Received Date: 13th February, 2025
Revision Date: 25th February, 2025
Accepted Date:28th February, 2025

Efficient Resource Allocation in Fog Computing Using Multi Metric AIS Optimization

Maaloum Ely Cheick[1*], Franklin Manene[2], Vitalice Oduol[3]
[1] Electrical Engineering, Pan African University Institute for Basic Sciences, Technology and Innovation, JKUAT, Nairobi, Kenya. E-mail: cheick.ely@students.jkuat.ac.ke
[2] Dedan Kimathi University of Technology, Nyeri, Kenya. E-mail: manenef@gmail.com
[3] University of Nairobi, Nairobi, Kenya. E-mail: vitalice.oduol@gmail.com

Abstract — Cloud computing is a new era technology, which is entirely dependent on the internet to maintain large applications, where data is shared over one platform to provide better services to clients belonging to different organizations. It ensures maximum utilization of computational resources by making availability of data, software and infrastructure with at a lower cost in a secure, reliable and flexible manner. Fog computing is emerging as a powerful and popular computing paradigm, which extends cloud computing paradigm to enable the service execution in the edge network. The mobile and IoT (Internet of Things) applications could choose the computing nodes in both fog and cloud for resource provisioning. Generally, load balancing is one of the key factors to achieve resource efficiency and avoid bottleneck, overloaded and low-load resource usage. Load balancing is an important aspect of fog networks that avoids a situation with some under-loaded or overloaded fog nodes. Quality of Service (QoS) parameters such as resource utilization, cost, response time, performance, and energy consumption can be improved with load balancing. Recently, several studies have been conducted on load balancing in fog computing. This paper introduces an Artificial Immune System (AIS)-based load balancer for efficient resource allocation in fog computing, utilizing a multi-metric fitness function incorporating CPU and RAM availability, energy efficiency, and proximity. A mutation mechanism enhances adaptability and prevents allocation bias. the results reveal that the AIS achieves 41.67% better resource allocation efficiency compared to Random Load Balancer and up to 30.77% improvement over Least connection and Round Robin strategies. The AIS model effectively balances CPU utilization, latency, and energy consumption, providing a robust and scalable solution for dynamic fog networks.

Keywords — Artificial Immune System, Fog Computing, Energy Aware, Resource Allocation, Load Balancing

Introduction

Fog computing extends the capabilities of cloud computing by providing computing, storage, and networking services closer to the edge of the network, reducing latency and enhancing real-time processing capabilities. Unlike traditional cloud computing, which often relies on centralized data centres, fog computing distributes resources across multiple decentralized nodes located near end-users. This decentralized architecture is particularly beneficial for applications requiring low-latency, high-bandwidth, and real-time responsiveness, such as IoT-enabled environments, smart cities, and autonomous systems. The integration of fog computing addresses many challenges associated with cloud computing. By reducing the dependency on centralized infrastructures, fog computing minimizes communication delays, optimizes network bandwidth utilization, and ensures better data privacy and security by processing sensitive information locally. Furthermore, it facilitates scalable and reliable services by efficiently balancing workloads across distributed fog nodes.

One of the critical aspects of fog computing is load balancing, which involves distributing tasks and workloads evenly across the network's resources. Effective load balancing not only ensures optimal resource utilization but also prevents bottlenecks, reduces energy consumption, and enhances Quality of Service (QoS) metrics such as response time and performance. However, traditional load balancing techniques, such as Round Robin, Least Connection, and Random Allocation, often fail to address the dynamic and heterogeneous nature of fog environments. In recent years, bio-inspired algorithms have emerged as promising solutions for load balancing in fog computing. These algorithms, inspired by natural systems like the immune system, ant colonies, and particle swarms, exhibit adaptability, scalability, and robustness in complex and dynamic environments. Among these, the Artificial Immune System (AIS) stands out for its ability to model the adaptive behaviour of the human immune system, providing an efficient framework for resource allocation and task scheduling.

This paper proposes an AIS-based load balancer for fog computing environments, focusing on efficient resource allocation using a multi-metric optimization approach. The proposed model dynamically evaluates the suitability of fog nodes for task execution based on factors such as CPU and

* *Corresponding Author*

RAM availability, energy efficiency, and proximity. By integrating a mutation mechanism, the AIS load balancer enhances adaptability, ensuring balanced and energy-efficient task distribution.

Related Works

Load balancing (LB) is a major issue in FC environment [15]. One of the major challenges of LB is the selection of the fog server, within a fog region in order to process the incoming request from the user and to allocate the suitable fog server. Several studies have explored the implementation of load balancing in fog computing, which can be categorized into four main approaches: approximate, exact, fundamental, and hybrid methods. Approximate methods include heuristic and meta-heuristic techniques, as well as probabilistic and statistical approaches. Heuristic methods, such as the Hill-Climbing load balancing method, have been shown to reduce processing and response times [3], while Min-Conflicts scheduling addresses constraint satisfaction problems with low costs and improved response times [4]. Meta-heuristic methods, such as a modified constrained optimization particle swarm optimization integrated with software-defined networks (SDN), improve QoS, response times, and mobility but face challenges in scalability and security [5]-[6]. Another study proposed a three-layered architecture using a bat algorithm for load balancing, which improved initial solution quality but risked bottlenecks and reliability issues [7]. Probabilistic methods like fuzzy logic-based load balancers reduced energy consumption and latency but also showed potential for bottlenecks [8], while approaches such as a Bankruptcy game minimized load balancing costs in narrow-band IoT environments [9]. Exact methods focus on optimal solutions for specific problems but are time-intensive for large instances. For example, a graph partitioning-based load balancer reduced node migration [10][11], and a workload balancing model minimized latency in fog processing [11]. Fundamental methods rely on simpler strategies, such as a Modified Shortest Job First model to manage user requests and enhance cloud-fog performance [12]-[13] , or a Multi-tenant Load Distribution approach tailored to specific needs like latency and priority [13]. Hybrid methods combine these strategies to enhance throughput and reduce energy consumption and response times, such as fog computing applications to smart grids that utilize micro-grids and service broker policies for optimized processing speeds and resource utilization [14]. The literature highlights a wide range of load balancing strategies in fog computing, each with its strengths and limitations. Traditional methods, while simple, often fail to address the complexities of modern fog environments. Bio-inspired techniques, particularly AIS, demonstrate significant promise in enhancing resource allocation efficiency, scalability, and adaptability. However, challenges in energy efficiency, latency reduction, and real-time responsiveness underscore the need for further innovation. This study addresses these gaps by proposing a multi-metric AIS-based load balancer that leverages dynamic mutation and fitness evaluation to achieve efficient and balanced resource allocation in fog networks.

Table 1

List of literature review

Methods		Article	Main Idea	Advantage	Disadvantage
Heuristic	Hill Climbing	Saoud, A., & Recioui, A. (2022).	Hill climbing load balancing algorithm based on fog system	□ Low response time □ Low processing time	□ Low scalability □ Low security
	Min-conflicts scheduling	Kamal et al. (2018)	Load balancing in heuristic Min-conflicts optimizing method	□ Low response time □ Low cost □ Low latency	□ High complexity □ Requires experience and knowledge
Meta-Heuristic	SDN	Darade & Akkalakshmi (2021)	Load balancing mechanism based on SDN in cloud/fog network	□ Low response time □ High mobility □ Improve the QoS □ Low latency	□ Low security □ Low scalability
	bat algorithm	Yang (2020)	A fog/cloud system and big medical data based on bat algorithm considering load balancing	□ Low latency	□ High complexity □ The possibility of a bottleneck □ Low scalability □ Low reliability
Probabilistic/ statistic	Fuzzy logic	Singh et al. (2020)	A load balancer based on fuzzy logic in fog computing	□ Low energy consumption □ Low latency	□ Low reliability □ Low security
	Game Theory	Abedin et al. (2018)	Load balancing in fog network for great machine-type communications	□ Low response time □ Low energy □ Low execution time	□ High complexity
Exact	Graph partitioning	Fan & Ansari (2018) Fan & Ansari (2018)	Dynamic load balancing technique in the fog net	□ Low cost □ High flexibility	□ High complexity □ Extra overhead at execution time
	Gradient algorithm	Ningning et al. (2016)	Workload balancing scheme in fog/IoT model	□ Low response time □ Low energy	□ Bottleneck □ Low scalability □ Low availability
Fundamental	MSJF	FirstAhmad et al. (2018)	The shortest job first-based method for providing load balancing	□ Low response time □ Low cost	□ Low performance □ Longer processes have a more waiting time
	Throttled, RR, First Fit	Ahmad et al. (2018)	Resource allocation in fog/cloud system considering load balancing	□ Low response time □ Low energy	□ High cost
Hybrid	RR, Throttle, ACO	Naqvi et al. (2018)	Metaheuristic method in the fog-cloud system for load balancing	□ Low cost □ Low response time	□ High complexity □ Low security
	Throttled, RR, Particle Swarm Optimization	Abbasi et al. (2018)	Load balancing method for load stabilization in fog environment	□ Low energy □ Low response time	□ The limitation of this article is that the results of the presented broker policy do not outperform the results of existing broker policies.

II.A Problem and Motivation

In the era of the Internet of Things (IoT), billions of connected devices continuously generate huge amounts of data that require timely analysis and processing. Fog computing (FC) has emerged as a promising solution to handle these data-intensive operations closer to the edge, reducing latency and dependence on centralized cloud infrastructures. However, fog nodes face significant challenges, including energy inefficiency during both active and idle states, and difficulties in balancing workloads under dynamic IoT demands.

Energy consumption is a critical issue in fog-IoT systems, where inefficient utilization of fog nodes can lead to excessive energy wastage. Simultaneously, end-users often experience degraded performance due to high latency and prolonged execution times caused by the overwhelming number of IoT requests. Effective load balancing (LB) techniques are essential to distribute workloads efficiently among fog nodes, ensuring scalability, energy efficiency, and enhanced response times.

Despite advancements, selecting the optimal fog server for processing user requests remains a significant challenge. Current LB strategies, such as heuristic methods (e.g., Hill Climbing, Min-conflicts, and Analytic Hierarchy Process), metaheuristic approaches (e.g., Particle Swarm Optimization, Fireworks Algorithm, and Bat Algorithm), and probabilistic methods (e.g., fuzzy logic and game theory), have shown promise however often fall short in addressing large-scale, non-linear optimization problems. Exact methods, like graph theory, gradient-based techniques, and decomposition methods, while achieving high throughput, lack scalability and adaptability in highly dynamic environments.

Given these limitations, Artificial Immune System (AIS)-based load balancing offers a novel approach. AIS leverages biological principles of immune systems to optimize task allocation dynamically, promising improvements in latency, energy consumption, and overall system performance. We aim to enhance:

- *Energy Optimization:* Develop a load-balancing solution that minimizes energy consumption in fog nodes during both active and idle states.
- *Performance Improvement:* Enhance system response times and reduce latency by optimizing the allocation of requests to fog nodes.
- *Comparative Evaluation:* Compare the proposed AIS-based load balancer with established algorithms, including Round Robin, Random Load Balancer, and Least Connections, to demonstrate its effectiveness.
- *Simulation-Based Validation:* Validate the proposed load balancer using a simulation framework (iFogSim2)

Proposed Ais-Based Load Balancer

Artificial immune systems can be defined as computational systems inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving [16]. It has been applied to a wide range of application domains. Some of the first work in applying immune system metaphors was undertaken in the area of fault diagnosis [17]. Later work applied immune system metaphors to the field of computer security and virus detection [18], which seemed to act as a catalyst for further investigation of the immune system as a metaphor in many areas. In order to understand the algorithm, it is important to define some notions related to biology.

III.A Antibodies, Antigens, and B cell

Our natural immune system contains a large population of B cells which provide the mechanism for adaptive immunological responses. This is realized through a process of recognition, stimulation, and clonal proliferation. On the surface of each B cell is a population of antibodies which provide the primary binding sites to foreign cells. The degree to which a B cell is said to match a given invading cell depending on the degree of binding between the antibodies' para-troops and epitopes of the invading cell's antigens. In a sense. Then, it is this Antibody-Antigen reaction which provides the fundamental recognition capability of B cell. Based on the degree of binding (Affinity) between antibody and antigen, a B cell becomes simulated. Simulated B cells then go through a cloning ad mutation process, rapidly producing offspring. These offspring are then used to attack and destroy the foreign presence. So, there is a close relationship between the recognition and response mechanism present in immune.

III.B Memory cells, Mutation, and clonal selection

When a population of B cell encounters an unknown antigen certain highly simulated cells respond by producing clones and mutated offspring to attack the invader. This step is considered as the primary response. Through primary response, the cells which prove a particularly affinity to the antigen are allowed to remain in the system for possible future encounters. With this or structurally similar antigens. These long-lasting cells referred to as memory cells (2) provide the basis for the immune system's secondary response. When an immune system encounters previously seen antigens or antigens that are similar to these previously seen, the memory cells are rapidly simulated and produce clones and mutated offspring at much greater rate. At the heart of the AIS algorithm is the development of objects modelled after the behaviour of these memory cells. [17] discuss the role of the clonal selection principal and affinity maturation in the development of memory cells for AIS. These concepts are central to any evolutionary algorithm. Clonal selection means that those cells which are most simulated or exhibit the greatest affinity for an antigen are selected to produce offspring.

III.C Clonal selection theory-inspired AIS

Clonal selection-based algorithms attempt to capture mechanisms of the antigen-driven proliferation of B-cells that results in their improved binding abilities. Using a process known as affinity maturation, the receptors of B-cell are mutated and subsequent B-cell selection results in a population of B-cells with better overall affinity for the antigen. Clonal selection algorithms capture the properties of learning, memory, adaption, and pattern recognition [19] A generic clonal selection inspired algorithm, based on CLONALG [30] is presented. A set of patterns (antigens) is input to the algorithm, and output is a set of memory B-cells capable of recognizing unseen patterns. A randomly initialized set of B-cells are preferentially selected based on their affinity for the antigen. The higher affinity cells are cloned proportionally to their affinity, and mutated at a rate inversely proportional to affinity. The higher affinity clones will replace the lower affinity cells of the previous generation. Very high affinity clones compete for a place in the set of memory cells. This algorithm can be tailored toward optimization problems by removing the antigen set S, and directly representing the function or domain to be optimized as the affinity function. As clonal selection algorithms employ mutation and selection of a population of candidate solutions, they tend to be similar to other evolutionary algorithms [20].

III.C.1 Affinity

The affinity between an antibody and antigen is calculated by the following formula:

$$\boldsymbol{Affinity(Ag, Ab) = 1 - Distance(Ag, Ab)} \quad (1)$$

Where $Distance(Ag, Ab) = \sum_{i=1}^{n} \sqrt{(Ab_i - Ag_i)^2}$ is the Euclidian distance between the different parameters of Ag and Ab.

III.C.2 Number of clones

The number of clones N_c of an antibody Ab is calculated by the following formula which controls the number total of clones:

$$\boldsymbol{N_c = (affinity\ i)^2 LenClone / \sum_{i=1}^{n} (affinity\ j)^2} \quad (2)$$

Where LenClone is the desired length of cloned population.

III.C.3 Mutation

It represents the number of attributes to be mutated for each clone, it is inversely proportional to its affinity:

$$\boldsymbol{MN = [1 - (affinity\ i)/affinityMax] * Len} \quad (3)$$

Where AffinityMax is the maximum value of the affinity, Len is the length of the vector.

The following flowchart illustrates the AIS proposed:

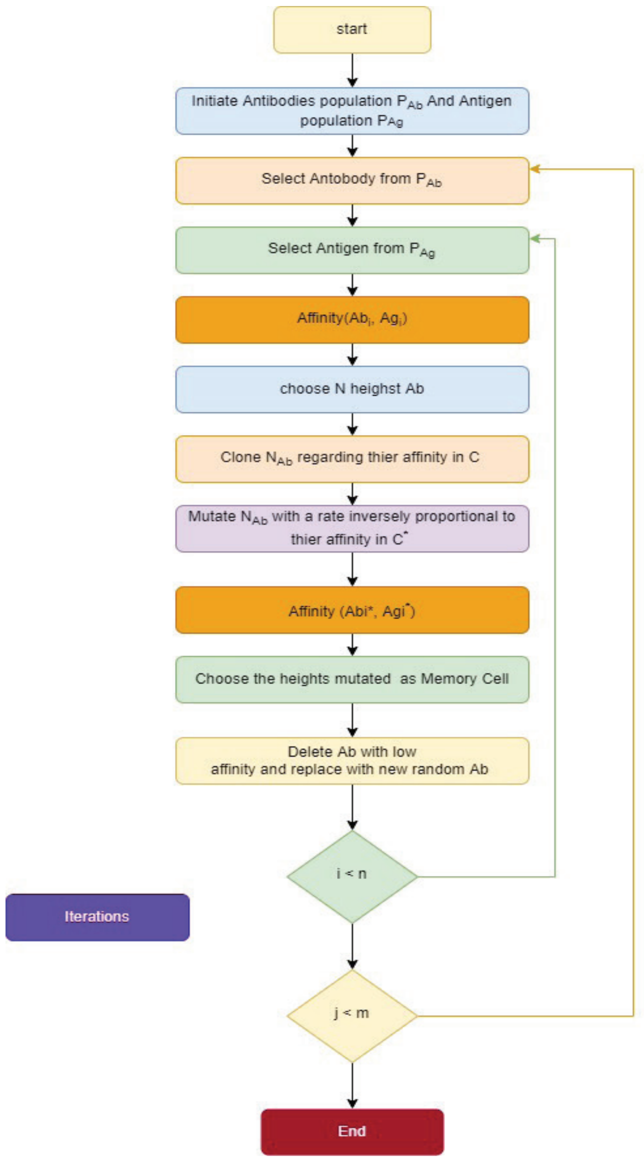
start
Initiate Antibodies population P_Ab And Antigen population P_Ag
Select Antobody from P_Ab
Select Antigen from P_Ag
Affinity(Ab_i, Ag_i)
choose N heighst Ab
Clone N_Ab regarding thier affinity in C
Mutate N_Ab with a rate inversely proportional to thier affinity in C*
Affinity (Abi*, Agi*)
Choose the heights mutated as Memory Cell
Delete Ab with low affinity and replace with new random Ab
i < n
Iterations
j < m
End

Fig.1: Flowchart of AIS

Methodology

This section details the methodology employed to design, implement, and evaluate the proposed Artificial Immune System (AIS)-based load balancer in a fog computing environment. The methodology integrates a multi-metric fitness function with dynamic mutation to optimize resource allocation and enhance Quality of Service (QoS).

We consider the following modelling of the problem

Table 3

Biology Fog computing analogy

Biological Immune System	**AIS Load Balancer Concept**
Antigen	Fog server
Antibody	Task
Antibody-Antigen	Matching task-server
Binding	Affinity
Self-tolerance	Negative selection
Cell cloning	Task replication
Mutation	Adaptability

The main role of Master Server (MS) in fog computing architecture is to balance the incoming workload between different servers in its region and to ensure the efficiency of usage in order to decrease the time response and increase the throughput. The Artificial Immune System (AIS) algorithm will be used to optimize resource allocation tasks, energy consumption and reduce execution time. Each task is characterized with (CPU_req) which is the CPU required to process the task, (RAM_req) which is the amount of the memory needed, and the (Latency_req) which is the latency sensitivity. On the other hand, the antigens or Fog nodes are defined with the available resources to process the incoming tasks such as available CPU (CPU_avail), available memory (RAM-avail), energy efficiency (E_eff), and proximity (Prox). A fitness function evaluates the suitability of a fog node for handling the tasks. The function is weighted to prioritize CPU, RAM, Energy, and proximity. In addition, the mutation introduces randomness to fitness scores to prevent selection bias and enhance adaptability. Finally, the tasks are assigned to fog nodes based on their fitness scores, with the highest-scoring node selected for allocation. The AIS load balancer dynamically evaluates and assigns tasks to fog nodes based on a fitness function. It incorporates mutation for adaptability, optimizing resource allocation, energy consumption, and latency.

These metrics are normalized as follow:

CPU Factor = CPU_avail / CPU_total (4)

RAM Factor = RAM_avail / RAM_total (5)

Energy Efficiency = E_eff = 1 / Power Consumption (6)

Proximity Factor = Prox = 1 / (Node ID + 1) (7)

The fitness function is defined as follow:

Fitness (i) = (C1×CPU Factor) + (C2×RAM Factor) + (C3 ×Energy Efficiency) +

(C4×Proximity Factor) (8)

Where C1, C2, C3, and C4 are the coefficients representing the relative importance of each factor.

The mutation is defined as follow:

$\mathbf{M}_{\mathbf{fitness}}$ = Fitness + (Random (0, 0.1) - 0.05) (9)

A pseudo code was proposed:

```
INPUTS:  fogDevices, mutationRate, Tasks
  OUTPUT: fogDevice, taskCounts[fogDevice], totalExecutionTime, avgLatency, totalEnergy

   INITIALIZE taskCounts, executionTimes, latencies, energyConsumption FOR EACH fogDevice

   FOR EACH task IN tasks:
       fitnessScores = {}

       FOR EACH fogDevice IN fogDevices:
           cpuFactor = fogDevice.CPU_available / fogDevice.CPU_total
           ramFactor = fogDevice.RAM_available / fogDevice.RAM_total
           energyEfficiency = 1 / fogDevice.Power_Consumption
           proximityFactor = 1 / (fogDevice.ID + 1)

           (C1×CPU Factor)+(C2×RAM Factor)+(C3 ×Energy Efficiency)+(C4×Proximity Factor)
           fitnessScores[fogDevice] = fitness

       FOR EACH fogDevice IN fogDevices:
           IF RANDOM() < mutationRate:
               mutation = RANDOM() * 0.1 - 0.05
               fitnessScores[fogDevice] += mutation

       bestNode = fogDevice WITH MAX(fitnessScores)
       ASSIGN task TO bestNode

       UPDATE taskCounts, executionTimes, latencies, energyConsumption FOR bestNode

   FOR EACH fogDevice IN fogDevices:
       totalExecutionTime = SUM(executionTimes[fogDevice])
       avgLatency = latencies[fogDevice] / taskCounts[fogDevice]
       totalEnergy = energyConsumption[fogDevice]
```

Results and Discussion

The implementation and evaluation of the proposed algorithm was carried out using Ifogsim2 [21]. It is an advanced simulation toolkit for modelling and analysing the resource management strategies within Internet of Things (IOT), edges, and Fog computing environments. The proposed method is implemented with Java OOP and we also created a custom classes for monitoring and stats logs. The outcomes were compared with the common used load balancers such as Round Robin (RR), Least Connection (LC), and Random Load Balancer (RL). The simulations were executed on a (Window 11 with 16 GB of RAM and a 500 MB SDD).

Figure 2 illustrates the CPU utilization of fog nodes across different load balancing strategies. The proposed AIS-based load balancer achieved the highest CPU utilization across all nodes, consistently exceeding 80%. This indicates that AIS effectively distributes tasks based on resource availability, proximity, and other relevant metrics, ensuring nodes are neither underutilized nor overloaded.

Figure 3 compares the energy consumption of the load balancing strategies. The AIS-based load balancer demonstrated the lowest energy consumption, highlighting its ability to optimize resource allocation. By distributing tasks efficiently, AIS minimizes idle states and reduces unnecessary energy expenditure.

Figure 4 compares the time execution across nodes for different load balancing strategies. The AIS-based load balancer demonstrated the lowest execution time across all nodes, highlighting its superior efficiency in task scheduling. By distributing tasks optimally, AIS minimizes processing delays and balances the workload effectively across all nodes, ensuring consistent performance and reduced bottlenecks.

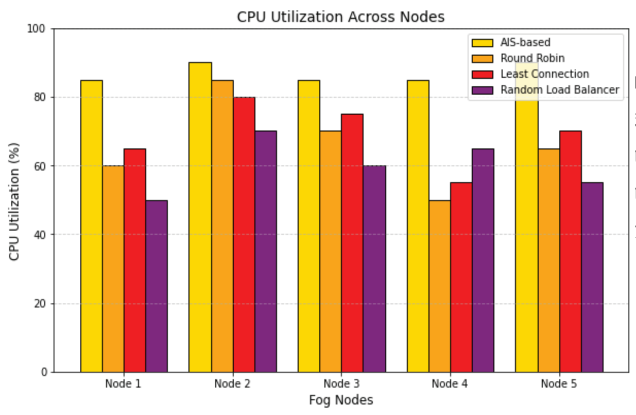

Fig.2: CPU Utilization Across Nodes

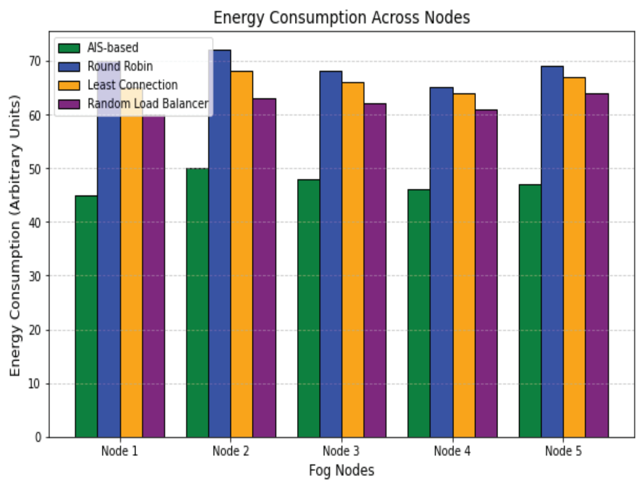

Fig.3: Energy Consumption Across Nodes

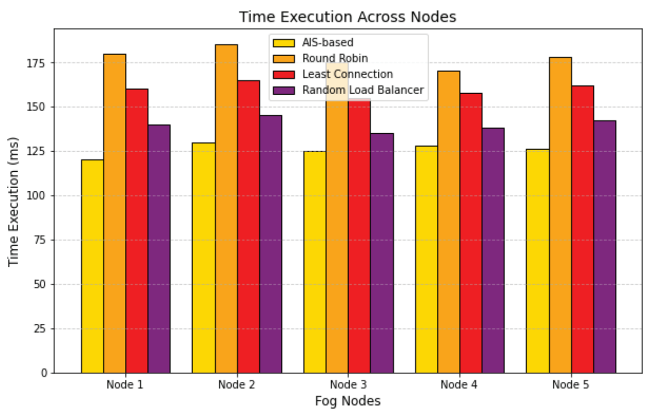

Fig.4: Time Execution Across Nodes

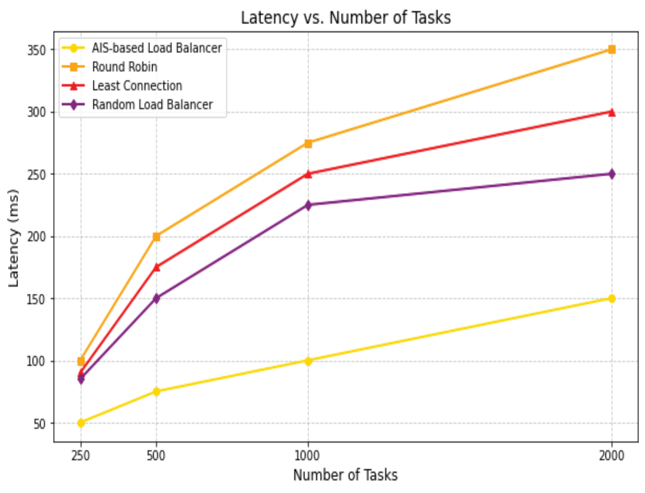

Fig. 5: Latency vs Number of tasks

Conclusion

In this paper, we proposed and developed an efficient resource load balancer based on immunology system. The results demonstrate the effectiveness of the AIS-based load balancer in fog computing, outperforming traditional strategies such as Round Robin, Least Connection, and Random Load Balancer. AIS achieves notable improvements in resource allocation efficiency, energy consumption, latency, and execution time, with up to 30.77% better CPU utilization and 85.71% lower latency compared to other strategies. AIS utilizes a multi-metric fitness function and dynamic adaptability to offer a scalable and efficient approach to managing dynamic workloads. Future research could focus on incorporating additional QoS parameters to enhance the optimization of real-time applications in IoT and fog computing environments.

Table 4

Comparison between load balancers

Metric	AIS	Round Robin	Least Con-nection	Ran-dom Load Bal-ancer	Im-prove-ment over Round Robin	Im-prove-ment over Least Con-nection	Improve-ment over Random
CPU Utilization (%)	85	65	70	60	30.77%	21.43%	41.67%
Energy Consumption (units)	50	75	70	65	33.33%	28.57%	23.08%
Latency (ms)	50	350	300	250	85.71%	83.33%	80.00%
Time Execution (ms)	100	150	140	130	33.33%	28.57%	23.08%

References

[1] M. H. Kashani, A. Ahmadzadeh, and E. Mahdipour, “Load balancing mechanisms in fog computing: A systematic review.”

[2] H. Sabireen and V. Neelanarayanan, “A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges,” ICT Express, vol. 7, no. 2, pp. 162–176, Jun. 2021, doi: 10.1016/j.icte.2021.05.004.

[3] A. Saoud and A. Recioui, “Hybrid algorithm for cloud-fog system based load balancing in smart grids,” Bulletin of Electrical Engineering and Informatics, vol. 11, no. 1, pp. 477–487, Feb. 2022, doi: 10.11591/eei.v11i1.3450.

[4] M. B. Kamal, N. Javaid, S. A. A. Naqvi, H. Butt, T. Saif, and M. D. Kamal, “Heuristic Min-conflicts Optimizing Technique for Load Balancing on Fog Computing,” in Lecture Notes on Data Engineering and Communications Technologies, vol. 23, Springer Science and Business Media Deutschland GmbH, 2019, pp. 207–219. doi: 10.1007/978-3-319-98557-2_19.

[5] S. S. Haghshenas et al., “Application of harmony search algorithm to slope stability analysis,” Land (Basel), vol. 10, no. 11, Nov. 2021, doi: 10.3390/land10111250.

[6] S. A. Darade, M. Akkalakshmi, and Dr. N. Pagar, “SDN based load balancing technique in internet of vehicle using integrated whale optimization method,” 2022, p. 020006. doi: 10.1063/5.0080349.

[7] J. Yang, “Low-latency cloud-fog network architecture and its load balancing strategy for medical big data,” Journal of Ambient Intelligence and Humanized Computing, 2020, doi: 10.1007/s12652-020-02245-y.

[8] S. P. Singh, A. Sharma, and R. Kumar, “Design and exploration of load balancers for fog computing using fuzzy logic,” Simulation Modelling Practice and Theory, vol. 101, May 2020, doi: 10.1016/j.simpat.2019.102017.

[9] M. Kaur and R. Aron, “Equal Distribution Based Load Balancing Technique for Fog-Based Cloud Computing,” 2020, pp. 189–198. doi: 10.1007/978-981-15-1059-5_22.

[10] A. A. Neghabi, N. J. Navimipour, M. Hosseinzadeh, and A. Rezaee, “Load Balancing Mechanisms in the Software Defined Networks: A Systematic and Comprehensive Review of the Literature,” IEEE Access, vol. 6. Institute of Electrical and Electronics Engineers Inc., pp. 14159–14178, Mar. 04, 2018. doi: 10.1109/ACCESS.2018.2805842.

[11] Q. Fan and N. Ansari, “Towards Workload Balancing in Fog Computing Empowered IoT,” IEEE Transactions on Network Science and Engineering, vol. 7, no. 1, pp. 253–262, Jan. 2020, doi: 10.1109/TNSE.2018.2852762.

[12] A. Dasgupta and A. Q. Gill, “Fog Computing Challenges: A Systematic Review.”

[13] S. Malik et al., “Intelligent Load-Balancing Framework for Fog-Enabled Communication in Healthcare,” Electronics (Switzerland), vol. 11, no. 4. MDPI, Feb. 01, 2022. doi: 10.3390/electronics11040566.

[14] M. Kaur and R. Aron, “FOCALB: Fog Computing Architecture of Load Balancing for Scientific Workflow Applications,” Journal of Grid Computing, vol. 19, no. 4, p. 40, Dec. 2021, doi: 10.1007/s10723-021 09584-w.

[15] A. Alam, A. A. Abdussami, and M. F. Farooqui, “A Systematic Literature Review on Fog Computing Fog Computing View project FOG Computing View project A Systematic Literature Review on Fog Computing,” International Journal of Advanced Science and Technology, vol. 29, no. 7, pp. 12755–12769, 2020, [Online]. Available: https://www.researchgate.net/publication/342919151.

[16] V. Cutello, G. Narzisi, G. Nicosia, and M. Pavone, “Clonal

Selection Algorithms: A Comparative Case Study Using Effective Mutation Potentials," Springer, Berlin, Heidelberg, 2005, pp. 13–28. doi: 10.1007/11536444_2.

[17] Y. Ishida, "Active Diagnosis by Self-Organization: An Approach by The Immune Network Metaphor." [18] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer immunology," Commun ACM, vol. 40, no. 10, pp. 88–96, Oct. 1997, doi: 10.1145/262793.262811.

[18] J. Timmis, "Artificial immune systems—today and tomorrow," Natural Computing, vol. 6, no. 1, pp. 1–18, Feb. 2007, doi: 10.1007/s11047-006-9029-1.

[19] Y. Cheung, Y. Wang, and H. Liu, CIS 2006 : 2006 International Conference on Computational Intelligence and Security : Guangzhou, China, November 3-6, 2006 : proceedings. IEEE, 2006.

[20] Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., & Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Software: Practice and Experience, 47(9), 1275–1296. https://doi.org/10.1002/spe.2509.J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[21] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11, 1997.