 based on comparison between our proposed method and the benchmarked bat algorithm and ACO algorithm. The same simulation scenario comprising of a DC and configuration settings depicted in Table 1 having task request of 10–200 MLi range. The lifetime

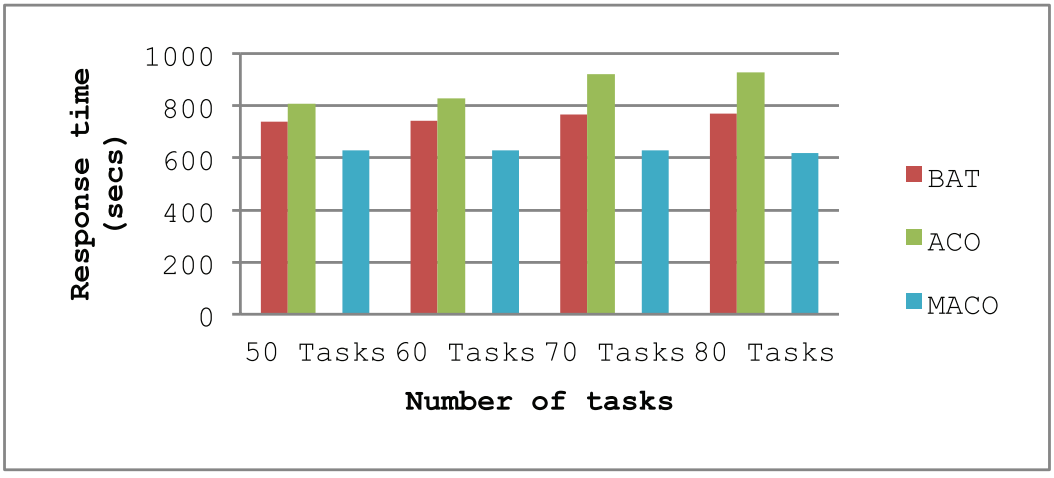

Figure 2. Response time for 20 VMs.

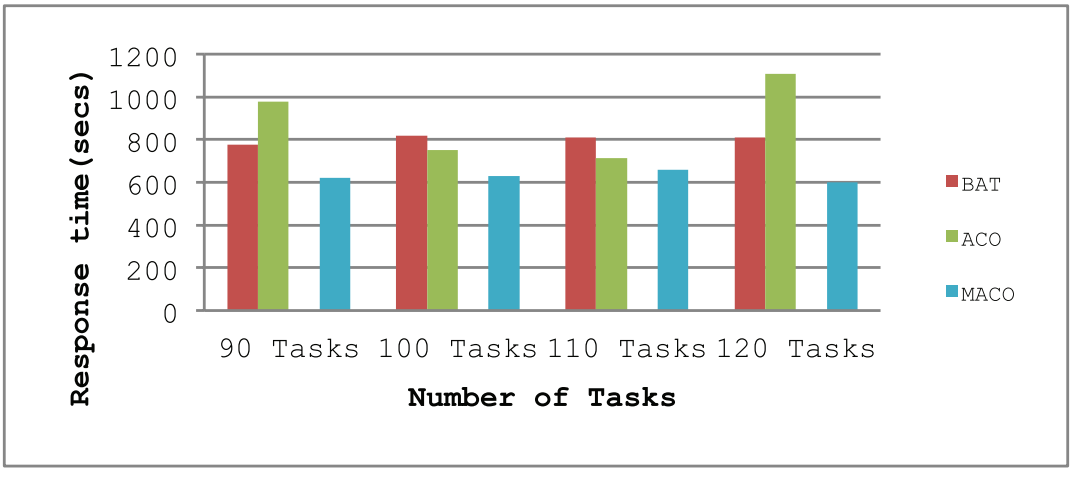

Figure 3. Response time for 30 VMs.

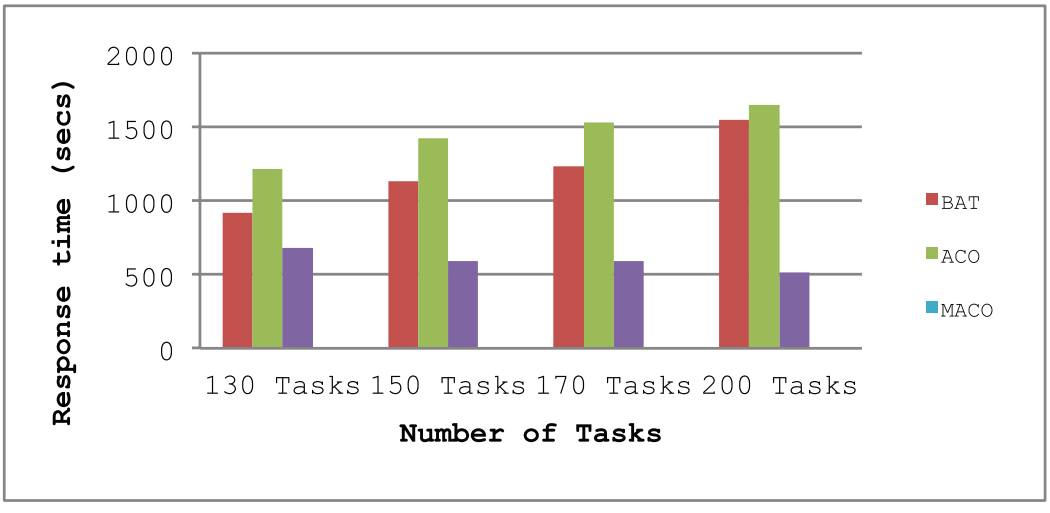

Figure 4. Response time for 40 VMs.

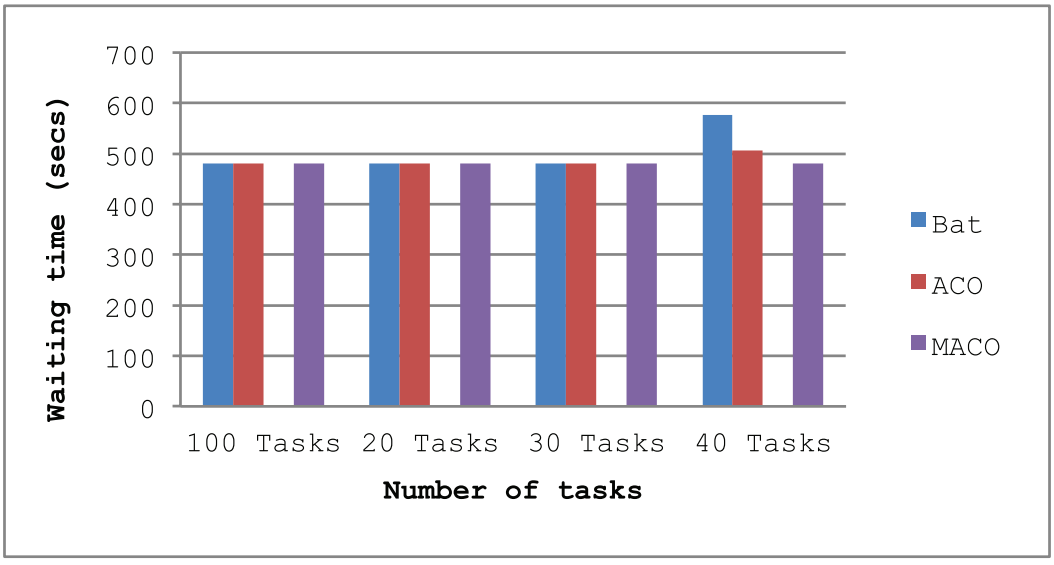

Figure 5. Waiting time for 10 VMs.

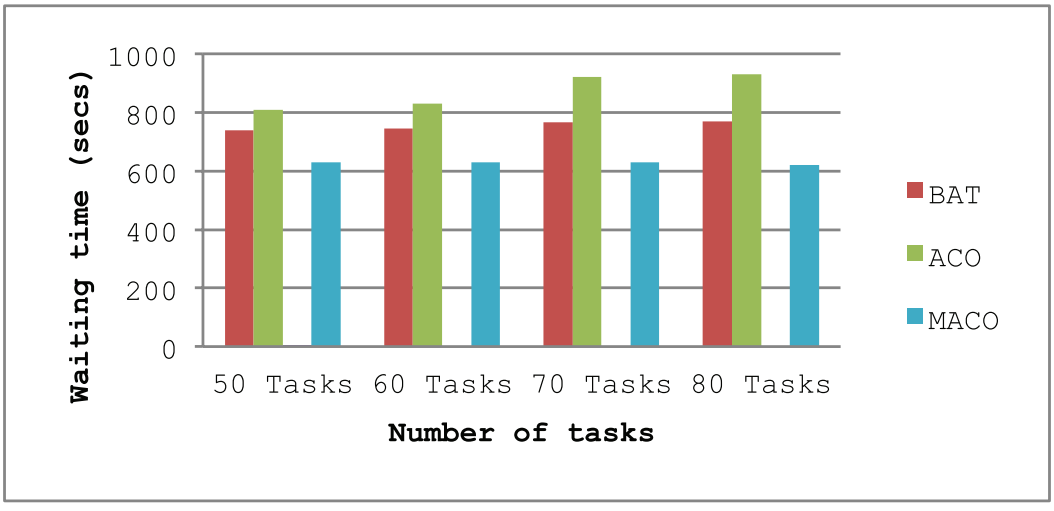

Figure 6. Waiting time for 20 VMs.

Figure 7. Waiting time for 30 VMs.

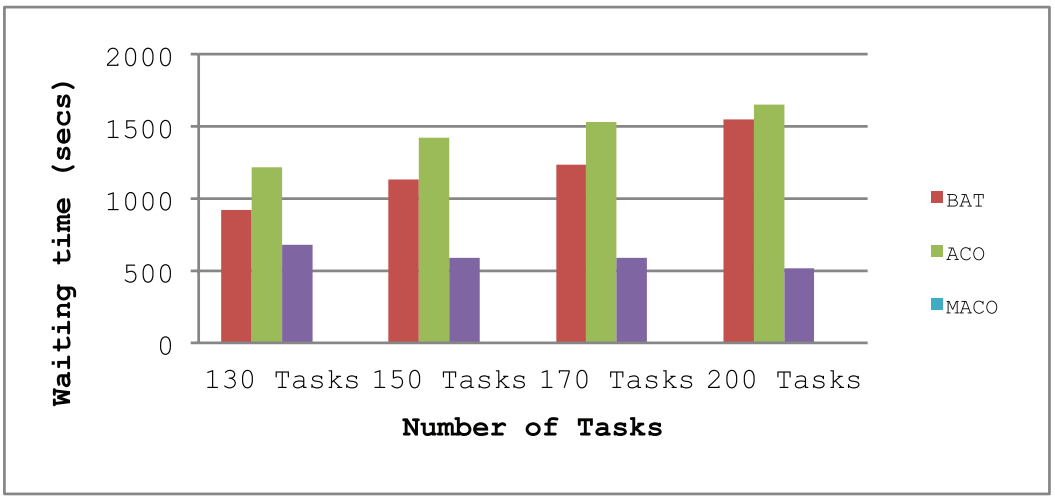

Figure 8. Waiting time for 40 VMs.

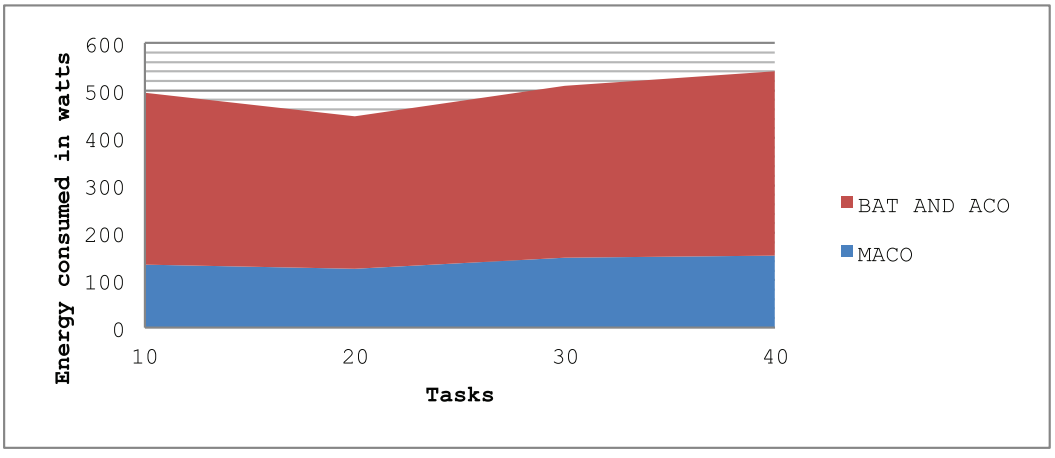

Figure 9. Energy consumed by 10 VMs.

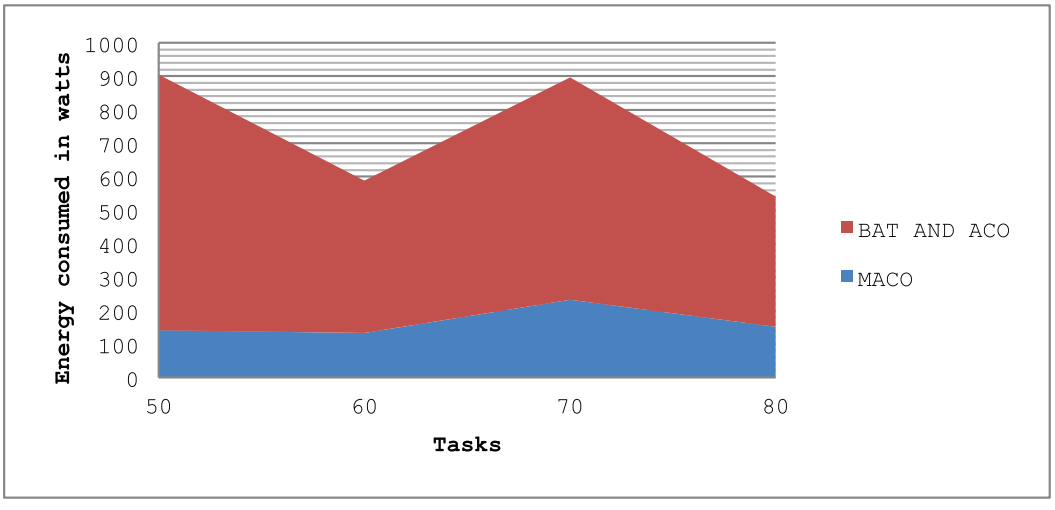

Figure 10. Energy consumed by 20 VMs.

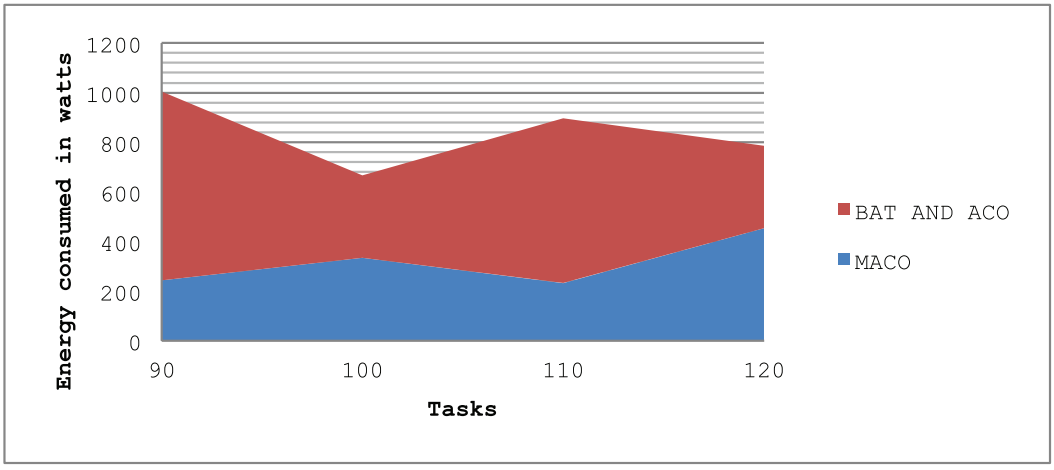

Figure 11. Energy consumed by 30 VMs.

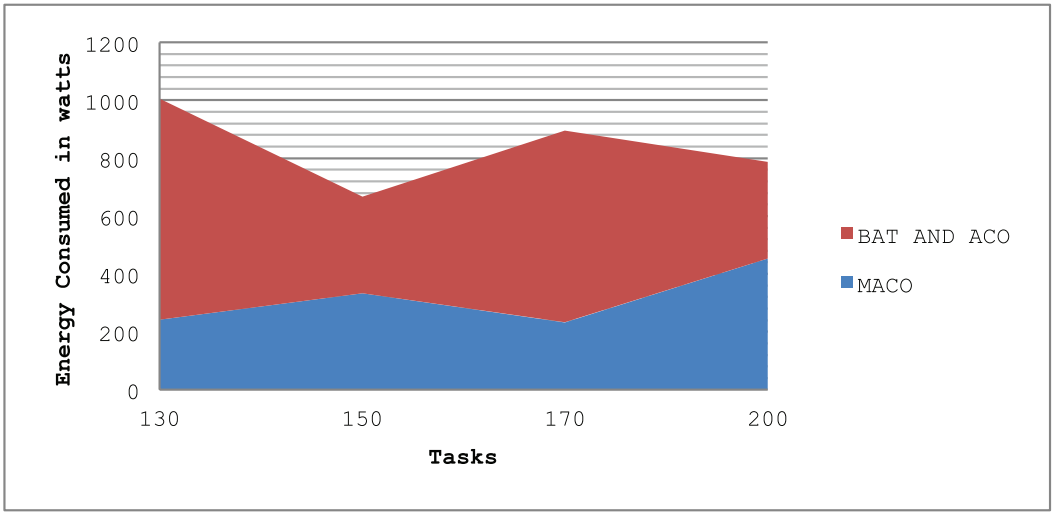

Figure 12. Energy consumed by 40 VMs.

of the VMs is uniform in [30 seconds, 180 seconds]. That is VM jobs last at least 30 seconds and will terminate in less than 180 seconds. The proposed MACO allocation algorithm as expected outperforms the Bat and ACO for over 150 times with respective time interval simulated and reported in the figures. ACO and Bat algorithms consume all PMs or servers (100% ordinate value in the figures) while the proposed algorithm manages to use fewer nodes with 10% to 50% more unused PMs.

Figures 9–12 extend the analysis for the proposed and benchmarked task scheduling algorithms by comparing its performance with and without consolidation. When the two heuristics are combined (bat and ACO) algorithm to perform the pre-emption, (to use migration to empty some nodes) it can significantly provide additional energy consumption reduction or power savings. The average gain can be estimated to be 10% to 20% more PMs that could be turned off. The average line for the exact algorithm is around 80% of PMs used while the average for the exact algorithm with pre-emption is more in the order of 60%.

6. Conclusion and Future Work

This research work demonstrated an alternative way of dealing with resource allocation in cloud computing environment. A hybridized heuristic approach that commingles the MACO and deterministic divide-and-conquer approach to perform resource allocation and task scheduling is presented. The results obtained clearly shows that, the proposed technique is found to be efficient with waiting time, response time and energy consumption as compared with bat and ACO algorithms. Researcher may wish to investigate the possibility of combining the MACO with any other

heuristic algorithms in order to improve resource allocation issues. Since combined (bat and ACO) algorithm to perform the pre-emption, (to use migration to empty some nodes) it can significantly provide additional energy consumption reduction or power savings.

Acknowledgment

This study was supported by the Tertiary Education Trust Fund (TETFund) Institutional Based Research (IBR) Fund, through the Directorate of Research and Innovation of Abubakar Tafawa Balewa University, Bauchi (2018).

References

Alboaneen DA, Pranggono B, Tianfield H. Energy-aware virtual machine consolidation for cloud data centers. In 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, IEEE, London, UK, pp 1010–5, 2014.

Gawali MB, Shinde SK. Task scheduling and resource allocation in cloud computing using a heuristic approach. J Cloud Comput 2018; 7:1–16.

Kalra M, Singh S. A review of metaheuristic scheduling techniques in cloud computing. Egypt Inf J 2015; 16(3):275–95.

Kaur A, Diwakar A, Vashisht R. Alternatives to VM consolidation techniques for energy aware cloud computing. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, Udupi, India, pp 2005–9, 2017.

Motwani RH, Pangal K. Use of error correction pointers to handle errors in memory. Google Patents, 2006.

Muhammad K, Lloret J, Baik SW. Intelligent and energy-efficient data prioritization in green smart cities: current challenges and future directions. IEEE Commun Mag 2019; 57:60–5.

Pagare MJD, Koli, NA. Performance analysis of an energy efficient virtual machine consolidation algorithm in cloud computing. Int J Comput Eng Technol 2015; 6:24–35.

Pang S, Zhang W, Ma T, Gao Q. Ant colony optimization algorithm to dynamic energy management in cloud data center. Math Probl Eng 2017; 2017, Article ID 4810514.

Sajitha A, Subhajini A. Dynamic VM consolidation enhancement for designing and evaluation of energy efficiency in green data centers using regression analysis. Int J Eng Technol 2018; 7:179–86.

Sharma O, Saini H. Vm consolidation for cloud data center using median based threshold approach. Procedia Comput Sci 2016; 89:27–33.

Shaw SB, Kumar JP, Singh AK. Energy-performance trade-off through restricted virtual machine consolidation in cloud data center. In 2017 International Conference on Intelligent Computing and Control (I2C2), IEEE, Coimbatore, India, pp 1–6, 2017.

Simao J, Esteves S, Pires A, Veiga L. GC-wise: a self-adaptive approach for memory-performance efficiency in Java VMs. Future Gener Comput Syst 2019; 100:674–88.

Sujata B. Energy efficient PEGASIS routing protocol in wireless sensor network. Int Res J Eng Tech 2017; 4:18.

Ullah A, Umeriqbal IAS, Rauf A, Usman O, Ahmed S, Najam Z. An Energy-efficient task scheduling using BAT algorithm for cloud computing. J Mech Cont Math Sci 2019; 14:613–27.

Usman MJ, Ismail AS, Chizari H, Gital AYU, Chiroma H, Abdullahi M, et al. Integrated resource allocation model for cloud and fog computing: toward energy-efficient infrastructure as a service (IaaS). Advances on Computational Intelligence in Energy. Springer, Cham, Switzerland, pp 125–45, 2019.

Wang H, Tianfield H. Energy-aware dynamic virtual machine consolidation for cloud datacenters. IEEE Access 2018; 6:15259–73.

Zhang J, Lu X, Panda DK. High-performance virtual machine migration framework for MPI applications on SR-IOV enabled InfiniBand clusters. In 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, Orlando, FL, pp 143–52, 2017.