International Journal on Emerging Technologies **11**(2): 929-934(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Optimized Distributed Resource Allocation for Cloud Computing Applications

Neeru Chauhan[1], Abhay Bansal[2] and Rakesh Matam[3]
[1]Research Scholar, Department of Computer Science and Engineering,
Amity School of Engineering and Technology, Amity University, Noida (Uttar Pradesh), India.
[2]Professor, Department of Computer Science and Engineering,
Amity School of Engineering and Technology, Amity University (Uttar Pradesh), India.
[3]Assistant Professor, Department of Computer Science and Engineering,
Indian Institute of Information Technology, Guwahati (Assam), India.

(Corresponding author: Neeru Chauhan)
(Received 20 January 2020, Revised 19 March 2020, Accepted 21 March 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Increase in digital technology will lead to the server resource allocation task to become more complex day by day. Server architecture checks the job requirement and allocate resources in form of virtual machines (VMs). Allocation of resources is done in such a way that it provides maximum utilization of CPU processing capacity with minimum energy. To execute jobs before their expiry time, system architecture needs to compute resource allocation more precisely. Recent work has shown that the distributed resource allocation gives better results as compared to centralized allocation schemes. Distributed allocation needs migration of tasks between different hosts. One of the main challenges in Distributed allocation is to predict best possible task scheduling with minimum scheduling time. This can be achieved using many ways out of which optimization is one of the methods. Proposed study includes optimization of resource allocation in distributed architecture using ant lion optimization technique to reduce optimization time for task scheduling. Evaluation of result with parameters like CPU and memory usage on benchmarked datasets shows significant difference with state of art methods.

Keywords: Ant lion Optimization, Cloud Computing, Cloud Power Load Balancing, Distributed Cloud Resource Allocation, Elasticity, Live-Migration, Task Schedule Management, Resource Allocation.

Abbreviations: VM, virtual machine, CPU, central processing unit, HA, Host Agent, GA, Global Agent, BFD, Best First Decreasing, CA, Centralized allocation approach, DA, Distributed allocation approach, UDA, Utility Distributed approach, OPT-DA, Optimized Distributed Allocation Approach.

I. INTRODUCTION

Computing services like networking, databases, servers, storages, intelligence and analytics are provided over internet and this whole system is termed as cloud computing [1-4]. With the advancement in the technology, the demand of cloud computing has also increased exponentially. Companies like Google, Amazon and Microsoft provides these clouds to users for various applications as per the requirements. The resources allocated are remotely accessed in the form of virtual machines (VM). The specifications of VM varies user to user depending upon the requirements. These virtual machines are the evolved feature of virtualization of technology [5]. A large demand also increases the energy load at the data centers. This not only affects the cost but also increases the carbon emissions. According to the author [6], 8% of the total world's power will be consumed by the year 2020. There are two architectures of cloud computing, one is centralized and other is distributed resource allocation [7]. The proposed model is based on distributed resource allocation. As the demand is increasing every day, so it is important that cloud computing should be fast and efficient. There are many models presented by various researchers for cloud computing architecture, but they were not effective enough to fasten the system and save the energy simultaneously. This study presents an optimized model of distributed resource allocation for cloud computing in distributed approach. One of the main advantages of this model over the previous models is "to get jobs scheduled in districted network with minimum computational time". Our focus is to make the model fast and efficient in terms of power usage.

This paper is organized as follows:

Section II consists of previous related works. Section III consists of resource architecture allocations which is further divided into two parts, centralized and distributed resource allocation. Section IV consists of optimization structure of the system and fitness function for optimization network. Section V consists of results and discussions.

II. RELATED WORK

In the previous decade, the field of cloud computing has evolved significantly. Many researchers have done some magnificent work in this field but still there is a need of improvement as the demand is still increasing day by day. This section briefly enlightens some previous works done in this field. Studies [3, 8, 9] proposed various algorithms for VM consolidations which were energy efficient. These works also kept the service level agreement violations in check. Wood *et al.*, (2009) proposed that virtual machine consolidation process is nothing but an optimization problem [10].

Nurmi *et al.*, (2009) presented a hierarchy of cloud computing frame work [11]. Farahnakian *et al.*, (2014) discussed the problems caused due to multi agent allocation of resources. They categorized these problems into two categories. In the first category, system checks the status of the host agent, whether it is overloaded or not using RL (Reinforcement Learning) technique and second is the optimization that took place at VM level for migrations [12]. Farahnakian *et al.*, (2014) have proposed a framework for the VM management for multi-agent systems. In the presented architecture the arrangement of the agents was done in three-levels, Global agents, clusters and local agents respectively [13].

Monil and Rahman (2016) proposed an adaptive model on the basis of data. The model judges the status of the host for overloaded and underloaded conditions and then migration is done accordingly [14]. Murtazaev and Oh (2011) proposed an algorithm which was capable for migrating VMs from least loaded nodes to most loaded nodes [8]. Mazrekaj *et al.*, (2017) a model was proposed which was capable of performing migration at runtime. It was based on distributed resource allocation for multi agents. Authors used the utilization function basing on the CPU utilization of the host for live migration [15].

Energy consumption is a great concern for researchers. Bui *et al.*, (2017) provides an optimized model to reduce the energy usage. This model was able to provide acceptable quality of service [16]. Nagamani *et al.*, (2019) suggested a model, which optimized the usage of both virtual and physical machines. This model was able to produce decent results on minimal energy [17].

Prasad *et al.*, (2019) proposed an optimized model for resource allocation. They also compared their results with basic optimization techniques like GA. However, there is still a room for improvement. Our study used ant lion optimization, which was able to produce better results than other optimizations. We also used task completion time as an important variable for fitness value. This not only helped in fastening this model but also managed energy utilization in a better way. As the stress is increasing on cloud computing, it is very much essential to bring up a new model, which not only fast but also helps in reducing the power consumptions. Proposed optimized model shows better results as compared to the state-of-art techniques used in this field [18].

III. TYPES OF RESOURCE ALLOCATION ARCHITECTURES

This section is divided into two parts, centralized and distributed resource allocation. These both are explained as follows:

A. Centralized Allocation Approach

In centralized allocation approach, Global Agent (GA) allocate jobs to Host Agents (HA). Host agents checks its database which consists of running processes in live state or historical data to conclude its saturation level. Fixed threshold value is used to categorize it in two states which are overloaded and underloaded. If host is overloaded, then it passes the status to global agent so that it will not receive any further job to process until the status is overloaded. Host is considered to be overloaded if historical three statuses are above the upper threshold values. Figure 1(a) shows a centralized allocation approach. Virtual Machine Monitor (VMM) allocates VMs to the host using algorithm given by [14] which is based on Best First Decreasing (BFD).

B. Distributed Allocation Approach

Distributed allocation approach can handle large number of processes on cloud. This approach gives the authority to every HA to communicate with another host agent directly and migrates the job if necessary without having permission from GA. Distributed allocation will have N number of HA interlinked with GA.

To handle large number of processes, this network needs optimization of tasks depending upon there priority and resource demands. Each HA has upper and lower threshold values to predict overloaded and underloaded conditions of its host. Live migration of host is possible in this approach. Figure 1(b) shows system architecture of distributed allocation approach. The proposed study uses Ant lion optimization to allocate jobs to HAs.

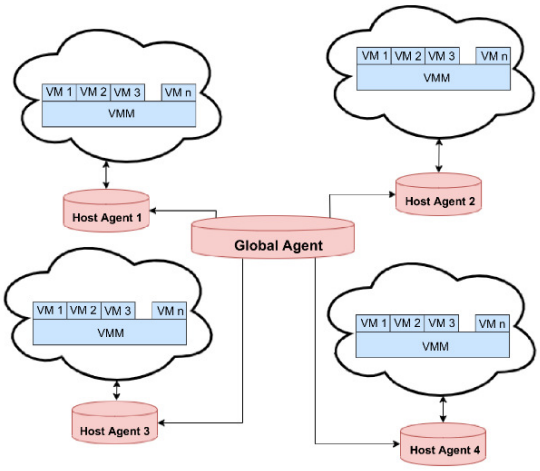


Fig. 1 (a) shows a centralized allocation approach

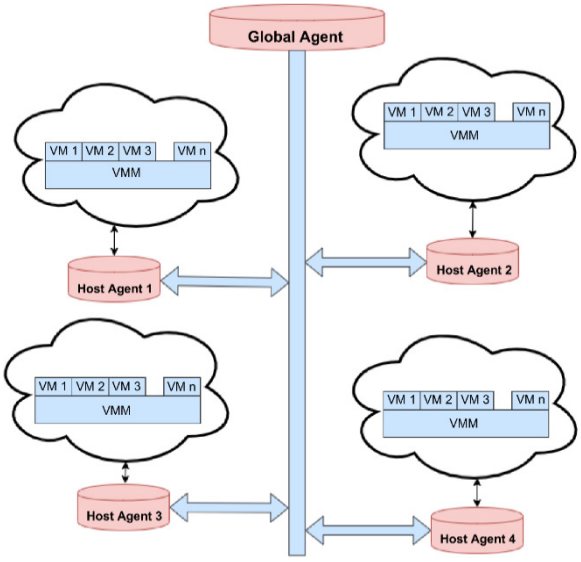


Fig. 1 (b) shows system architecture of distributed allocation approach.

IV. OPTIMIZED DISTRIBUTED ALLOCATION APPROACH (OPT-DA) USING ANT LION OPTIMIZATION

Optimization of resource allocation for processes will be divided in two sections, optimization structure of the system and fitness function for optimization network.

A. Optimization Structure

To maximize the utilization of resources considering all constraints, we propose a novel architecture, which uses Ant lion optimization [19] for task scheduling on Host Agents (HA). Every process has to pass from GA and it prepares a list for time frame (T) which includes processors required, memory requirement, estimated run time and expiry time of the jobs. Number of jobs are directly proportional to T. It's a good practice to not take a large value of T to reduce the complexity of the system.

Global agent has the information of tasks, following are some annotations used to understand the process.

$$T \in \{T_o, T_1 \ldots . T_n\}$$

Where T_o and T_n denotes start and end time of time frame T.

$$J \in \{J_o, J_1, J_2, J_3 \ldots . J_n\}$$

J denotes the number of jobs such that its run time limits are $T_o \leq J \leq T_n$

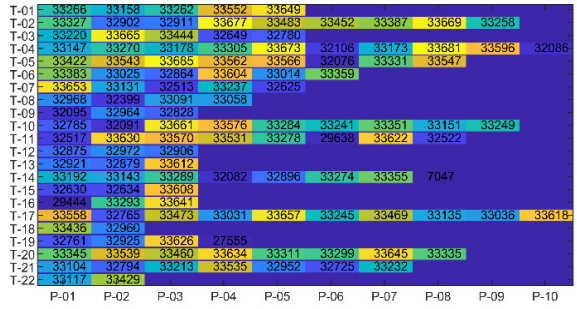
$$PR \in \{PR_1, PR_2, PR_3, \ldots . . PR_n\}$$

Requested processors by job J are taken as PR

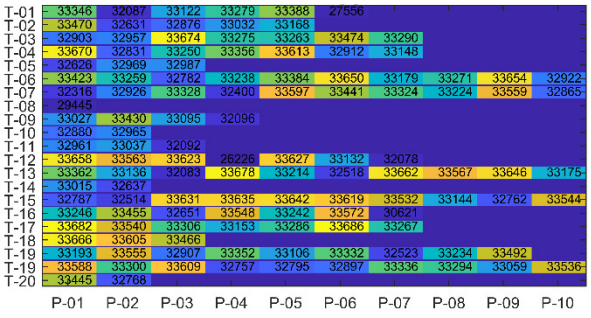
$$M \in \{m_1, m_2, m_n \ldots . m_n\}$$

Requested memory by job J are taken as M

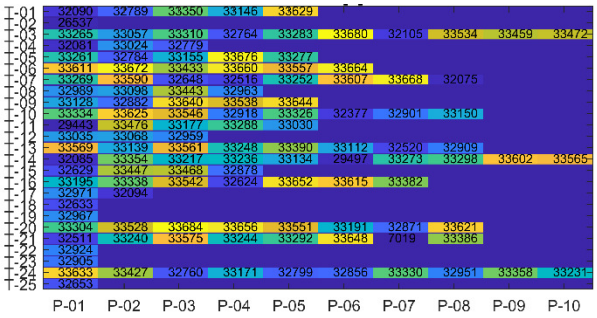
Proposed model uses Ant lion optimization to run on GA to decide which host agent will get which task and also the task scheduling list. It takes some time to make a list which is known as scheduler list (SL) and passes to every host to start the process. Fig. 2(a-d) shows optimized SL for four hosts.



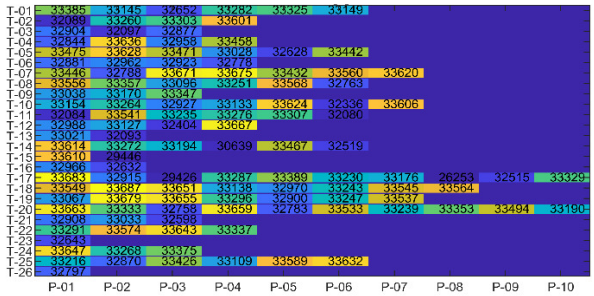
(a) Optimized Task for Host 1.



(b) Optimized Task for Host 2.



(c) Optimized Task for Host 3.



(d) Optimized Task for Host 4.

Fig. 2 (a-d) shows optimized SL for four hosts.

Pseudocode of optimizer is given below:

Algorithm for Global Agent

1. *For i =1 to i = len_T*
2. *P_i= $T_i \leq P \leq T_{i+1}$*
3. *For i =1 to i = host*
4. *Initialize optimization parameters No. of*
5. *Ants Lions, No. of Ants, Max. iterations*
6. *Calculate fitness values of Ant Lions*
7. *Determine best fitness value for iteration=1*
8. *While !(termination condition)*
9. *for k= No. of Antlions*
10. *Selecting Ant lion using Roulette wheel*
11. *Updating of its lower and upper bound*
12. *Update Ant lion*
13. *end for*
14. *for k = No. of Ant lions. Fit(Ant$_k$)*
15. *Calculate fitness value of and Ant lion*
16. *If fit (Ant$_k$) < fit (Ant*$_k$)*
17. *Replace kth ant lion with updated ant*
18. *end if*
19. *end for*
20. *Determine best fitness value*
21. *end while*
22. *Save job scheduler for ith host*
23. *End for*

$$P_r = Rand\{\text{pending jobs for } i^{th} \text{ host}\}m$$

$$T_{max} = \begin{cases} T_{max} = T\{P_{rj}\}\ if & T_{max} < T\{P_{rj}\} \\ \text{else} & \text{do nothing} \end{cases}$$

B. Fitness Function

Fitness function plays an important role in any optimization method, better the relation between fitness variables and problem, better will be the output achieved. In this study, we assume that GA has the data of jobs which involves number of processors required, amount of memory needed and expected run time.

Below are the equations of fitness function based on these known variables.

$$fit\_val = \left(\frac{\sum_{k=1}^{k=T_{max}} (CPU_{max} - CP_u - Pr_j}{T_{max}} \right)$$

Where m is random number of jobs such that 0 ≤ m ≤ 10. Maximum 10 jobs are allowed to process in parallel for one host. $T\{P_{rj}\}$ denotes runtime for j[th] job. CPU_{max} is the upper threshold value of CPU for a host agent.

V. EXPERIMENTS RESULTS AND DISCUSSION

A. Evaluation Parameters

Proposed study uses three performance evaluation parameters, which are discussed below.

—CPU usage: CPU usage [15] also known as CPU load, this is defined as the total CPU usage of a HA when processing the jobs. This study involves two threshold levels, upper and lower.

CPU usage below the lower threshold level (LT) is considered to be underloaded and CPU usage above the upper threshold (UT) will be considered as an overloaded situation. Accounting that each host should have CPU usage, which is $LT \leq \text{cpu usage} \leq UT$.

— Memory consumption (MC): Another parameter is memory consumption [20] by the jobs on a HOST. MC can also be taken as the constraints to limit over usage of the server. If a host is in the condition of upper limits of MC, then GA should not allocate more jobs to that host to avoid negative effects on system's performance.

—Run Time (RT): This is the most important parameter for resource allocation for any network. Every network architecture tends to process the jobs within their expiry time limit to save job from failure and prevent it from live migration to reduce extra energy consumption. Lesser the RT [18] value of the host, better is the scheduler algorithm.

B. Results

This section covers the experimental results of proposed model OPT-DA compared with centralized allocation approach (CA), distributed allocation approach (DA) and utility distributed approach (UDA) [15].

Centralized allocation approach (CA): In this approach a communication is performed between GA and HAs as shown in the figure 1(a). HA checks the overloaded and underloaded conditions from history. If last three history is overloaded and underloaded then it is considered as overloaded and underloaded respectively. This status report is then sent to GA. When VM is considered as overloaded then GA stops taking further jobs for HA and if VM is underloaded then GA performs live migration of job.

Distributed allocation approach (DA): In this approach HAs are capable to contact other HAs directly without any involvement of GA as shown in figure 1(b). When host finds HA to be overloaded then HA holds all the upcoming jobs for the VMs. When host finds out HA to be underloaded then HA performs live migration with other HA on the basis of availability of resources. This approach is considered to be faster than centralized allocation approach.

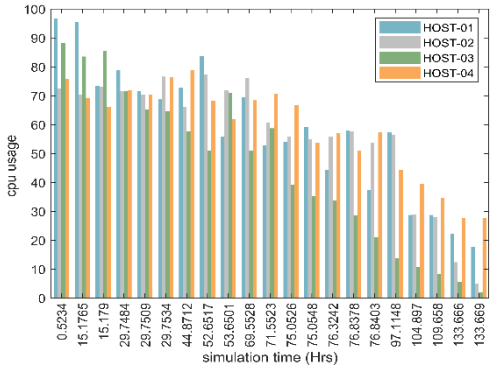
Utility distributed approach (UDA):UDA is considered to be more stable and faster than above given approaches. In this approach live migration is based upon the utility factor of last four history. The trend of utility factor allows VM to make decision of performing live migration or not.

Dataset HPC2N [21] is used to evaluate the results. We have taken first 500 jobs to run with above-mentioned approaches and compared its parameters. Attributes like processors required, memory required and start stop times have been used from the dataset.

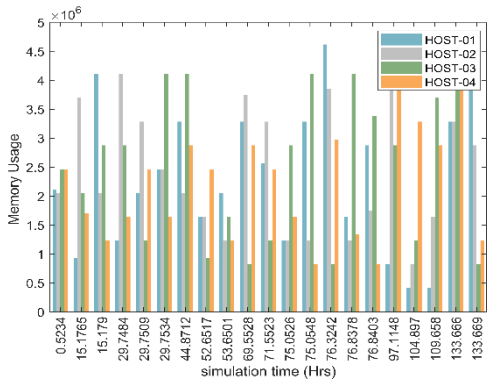
Table 1: Comparison of OPT-DA with other techniques.

Sub Datasets	CA				DA				UDA				OPT-DA			
	Avg. CPU (%)	Avg. MC (10^6)	Total (RT) Hrs	Total Migration	Avg. CPU (%)	Avg. MC	Total (RT) Hrs	Total Migration	Avg. CPU (%)	Avg. MC	Total (RT) Hrs	Total Migration	Avg. CPU (%)	Avg. MC	Total (RT) Hrs	Total Migration
DS-1	19.58	1.69	427.82	0	32.43	2.01	232.99	04.0	36.78	2.11	200.29	12.00	54.66	2.35	133.66	04.00
DS-2	23.61	1.65	442.13	0	33.10	2.05	377.71	10.0	39.14	2.30	268.89	09.00	48.78	2.31	251.00	08.00
DS-3	19.65	1.77	442.58	0	26.10	2.24	275.93	18.0	32.80	2.34	214.77	10.00	42.66	2.22	245.44	09.00
DS-4	16.15	2.18	179.29	0	21.21	2.61	141.13	30.0	27.40	2.72	126.22	18.00	29.28	2.64	92.62	24.00
Average	**19.74**	**1.82**	**372.95**	**0**	**28.21**	**2.22**	**256.94**	**15.5**	**34.03**	**2.36**	**203.03**	**12.25**	**43.84**	**2.38**	**18.68**	**11.25**

Ant lion optimization uses 1000 iterations with 20 Ant lions as input parameter to optimize task scheduling over the HA. Four hosts are taken into account with 10 VM for every host for this study. Figure 2(a-d) shows OPT-DA jobs scheduled over four host along with their job ids. Y axis shows the time frames for jobs running in parallel, though every time frame is different but to show as a heat map, we used annotation $T_1, T_2 \ldots\ldots, T_n$ X axis represents VM running for different jobs. Fig. 3 (a-b) shows results of OPT-DA for HPC2N dataset. As mentioned above, proposed scheme gets compared with state of art as discussed in Table 1. Four random sub datasets are taken from HPC2N to evaluate system performance having 500 jobs each, these are represented with DS1, DS2, DS3 and DS4.



(a) CPU usage vs simulation time for DS-1 dataset.



(b) Memory usage vs simulation time for DS-1 dataset

Fig. 3.

VI. CONCLUSION

In the paper, bio-inspired ant lion optimization is used, this approach is used for resource allocation in cloud computing. Evaluation results show that the migration of hosts due to overloading did improve by 11.30% for OPT-DA over UDA approach, whereas run time for jobs for the taken dataset was lowered by 23hrs over the cloud, which enhances the power consumption of the server directly. Moreover, two implications have been made after evaluating the results, firstly, it was observed that CPU usage is increased in OPT-DA with same memory consumptions, which showed better load balancing over the server and secondly, decrease in optimization time for job scheduling. Study concludes that using OPT-DA before allocating jobs to host agent (HA) improves overall performance over traditional approaches like UDA, DA and CA, which uses only threshold values.

VII. FUTURE SCOPE

This work includes parallel optimization for HA jobs that can be taken as a base study for future work to reduce the optimization task over GA. Hybrid optimization techniques can be used as a future work to further optimization of parameters. Another possibility is to work on migration optimization of jobs, as we have used threshold value to migrate live host in OPT-DA. However, cloud computing is a field which is not yet completely explored. There are various other perspectives in this field, which require further evaluation. This unravelling is likely to pose challenges as well. One of the major challenges that we can already anticipate is dealing with “big data”.

ACKNOWLEDGEMENTS

The authors are grateful to the Department of Computer Science and Engineering, School of Engineering and Technology, Amity University, Noida (UP), for providing facilities for completion of this work.

Conflict of Interest. There is no conflict of interest.

REFERENCES

[1]. Sharkh, M. A., Jammal, M., Shami, A., & Ouda, A. (2013). Resource allocation in a network-based cloud computing environment: design challenges. *IEEE Communications Magazine*, *51*(11), 46-52.

[2]. Omara, F. A., Khattab, S. M., & Sahal, R. (2014). Optimum resource allocation of database in cloud computing. *Egyptian Informatics Journal*, *15*(1), 1-12.

[3]. Marzolla, M., Babaoglu, O., & Panzieri, F. (2011). Server consolidation in clouds through gossiping. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (pp. 1-6). IEEE.

[4]. Kolhar, M., El-Atty, S. M. A., & Rahmath, M. (2017). Storage allocation scheme for virtual instances of cloud computing. *Neural Computing and Applications*, *28*(6), 1397-1404.

[5]. Selim, G. E. I., El-Rashidy, M. A., & El-Fishawy, N. A. (2016). An efficient resource utilization technique for consolidation of virtual machines in cloud computing environments. In *2016 33rd national radio science conference (NRSC)* (pp. 316-324). IEEE.

[6]. Walley, N., & Whitehead, B. (1994). It's not easy being green. *Reader in Business and the Environment*, *36*(81), 211-222.

[7]. Endo, P. T., Sadok, D., & Kelner, J. (2011). Autonomic cloud computing: Giving intelligence to simpleton nodes. In *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, 502-505. IEEE.

[8]. Murtazaev, A., & Oh, S. (2011). Sercon: Server consolidation algorithm using live migration of virtual machines for green computing. *IETE Technical Review*, *28*(3), 212-231.

[9]. Feller, E., Morin, C., & Esnault, A. (2012). A case for fully decentralized dynamic VM consolidation in clouds. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, 26-33.

[10]. Wood, T., Shenoy, P., Venkataramani, A., & Yousif, M. (2009). Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, *53*(17), 2923-2938.

[11]. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., & Zagorodnov, D. (2009). The eucalyptus open-source cloud-computing system. In *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 124-131.

[12]. Farahnakian, F., Pahikkala, T., Liljeberg, P., Plosila, J., & Tenhunen, H. (2014). Multi-agent based architecture for dynamic VM consolidation in cloud data centers. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications,* 111-118.

[13]. Farahnakian, F., Liljeberg, P., Pahikkala, T., Plosila, J., & Tenhunen, H. (2014). Hierarchical vm management architecture for cloud data centers. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, 306-311.

[14]. Monil, M. A. H., & Rahman, R. M. (2016). VM consolidation approach based on heuristics, fuzzy logic, and migration control. *Journal of Cloud Computing*, *5*(1), 1-18.

[15]. Mazrekaj, A., Minarolli, D., & Freisleben, B. (2017). Distributed resource allocation in cloud computing using multi-agent systems. *Telfor Journal*, *9*(2), 110-115.

[16]. Bui, D. M., Yoon, Y., Huh, E. N., Jun, S., & Lee, S. (2017). Energy efficiency for cloud computing system based on predictive optimization. *Journal of Parallel and Distributed Computing*, *102*, 103-114.

[17]. Nagamani, T. S., Lakshmi, K. V., & Bhavani, B. L. (2019). A new dynamic and enhanced resource allocation algorithm in cloud computing. In *Journal of Physics: Conference Series* 1-8. IOP Publishing.
[18]. Prasad, V. K., Nair, A., Tanwar, S., & Tracking, V. (2019). Chapter 10 : Resource Allocation in Cloud Computing.
[19]. Mirjalili, S. (2015). The ant lion optimizer. *Advances in engineering software*, *83*, 80-98.
[20]. Bronson, F. A. X., Rajagopalan, S. P., & Raja, V. S. S. (2018). A Dynamic Memory Allocation Strategy for Virtual Machines in Cloud Platform, 1423–1444.
[21]. Madni, S. H. H., Latiff, M. S. A., Abdullahi, M., Abdulhamid, S. I. M., & Usman, M. J. (2017). Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. *PloS one*, *12*(5), 1-26.

How to cite this article: Chauhan, N., Bansal, A. and Matam, R. (2020). Optimized Distributed Resource Allocation for Cloud Computing Applications. *International Journal on Emerging Technologies*, *11*(2): 929–934.