Best-Fit Virtual Machine Placement Algorithm for Load Balancing in a Cloud Computing Environment

Fale Mantim Innocent, Mwanret Alphonsus, Lar Nansel, Edwin Freedom Titus, Abdulmalik Dashe

Abstract— In Cloud Computing, load balancing is key to the maximization of profit as load imbalance leads to excessive power consumption and inefficiency in terms of computational power. Load Balancing goes in two directions: firstly, it is required to distribute tasks across Virtual Machines; and secondly, to efficiently place Virtual Machines on physical servers or physical machines such that resource and energy consumption is minimized. This paper proposes a new Virtual Machine Placement algorithm: 'Best-Fit Virtual Machine Placement Algorithm'. This algorithm computes tasks resource demands, models a Virtual Machine that fits those demands, and places the Virtual Machine on a physical server that has the minimum remaining resources that is large enough to accommodate such a Virtual Machine. This technique ensures that resources are not over-utilized nor are they under-utilized. Simulated Data was obtained using CloudSim and Microsoft Visual Studio's C#.NET was used to experiment and test the performance of this algorithm side-by-side its counterparts: 'Multi-Objective Ant Colony System Virtual Machine Placement Algorithm', 'Max-BRU' and 'Enhanced Firefly Algorithm'. The result of this experiment shows that the proposed 'Best-Fit Virtual Machine Placement Algorithm' is not only efficient but competitive.

Index Terms— Best-Fit, Cloud Computing, Load Balancing, Virtual Machine Placement

1 Introduction

Any system providing access to processing power, storage, software, or other computing services via the internet is referred to as a Cloud [1]. Typically, a cloud could be Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS), or Software-as-a-Service (SaaS). The services provided by the cloud are on rental basis – clients pay as they use any of these services. This, of course, is a great idea but it accompanies with it its own challenges. The major issue faced with cloud computing is traffic and this is caused mainly by load imbalance. The technique for handling load imbalance is called Load Balancing. Load balancing is a method used for distributing workload on multiple computers or a computer cluster through network links to achieve optimal resource utilization which maximizes throughput and minimizes overall response time [2]. Load Balancing goes in two directions: Task Scheduling; and Virtual Machine (VM) placement [3] [4] [5] [6] [7] [8] [9]. A VM abstracts a physical machine (PM) via some software means (e.g. processor speed, memory capacity, and disk size). Some approaches for solving VM placement problem are: Constraint Programming; Stochastic Integer Programming; Bin Parking; and Genetic Algorithms [6]. The problem with the aforementioned approaches is that they lack direct implementation on the cloud infrastructure. This research proposes an algorithm called "Best-Fit VM Placement Algorithm". See Sections 2 and 4 for a brief review of the algorithm and for description of the algorithm respectively.

Section 3 of this work covers 'SYSTEM MODEL', Section 5 presents 'EXPERIMENTS AND EVALUATION', Section 6 covers 'CONCLUSION', and Section 7 is 'REFERENCES'.

2 Literature Review

When a virtual machine is deployed on a host, the process of selecting the most suitable host for the virtual machine is known as virtual machine placement, or simply, placement. During placement, hosts are ranked based on the virtual machine's resource requirements and the projected usage of resources. Host ranking also take into consideration the placement objective: either resource maximization on distinct hosts or load balancing between hosts. The administrator selects a host for the virtual machine based on the host rankings [10]. Automatic Placement is the process used to automatically place a VM on the most suitable host. This occurs either in self-service or VM migration. In VM self-service, users' VMs are automatically placed on the most suitable host in the self-service host group. Automatic Placement is also possible in the drag-and-drop method which is used to migrate a VM to a host group in VM view. VM configuration files are moved to the volume judged to be the most suitable on the selected host during automatic placement. A few VM placement algorithms that address the same need but uses different approaches shall be reviewed in the following paragraphs.

'A Multi-Objective Ant Colony System Algorithm for Virtual Machine Placement in Cloud Computing' [8] is a load balancing algorithm which aims at optimal placement of virtual

- *Fale Mantim Innocent is with Federal College of Education, Pankshin, Plateau State, Nigeria. Email: thefmicorporation@gmail.com*
- *Mwanret Alphonsus* **is** *with Federal College of Education, Pankshin, Plateau State, Nigeria. Email: mwanretalphonsus @yaoo.com*
- *Lar Nansel is with Federal College of Education, Pankshin, Plateau State, Nigeria. Email: larnansel@gmail.com*
- *Edwin Freedom Titus is with Federal College of Education, Pankshin, Nigeria. Email: drumlineroyalbeat@gmail.com*
- *Abdulmalik Dashe is with Federal College of Education, Pankshin, Nigeria. Email: abdulmalikdashe@gmail.com*

machines. This is important for improving power efficiency and resource utilization in a cloud computing environment. Its goal is to efficiently obtain a set of non-dominated solutions (the Pareto set) that simultaneously minimize total resource wastage and power consumption [4] [8]. This algorithm is metaheuristic. It is inspired by the observation of real ant colonies and it is based upon their collective foraging behavior.

'A virtual machine placement algorithm for balanced resource utilization in cloud data centers' [7] proposes an algorithm called Max-BRU. This algorithm is based on multiple resource-constraint metrics that help to find the most suitable server for deploying VMs in large cloud datacenters. Its drive is that most algorithms consider a limited number of resource types, thus resulting in unbalanced load or results in the unnecessary activation of physical servers [7].

'Energy-efficient virtual machine placement using enhanced firefly algorithm' is a load balancing algorithm. It addresses VM placement issues by proposing two meta-heuristic algorithms namely, the enhanced modified firefly algorithm (MFF) and the hierarchical cluster based modified firefly algorithm (HCMFF) [3]. In this paper, [3] poses a claim that many algorithms lack the use of exploitation mechanism efficiently.

'Best-Fit VM Placement Algorithm' is an automatic VM placement algorithm. The best-fit problem, like the knapsack problem, is a combinatorial optimization problem. This algorithm makes use of tasks requirements to model a VM to execute such tasks. It chooses the maximum tasks $\text{CPU}_{\textit{speed}}$, $\text{RAM}_{\textit{capacity}}$, $\text{DISK}_{\textit{size}}$, and $\text{NET}_{\textit{bandwidth}}$ as requirements for VM. For each newly created VM, it gets all activated Physical Machines (PMs), identifies the PM with available processor/core and marks them as candidate PMs. It then chooses the PM that has the least available resources that can match the requirements of the VM and then assigns the VM to such a PM. If no PM has resources to match the VM's requirements, a new PM is activated and the process starts all over again.

Let's take a scenario to better understand the Best-Fit Virtual Machine Placement Algorithm. There is one datacenter consisting of four PMs that have the same capacity and four tasks with varying requirements. The proposed algorithm is expected to: model a VM to execute these tasks; and automatically place the VM on a PM such that resources are efficiently utilized. It is also assumed that PM1 is already executing three VMs with the following resource requirements: VM1 <$\text{CPU}_{1.5\text{GHz}}$, $\text{RAM}_{200\text{MB}}$, $\text{DISK}_{50\text{GB}}$, $\text{NET}_{100\text{Mbps}}$>, VM2 <$\text{CPU}_{2.5\text{GHz}}$, $\text{RAM}_{100\text{MB}}$, $\text{DISK}_{150\text{GB}}$, $\text{NET}_{50\text{Mbps}}$>, and VM3 <$\text{CPU}_{3.5\text{GHz}}$, $\text{RAM}_{100\text{MB}}$, $\text{DISK}_{200\text{GB}}$, $\text{NET}_{50\text{Mbps}}$>. And PM2 is executing two VMs with the following requirements: VM1 <$\text{CPU}_{1.0\text{GHz}}$, $\text{RAM}_{100\text{MB}}$, $\text{DISK}_{20\text{GB}}$, $\text{NET}_{100\text{Mbps}}$> and VM2 <$\text{CPU}_{1.5\text{GHz}}$, $\text{RAM}_{50\text{MB}}$, $\text{DISK}_{100\text{GB}}$, $\text{NET}_{50\text{Mbps}}$>. The maximum tasks resource requirements is used to model the new VM resource requirement since this VM is to execute these tasks efficiently. Task 2 has the maximum CPU requirement; Task 2 has the maximum RAM requirement; Task 1 has the maximum DISK requirement; and Task 1 has the maximum network requirement. These resource requirements were used to model the VM to execute the four tasks (see *Fig. 2*).

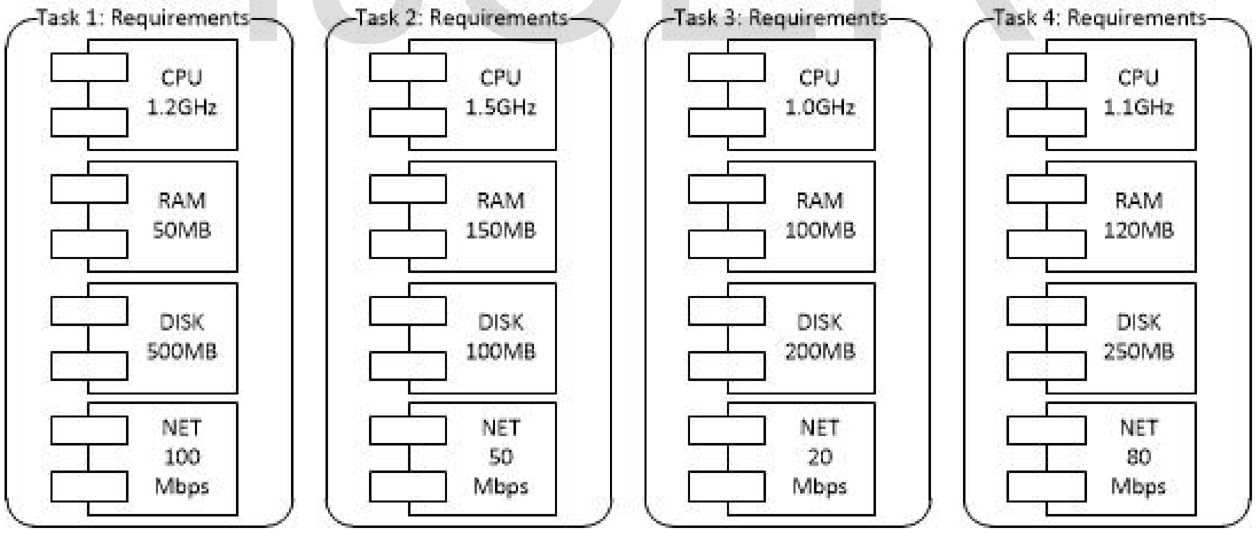

FIG. 1: DIAGRAM SHOW FOUR TASKS WITH THEIR RESOURCE REQUIREMENTS

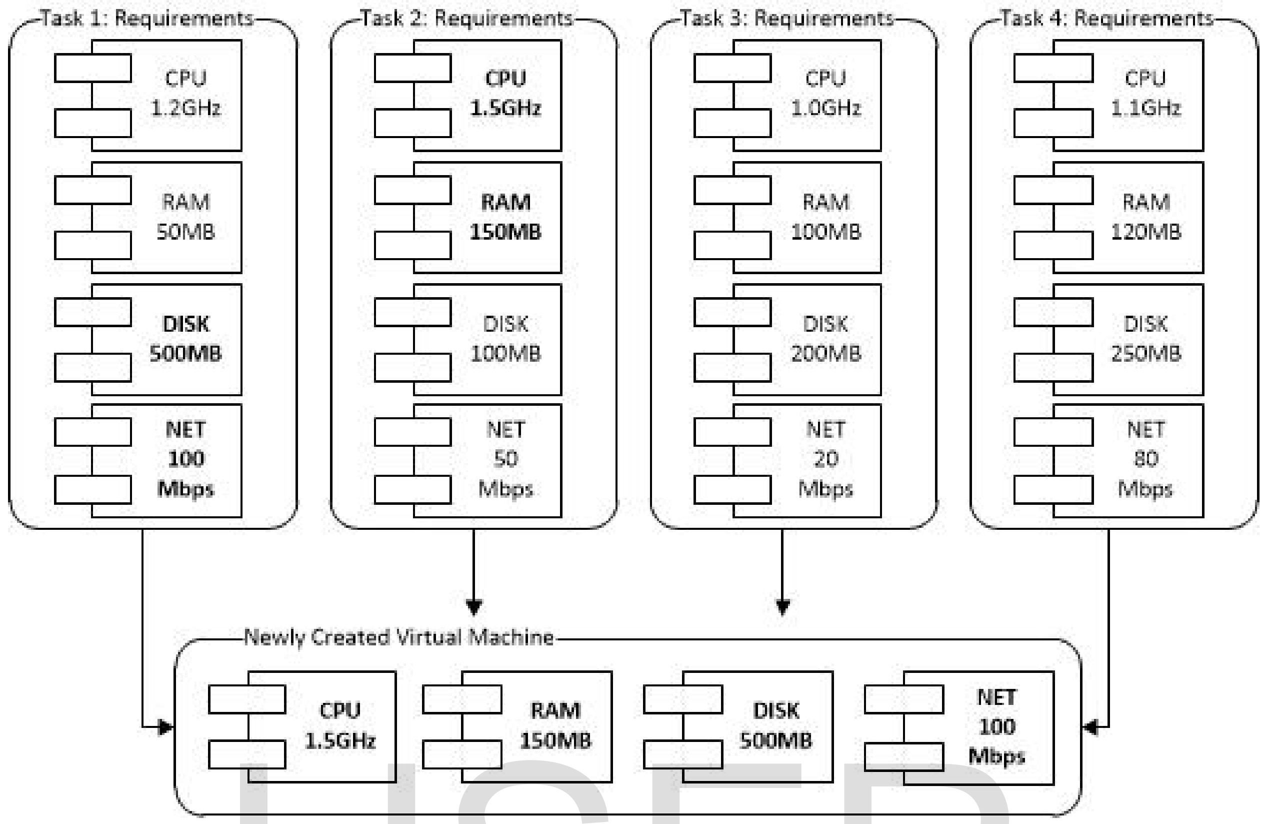

FIG. 2: DIAGRAM SHOWING NEWLY CREATED VM USING MAXIMUM TASKS REQUIREMENTS

The new VM requirement was tested against resource availability in accordance to the workability of the algorithm. PM1 and PM2 have been activated and are running 3 and 2 VMs respectively. They both have Quad Core processors, meaning that PM1 still has one available core to execute an additional VM while PM2 has two available cores to execute two additional VMs. The capacity of each core is 3.5GHz which is the smallest available core/processor in the activated category but large enough to execute the VM. The resources being consumed by the VMs being executed on either machines is a simple summation of the VMs requirements excluding CPU and NET because a Core is either available or not while NET is a shareable resource. For PM1, the RAM being consumed by the 3 VMs amounts to 400MB while the DISK being consumed totals 400GB. Hence, the RAM, DISK, and NET capacity of PM1 is very sufficient to accommodate the requirements of the new VM. For PM2, the RAM being consumed by the two VMs amounts to 150MB while the DISK being consumed totals 120GB. The RAM, DISK, and NET capacity of PM2 is also very sufficient to accommodate the requirements of the new VM. The new VM is then placed on PM1 because it has the least available resources that are large enough to match the requirements of the VM (see *Fig. 3*). In Section 5, the proposed algorithm, alongside side its competitive counterparts, will be subjected to more complex scenarios where a posteriori analysis will be performed in order to compare results and ascertain the most efficient of them all.

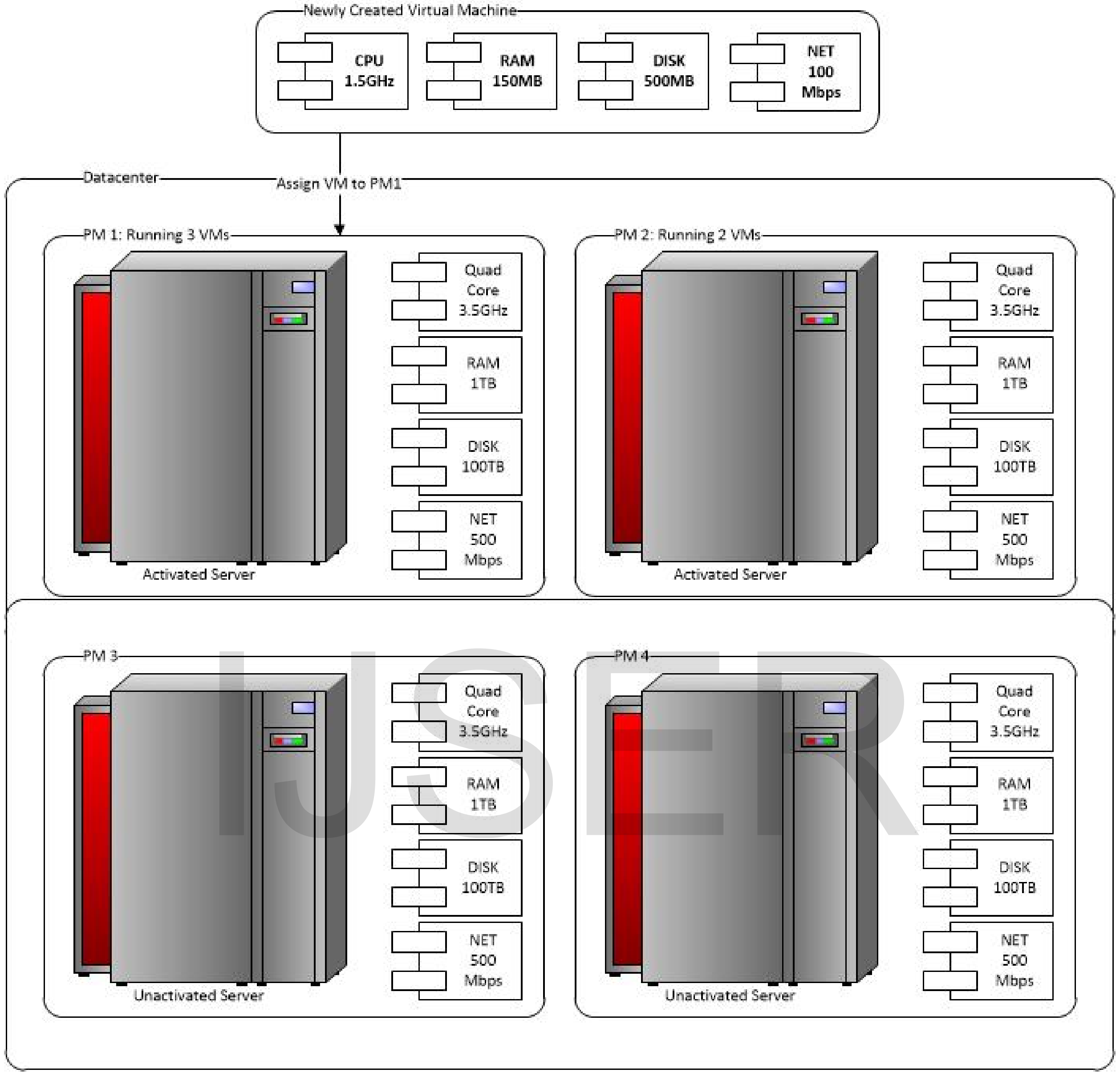

FIG. 3: DIAGRAM SHOWING HOW NEWLY CREATED VM GOT PLACED ON PM1

3 SYSTEM MODEL

Task Model

Definition 1 Each time a VM executes a task, the task does consume resources. This task resource consumption is in accordance with the availability of resources at its respective node (VM).

$T_i = <CPU_{speed}, RAM_{capacity}, DISK_{size}, NET_{bandwidth}>$ (1)

Resource Model

Definition 2 The resources required by each VM to exe-cute
tasks assigned to it is given by the relationship below. The CPU_{speed}, $RAM_{capacity}$, $DISK_{size}$, and $NET_{bandwidth}$ of the VM is the maximum requirement of the tasks assigned to such VM.

$$VM_{iresource} = Max(<T_{ij}CPU_{speed}, T_{ij}RAM_{capacity}, T_{ij}DISK_{size}, T_{ij}NET_{bandwidth}>) \quad (2)$$

4 BEST-FIT VM PLACEMENT ALGORITHM

STEP 1: Sort task requirements (i.e. tasks to be executed by VM) in descending order of RAM, then DISK.
STEP 2: Model a VM such that the resource requirement of task T_0 becomes the resource requirement of the VM. (This is in conformance with *Equation 2*).
STEP 3: For every newly created VM, do the following:

- Get all activated PMs.

- Identify PMs with available processor/core to execute VM and call them candidate PMs.
- Sort candidate PMs in ascending order of available RAM, then DISK.
- Choose candidate PM$_i$ which has the least available resources that are large enough to match the requirements of the VM.
- Assign VM to such PM.
- If VM has no matching PM, activate new PM and go to step 3.1.

STEP 4: For every completed task T$_i$, do the following:

- Remove task from VM.
- Repeat STEPS 1 to 4 until VM completes execution of all tasks.

STEP 5: Remove VM from PM.
STEP 6: Terminate algorithm.

5 Experiments and Evaluation

5.1 Experimental Conditions for Runs

The overall goal of this experiment is to determine the most efficient algorithm based on some predefined efficiency metrics. This entails the identification of the algorithm that allocates a VM to a PM in the least possible time. In order to achieve this goal, a randomized selection of algorithms that are not necessarily of the same asymptotic time complexity as 'Best-Fit Virtual Machine Placement Algorithm' was made. After these algorithms were implemented, a posteriori analysis was conducted as an algorithm's real performance might not really be determined by a priori analysis due to the different performance metrics being considered. *Fig. 4* presents the result of the experiment.

5.2 Discussion of Results

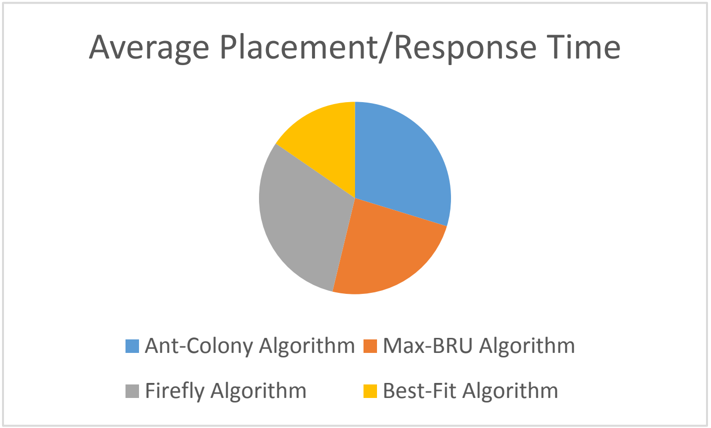

Fig. 3: Average VM Placement/Response Time for Ant-Colony, Max-BRU, Firefly, and Best-Fit Algorithms

From the chart in Fig. 4, the Best-Fit algorithm has the least Average VM Placement/Response Time followed by Max-BRU, Ant-Colony, and Firefly algorithms in this respective order. By implication of this result, the 'Best-Fit Virtual Machine Placement Algorithm' is more efficient in terms of resource wastage and power consumption. In fact, the Average VM Placement/Respnse Time of the proposed algorithm is directly proportional to the number of VMs.

6 Conclusion

This paper introduced the 'Best-Fit Virtual Machine Placement Algorithm'. This algorithm is easy to understand and implement. It is very efficient as it ensures that resources are adequately chosen to match the demands of VMs, hence resource wastage and power consumption, which are a function of load balancing are minimized. However, further research is recommended. An introduction of Binary Search Trees will reduce the VM placement time.

References

[1] H. Alexa and C. James, "The Basics of Cloud Computing," Carnegie Melon University, 2011.

[2] K. M. Nitin and M. Nishchol, "Load Balancing Techniques: Need, Objectives and Major Challenges in Cloud Computing - A Systematic Review," *International Journal of Computer Applications,* vol. 131, pp. 11-19, 2015.

[3] E. Barlaskar, Y. J. Singh and B. Isaac, "Energy-efficient virtual machine placement using enhanced firefly algorithm," *Multiagent and Grid Systems,* vol. 12, no. 3, pp. 167-198, 2016.

[4] J. S. Benita, "Survey on VM Placement Algorithms," *International Journal of Engineering Trends and Technology,* vol. IV, no. 7, pp. 349-352, 2013.

[5] Z. Linquan, Y. Xunrui, L. Zongpeng and W. Chuan, "Hierarchical Virtual Machine Placement in Modular Data Centers," in *IEEE 8th International Conference on Cloud Computing*, 2015.

[6] H. K. Md, C. S. Gholamali and G. Sudhakar, "VM Placement Algorithms for Hierarchical Cloud Infrastructure," in *Cloud Computing Technology and Science (CloudCom)*, Singapore, 2014.

[7] T. H. Nguyen, D. F. Mario and Y. J. Antti, "A virtual machine placement algorithm for balanced resource utilization in cloud data centers," in *Cloud Computing (CLOUD)*, Anchorage, 2014.

[8] G. Yongqiang, G. Haibing, Q. Zhengwei, H. Yang and L. Liang, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences,* vol. 79, pp. 1230-1242, 2013.

[9] M. Mayank and S. Anirudha, "On Theory of VM Placement: Anomalies in Existing Methodologies and Their Migration Using a Novel Vector Based Approach," in *IEEE International Conference on Cloud Computing*, 2011.

[10] TechNet, "About Virtual Machine Placement," Microsoft, [Online]. Available: https://technet.microsoft.com/en-us/library/bb740817.aspx. [Accessed 31 10 2017].

[11] Torry Harris, "Cloud Computing - An Overview," [Online]. Available:

http://www.thbs.com/downloads/Cloud-Computing-Overview.pdf. [Accessed 21 December 2017].

[12] K. C. N. M. Mosharaf and B. Raouf, "A survey of network virtualization," *Computer Networks,* vol. 54, no. 5, pp. 862-876, 2010.

[13] B. Paul, D. Boris, F. Keir, H. Steven, H. Tim, H. Alex, N. Rolf, P. Ian and W. Andrew, "Xen and the art of virtualization," in *SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003.

[14] L. Flavio and D. P. Roberto, "Secure virtualization for cloud computing," *Journal of Network and Computer Applications,* vol. 34, no. 3, pp. 1113-1122, 2011.

[15] L. K. Ronald and D. V. Russel, Cloud Security: A Comprehensive Guide to Secure Cloud Computing, Wiley Publishing, 2010.

[16] J. Y and M. K, "Cloud computing - concepts, architecture and challenges," in *Computing, Electronics and2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, Kumaracoil, India, 2012.

[17] A. Mohammad and H. Eui-Nam, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," in *2014 International Conference on Future Internet of Things and Cloud*, Barcelona, 2014.

[18] G. Chunye, L. Jie and Z. Qiang, "The Characteristics of Cloud Computing," in *2010 39th International Conference on Parallel Processing Workshops*, San Diego, CA, USA, 2010.

[19] Z. Qi, C. Lu and B. Raouf, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications,* vol. 1, no. 1, pp. 7-18, 2010.

[20] K. Dzmitry, B. Pascal and U. K. Samee, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing,* vol. 62, no. 3, pp. 1263-1283, 2012.

[21] R. Martin, L. David and T.-B. A, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing," in *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, Perth, WA, Australia, 2010.

[22] W. Lee, J. S. Howard, P. R. Vwani and A. M. Anthony, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach," *Journal of Parallel and Distributed Computing,* vol. 47, no. 1, pp. 8-22, 1997.

[23] S. Pinal, "A SURVEY OF VARIOUS SCHEDULING ALGORITHM IN CLOUD," *International Journal of Research in Engineering and Technology,* vol. 2, no. 2, pp. 131-135, 2013.

[24] M. M, M. N, K. Y, C. L. Y, G. T. E, Y. Z. A and T. D, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *Journal of Parallel and Distributed Computing,* vol. 71, no. 11, pp. 1497-1508, 2011.

[25] F. B. Luiz and R. M. M. Edmundo, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," *Journal of Internet Services and Applications,* vol. 2, no. 3, pp. 207-227, 2011.

[26] T. M. Siva, S. R and Y. Lei, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *2012 Proceedings IEEE INFOCOM*, Orlando, FL, USA, 2012.

[27] M. Yinchi, C. Xi and L. Xiaofang, "Max-Min Task Scheduling Algorithm for Load Balance in Cloud Computing," in *Proceedings of International Conference on Computer Science and Information Technology, Advances in Intelligent Systems and Computing 225*, India, 2014.

[28] B. K. Remesh and P. Samuel, "Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud," *Innovations in Bio-Inspired Computing and Applications,* vol. 424, pp. 67-78, 2016.

[29] L. Zhanghui and W. Xiaoli, "A PSO-Based Algorithm for Load Balancing in Virtual Machines of Cloud Computing Environment," in *International Conference in Swarm Intelligence*, Berlin, 2012.

[30] M. Gao and L. Hao, "An Improved Algorithm Based on Max-Min for Cloud Task Scheduling," *Recent Advances in Computer Science and Information Engineering,* vol. 125, pp. 217-223, 2012.

[31] S. Pooja and M. Pramati, "Analysis of variants in Round Robin Algorithms," *International Journal of Computer Science and Information Technologies,* vol. 4, no. 3, pp. 416-419, 2013.