A Novel Firefly algorithm based Load Balancing approach for Cloud Computing

J. Aswini, N. Malarvizhi, T. Kumanan

Abstract*: This research work discusses the methodology for cloud workloads prediction and simulation for developing a load balancing algorithm. The main goal of the methodology is to develop an efficient load balancing algorithm which would require less processing power. In-order to enhance load balancing, these following parameters should incorporated, namely, determining as well as comparing the load and efficiency of the system, connection between the nodes, transmission nature and obtaining the proper nodes. Therefore. to enhance the performance of the load balancing, a firefly based algorithm was developed. The proposed algorithm main task is to increase the utilization of resource among cloud servers, thereby enhancing performance of the cloud servers by proper load balancing.

Index Terms: Firefly algorithm, Cloud computing, Load balancing, Global Optimization

I. INTRODUCTION

The concept behind the Artificial Intelligence (AI) is the inspiration from the nature. In the past, the intelligence program is completely based on the survival of the fittest and the idea of evolution. Most inspiration is behind the colonies that could be found in the nature through the ants, swarm, bees, and immune system of our body as well as most other natural progress including the metallurgy[1]. The behaviour of the swarm has raised a technique named the particle swarm optimization and the firefly algorithm has been developed by the behaviour of fireflies[2]. Our main work involves in comparing these both algorithms and put them into test to know the best that could be suited and get the best work. To known about these algorithms, experiments should be conducted that could involve the use of these algorithms in the large scale application[3]. One of the applications could include the detection of the emission source from the large area that could be either chemical or biological sources identified by various sensors fitted around the environment. Sensors in the real-time application could have a huge amount of noises and error and these data should be gathered and simulated [4]. On putting this experiment into consultation it is identified that the firefly algorithm works well when compared to the particle swarm optimization in small scale. Firefly algorithms work well with the noisy sensors. But in the large scale application, though the firefly algorithm works with the extreme speed, the results are not better.

II. GLOBAL OPTIMIZATION

Heuristic is a search technique that makes us to have the next possible move but it could be either good or not good. It is not guaranteed for the optimal solution to be obtained but it is a key to speed AI. Although heuristic could yield the best solution based on the technique, it could considerably break in certain strategies for the large complex problems. If certain application requires large processing time or if a fitting solution is acceptable in which the obtained heuristic solution is near to the optimized solution then this approach could be suitable[5].

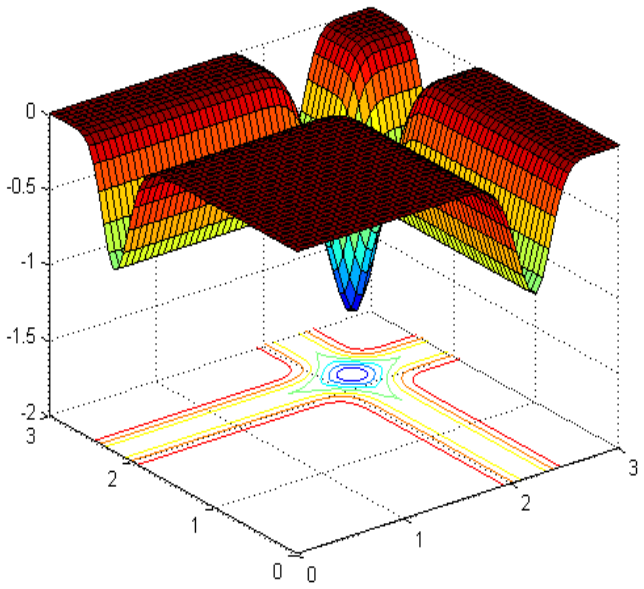

Figure 1 : Michalewicz's function of two dimensions

The figure 1 shows the Michalewicz's function, with two parameters. The parameters have been represented as the dimensions in the graph where on the other hand, the function output serves as the third dimension.

Revised Manuscript Received on March 04, 2019.

J. Aswini, Research Scholar,Department of Computer Science and Engineering,, Meenakshi Academy of Higher Education and Research,Chennai – 600 069, Tamil Nadu, India. And Assistant Professor, Department of Information Technology, Jawahar Engineering College, Chennai – 600093, Tamil Nadu, India.

N. Malarvizhi , Professor & Head,Department of Computer Science and Engineering, School of Computing, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai-600062, Tamil Nadu, India.

T. Kumanan, Professor,Department of Computer Science and Engineering, Meenakshi Academy of Higher Education and Research, Chennai – 600 069, Tamil Nadu, India.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

There could be any number of dimensions for the objective function. To make the visualization process easy we could adapt 2-D version rather than using any D-dimension problem with the larger D. For every parameter, a certain range could be bounded to them[6].

III. PROPOSED ALGORITHM

In-order to enhance load balancing, these following parameters should incorporated, namely, determining as well as comparing the load and efficiency of the system, connection between the nodes, transmission nature and obtaining the proper nodes. Therefore. to enhance the performance of the load balancing, a firefly based algorithm was developed. The proposed algorithm main task is to increase the utilization of resource among cloud servers, thereby enhancing performance of the cloud servers by proper load balancing[7].

A. Firefly Algorithm

In order to go on with the algorithm, the fireflies are placed in various locations initially[7]. The objective function that has the value of the parameters is related to the position and location of the firefly. In the 3-D environment it will be detect the firefly's positions for the function of 3 variables. But the visualization becomes tough once the function of variables increase beyond 10. The evaluation of the objective function could be done by detecting the firefly's current position and their light intensity could be detected by their inverse evaluation[8]. Their main goal of inverse their intensity is to minimize the objective functions. This leads to the higher intensity from the minimum evaluation function. In the figure 2 , the demonstration is done with the 12 fireflies in the 2-D sphere function problem. The origin (0,0) is the global minimum of this problem. The functional value will be higher once it is far away from the origin that has been shown through the contour lines on the graph[9].

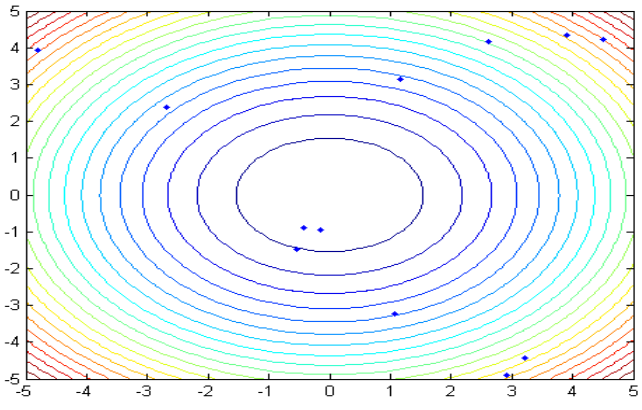

Figure 2 : Firefly algorithm within the 2 dimensional sphere function problem for 12 fireflies

The fireflies are being compared with each other and they tend to move towards the brighter firefly that has been detected as shown in figure 2. Several happening will take place once the firefly tends to move towards their brighter one. 1) The distance among the fireflies should be determined. At certain cases Cartesian distance calculation is adapted but in some cases any form of distance calculation could be adopted[10].

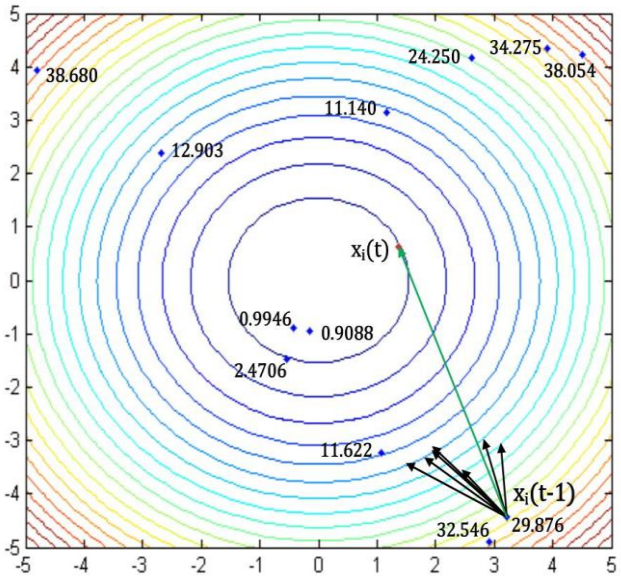

Figure 3: The vector representation for the updated position of the Firefly

Once the entire fireflies moves towards the brighter set, the objective function should be updated towards their new positions as shown 3 . The new estimation should be compared with the old one and it should be verified whether the new position is better than the old set. In this way, if a firefly goes through a position that seems to be good than any other detected locations, but for whatever cause it or any other firefly does not end up there by the last iteration of the algorithm, that best location is still noted[11]. The update is done for the specified firefly after the re-evaluation and the algorithm moves to the succeeding fireflies (each at a time). An example for the sphere function after a complete generation has been shown in the figure 4.

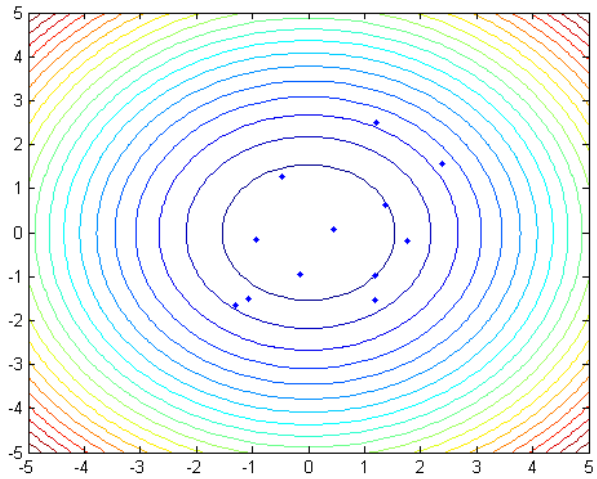

Figure 4: After one iteration

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

It is noticed that after 5 successful generations there is no firefly left in the sphere in the search space that is shown in the figure 5. It also shows the fireflies that have been clustered in the origin[12].

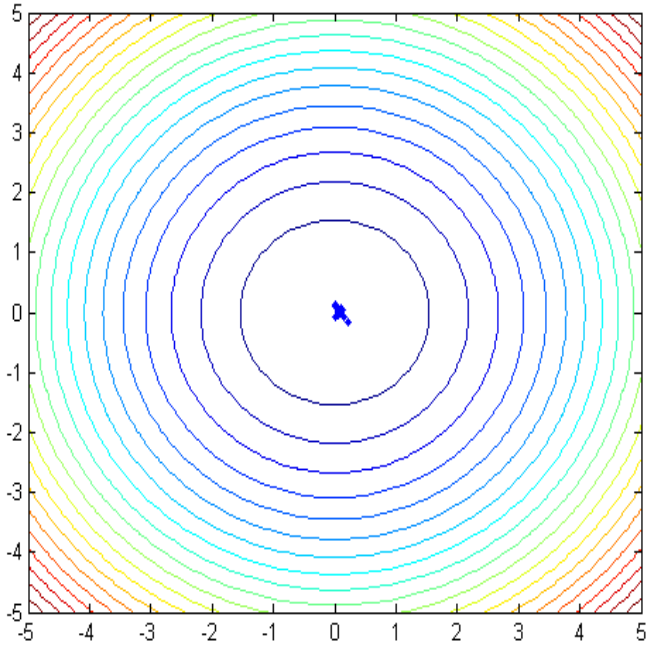

Figure 5: After five iterations

Proposed algorithm for load balancing:
function f(x), x(x1,…,xd)T
Produce initial population of fireflies xi (i = 1,2,…,n)
Light intensity Ii at xi si obtained by f(xi)
while (t< Max_Generation)
fori = 1: n all n fireflies
forj = 1: i all n fireflies
if (Ij>Ii), transfer firefly i to j in d-dimension;
end if Attractiveness differs with distance r via exp [-r]
Check solutions and refresh light intensity
end for j
end for i
Rank the fireflies and find the current best
end while

The proposed firefly algorithm for load balancing first schedule the nodes and then scheduling process initiates, The process of scheduling is allocated to a group of nodes consisting of minimum amount of load possession[13]. The nodes with less minimum amount of load balancing are selected for extra process by the cloud network. The virtual machine with more than two servers and every server posses three nodes and the scheduling process collects the attributes of each nodes[14]. Population generation , scheduling index calculation and least node selector generates the scheduling technique. The population generation is determined by the set of nodes based on the users request to each server. The server gathers the node and it is mapped based on the nodes and their processing time is scheduled and then a scheduled list is generated[15]. The initial population represents the scheduling list. The server which receives the request, determines the unused nodes among all the nodes and as soon as the free node is determined, it gets mapped on the schedule list and the unused free node on top element of the queue by processing the cloud user request. The schedule list is created by the virtual machine by each complete cycle on processing time and its availability. The processing in the firefly algorithms are applied in this particular population to generate an effective scheduling strategy for the cloud system by giving priority to load balancing[16][17]. Since our proposed algorithm involves reassigning users at every iteration and a main component of the objective function is the total system load, we will consider how the total system load changes when a user u is moved from server i to server j.

A soon as the user b is transferred from server s to another server e and modification in load for servers s and e, but server is still the same. The servers load will be s+1 will be:

- The fresh load in server s is:

$$Ts(w + 1) = Ts(w) - \sum AulXts(w) \quad \text{eqn (1)}$$

- The fresh load in server e is:

$$Te(w + 1) = Te(w) + \sum AulXls(w) \quad \text{eqn (2)}$$

- The load in any server T = s, e does not change:

$$Tw(t + 1) = Ts(w) \quad \text{eqn (3)}$$

- Accordingly, the fresh total system load is:

$$T(w + 1) = T(w) + \sum Aul(Xls(w) - Xl\ e(w)) \quad \text{eqn (4)}$$

The primary statement in (1) is true since the double sum depicts the load that user b received on server s because of the interconnection with other users in other servers.

IV. EVALUATION AND RESULTS

The use of CloudSim simulator in this research work is justified because the simulator allows developers to focus on the design issues specific to a particular system, without concerns over the cloud-based infrastructure and services.

According to Calheiros et al., the CloudSim toolkit can perform both system and behaviour modelling of cloud components like virtual machines, data centers, and policies for resource provisioning. In particular, simulation of the cloud computing environments can provide insights into the performance of cloud components. The main advantages of cloud simulations are, improved flexibility in application configurations, ease of use and enhanced customization, as well as the cost savings achieved by reusing the models created during the design phase. CloudSim provides a robust tool for simulating datacenters because the toolkit provides the basic classes for defining datacenters.

The first step for load balancing is to find and define a reliable method of prediction.

Methodology Step 1: Instruments of Prediction

The first step of the methodology is to prove that whether Cicada and Choero extension can generate reliable prediction data. Choero is a network measurement extension to Cicada which allows users to perform network measurement without access to the network infrastructure.

Workload prediction of a cloud demands a highly efficient tool that can predict workload of a cloud network within a few minutes and can be compatible with Cicada. This paper presents a network measurement extension to Cicada called Choreo. This network measurement tool estimates TCP throughput by simply analyzing packet trains. In this paper, data collected from hypothetical deployed networks will be used to simulate CloudSim. To demonstrate that Cicada can detect whether its predictions are reliable and can generate an alert in case if the prediction is unreliable. The following literature will describe the fundamental of Cicada and Choero extension. According to the research, state-of-the-art Cicada's workload prediction algorithm has success rate up to 90% for static placement because for workload prediction, Cicada load balances applications traffic by measuring both spatial and temporal variations of every application. Cicada is capable of workload prediction for different type and class of cloud applications and can provide a reliable feedback indicating whether the prediction is incorrect or the prediction is reliable saving users from uncertainty. Cicada is capable of improving the average completion time of application from 8% to 14% per cent in some cases up to 61%. All these improvements are achieved without any modifications of network infrastructure. Cicada predictions for networks less than 5 virtual machines are unreliable. To eliminate the possibility of any unreliable predictions networks with more than 15 VMs will be considered for predictions and any with less than 15 will not be considered. In the data sample of this paper, a minimum of 20 virtual machines will be used for each host machines. Another research paper objects the reliability of Cicada and suggests that Cicada predictions can be unreliable during the peak hours. In all cases, Cicada can provide a reliable feedback of its own predictions and detect whether or not its predictions are reliable. The conditions mentioned above address the first research question. It is important to understand that Cicada predictions focus on individual pairs, which require VM-to-VM traffic matrices in the cloud infrastructure. It is expected that this type of data from IaaS clouds might provide rich applications compared to data centers or other cloud computing environments. The dataset (sFlow) will be collected from a hypothetical data gathered from Cicada predicted data. The dataset collection process requires sFlow-enabled network switches to gather datagrams that come with the information such as the source and destination IPs, sample timestamp, and MAC address. The data collection process entails three primary steps:

1. The sFlow-enabled switches send the samples to a centralized server.
2. The centralized server collects sample information such as the source and destination IPs, timestamp, and transferred bytes.
3. 3.The database stores sample aggregate data

Methodology Step 2: Sample Data

The second step of the methodology the gathering of sufficient workload data get enrage a simulation in CloudSim and different data of real parallel workloads however, due to the fact that the mentioned data demands a massive and a complicated simulation, in this, a hypothetical set of small data will be used for CloudSim to generate a feasible algorithm for workload balancing. The hypothetical data consists of a predicted amount of workload gathered by Cicada.

Methodology Step 3: Importation of sample data

The 3rd step of the methodology is to import sample data into CloudSim. This can be achieved by exporting the data into a file and later that file can be imported by CloudSim simulator. All predictions of Cicada and Choero can be exported into a .swf file. At the end of the prediction, that file can be imported into CloudSim.

Methodology Step 4: Instrument for simulation

The 4th step of the methodology is to simulate the data in the CloudSim simulator. In this paper, CloudSim will be used to predict and simulate the network. Generally, CloudSim refers to a set of simulation tools that can assess the performance of cloud services within a controllable or a rule-based environment. The simulation toolkit provides classes for describing users, applications, computational purposes, resources management, and data centers in order to facilitate the management and utilization of these components. That is, CloudSim provides a system and a behaviour modelling cloud computing environments. The simulation of cloud environments and applications can facilitate the evaluation of performance in dynamic and distributed environments. The main advantages of simulation include enhanced flexibility in terms of defining cloud configurations, ease of customization, and the cost savings that come with customized simulations. The CloudSim framework is a layered architecture comprising of three layers of components.

The lowest layer comprises of the SimJava simulation engine, which implements the core functionalities for enabling simulation of queuing and processing of events and enabling creation and communication among system components. The next layer is the GridSim layer, which consists of a toolkit for modelling Grid networks and components. This layer comprises of two sets of components: grid services such as datasets, grid information service, resource allocation, workload traces, and core elements such as resources, traffic generator and the network. The next layer is the CloudSim, which extends the functionalities of the GridSim layer. The CloudSim enables the modelling and simulation functions in virtualized cloud environments. It also manages the execution of the core entities such as Virtual Machines (VM), applications, data centers and hosts during simulation. The CloudSim layer comprises user interface structures, VM services, cloud services, and cloud resources. The top layer is the User Code layer, a simulation stack supporting the configuration of hosts, VMs, and applications.

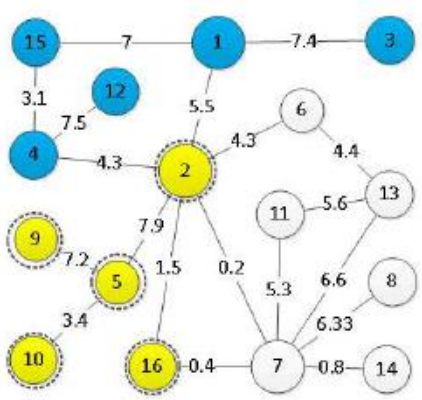

Figure 6: Result of graph node allocation

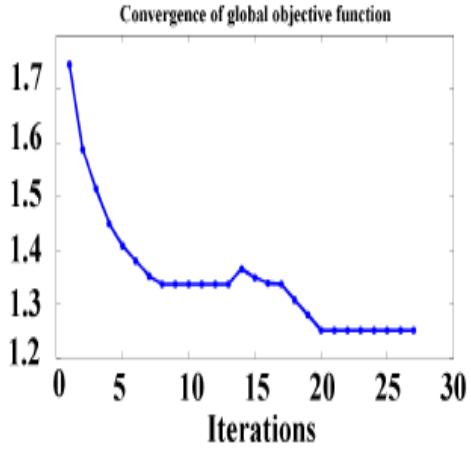

Figure 7: Convergence rate of proposed algorithm

Figure 7 presents convergence of rate the global function objective value. The steady region are depicted in the local optimal values. The global objective function reveals the local point which posses greater value and optimization process occurs in the system.

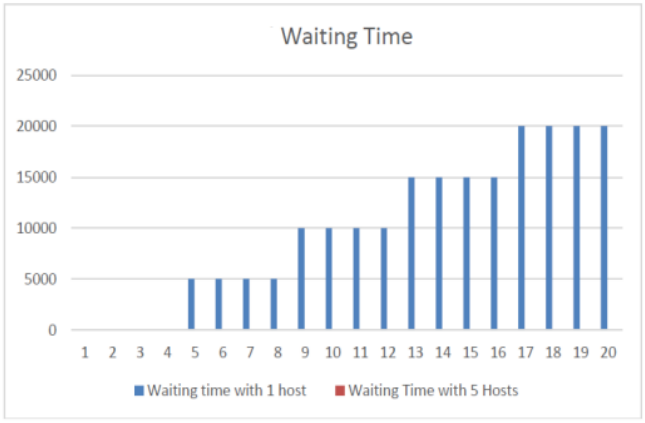

Figure 8: Chart of Final CPU Waiting Time after adding 5 host machines.

If no previous simulation data were to be found then proposed algorithm would run its own simulation to workload balance if there was no previous predictions, random algorithm is used temporary for load balancing while CloudSim runs a simulation to find the optimal amount of resource allocation for the workload as shown in figure 8. The objectives of this work are three-fold: to investigate under what conditions to use Cicada for predicting workloads in dynamic Internet hosting platforms, to determine the conditions to use CloudSIM for reliable workload simulation, and to identify the challenges of firefly based algorithm for load balancing for Internet-based platforms compared to Cicada predictions. The methodology envisaged in this work entails three phases: workload prediction using Cicada, simulation using CloudSIM framework, and the development of a space-shared algorithm for dynamic workload balancing in cloud environments. The final objective is to reduce cloud resource assignment for a specific workload while reducing the waiting time of servers

V. CONCLUSION

In this research paper, we introduced the Load balancing employing firefly algorithm. The proposed algorithm focus on balancing the loads in a cloud network. The load balancing is developed from the firefly algorithm since it posses several advantages. The simulation envisaged in this research work utilizes CloudSim simulation tool, a generalized framework that allows a controllable environment for the simulation and modelling of application performance (Cloud Computing and Distributed Systems (Clouds) Laboratory).

REFERENCES

1. Al-Ta'i, Z., & Al-Hameed, O. A. (2013). Comparison between pso and firefly algorithms in fingerprint authentication. International Journal of Eng. and Innovative Technology (IJEIT), 3, 421–425.
2. Apostolopoulos, T., & Vlachos, A. (2010). Application of the firefly algorithm for solving the economic emissions load dispatch problem. International Journal of Combinatorics, 2011.
3. Arora, J. (2004). Introduction to optimum design. Academic Press.
4. Richard Málek. Seage: Search agents for optimization. http://www.seage.org, July 2009.
5. Magnus Erik Hvass Pedersen and Andrew John Chipperfield. Simplifying particle swarm optimization. Applied Soft Computing Journal, 2010.
6. Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. Journal of the Association for Computing Machinery, 1976.
7. Michal Smith. Simple Firefly Synchronization. PhD thesis, University of Daleware, 2008.
8. Szymon Łukasik and Sławomir Żak. Firefly algorithm for continuous constrained optimiza- tion tasks, 2009. Lecture Notes in Computer Science
9. R.Udendhran. A hybrid approach to enhance data security in cloud storage.Proceeding ICC '17 Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing, Cambridge University, United Kingdom — March 22 - 23, 2017 ACM ISBN: 978-1-4503-4774-7 doi>10.1145/3018896.3025138
10. Tim Mather, Subra Kumaraswamy, and Shahed Latif. Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance. O'Reilly Media, Inc., 2009.
11. Matthias, S., Klink, M., Tomforde, S., & Hahner, J. (2016). Predictive Load Balancing in Cloud Computing Environments based on Ensemble Forecasting (4 ed., Vol. Augsburg,: IEEE International Conference on Autonomic Computing.

12. Alexandre, D., Tomasik, J., Cohen, J., & Dufoulon, F. (2017). Load prediction for energy-aware scheduling for Cloud computing platforms. Orsay: IEEE 37th International Conference on Distributed Computing System
13. LaCurts, K. L. (2014, June). Application workload prediction and placement in cloud computing systems (Unpublished doctoral dissertation). Massachusetts Institute of Technology, Cambridge Massachusetts.
14. Lee, R., & Jeng, B. (2011). Load-balancing tactics in cloud. In Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge CyberC Discovery, pp. 447-454.
15. Mahmood, Z. (2011). Cloud computing: characteristics and deployment approaches. In the 11th IEEE International Conference on Computer and Information Technology, pp. 121-126.
16. Mathur, S., Larji, A. A., & Goyal, A. (2017). Static load balancing using SA Max-Min algorithm. International Journal for Research in Applied Science & Engineering Technology, 5(4), 1886-1893.
17. Xin-She Yang. Nature-Inspired Metaheuristic Algorithms. Luniver Press, 2008.

AUTHORS PROFILE

Ms. J. Aswini, currently pursuing Ph.Din theDepartment of Computer Science and Engineering at Meenakshi Academy of Higher Education and Research (Deemed to be University), Chennai. She did her B.Sc Computer Science, Master of Computer Application and Master of Computer Science and Engineering in 2000 , 2003 and 2011 respectively from Madras University and Anna University, India. At present she is working as an Assistant Professor in Department of Information Technology at Jawahar Engineering College, Chennai, Tamil Nadu and She has eight years of teaching experience. She has published several papers in International Conferences and Journals. Her research interests include Cloud Computing and Internet of Things.

Dr. N. Malarvizhi, currently working as Professor & Head in the Department of Computer Science and Engineering at Vel Tech RangarajanDr.Sagunthala R&D Institute of Science and Technology, Chennai-62, Tamilnadu, India. She is having more than 15 years of teaching experience. She has written a book titled “Computer Architecture and Organization”, Eswar Press, The Science and Technology Book Publisher, Chennai. She serves as a reviewer for many reputed journals. She has published numerous papers in International Conferences and Journals. Her area of interest includes Parallel and Distributed Computing, Grid Computing, Cloud Computing, Big Data Analytics, Internet of Things, Computer Architecture and Operating Systems. She is a life member of Computer Society of India (CSI), Indian Society for Technical Education (ISTE), IARCS and IAENG. She is a Senior Member of IEEE and IEEE Women in Engineering (WIE). She is a Member of Association for Computing Machinery (ACM).

Dr. T. Kumanan, currently working as Professor in Department of Computer Science and Engineering at Meenakshi Academic of Higher Educationand Research(Deemed to be University), Chennai. He received M.E and Ph.D Degrees in 2005 and 2014 from Anna University, Chennai. He has published 10 paper in International Level Conferences and Two papers in National Level Conferences. He has published 15 paper in Internal National Journal. His areas of interests are in Computer Networks, Mobile Computing, Cryptography and Network Security, Image Processing and High Speed Network. He is member of ISTE and CSI.