Whale Optimization-based Double Reinforcement Learning Approach (WODRL) for Load Balancing in Fog Computing

Bhargavi K[1], B Sathish Babu[2*], K N Subramanya[3]

[1]*Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumakuru-572103*

[2]*Department of Computer science and Engineering, RV College of Engineering®, Bengaluru-560059*

[3]*Department of Industrial Engineering and Management, RV College of Engineering®, Bengaluru-560059*

Abstract

Fog computing provides a flexible architecture for edge devices to perform substantially large amount of computation, data storage, and inter-communication. The fog computing also extends support to Internet of Things (IoT) in which entities get connected through their end devices like mobile phones, smart drone swarms, wearable health monitoring devices, and others. Fog computing environment is prone to several performance challenges, to mention few, difficulty in handling heterogeneity in fog nodes, mobility of fog nodes, load balancing, task scheduling, resource provisioning, and conflicting service level requirements. In this paper the load balancing performance issue in fog computing is studied by applying the whale optimization enabled double reinforcement learning, two level optimized load balancing policies are drawn to increase the overall performance of the computation intensive applications. The results obtained are found to be good in comparison with the recent work in terms of performance metrics like total execution time, response time, and resource utilization rate.

Keywords: *Fog computing, WODRL, load balancing, reinforcement learning, whale optimization, expected value analysis, performance, IOT applications.*

1.0 Introduction

Fog computing is popularly referred as fogging, edge computing or fog network which provides a flexible architecture for edge devices to perform substantially large amount of computation, data storage, and communication over the network. Fog basically stands as a peripheral computing end for cloud, rather than sending each and every request to

**Mail Address: B. Sathish Babu, Associate Professor, Department of Computer Science and Engineering, RV College of Engineering®, Bengaluru-59, Email: bsbabu@rvce.edu.in*

cloud for processing, the fog nodes will be used for computation which in turn reduces the response time and bandwidth requirements [1, 2]. Even the fog computing extends support to Internet of Things (IoT) in which individuals get connected through their end devices like mobile phones, smart drone swarms, wearable health monitoring devices, and so on. Some of the advantages offered by fog computing include reduction in the response time, provides a high level of security, achieves high speed of operation, less reliance over the wired services, less consumption of bandwidth, and so on. However fog computing environment is prone to several performance challenges which include difficulty to handle heterogeneous fog nodes, mobility support is tedious to manage, difficulty in managing massively geographically distributed fog nodes, lack of location awareness, fluctuation in the power consumption level, exhibits sensitivity towards the effect of outliers, over provisioning of resources, conflicting quality of service requirements, improper scheduling of incoming customer requests, and so on [3, 4]. Among the performance challenges faced by the fog computing environment proper load balancing among the fog nodes is one of the primary requirements. Load balancing can be implemented either in centralized or distributed approaches. In centralized approach the load balancing is carried out by a centralized controller which is easily implementable and operates by gathering the load status information from all the nodes, analyze and then initiates load balancing action. Whereas, in distributed approach, a distributed controller, is responsible to establish coordination among local controllers which results in improved scalability of the approach [5, 6].

Reinforcement learning is one of the prominent areas of machine learning in which the reinforcement learning agent take random actions to achieve maximum number of rewards. However, the reinforcement learning suffers from poor results due to overloaded states, high tendency to forget the previous learning episodes, inability to handle large number of states and discrete events, significant increase in the variance of policy gradient value, and so on [7, 8]. These drawbacks are overcome by considering the double Q state values of the reinforcement learning agent which makes critical decision without the necessity to conduct exploration.

The whale search optimization algorithm mimics the hunting strategy of humpback whale animal which finds global solution through encircle updating and encircle shrinking operations and is commonly used to solve complex optimization problems [9, 10]. Basically, the algorithm

mimics the bubble net hunting strategy of the whale animal to track the position of the prey at distant location. The best part of the whale animal is, it always tries to find best meal and never gets satisfied by average or below average meals which does not let to converge to sub-optimal solutions. Some of the advantages offered by whale search algorithm includes producing of high quality solutions, good at exploring larger state space of the computing domain, converges to the global optimal solutions in minimum epochs of training, provides simple structure for operations, speed up the local search mechanism, provides multi objective solutions, results in promising solutions at a faster search space, and so on [11, 12]. The whale algorithm is used in variety of applications like design of photonic filters, allocation of resources in wireless multimedia network, tracking the position of photovoltaic system, identification of skeletal structure in solid objects, tunning the parameters of fuzzy controller, classification of cancer images, and so on.

In this paper the benefits of both double reinforcement learning and whale swarm optimization is combined to address the load balancing problem in fog computing environment. Here by applying the whale optimization enabled double reinforcement learning, two level optimized load balancing policies are drawn which increases the overall performance of the computation intensive applications.

Some of the research works carried out in literature for balancing the load across the fog nodes is discussed as follows. A novel dynamic load balancing algorithm by name tabu search is proposed for fog computing environment [13]. The tabu search mechanism basically uses nearest neighborhood mechanism to produce global optimal solution by moving from one local optimal solution to another local optimal solution. Two objectives are kept for the purpose of optimization purpose first one is computational coast incurred while distributing the load among fog nodes and the second one denotes the computational coast incurred while distributing the load among cloud nodes. The technique converts the multi objective problem into single objective problem to determine the best fit fog node or cloud node for the incoming tasks requests. There is considerable reduction in the total execution time of fog nodes but it suffers from several limitations like there is no guarantee for producing the global optimal solutions, effective searching of the input search space not used to happen, and too many parameters are involved in the optimization process.

A modified min-min load balancing algorithm for processing the tasks in fog computing environment is discussed [14]. The available fog computing nodes in the fog environment is taken into account which includes end user resources, network related resources, and resources of the cloud nodes. The load balancing algorithm tries to evenly distribute the incoming load among the fog nodes which are available thereby reducing the response time of the tasks. The best fit task for the fog nodes is selected based on the length of the incoming tasks. The tasks get clustered and the cluster having tasks with maximum makespan time is selected first for the operation. Compared to the round robin algorithm, the performance of the proposed technique is found to be good. But the time involved in context switching is more, and higher priority tasks may suffer from starvation.

A load balancing algorithm based on hill climbing is proposed [15], hill climbing is one of the popular algorithms used to perform searching using mathematical optimization method. It begins with random distribution of tasks among fog nodes and through several iterations of training the best fit fog nodes are determined for the incoming tasks. The loop executes until best fit solution is found for the incoming tasks in the closest available location. The algorithm continuously improves stepwise by updating the size of the elevation value to determine the peak of the mountain. With respect to load balancing the hill climbing algorithm suffers from limitations like rigidness towards higher steep regions, availability of limited number of step moves, frequent restarting of the algorithm from random locations, produces suboptimal solutions, and so on.

Load balancing in cloud using dynamic resources allocation mechanism for fog is proposed [16]. IoT applications requests for varying amount of resources, responding to such kind of requests suffers from several drawbacks like bottleneck of nodes, overloading of fog nodes, underloading of fog nodes, and so on. A Dynamic Resource Allocation Method (DRAM) is proposed which does analysis of the load balancing condition of various computing nodes first then calls appropriate static or dynamic resource allocation mechanisms. The load balancing is performed in four steps which includes partitioning of the fog services, identification of sparse space for computing nodes, perform static allocation of resources, and then do global allocation of resources. However, the approach suffers from several drawbacks in terms of poor resource allocation for computation intensive applications, decline in throughput, and so on.

The main drawbacks observed in the existing works towards load balancing in fog computing are premature convergence towards suboptimal solutions, often misallocation of the resources, starvation of higher priority tasks, and high computation overhead due to the involvement of too many parameters in decision making. Keeping these drawbacks, in this paper the benefits of both double reinforcement learning and whale swarm optimization is combined to address the load balancing problem in fog computing environment. Here by applying the whale optimization enabled double reinforcement learning, two level optimized load balancing policies are drawn which increases the overall performance of the computation intensive applications in terms of global optimal policies formation, lowered response time, improved efficiency in the task scheduling among fog nodes, and better tuning of the decision-making parameters.

2.0 Definitions

The mathematical definitions for the performance metrics considered for evaluation of the proposed WODRL load balancer is given below.

Total execution time: The total execution time of WODRL load balancer with set of end devices and fog nodes $TET(ED, WODRL, FN)$ is defined as the summation of the time taken to perform first level optimization using double reinforcement learning and second level optimization using whale optimization for scheduling the requests from end devices among best fit appropriate fog nodes.

$$TET(ED, WODRL, FD) = \sum_{i=1}^{i=m} \left[T_{ED_i} + T_{FN_i} \right] + T_{WODRL}$$

Where, T_{ED_i} is the time taken to process end device request, T_{FN_i} is the time taken to process the load on fog nodes, and T_{WODRL} is the time taken by the WODRL load balancer to distribute the end devices requests among the fog nodes.

Response time: The response time of WODRL load balancer with set of end devices and fog nodes $RT(ED, WODRL, FD)$ is the time difference between submission of the request of the end devices and obtaining the response from the WODRL load balancer.

$$RT(ED, WODRL, FD) = \sum_{i=1}^{i=m} \left[T_{FN_i} - T_{ED_i} \right]$$

Resource utilization rate: The resource utilization rate of WODRL load balancer with set of end devices and fog nodes $RU(ED, WODRL, FD)$ is the ratio of the amount of resources used among the allocated resources by the WODRL load balancer.

$$RT(ED, WODRL, FD) = \sum_{i=1}^{i=m} \left[\frac{RU_i}{RA_i} \right] * 100$$

Where $RA_i$the amount of resources is allocated to the fog node and RU_i is the amount of resources used by the fog node.

3.0 Proposed Design of the WODRL

A high-level architecture of the proposed WODRL load balancer is shown in Fig. 1. It consists of set of end devices which are connected through internet and are fed as input to the WODRL fog load balancer. The WODRL fog load balancer processes the input requests from the end devices then applies double reinforcement learning, the policies formed by the reinforcement learning is further optimized using whale optimization technique. The policies formed by WODRL fog load balancer maps the requests generated by the end devices to appropriate fog nodes efficiently. The main purpose of applying double reinforcement's is one action complements another action and benefits in producing optimal policy by harnessing memory lessness property. By applying double reinforcement to the actor critic method of reinforcement learning leads to better estimate of the Q values and also increases the overall convergence rate of the algorithm. The high level working of WODRL is given in Algorithm 1.

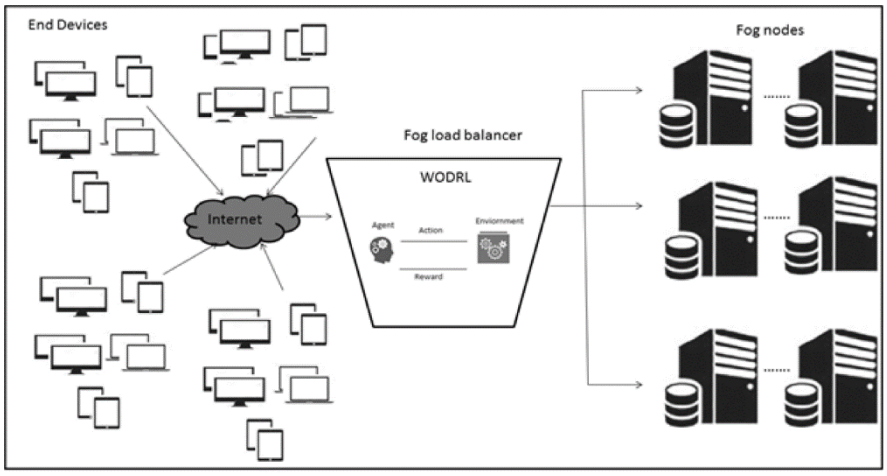

Fig. 1. High level architecture of WODRL load balancer

Algorithm: Working of WODRL

1: **Start**
2: **Input:** Requests from end devices, $ED = \{ED_1, ED_2, ED_3, \ldots, ED_m\}$, set of fog nodes $FN = \{FN_1, FN_2, FN_3, \ldots, FN_m\}$,
3: **Output:** Second level optimized Load balancing policies,
$S - LDP(ED \rightarrow FD) = \{LDP_1, LDP_2, \ldots, LDP_m\}$
4: **For** every ED_i in ED **do**
5: Initialize two Q states $Q^A(S,a), Q^B(S,a)$, reward r, Action set *A*
6: Perform a random action a and update either $Q^A(S,a)$ or $Q^B(S,a)$
7: **If** update $Q^A(S,a)$ **then**
8: Compute $a^* = argmax_a\ Q^A(S`, a)$
9: Update $Q^A(S,a) = Q^A(S,a) + \alpha(S,a) * (r + \gamma Q^B(S`, a) - Q^A(S,a))$
10: **Else If** update $Q^B(S,a)$ **then**
11: Compute $b^* = argmax_a\ Q^B(S`, a)$
12: Update $Q^B(S,a) = Q^A(S,a) + \alpha(S,a) * (r + \gamma Q^A(S`, b^*) - Q^B(S,a))$
13: **End IF**
14: Update s to $S`$
15**: End For**
16: Generate first level optimized load balancing policies,
$F - LDP(ED \rightarrow FD) = \{LDP_1, LDP_2, \ldots, LDP_m\}$
17: Initialize the population of whale search agents
$\mathrm{WSA} = \{\mathrm{WSA}_1, \mathrm{WSA}_2, \mathrm{WSA}_3, \ldots \mathrm{WSA}_\mathrm{m}\}$
18: Identify the best whale search agent WSA_i^*
19: Initialize the whale action set WA and whale decision set WD to NULL
20: **For** minimum number of iterations **do**
21: **For** each whale search agent $\mathrm{WSA_i}$ **do**
22: **If** $|\mathrm{WSA_i}| \leq 1$ **then**
23: Update the position of the current whale search agent
$\mathrm{P(WSA_i)} = \mathrm{WSA}_i^* - \mathrm{WA*WD}$
24: **Else**
25: Update the position of the random whale search agent
26: $\mathrm{P(WSA_i)} = \mathrm{WSA}_i^{rand} - \mathrm{WA*WD}$
27: **End IF**
28: **End For**
29: **End For**
30: Update the whale search agent position $\mathrm{P(WSA_i)}$ to $P`(\mathrm{WSA_i})$

31: Generate second level optimized load balancing policies,
$\Pi(ED \rightarrow FD) = \{\Pi_1, \Pi_2, \ldots., \Pi_m\}$
32: **Stop**
33: **End**

4.0 Results and Discussion

Expected value analysis

The performance of the Proposed Work (PW) WODRL load balancer is evaluated on metrics such as total execution time, response time, and resource utilization under limited and unlimited fog computing environment against the recent Existing Work (EW) [12]. In limited fog computing environment limited number of requests from end devices and limited number of fog nodes are considered for evaluation purpose. The expected value analysis towards the is carried out in three-time intervals $< T >, < T - \delta >,$ and $< T + \delta >$

Consider a typical finite fog computing environment with m number of input requests from end devices $ED = \{ED_1, ED_2, \ldots., ED_m\}$, m number of fog nodes $FN = \{FN_1, FN_2, FN_3, \ldots., FN_m\}$, and 'm' number of load balancing policies $LDP(ED \rightarrow FD) = \{LDP_1, LDP_2, \ldots., LDP_m\}$.

Total Execution Time

The $\mathbb{EV}(TET(ED, WODRL, FD))$ of the *WODRL* load balancer is influenced by the different values of the T_{ED_i}, T_{FN_i}, and T_{WODRL}. In which T_{WODRL} influences more on the execution time of the load balancer.

$$\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{T_{ED_i},T_{FN_i},T_{WODRL}}, T\right) = \int_x^y \frac{\sum \Pi_i \in \Pi \, TET(ED,WODRL,FD)}{|T_{ED_i},T_{FN_i},T_{WODRL}|}$$

$$\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{T_{ED_i},T_{FN_i},T_{WODRL}}, T\right) = \sum_{a \in A} a \int_x^y \frac{\sum \Pi_i \in \Pi \, TET(ED,WODRL,FD)(\Pi_i)}{|T_{ED_i},T_{FN_i},T_{WODRL}|}$$

$$\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{T_{ED_i},T_{FN_i},T_{WODRL}}, T\right) = \frac{E\mathbb{V}(1 * T_{st}, T_{pm}, T_{\pi} * T\_ET(\mathrm{t_k}, \mathrm{pm_k}, \mathrm{VM}))}{P(T_{ED_i},T_{FN_i},T_{WODRL})}$$

$$\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{T_{ED_i},T_{FN_i},T_{WODRL}}, T\right) = \int_p^z Z * dP \frac{p}{T_{ED_i} + T_{FN_i} + T_{WODRL}}$$

PW: $\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{\Pi}, T - \delta\right) \cong low > 0$, $\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{\Pi}, T\right) \cong low = 0$, and $\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{\Pi}, T + \delta\right) \cong low < 0$.

EW: $\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{\Pi}, T - \delta\right) \cong low < 0$, $\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{\Pi}, T\right) \cong low > 0$, and $\mathbb{EV}\left(\frac{TET(ED,WODRL,FD)}{\Pi}, T + \delta\right) \cong low < 0$.

The expected value of the PW towards total execution time performance metric is found to be much lower compared to the total execution time of the EW.

Response time

The $\mathbb{EV}(RT(ED, WODRL, FD))$ of the *WODRL* load balancer is influenced by the different values of the T_{ED_i}, T_{FN_i}. In which T_{FN_i}influences more on the response time of the load balancer.

$$\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{T_{ED_i},T_{FN_i}}, T\right) = \int_x^y \frac{\sum \Pi_i \in \Pi \, RT(ED,WODRL,FD)}{|T_{ED_i},T_{FN_i}|}$$

$$\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{T_{ED_i},T_{FN_i}}, T\right) = \sum_{a \in A} a \int_x^y \frac{\sum \Pi_i \in \Pi \, RT(ED,WODRL,FD)(\Pi_i)}{|T_{ED_i},T_{FN_i}|}$$

$$\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{T_{ED_i},T_{FN_i}}, T\right) = \frac{E\mathbb{V}(1 * T_{st}, T_{pm}, T_{\pi} * RT(\mathrm{t_k}, \mathrm{pm_k}, \mathrm{VM}))}{P(T_{ED_i},T_{FN_i})}$$

$$\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{T_{ED_i},T_{FN_i}}, T\right) = \int_p^z Z * dP \frac{p}{T_{ED_i},T_{FN_i}}$$

PW: $\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{\Pi}, T - \delta\right) \cong low > 0$, $\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{\Pi}, T\right) \cong low = 0$, and $\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{\Pi}, T + \delta\right) \cong low < 0$.

EW: $\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{\Pi}, T - \delta\right) \cong low < 0$, $\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{\Pi}, T\right) \cong low > 0$, and $\mathbb{EV}\left(\frac{RT(ED,WODRL,FD)}{\Pi}, T + \delta\right) \cong low < 0$.

The expected value of the PW towards response time performance metric is found to be much lower compared to the response time of the existing work.

Resource utilization rate

The $\mathbb{EV}(RU(ED, WODRL, FD))$ of the *WODRL* load balancer is influenced by the different values of the RU_i *and* RA_i. In which RA_i influences more on the resource utilization rate of the load balancer.

$$\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T\right) = \int_x^y \frac{\sum \pi_i \in \pi \, RU(ED,WODRL,FD)(\pi_i)}{|RA_i|}$$

$$\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T\right) = c \int_x^y \frac{\sum_{\pi_i \in vm_i^{res\_al}} RU(ED,WODRL,FD)(\pi_i)}{|RA_i|}$$

$$\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T\right) = \frac{\mathbb{EV}(1 * N_{UU_i} + N_{OU_i} * RU(ED,WODRL,FD)}{P(RA_i)}$$

$$\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T\right) = \int_p^z Z * dP \frac{p}{\pi}$$

$$\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T\right) = \int_p^z Z * dP \frac{p}{RA_i}$$

PW: $\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T-\delta\right) \cong high > 0$, $\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T\right) \cong high > 0$, and $\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T+\delta\right) \cong high < 0$.

EW: $\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T-\delta\right) \cong high > 0$, $\mathbb{EV}\left(\frac{RU(ED,WODRL,FD)}{RA_i}, T\right) \cong high > 0$, and $\mathbb{E}\left(\frac{RU(\mathrm{t_k,pm_k,VM})}{RA_i}, T+\delta\right) \cong high == 0$.

The expected value of the PW towards resource utilization rate performance metric is found to be much higher compared to resource utilization rate of the EW.

Simulation results

The iFogSim simulator is used for simulating the scenario, which allows modeling and simulation of applications related to IoT devices, mobile devices, and edge computing devices. The number of fog nodes are fluctuated from 10 to 100 which are distributed among larger geographical area and the number of incoming requests from the end devices are fixed to 20 thousand.

Total Execution Time

A graph of number of fog nodes versus total execution time is given in Fig. 2. The total execution time of the proposed work remained consistently lower when they are exposed more number of fog nodes as the algorithm designed exhibits high exploitation ability and does not perform unnecessary search operation over the large state space of fog nodes. But the total execution time of the existing work is lower when the number of fog nodes considered for evaluation is less but as soon as the number of fog nodes increases the total execution time incurred also increased as the existing work does not achieve optimal tradeoff between exploration and exploitation phases of the search operation to find optimal fog node for the incoming task requests.

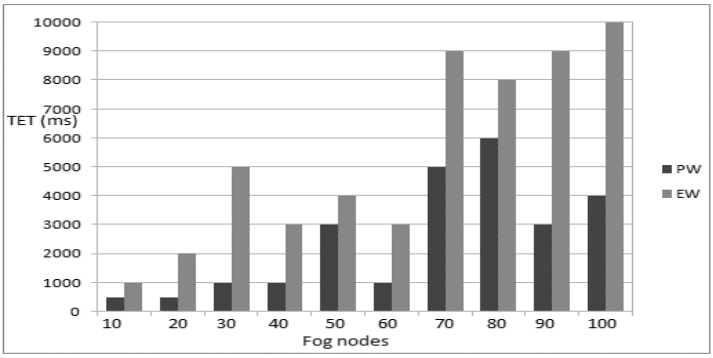

Fig. 2. Number of fog nodes versus total execution time

Response time

A graph of number of fog nodes versus response time is given in Fig. 3. It is observed from the graph that the response time of the proposed work is lower when they are exposed to lower number of fog nodes and the response time remained consistently lower even after they are exposed to higher number of fog nodes as the convergence rate of the designed algorithm is high and improves the quality of solution over each epoch of training. Whereas the response time of the existing work is found to be lower when they are exposed to lesser number of fog nodes but as soon the number of fog nodes increases the response time increases steeply as the convergence rate of the existing algorithm is very high and also exhibits tendency of converging to suboptimal solutions.

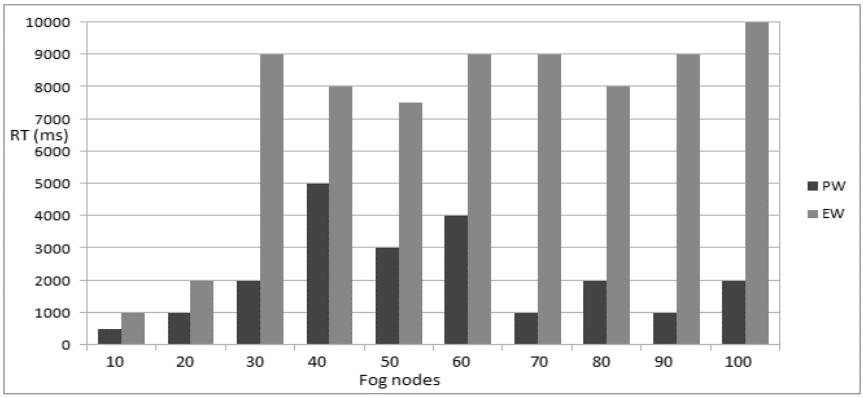

Fig. 3. Number of fog nodes versus response time

Resource utilization rate

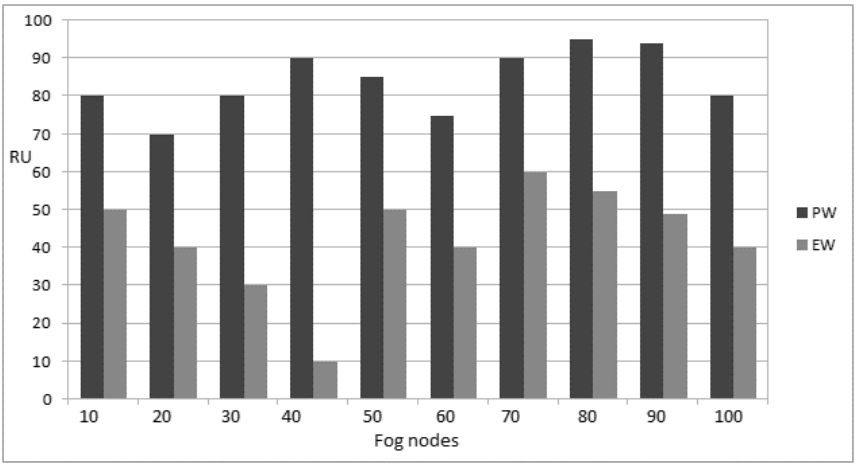

Fig. 4. Number of fog nodes versus resource utilization rate

A graph of number of fog nodes versus resource utilization rate is given in Fig. 4. It is observed from the graph that the resource utilization rate of the proposed work is consistently high over the increase in the number of fog nodes as the designed algorithm involves minimum number of parameter adjustment which results highly accurate load balancing decisions. Whereas the resource utilization rate of the existing work is found to be lower even when they are exposed to a smaller number of fog nodes or a greater number of fog nodes as the existing algorithm involve a greater number of parameter adjustments which hinders the accuracy of load balancing and also causes more computation overhead.

The performance of the proposed WODRL load balancer is found from the expected value analysis and simulation results as the total execution time and response time incurred is less and resource utilization rate is high. The use of double reinforcement learning to formulate first level load distribution policies is useful in terms of efficiency as the policy value estimated finds a defensive strategy to handle the load imbalance situations. On top of first level optimized load distribution policies, whale optimization methodology is applied to draw second level optimized load balancing policies is useful in terms of accurate mapping of requests to fog nodes as the whale search agents are good at exploring the large state space and pick only the potential and global optimal solutions.

4.0 Conclusion

The paper discusses the load balancing in fog computing and its challenges. A novel WODRL based load balancing scheme is proposed for fog computing environment which prevents the situations of overloading or underloading resources through efficient mapping of tasks among the fog nodes. Expected value analysis of the proposed WODRL is carried out towards the performance metrics like total execution, response time, and resource utilization. The performance of the proposed WODRL is tested using the iFogSim simulator by comparing it with the recent existing load balancers. The overall results obtained are found to satisfactory as two levels of optimization are performed by the load balancer to arrive at accurate load balancing policies. Further the proposed WODRL load balancing scheme can extended to address the scheduling, and resource provisioning issues in fog computing.

References

1. A Yousefpour, C Fung, T Nguyen, K Kadiyala, F Jalali, A Niakanlahiji, J Kong, J P Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, *Journal of Systems Architecture*-2019, 98, 289-330
2. R Mahmud, R Kotagiri, R Buyya, Fog computing: A taxonomy, survey and future directions, *In Internet of everything, Springer* - 2018, 103-130
3. F Haouari, R Faraj, J M AlJa'am, Fog computing potentials, applications, and challenges, In *2018 International Conference on Computer and Applications*-2018, 399-406
4. M I Bala, M A Chishti, Survey of applications, challenges and opportunities in fog computing, *International Journal of Pervasive Computing and Communications*-2019, 15 (2), 80-96
5. D Puthal, M S Obaidat, P Nanda, M Prasad, S P Mohanty, A Y Zomaya, Secure and sustainable load balancing of edge data centers in fog computing, *IEEE Communications Magazine*-2018, 56 (5), 60-65
6. S P Singh, R Kumar, A Sharma, A Nayyar, Leveraging energy- efficient load balancing algorithms in fog computing, *Concurrency and Computation: Practice and Experience*-2020, e5913

7. B Recht, A tour of reinforcement learning: The view from continuous control, *Annual Review of Control, Robotics, and Autonomous Systems*-2019, 2, 253-279
8. M Botvinick, S Ritter, J X Wang, Z K Nelson, C Blundell, D Hassabis, Reinforcement learning, fast and slow, *Trends in cognitive sciences*-2019, 23 (5), 408-422
9. S Mirjalili, A Lewis, The whale optimization algorithm, *Advances in engineering software* 2016, 95, 51-67
10. J Nasiri, F M Khiyabani, A whale optimization algorithm (WOA) approach for clustering, *Cogent Mathematics & Statistics*-2018, 5 (1), 1483565
11. K Bhargavi, B Sathish Babu, Soft-set based Double-Deep-Q Scheduler for Optimal Task Scheduling under Long-term Uncertainty in the Cloud, *International Journal of Science, Technology, Engineering and Management-A VTU Publication*-2019, 1(2), 43-55
12. K Bhargavi, B Sathish Babu, Uncertainty Aware Resource Provisioning Framework for Cloud Using Expected 3-SARSA Learning Agent: NSS and FNSS Based Approach, *Cybernetics and Information Technologies*-2019, 19 (3), 94-117
13. N Tellez, M Jimeno, A Salazar, E Nino-Ruiz, A tabu search method for load balancing in fog computing, *Int. J. Artif. Intell*-2018, 16 (2), 78-105
14. A B Manju, S Sumathy, Efficient load balancing algorithm for task preprocessing in fog computing environment, In *Smart Intelligent Computing and Applications, Springer* -2019, 291-298
15. M Zahid, N Javaid, K Ansar, K Hassan, M K U Khan, M Waqas, Hill climbing load balancing algorithm on fog computing, In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Cham*-2018, 238-251
16. X Xu, S Fu, Q Cai, W Tian, W Liu, W Dou, X Sun, A X Liu, Dynamic resource allocation for load balancing in fog environment, *Wireless Communications and Mobile Computing*-2018