WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

"A Collective methodology for Predicting IP Address for end to end Cloud & Edge Computing"

Attar Mohammad Younus[1], Avijoy Das Adhikari[2], Jagannath Rath[3], Dr. Sujatha K., M.E., Ph.D.[4]

[1] ay2229@srmist.edu.in
[2] ad3361@srmist.edu.in
[3] ju8516@srmist.edu.in
[4] sujathak@srmist.edu.in
[1, 2, 3, 4] SRM University, Chennai (Ramapuram campus), India

ABSTRACT:

Fog computing, situated at the edge of the network, offers potential as an extension of cloud computing, especially for a wide array of Internet of Things (IoT) applications. Despite its promise to reduce application response times, its widespread implementation relies heavily on the availability and capabilities of resources within the fog infrastructure. Consequently, efficiently utilizing fog systems to execute various IoT type applications seeing their quality of service (QoS) requirements becomes imperative. This task is surely challenging when applications are broken down into multiple modules with varying latency sensitivities. Moreover, scattering application modules across distributed fog nodes exacerbates the issue by elevating the overall energy consumption of the fog environment. Study introduces a policy for modular application placement in fog computing environments that is deadline and energy-conscious. This policy prioritizes placing critical applications in the fog infrastructure while also consolidating active fog nodes for energy management. The performance of this policy was assessed and compared with various contemporary solutions. Analysis indicate that it can be checked through several prediction models from time series data of collected ip addresses

1. Introduction

Internet of Things (IoT) necessitate rapid computation services, particularly helps managing real-time applications. In IoT, devices like sensors and mobile devices generate substantial data, ideally processed in cloud systems due to their cost-effectiveness and scalability. However, for certain IoT applications requiring swift responses and minimal latency, cloud systems may not suffice. Addressing this concern, Cisco introduced fog computing systems to cloud services but situated closer to users and devices, thereby improving service qualities and gradually reducing computing and communication costs.

Fog nodes, are positioned near data sources, mitigate communication delays, serving as an intermediary layer between IoT and cloud computing, ideal for latency-sensitive applications. Nonetheless, fog nodes are technically used and displaced and have fewer type of resources compared to cloud servers, rendering them incapable of handling all applications. Consequently, determining the optimal placement of applications in fog/cloud systems is seen a threat by microbial services applications characterized by interdependent variant of modules computing nodes. Each module necessitates specific resources to function within designated deadlines.

A fresh type of method is used for positioning applications within fog-cloud systems. We differentiate applications into modules and then accordingly priortize them to deadlines. This facilitates parallel processing of modules and their efficient distribution across fog/cloud servers, thereby decreasing latency through local or parallel processing on servers and conserving energy.

Our contributions are:

- Introduction of two placement strategies aimed at enhancing service quality and diminishing power usage.
- Accounting for the dynamic arrival of time-sensitive applications, represented as a Distributed Data Flow model.
- Assessment of our methodologies within the simulated fog environment, iFogSim, demonstrating superior performance compared to alternative approaches.

The goal of the system is to efficiently allocate computing resources to different modules of an application. This aims to ensure that:

1. Deadlines are adhered to: Every module in an application must be processed within a designated timeframe. Through strategic module placement, the system guarantees timely processing of all modules, thereby meeting application deadlines collectively.
2. Energy usage is reduced: Through strategic distribution of modules across fog and cloud servers, the system strives to minimize energy consumption. This includes situating modules near data sources when feasible, employing local processing to cut down on data transmission, and optimizing resource allocation to prevent needless energy expenditure.
By achieving these goals, the system can enhance the overall performance and efficiency of fog-cloud systems, providing better quality of service while reducing operational costs and environmental impact.

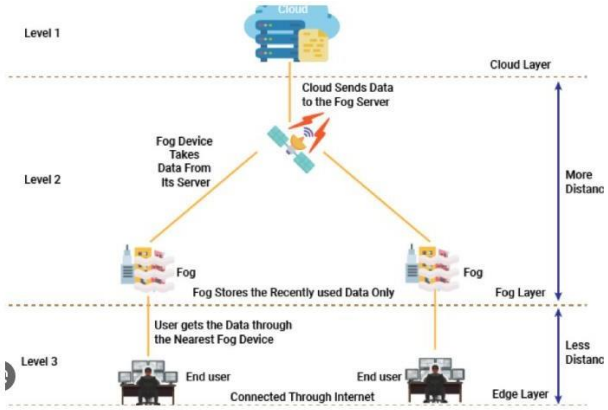


Figure 1: IoT with Fog-cloud Environment..

Related Works

Task distribution and resource management in fog computing are new topics that combine aspects of cloud computing, mobile computing, and sensor networks. Processing of data from sensors, actuators, and similar devices requires fog nodes and/or clouds. The writers presented a plan for dividing work across a fog-cloud system, taking into account both power usage and latency issues. The problem of workload allocation is divided into two main problems. The framework addressed this issue, showing how fog and cloud IP systems complement each other. There hasn't been much investigation into the interplay architecture of workloads and resources. The study presented a technique for mapping modules in IoT applications in a fog-cloud setting to enhance resource utilization. This policy deals with network problems by giving priority to network nodes and application modules depending on the capacity available..

As they are met, module mapping occurs. However, this endeavor solely considers computing resources like CPU and RAM for node selection. Negligible attention is given to other crucial IoT ecosystem factors such as latency and resource availability. Each fog device is assessed based on idle energy, maximum frequency, and maximum energy parameters. Penalty and reward mechanisms minimize energy consumption, preventing exponential increases.., considering latency and deadline requirements. Proposed a policy for application module management, aiming to optimize working node count to reduce power consumption without violating QoS constraints. Evaluated using it excelled in satisfying latency for applications with strict deadlines. Developed a application placement policy prioritizing requests according to user expectations, considering fog instance capabilities. Similarly, examined . Proposed an optimized placement approach for fog computing applications using genetic algorithms (GA), minimizing computational and communication costs while ensuring resource efficiency. Likewise, used GA for cost-effective task scheduling in cloud architecture, enhancing cost efficiency for time-sensitive applications. It is a real-time task schedule type system considering deadline type constraints to enhance task mapping, processing.Module mapping takes place when requirements are fulfilled. Nevertheless, this project only takes into account computing resources such as CPU and RAM when choosing nodes. Other important IoT ecosystem factors such as latency and resource availability are often overlooked. Researchers developed an algorithm for efficient resource distribution in fog computing by addressing the allocation problem as a penalty-aware bin-packing issue. Every fog device is evaluated according to idle energy, maximum frequency, and maximum energy parameters. Punishment and incentive systems reduce energy usage, avoiding exponential growth. Implemented the Fog Service Provisioning Policy (FSPP) to effectively allocate resources among IoT services on fog nodes, taking into account latency and deadline constraints. Suggested a strategy for managing application modules to minimize the number of active nodes, thereby decreasing power usage while still upholding QoS requirements. Assessed in terms of meeting latency requirements for time-sensitive applications, it performed exceptionally well. Created an app placement policy that prioritizes requests based on user expectations and takes into account the capabilities of fog instances. Suggested a refined method for positioning fog computing applications by using genetic algorithms (GA) to lower computational and communication costs while maintaining resource efficiency. Similarly, GA was utilized for economical task organization in cloud structure, improving cost effectiveness for time-critical programs. It is a system that schedules tasks in real-time, taking deadline constraints into consideration

to improve task mapping and processing. The service function chain concept aims to enhance speed, resource use, and efficiency in fog computing through a flexible planning model. It improved resource use by introducing an SFC queue network. introduced the Alloc algorithm for big data applications, focusing on cost and load balancing while maintaining performance. created Fog-Care, a healthcare software prototype, to decrease latency and increase throughput in distributed locations. proposed a method in Federated Learning, selecting quality data with border entropy evaluation.

Proposed Work

This section introduces a new method for arranging application modules in fog-cloud systems, placing emphasis on performance and power efficiency. The goal is to reduce the make span time (MST) of application modules by strategically placing them on fog nodes. Additionally, the goal is to reduce the number of active fog nodes by consolidating multiple modules onto a smaller number of nodes, in turn improving resource usage and decreasing power consumption by fog nodes.
Furthermore, we take into account the option of deploying specific applications on cloud servers without exceeding time limits. In these cases, we decide to run those applications on cloud servers. This not just improves efficiency but also reduces the need for fog nodes in the system.

MECHANISM FOR PEAPM

Efficient power consumption may hold importance in both cloud and fog computing environments, directly influencing operational expenses for providers and users. The key to achieving power savings lies in maximizing resource utilization, thereby diminishing the necessity for multiple active computing nodes in fog or cloud configurations. Studies have indicated that. Building upon this insight, this section presents an innovative approach to reducing power usage in fog environments by minimizing the number of active fog nodes, while transitioning inactive nodes to a power-saving mode.

The level of utilization of a fog node, denoted as directs the objective of this mechanism to decrease the count of fog devices hosting application modules. Accordingly, we introduce a power-conscious placement strategy termed Power-Aware Placement Mechanism (POAPM), elucidated in Algorithm 2. The algorithm commences by evaluating the utility level of each fog node, initiating a migration process for application modules if the utility falls below a specified threshold.

The utility threshold, a parameter of the algorithm, plays a crucial role in determining the optimal threshold for achieving maximal power savings. Lower thresholds, like 10%, result in minor migrations, offering limited power benefits. Conversely, higher thresholds may induce excessive migrations. Discovering the optimal threshold necessitates an iterative approach, experimenting with different values to identify the most suitable one.

The algorithm organizes fog layer nodes in ascending order based on their current utilization levels, with the most utilized node selected first. The strategy aims to consolidate application modules on fewer nodes to bolster utilization and, consequently, enhance power savings. The migration of application modules from underutilized nodes (n) to candidate nodes (nc) is facilitated by the update Cap and remove methods. Ensuring compliance with fog environment performance, line 8 verifies if the latency of the candidate node aligns with latency requirements; otherwise, the module is placed elsewhere. Modules are allocated to nodes to optimize compactness, reducing the need for multiple fog nodes and resulting in lower energy consumption.

index = nodeList.getIndex(n)

Remove processed module from the queue Q.pop(0)

Check if a suitable node was found if index == -1:
return False else:
Update node capacity and return true

MECHANISM FOR POAPM

Algorithm outlines the Performance-Aware Placement Mechanism (PEAPM). It initiates by arranging application modules based on their deadlines and then proceeds to select modules for each application in accordance with their dependency constraints, prioritizing modules with no dependencies within the application.

Applications are sequenced by their deadlines, and within each application, modules are checked thoroughly. Modules with dependencies are assessed based on their data dependency delay type of system , favoring nodes with minimal delay. Nodes in closer proximity are preferred to reduce data dependency delay, considering node processing speed. The node with the shortest minimum spanning tree is chosen to accommodate , as it possesses sufficient CPU and RAM resources. The placement ensures compliance with the application's delay requirement.

The algorithm predicts the completion time for each task before placing the module on the designated node. If node is seen to be affecting the module, it gives out discrepant results, and it may be redirected .

Algorithm1:

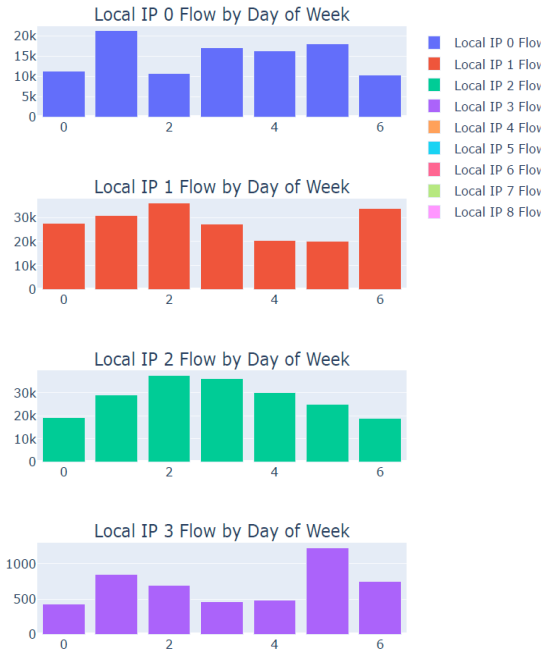
Define the placement algorithm with parameters \(m \), \
(l_{\text{max}} \), \(\text{modules} \), and \(\text{nodeList} \):
1. Sort modules by deadline:
  - \(Q = \text{sorted(modules, key=lambda x: x.deadline)} \)
2. Initialize \(\text{MSTmin} \) to positive infinity:
  - \(\text{MSTmin} = \text{float('inf')} \)
3. Initialize index to -1:
  - \(\text{index} = -1 \)
4. Iterate while the queue is not empty:
  - \(\text{while Q:} \)
    - Iterate over nodes:
      - \(\text{for n in nodeList:} \)
        - Check resource availability and delay constraint:
          - \(\text{if Req}(m) \leq \text{Cap}(n) \) and
\(n.\text{delay} \leq l_{\text{max}} \):
            - Calculate temporary MST:
              - \(\text{MSTtmp} = \text{MST}(m, n) \)
            - Update \(\text{MSTmin} \) if temporary MST is smaller:
              - \(\text{if MSTtmp} \leq \text{MSTmin}: \)
                - \(\text{MSTmin} = \text{MSTtmp} \)
                - \(\text{index} = \text{nodeList.getIndex}(n) \)
            - Remove processed module from the queue:
              - \(\text{Q.pop(0)} \)
            - Check if a suitable node was found:
              - \(\text{if index == -1:} \)
                - \(\text{return False} \)
              - \(\text{else:} \)
                - Update node capacity and return true:
                  - \(\text{updateCap}(n, m) \)

Algorithm2:

```
def minimizePowerUsage(threshold, nodeList):
    sortedNodes = sorted(nodeList, key=lambda node: node.utilization, reverse=True)
    for node in nodeList:
        if node.utilization < threshold:
            index = 0
            for module in node.modules:
                candidateNode = sortedNodes[index]
                if requiredResources(module) <= candidateNode.capacity and candidateNode.delay <= module.max_latency:
                    updateCapacity(candidateNode, module)
                    remove(node, module)
                else:
                    index += 1
```

Results and Discussions

Exploratory data analysis's (EDA) objective is to get as much information as possible from a dataset using a variety of approaches, most notably graphical ones. Learn the underlying principles. Eliminate the critical elements. Look for outliers and oddities. Analyze the guiding assumptions. Build cost-effective models and identify the ideal factor values. Despite the common confusion between the names, statistical graphics and EDA are not the same. Statistical graphics is a group of graphically focused methods focused on particular facets of data characterisation. Exploratory data analysis (EDA), which has a broader reach, is a data analysis technique that eliminates conventional assumptions about the type of model the data fits into and instead allows the data speak for itself. EDA is a philosophy about how we analyze a data set, what we search for, how we search, and how we interpret it. It is not just a set of methods.



Conclusion

The study described a novel way for delivering application modules in a fog-cloud environment. The goal was to reduce processing time while retaining acceptable delays. We created two approaches, PEAPM and POAPM, and tested them through simulations under various conditions. According to our findings, our methods outperformed others in parameters like as TGR, make span, power consumption, and violation cost. Performance varies based on

However, determining the ideal number of servers for each tier is critical for system optimization, which will be investigated further in future study. Furthermore, utilizing various meta-heuristic optimization frameworks may improve system performance, indicating intriguing paths for future research in this subject.

REFERENCES

1. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, ‘‘Fog computing and its role in the Internet of Things,’’ in Proc. 1st, Ed., MCC Workshop Mobile Cloud Comput. (MCC), vol. 131. New York, NY, USA: ACM Press, Aug. 2012, p. 13.
2. R. Deng, R. Lu, C. Lai, and T. H. Luan, ‘‘Towards power consumption-delay tradeoff by workload allocation in cloud- fog computing,’’ in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2015, pp. 3909–3914.
3. M. A. Al Faruque and K. Vatanparvar, ‘‘Energy management-as-a-service over fog computing platform,’’ IEEE Internet Things J., vol. 3, no. 2, pp. 161–169, Apr. 2016.
4. S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, ‘‘Fog computing: From architecture to edge computing and big data processing,’’ J. Supercomput., vol. 75, no. 4, pp. 2070–2105, Apr. 2019.

5. P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, ‘‘A survey on fog computing for the Internet of Things,’’ Pervasive Mobile Comput., vol. 52, pp. 71–99, Jan. 2019.
6. S. Jošilo and G. Dán, ‘‘Decentralized algorithm for randomized task allocation in fog computing systems,’’ IEEE/ACM Trans. Netw., vol. 27, no. 1, pp. 85–97, Feb. 2019.
7. R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, ‘‘Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption,’’ IEEE Internet Things J., vol. 3, no. 6, pp. 1171– 1181, Dec. 2016.
8. M. Taneja and A. Davy, ‘‘Resource aware placement of IoT application modules in fog-cloud computing paradigm,’’ in Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM), May 2017, pp. 1222–1228.
9. Z. Pooranian, M. Shojafar, P. G. V. Naranjo, L. Chiaraviglio, and M. Conti, ‘‘A novel distributed fog-based networked architecture to preserve energy in fog data centers,’’ in Proc. IEEE 14th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS), Oct. 2017, pp. 604–609.
10. L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, ‘‘Mobility-aware application scheduling in fog computing,’’ IEEE Cloud Comput., vol. 4, no. 2, pp. 26–35, Mar. 2017.
11. O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, ‘‘Towards QoS-aware fog service placement,’’ in Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC), May 2017, pp. 89–96.
12. R. Mahmud, K. Ramamohanarao, and R. Buyya, ‘‘Latency-aware application module management for fog computing environments,’’ ACM Trans. Internet Technol., vol. 19, no. 1, pp. 1–21, Feb. 2019.
13. R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, ‘‘Quality of experience (QoE)-aware placement of applications in fog computing environments,’’ J. Parallel Distrib. Comput., vol. 132, pp. 190–203, Oct. 2019.
14. Brogi, S. Forti, C. Guerrero, and I. Lera, ‘‘Meet genetic algorithms in Monte Carlo: Optimised placement of multi-service applications in the fog,’’ in Proc. IEEE Int. Conf. Edge Comput. (EDGE), Jul. 2019, pp. 13–17.
15. T. S. Nikoui, A. Balador, A. M. Rahmani, and Z. Bakhshi, ‘‘Cost-aware task scheduling in fog-cloud environment,’’ in Proc. CSI/CPSSI Int. Symp. Real-Time Embedded Syst. Technol. (RTEST), Jun. 2020, pp. 1–8.
16. M. Louail, M. Esseghir, and L. Merghem-Boulahia, ‘‘Dynamic task scheduling for fog nodes based on deadline constraints and task frequency for smart factories,’’ in Proc. 11th Int. Conf. Netw. Future (NoF), Oct. 2020, pp. 16–22.
17. F. Faticanti, F. De Pellegrini, D. Siracusa, D. Santoro, and S. Cretti, ‘‘Throughput-aware partitioning and placement of applications in fog computing,’’ IEEE Trans. Netw. Service Manage., vol. 17, no. 4, pp. 2436–2450, Dec. 2020.
18. R. Mahmud, A. N. Toosi, K. Ramamohanarao, and R. Buyya, ‘‘Context-aware placement of Industry 4.0 applications in fog computing environments,’’ IEEE Trans. Ind. Informat., vol. 16, no. 11, pp. 7004–7013, Nov. 2020.
19. A.-V. Postoaca, C. Negru, and F. Pop, ‘‘Deadline-aware scheduling in cloud-fog-edge systems,’’ in Proc. 20th IEEE/ACM Int. Symp. Cluster, Cloud Internet Comput. (CCGRID), May 2020, pp. 691–698.
20. M. Salimian, M. Ghobaei-Arani, and A. Shahidinejad, ‘‘Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment,’’ Softw., Pract. Exp., vol. 51, no. 8, pp. 1745–1772, Aug. 2021
21. T. Lähderanta, T. Leppänen, L. Ruha, L. Lovén, E. Harjula, M. Ylianttila, J. Riekki, and M. J. Sillanpää, ‘‘Edge computing server placement with capacitated location allocation,’’ J. Parallel Distrib. Comput., vol. 153, pp. 130–149, Jul. 2021.
22. E. Batista, G. Figueiredo, and C. Prazeres, ‘‘Load balancing between fog and cloud in fog of things based platforms through software-defined networking,’’ J. King Saud Univ. Comput. Inf. Sci., vol. 34, no. 9, pp. 7111–7125, Oct. 2022.
23. N. Godinho, H. Silva, M. Curado, and L. Paquete, ‘‘A reconfigurable resource management framework for fog environments,’’ Future Gener. Comput. Syst., vol. 133, pp. 124–140, Aug. 2022.
24. X. Gao, R. Liu, and A. Kaushik, ‘‘Virtual network function placement in satellite edge computing with a potential game approach,’’ IEEE Trans. Netw. Service Manage., vol. 19, no. 2, pp. 1243–1259, Jun. 2022.
25. J. C. S. D. Anjos, K. J. Matteussi, P. R. R. De Souza, G. J. A. Grabher, G. A. Borges, J. L. V. Barbosa, G. V. González, V. R. Q. Leithardt, and C. F. R. Geyer, ‘‘Data processing model to perform big data analytics in hybrid infrastructures,’’ IEEE Access, vol. 8, pp. 170281– 170294, 2020.
26. P. R. R. De Souza, K. J. Matteussi, A. D. S. Veith, B. F. Zanchetta, V. R. Q. Leithardt, Á. L. Murciego, E. P. De Freitas, J. C. S. D. Anjos, and C. F. R. Geyer, ‘‘Boosting big data streaming applications in clouds with BurstFlow,’’ IEEE Access, vol. 8, pp. 219124–219136, 2020.
27. H. J. D. M. Costa, C. A. D. Costa, R. D. R. Righi, R. S. Antunes, J. F. D. P. Santana, and V. R. Q. Leithardt, ‘‘A fog and blockchain software architecture for a global scale vaccination strategy,’’ IEEE Access, vol. 10, pp. 44290–44304, 2022.
28. F. C. Orlandi, J. C. S. D. Anjos, V. R. Q. Leithardt, J. F. D. P. Santana, and C. F. R. Geyer, ‘‘Entropy to mitigate non-IID data problem on federated learning for the edge intelligence environment,’’ IEEE Access, vol. 11, pp. 78845–78857, 2023.