I.J. Wireless and Microwave Technologies, **2019, 6, 1-10**
Published Online November 2019 in MECS(http://www.mecs-press.net)
DOI: 10.5815/ijwmt.2019.06.01

Available online at **http://www.mecs-press.net/ijwmt**

Load Balancing in Cloud Computing Using Hungarian Algorithm

[1]Mohammad Irfan Bala, [2]Mohammad Ahsan Chishti

[1]National Institute of Technology, Srinagar, 190006, India
[2]National Institute of Technology, Srinagar, 190006, India

Received: 09 June 2019; Accepted: 15 October 2019; Published: 08 November 2019

Abstract

Cloud computing is a highly popular computing paradigm providing on-demand resources with high reliability and availability. The user requests are fulfilled by providing a virtual machine with the requested configuration. However, with the ever-increasing load on the cloud resources, the need for optimal resource utilization of the cloud resources has become the need of the hour. Load balancing has been identified as one of the possible ways to improve resource utilization in the cloud and the current state-of-the-art algorithms indicate the numerous attempts made to find the approximate solution for this NP-hard problem. In this work, we have focused on evaluating the efficiency of the Hungarian algorithm for load distribution in the cloud and compared its performance with First-come-first-serve (FCFS). The simulations were carried out in CloudSim and show remarkable improvement in various performance parameters. Finish time of a given task schedule was reduced by 41% and average execution time was reduced by 13% in the Hungarian algorithm when compared with FCFS. The simulations were carried out under different workload conditions to validate our results.

Index Terms: Cloud Computing, CloudSim, Load balancing, Resource Scheduling, Virtual machine

© 2019 Published by MECS Publisher. Selection and/or peer review under responsibility of the Research Association of Modern Education and Computer Science

1. Introduction

The idea of cloud computing has been around for almost the last two decades and has achieved a state of maturity since then finding applications in multiple domains. Cloud computing is defined as a practice of using the computing services which are made available to the end-users over the internet and can be used to store and process the data. It is unlike a local server or a personnel computer and the end-user does not need to purchase

* Corresponding author.
E-mail address: mirfan508@gmail.com; ahsan@nitsri.net

the resources for his use. The resources are shared instead of being dedicated. Cloud computing leases its resources to the end-users and provides services like compute, networking, storage, analytics, and various other services under a usage-based payment model.

Cloud computing also acts as the backbone for the Internet of Things (IoT) by providing the necessary computational power for data processing. The data generated by the resource-constrained IoT devices is stored in the cloud and also processed there. IoT has witnessed a tremendous increase in the number of devices that are being connected to the internet, resulting in increased load on the cloud resources for data processing. Scalability in cloud computing, being one of its touted characteristics, has been pushed to extreme limits because of continuous and sustained growth in IoT devices. Given the current scenarios of cloud and IoT, the day is not far when the cloud will fail to meets the demands of the IoT devices. One of the possible ways to handle this situation, before it goes out of hand, is to improve the resource utilization of cloud resources. Optimal load balancing on the cloud resources can greatly increase the utilization of cloud resources but it is regarded as one of the important challenges in cloud computing and is an NP-Hard problem [1]. Optimal load balancing can be reduced from a multiprocessor scheduling problem which is a well known NP-hard problem. Considering the fact that it is not possible to find an optimal solution for NP-hard problems in polynomial time, numerous efforts have been made to find the approximate or near-optimal solutions [2] and the review of such efforts has been given in next section. A supplementary computing paradigm, called Fog computing, has emerged that aims to provide the cloud services at the edge of the network [3]. Although Fog computing is still in the stages of infancy but may acquire primary importance in the near future.

Our work focuses on the load balancing problem and we have proposed the use of the Hungarian algorithm for load balancing. Various simulations were performed in CloudSim and the results have proved the efficiency of the Hungarian algorithm for load balancing problems in cloud computing. It reduces the total execution time for the group of submitted jobs which in turn increases the utilization of the cloud resources. The paper is organized as follows: Section 2 gives the literature review of the state-of-the-art algorithms proposed for load balancing. Section 3 briefly describes the architecture of the CloudSim, simulation tool used in cloud computing. Our proposed algorithm is given in section 4. The simulation results and their detailed analysis are given in section 5. Section 6 concludes the paper and specifies future work directions.

2. Related Work

The simulation toolkit for cloud computing, called CloudSim, was proposed in [4] and it is widely used by the researchers for simulations in cloud computing as it allows repeatability of experiments in a controlled manner. Most of the load balancing algorithms proposed in the literature have been simulated using CloudSim. This section presents a brief overview of the work done on load balancing in cloud computing.

The comparative analysis of different variants of the round-robin algorithm is discussed in [5] concluding that Time Slice Priority Based Round Robin (TSPBRR) is superior to the other variants. Dynamic time quantum was proposed in [6] which managed to remove the inherent disadvantages of the round-robin algorithm i.e. long waiting time and long response time. It also managed to reduce the number of context switches which is seldom researched performance parameter. Different operating system scheduling algorithms like First-come first-serve [7], Shortest job first, round-robin, etc were analyzed and evaluated using CloudSim in [8]. A task scheduling algorithm for autonomous and dividable tasks based on genetic algorithm was proposed in [9] taking into account the computation and memory requirements of tasks. The work depicted in [10] has compared the performance of 11 different load balancing algorithms like opportunistic load balancing, Min-min, Max-min, genetic simulated annealing, etc and concluded that Min-min performs well compared to other techniques. A modified Min-min algorithm called user-priority guided Min-min algorithm has also been proposed. Its simulations were carried out in Matlab and it achieved a 20% improvement in utilization ratio than the simple Min-min algorithm. Reference [11] proposed the use of the Max-min algorithm for load balancing in the elastic cloud. It improved resource utilization and reduced the response time of tasks. The work in [12] proposed a task scheduling algorithm exploiting the advantages of both Min-min and Max-min algorithms and covering their disadvantages. It achieved a comparatively lower makespan.

A task scheduling algorithm was proposed in [13] which focussed on QoS requirements of the tasks. The algorithm was based on a modified version of a genetic simulated annealing algorithm. Ant Colony optimization was proposed in [14] and its performance was compared with FCFS and RR. Nature-inspired BAT algorithm has been proposed in [15] which reduces the idle time of the virtual machines and thus improving their efficiency. A combination of Particle Swarm Optimization and Grey Wolf Optimization (PSO_GWO) has been compared with the BAT algorithm in [16] with the proposed PSO_GWO algorithm resulting in reduced total execution time and total execution cost. PSO strives for local optimization while GWO optimizes locally.

There are many more load balancing algorithms presented in the literature [17] like Artificial Bee Colony Algorithm [18], Firefly algorithm, Honey Bee algorithm, Adaptive incremental genetic algorithm [19], and also some hybrid based algorithms [20]. The critical analysis of literature review has been given in Table 1. Each algorithm focuses on one or multiple quantitative metrics like throughput, fault tolerance, response time, makespan, migration time, resource utilization, etc trying to improve these metrics. Moreover, many of the works in the literature have measured these performance parameters in a single scenario, making the result analysis unreliable. Due to the unavailability of the actual load balancing algorithms used in the cloud by Amazon or Microsoft, there is no benchmark or standard for the performance measurement of our algorithm. We have proposed Hungarian algorithm for load balancing in cloud and analysed its performance using CloudSim. We have also analysed the performance of FCFS and compared their results.

3. CloudSim Architecture

CloudSim is a well-known tool used for the simulation of cloud computing environments [4]. There are few more cloud simulation tools available like GridSim, SimJava, CloudAnalyst, etc but we have used CloudSim because of its various advantages and flexibility it offers. Moreover, it is the most commonly used simulation tool in the literature because it is capable of monitoring multiple performance parameters like latency, energy consumption, network congestion, etc. CloudSim allows provisioning of resources and their management in virtual environments. The capabilities and functionalities of CloudSim are being continuously upgraded allowing users to simulate more and more complex scenarios.

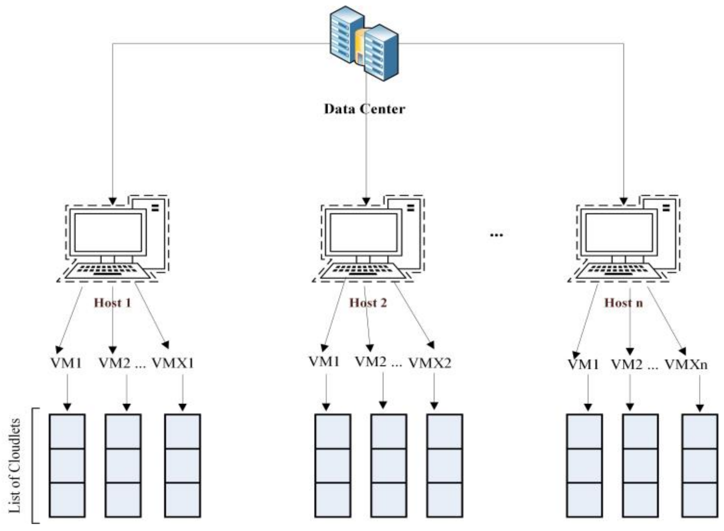

Fig.1. Computational hierarchy in CloudSim

Cloud consists of a shared pool of resources that can be rapidly and easily provisioned among the requesting users. It consists of huge data centers, each consisting of tens of thousands of computers. Physical machines are called hosts and each host is used to create multiple virtual machines on it which can later be used for the execution of tasks, termed here as cloudlets. Both cloudlets and virtual machines have some configurable properties like size, ram, MIPS, bandwidth, etc.

The number of virtual machines that can be created on each host depends on the configuration of the host machine and the VMs created on it. The cumulative resources of all the virtual machines on a particular host cannot exceed the total resources available in a host. Any attempt to create more VMs will fail. However, the number of cloudlets assigned to any particular VM is not restricted and they are stored in a queue and executed as per the specified policy.

Table 1. Survey of different load balancing algorithms.

Algorithm	**Simulator**	**Optimization parameter**	**Comparison with**	**Benefits/Limitations**
First come first serve [7,14,21]	CloudSim	Makespan	RR, Throttled	Better than RR but causes convoy effect
Round Robin [5]	CloudAnalyst	Execution time	Different variants of RR	Works well for a smaller number of jobs
Min-Min [10]	Matlab	Task completion time and resource utilization	11 different algorithms	Improves the resource utilization but SLA is not considered
Max-Min [21]	CloudSim	Makespan	RASA, Min-Min, RR	Performs better than RR and improved Max-Min can further improve its performance
Hungarian algorithm [22]	CloudSim	Resource utilization and execution time	-	The algorithm is computationally expensive and comparison with other algorithms is not given.
Ant colony optimization [14]	CloudSim	Load balancing and makespan	FCFS and RR	Uses feedback mechanism but convergence is slow
Genetic algorithm [23]	CloudAnalyst	Response time	FCFS, RR and local search algorithm Stochastic Hill Climbing	High computational complexity which leads to poor performance for large inputs
BAT algorithm [15]	Matlab	VM utilization and response time	FIFO, Particle Swarm optimization and harmonic search	The convergence rate is high
League Championship Algorithm [24]	-	Makespan, Average response time, average completion time	FCFS, Last job first	Algorithmic parameters require manual tuning
Particle Swarm optimization [25]	CloudSim	Energy consumption	Modified best fit decreasing, Modified best fit heuristic algorithm	PSO performance is application dependant
Particle Swarm optimization and Grey wolf optimization (Hybrid) [20]	CyberShake, LIGO	Total execution time, total execution cost	BAT algorithm	PSO is responsible for global optimization while GWO performs local optimization

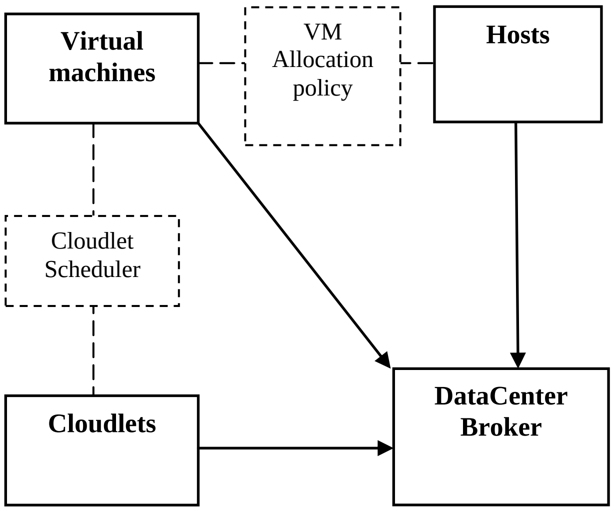

Fig.2. CloudSim components

There are 3 major entities in CloudSim which are responsible for handling the execution of jobs in CloudSim. They are: Broker, Management Information System (MIS) and DataCenter [22]. Figure 2 shows the important components of CloudSim which we will be dealing with. DataCenterBroker receives the set of virtual machines and cloudlets as input and it is responsible for mapping the cloudlets to the virtual machines. Different mapping policies may be used, each of which can have different objectives like reduction in execution time, shortest finish time, energy efficiency lesser waiting time, etc. We will be using the Hungarian algorithm for cloudlet to VM mapping.

4. Hungarian Algorithm

Hungarian algorithm is a well-known optimization algorithm used to solve the assignment problem in polynomial time. We have modeled our Cloudlet-VM mapping problem as an assignment problem and applied the Hungarian algorithm to find the optimal mapping and the results have been compared with FCFS. The performance of our algorithm cannot be compared with the standard algorithms used by Microsoft or Amazon because of their unavailability in the public domain [4]. As a result performance of all the proposed algorithms is usually compared with each other instead of comparing them with the algorithms used by Amazon or Microsoft.

The working of the Hungarian algorithm is based on a theorem which states that, given a matrix for which the optimal assignment is to be found, if a constant value is added or subtracted from all the elements of any one row or column, then the optimal assignment for the newly obtained matrix is also an optimal assignment for the original matrix.

The Hungarian algorithm receives a square matrix (n x n) as input and generates a row-column optimal mapping as output in the form of an array. Following are the steps involved in the Hungarian algorithm

Step 1: Find the minimum element from each row and subtract it from all the elements of the corresponding row.

Step 2: Now find the minimum element from each column and subtract it from all the elements of the corresponding column.

Step 3: Identify the zero elements in the matrix and try to draw minimum possible horizontal and vertical lines which will cover the identified zero elements.

Step 4: If the number of the lines drawn in step 3 is equal to the order of the matrix then the optimal assignment has been found. However, if the number of lines drawn is less than the order of the matrix (n) then the optimal assignment has not been found. In the latter case proceed to step 5.

Step 5: Find out the smallest element which was not covered by the lines in step 3. Subtract this value from all uncovered rows and then add the same value to all the covered columns. Go back to step 3.

5. Experimental analysis

Our objective in this research is to allocate the cloudlets to the Virtual machine in order to minimize the total execution time or makespan. Suppose we are given “m” cloudlets and “n” virtual machines. Then the cost matrix will be of size (m x n) where costMatrix[i][j] is the time taken to execute cloudlet[i] on virtualMachine[j] and can be calculated as:

costMatrix[i][j] = length of cloudlet[i]/mips of virtualMachine[j]

We generate cloudlets and virtual machines with each cloudlet and virtual machine being assigned length and computational power respectively by a random function. Length of the cloudlets and computational power of the virtual machines are generated by the following code:

```
Random r=new Random();
LengthOfCloudlet[i]=10000 + ((r.nextInt(100)/m)*1000);
mipsOfVM[j]=100+ ((r.nextInt(100)/n)*5);
```
where “m” and “n” are the number of cloudlets and VMs respectively.

Since the Hungarian algorithm requires a square matrix, so we have used the same number of cloudlets and VMs in each simulation. However, we can add dummy cloudlets or VMs each with zero-length or MIPS respectively if the number of cloudlets and VMs are not the same without affecting the working of the algorithm. We have run 6 different simulations with the number of cloudlets varying between 5 and 200. Increasing the number of cloudlets can greatly increase the complexity of the algorithm because of its O(n^3) time complexity. Our simulation results are given in Table 2.

Table 2. Simulation results

	Hungarian algorithm				**FCFS**			
No. of Cloudlets	**Finish time**	**Avg. time**	**Throughput**	**Std. dev**	**Finish time**	**Avg. time**	**Throughput**	**Std. dev**
5	108.43	93.76	0.04611	7.96	113.14	93.9	0.04419	9.77
10	135.81	111.79	0.07363	18.15	156.61	114.01	0.06385	28.64
20	166.76	141.15	0.11993	17.36	234.88	150.08	0.08515	51.06
50	196.71	157.08	0.25418	19.22	513.14	179.24	0.09744	98.36
100	196.46	159.25	0.509	23.59	644.54	203.92	0.15515	146.11
200	198.04	163.13	1.00991	21.37	710.09	210.14	0.28165	150.71

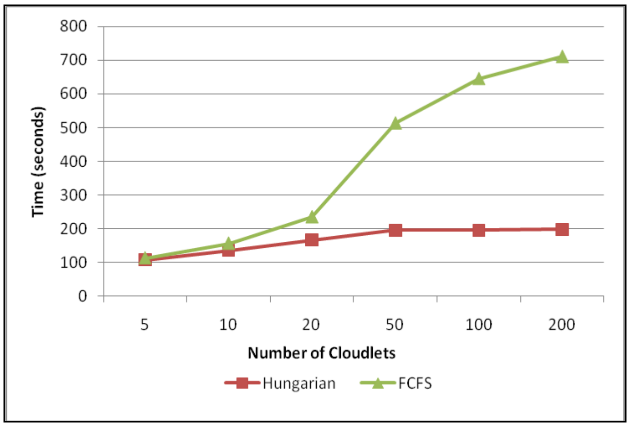

Fig.3. Finish time in Hungarian algorithm and FCFS

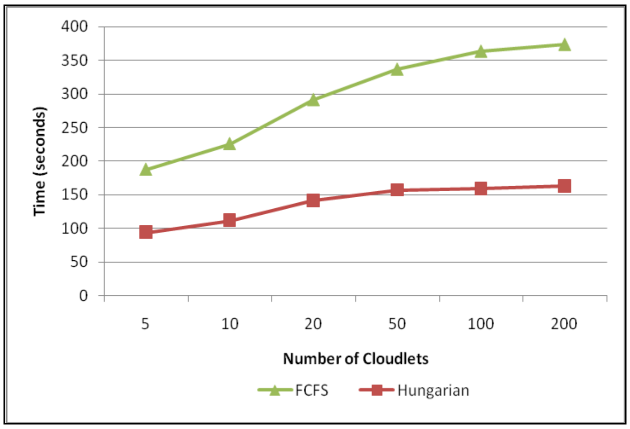

Fig.4. Average execution time for the Hungarian algorithm and FCFS

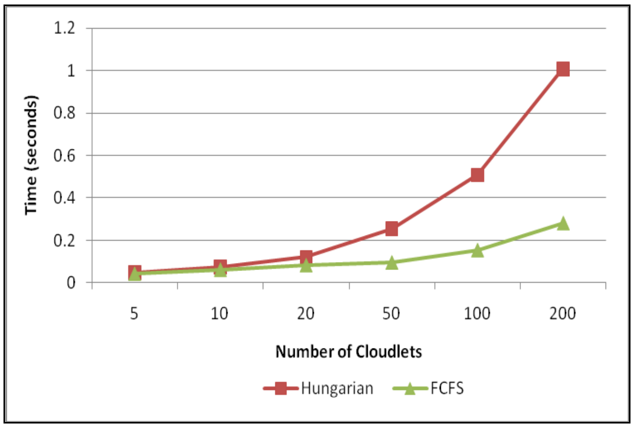

Fig.5. Throughput for Hungarian algorithm and FCFS

As observed from figures 3-5, there is a noteworthy improvement in the various performance parameters of the cloud using the Hungarian algorithm over FCFS. The results when analyzed across all the simulations show a reduction in finish time by approximately 41% and the average execution time has also reduced by 13%.

6. Conclusions

The growing popularity of cloud computing across various application domains and the ever-increasing load on the cloud resources has magnified the importance of the optimal use of cloud resources. Our work focused on evaluating the performance of the Hungarian algorithm, when used for load balancing, and compared with the performance of FCFS. Multiple simulations carried out in CloudSim supported our claim that the Hungarian algorithm easily outperforms the FCFS in multiple performance parameters. Simulations were repeated under increasing load conditions to ensure scalability and heterogeneity of our proposed algorithm. While many algorithms tend to be situation-specific, generating optimal results only for a specific scenario, we have tested the Hungarian algorithm in multiple scenarios to verify the authenticity of our results. Our work concentrated on cloudlet-VM mapping only and does not include VM-Host mapping policies. In the future, we aim to include “energy consumption” for evaluating the performance of the algorithms.

References

[1] X. Chen, L. Jiao, W. Li, and X. Fu, “Efficient multi-user computation offloading for mobile-edge cloud computing,” *IEEE/ACM Trans. on …*, vol. 24, no. 5, pp. 2795–2808, 2016.

[2] M. A. Rodriguez and R. Buyya, “Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds,” *IEEE Trans. Cloud Comput.*, 2014.

[3] M. I. Bala and M. A. Chishti, “Survey of applications, challenges and opportunities in fog computing,” *Int. J. Pervasive Comput. Commun.*, vol. 15, no. 2, pp. 80–96, Jun. 2019.

[4] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Softw. - Pract. Exp.*, 2011.

[5] P. Samal and P. Mishra, “Analysis of Variants in Round Robin Algorithms for Load Balancing in Cloud Computing,” *Int. J. Comput. Sci. Inf. Technol.*, 2013.

[6] S. Ghosh and C. Banerjee, “Dynamic Time Quantum Priority Based Round Robin for Load Balancing In Cloud Environment,” in *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2018, pp. 33–37.

[7] F. Saeed, N. Javaid, M. Zubair, and M. Ismail, “Load Balancing on Cloud Analyst Using First Come First Serve Scheduling Algorithm,” no. June, 2019.

[8] M. Gahlawat, “Analysis and Performance Assessment of CPU Scheduling Algorithms in Cloud using Cloud Sim,” *Int. J. Appl. Inf. Syst.*, vol. 5, no. 9, pp. 5–8, 2013.

[9] Z. Chenhong, Z. Shanshan, L. Qingfeng, X. Jian, and H. Jicheng, “Independent tasks scheduling based on genetic algorithm in cloud computing,” in *Proceedings - 5th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2009*, 2009.

[10] T. D. Braun *et al.*, “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems,” *J. Parallel Distrib. Comput.*, 2001.

[11] Y. Mao, X. Chen, and X. Li, “Max–Min Task Scheduling Algorithm for Load Balance in Cloud Computing,” Springer, New Delhi, 2014, pp. 457–465.

[12] S. Parsa and R. Entezari-Maleki, “RASA: A New Task Scheduling Algorithm in Grid Environment,” *World Appl. Sci. J.*, vol. 7, pp. 152–160, 2009.

[13] G. N. Gan, T. L. Huang, and S. Gao, “Genetic simulated annealing algorithm for task scheduling based on cloud computing environment,” in *Proceedings - 2010 International Conference on Intelligent Computing*

and Integrated Systems, ICISS2010, 2010.
[14] A. E. keshk, A. B. El-Sisi, and M. A. Tawfeek, "Cloud Task Scheduling for Load Balancing based on Intelligent Strategy," *Int. J. Intell. Syst. Appl.*, vol. 6, no. 5, pp. 25–36, 2014.
[15] T. S. Rani, "Task Scheduling on Virtual Machines using BAT Strategy for Efficient Utilization of Resources in Cloud Environment," vol. 12, no. 17, pp. 6663–6669, 2017.
[16] R. Kaur and K. S. Dhindsa, "Efficient Task Scheduling using Load Balancing in Cloud Computing," *Int. J. Adv. Netw. Appl.*, vol. 10, no. 03, pp. 3888–3892, 2018.
[17] A. Hota, S. Mohapatra, and S. Mohanty, "Survey of Different Load Balancing Approach-Based Algorithms in Cloud Computing: A Comprehensive Review," in *Computational Intelligence in Data Mining*, 2019, pp. 99–110.
[18] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Appl. Math. Comput.*, 2009.
[19] K. Duan, S. Fong, S. Siu, W. Song, and S. Guan, "Adaptive Incremental Genetic Algorithm for Task Scheduling in Cloud Environments," *Symmetry (Basel).*, vol. 10, no. 5, p. 168, May 2018.
[20] M. Berwal and C. Kant, "Load Balancing in Cloud Computing," *Int. J. Comput. Sci. Commun.*, vol. 6, pp. 52–58, 2015.
[21] D. A. Agarwal and S. Jain, "Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment," *Int. J. Comput. Trends Technol.*, vol. 9, no. 7, pp. 344–349, 2014.
[22] K. Parikh, N. Hawanna, P. K. Haleema, R. Jayasubalakshmi, and N. C. S. N. Iyengar, "Virtual Machine Allocation Policy in Cloud Computing Using CloudSim in Java," *Int. J. Grid Distrib. Comput.*, vol. 8, no. 1, pp. 145–158, 2015.
[23] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing," *Procedia Technol.*, 2013.
[24] A. Husseinzadeh Kashan, "League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships," *Appl. Soft Comput. J.*, vol. 16, no. August, pp. 171–200, 2014.
[25] A. P. Xiong and C. X. Xu, "Energy efficient multiresource allocation of virtual machine based on PSO in cloud data center," *Math. Probl. Eng.*, vol. 2014, 2014.

Authors' Profiles

Mohammad Irfan Bala, is pursuing his PhD at the department of Computer Science & Engineering, National Institute of Technology Srinagar, India. He has done his bachelors and masters in Computer Engineering from Pune University in 2011 and 2013 respectively. His research focuses on Cloud Computing, Fog Computing and Computational offloading.

Mohammad Ahsan Chishti, has done his Bachelor of Engineering and M.S. in Computer and Information Engineering from International Islamic University Malaysia with specialization in Computer Networking. He has received his Ph.D. from National Institute of Technology Srinagar, India. Presently he is working as Assistant Professor and Head of the Department of Computer Science & Engineering, National Institute of Technology Srinagar, India. He has more than 60 research publications to his credit and 10 patents with two granted International Patents. He is a Young Scientist awardee from Department of Science & Technology, Government of Jammu and Kashmir for the year 2009-2010 and

IEI young engineer awardee 2015. He is Senior Member-Institute of IEEE, Member IEI, Life Member CSI, and Member IETE. He is a certified White belt in Six Sigma by Six Sigma Advantage Inc. of USA (SSAI).

How to cite this paper: Mohammad Irfan Bala, Mohammad Ahsan Chishti, " Load Balancing in Cloud Computing Using Hungarian Algorithm ", International Journal of Wireless and Microwave Technologies(IJWMT), Vol.9, No.6, pp. 1-10, 2019.DOI: 10.5815/ijwmt.2019.06.01