SN Computer Science (2023) 4:745
https://doi.org/10.1007/s42979-023-02187-0

ORIGINAL RESEARCH

Optimal Cloudlet Selection in Edge Computing for Resource Allocation

Bablu Kumar[1] · Mohini Singh[1] · Anshul Verma[1] · Pradeepika Verma[2]

Received: 12 April 2023 / Accepted: 23 July 2023

Abstract
Mobile and Edge Computing devices have limited resources to perform computationally intensive jobs, and hence, there is a need for task offloading. In Mobile Cloud Computing, cloud servers are placed far from the user devices; as a consequence, many challenges are faced, such as security, limited bandwidth, network latency, and storage. Whereas edge servers are placed near the user devices in Edge Cloud Computing; however, issues of Cloud computing are also faced in Edge computing due to the huge number of devices, which also generates a significant load on edge servers. Some resource optimization approaches help in achieving optimal Cloudlet selection at the edge servers. When users access edge resources, such as CPU, memory, and hard disk, load balancing helps in distributing tasks among edge servers and achieving efficient results. The user devices communicate either within a Cloudlet or between Cloudlets using resource sharing, in which one of the main issues is optimal Cloudlet selection. This paper presents an optimal Cloudlet selection algorithm in which, first of all, an index value for each resource is calculated using parameters like weight, cluster of Cloudlets, availability, and total resource usage. Thereafter, the resource level and available resources of this level are calculated for each Cloudlet. Finally, an algorithm is proposed to help in finding the optimal Cloudlet for the cloud broker. The proposed approach is implemented in Cloud-Sim. The simulation results have shown efficiency of the proposed approach.

Keywords Cloud computing · Edge computing · Cloudlet computing · Load balancing · Cloud federation · Resource allocation

Introduction

Two themes (virtualization and application tool) are combined to form the main pillars of the field of mobile cloud computing. On one hand, hypervisors in data centers are being impacted by virtualization, and on the other hand, mobile gadgets like smartphones, laptops, and tablets have made a significant impact on everyday life. Resources, such as CPU, memory, energy, and network resources, are constrained on mobile devices which is worsening due to the development of resource-intensive applications [1]. Today, mobile devices have access to a resource-rich environment through the cloud computing paradigm for resource sharing and load balancing. A simulated environment created by virtualization means software that allows a single host to run one or more guest operating systems. A user builds their federation from various clouds and organizations according to the National Institute of Standards and Technology (NIST) [2]. Cloud computing refers to the delivery of on-demand computing services, such as servers, storage, databases, networks, and software via the Internet to enable rapid innovation and economic scale [3]. Cloud computing provides various services like Software as a Service (SAAS), Platform as a Service (PAAS), and Infrastructure as a Service (IAAS) [4]. In cloud computing, IAAS is used for infrastructure as well as security

This article is part of the topical collection "Machine Intelligence and Smart Systems" guest edited by Manish Gupta and Shikha Agrawal.

✉ Bablu Kumar
bablu.kumar2k23@gmail.com

Mohini Singh
mohini.singh4@bhu.ac.in

Anshul Verma
anshulverma87@gmail.com

Pradeepika Verma
pradeepikav.verma093@gmail.com

[1] Department of Computer Science, Banaras Hindu University, Varanasi, Uttar Pradesh 221005, India

[2] TIH, Indian Institute of Technology, Patna, Bihar 801106, India

Published online: 27 September 2023

purposes. In this study, the IaaS model is applied to a use case where the job is packaged as a Virtual Machine (VM) and installed on a physical server. Virtualization is the simulation of the software or hardware upon which other software runs smoothly and this environment is called a VM. A task that requires a lot of computation is transferred from a mobile device to the cloud environment using the Mobile cloud computing (MCC) model [5]. By moving a computing-intensive operation from a mobile device to the cloud environment, the MCC idea tackles resource limitations [6]. Edge-based solutions aim to offer the nearest proximity-based answer and remove these restrictions, such as Mobile Edge Computing (MEC) [7, 8], fog computing, and cloudlet computing [9]. In comparison to MEC and fog computing [10], cloudlet-based computing is more versatile, since it provides big computational resources, diverse functions, and better bandwidth without requiring specific equipment. Now, there have been additional devices interacting with the cloudlets, increasing the workload beyond what the cloudlets can handle. In this situation, Cloudlet replicates a traditional MCC architecture and sends the tasks, that are too far away to handle locally to the cloud [11], negating the advantage of edge computing [12]. As a result, this issue must be solved in a way that allows the majority of requests to be fulfilled without being routed to a distant cloud server.

In edge-based resolutions, when only an individual's desktop or mobile device contributes to the edge device, the user's choices are highly crucial to explicitly prevent specific additional resources that can be shared [13]. With an oversupply of necessary resources in each and every one of them, neither the user nor the event bears equal weight in this regard. Consequently, each edge device can provide a varying capacity of resources to only respond to a certain range of queries to satisfy the resource needs (see Fig. 1). An ideal solution should take into account resource sharing, load balancing, and task distribution [14–17], considering the resource circumstances of edge devices.

The proposed research work involves the communication between cloudlets and edge devices via a network, such as PAN1, PAN2, or PAN3, etc., as depicted in Fig. 2. The data flow (see in Fig. 3) begins when a new user requests a resource to create a new subscription. Additionally, existing users may request to terminate execution or extend the timeline. All these types of requests are sent to the cloud broker through a request, and a virtual machine is created on the cloudlet host. The cloudlet host then forwards the request to the cloud broker, which checks its data related to the cloudlets. This paper also calculates and considers the information that the cloud broker must identify the optimal cloudlet and assign it to the cloud host as the chosen cloudlet. The problem lies in how the cloud broker classifies which cloudlet

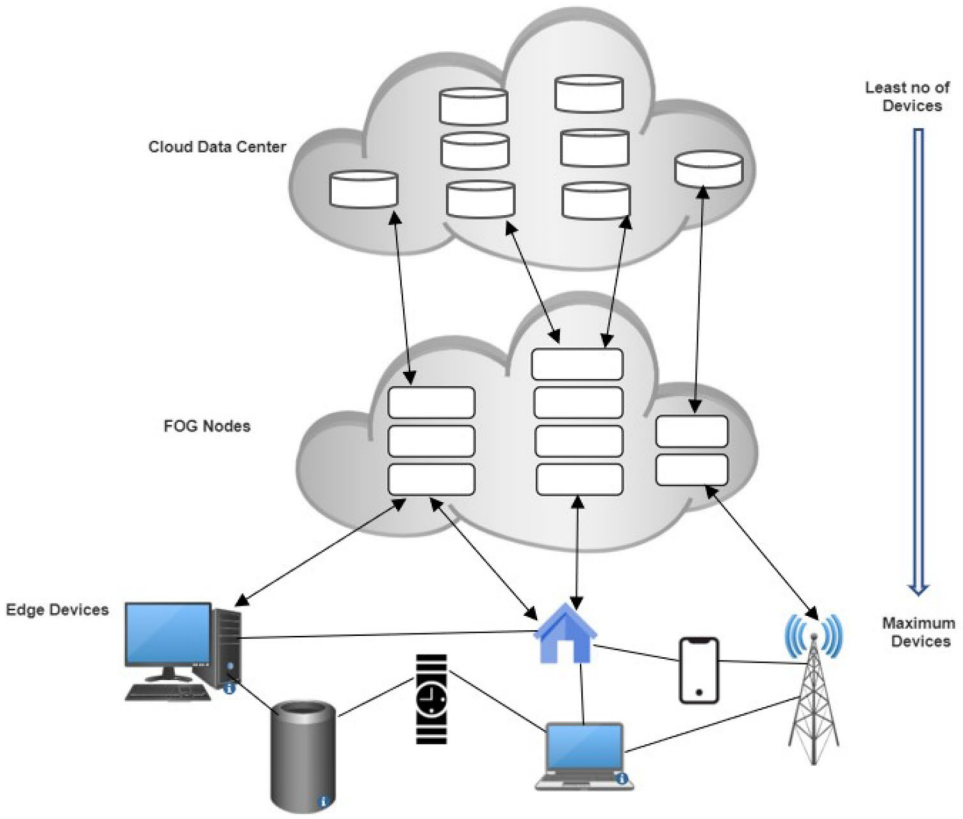


Fig. 1 Fog and Edge Computing

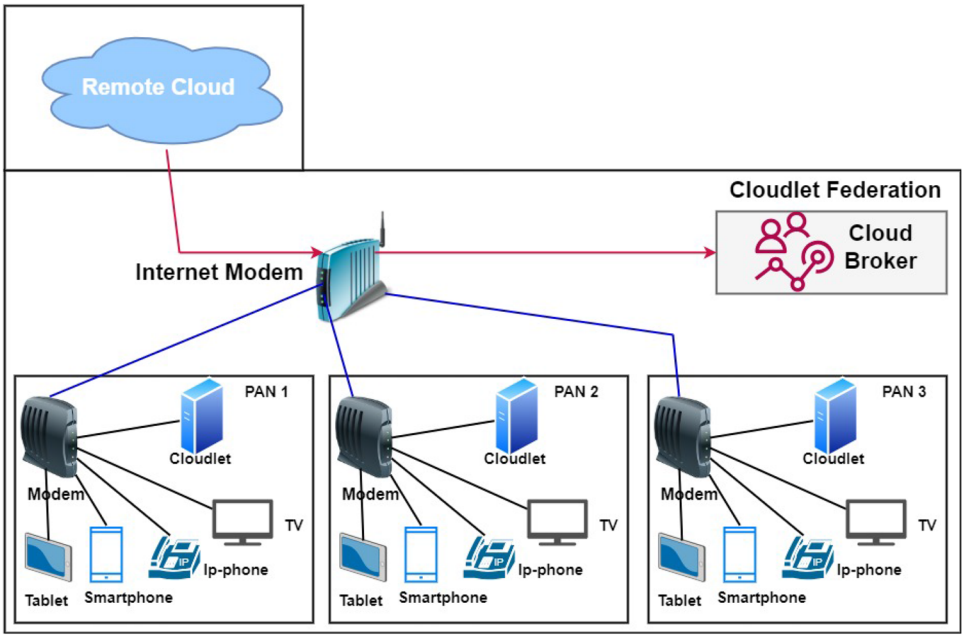


Fig. 2 Collective model

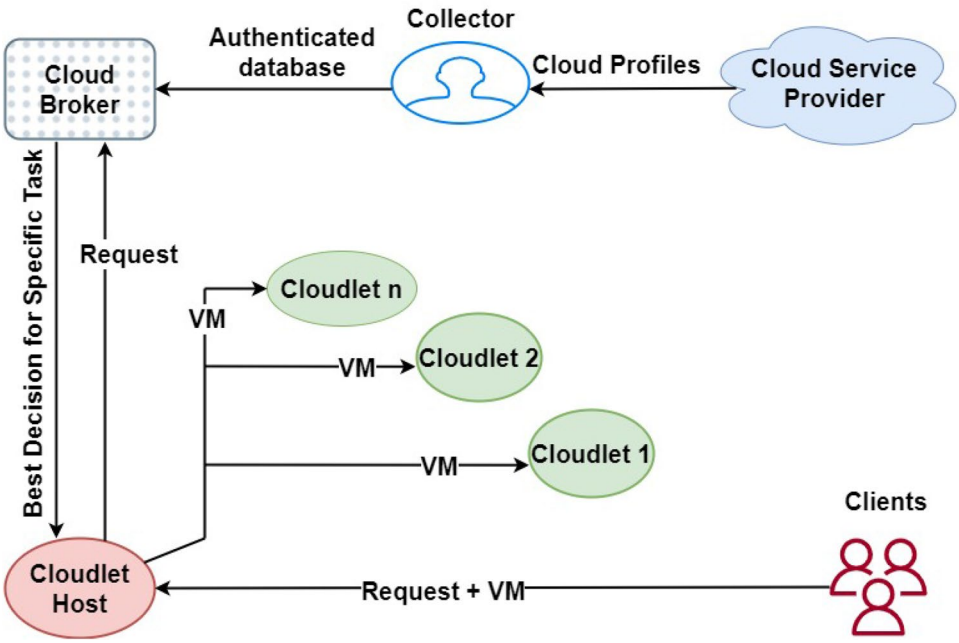


Fig. 3 Data flow

is best according to the resource sharing and how to select another cloudlet if the currently assigned goes down. The methodology section discusses how to solve these problems and presents the solutions.

The first step is calculating the resource index which determines the value of the metrics and this index value represents the resources. The level value for resources is calculated based on the total resource of the segment cloudlet. The optimal selection for resource cloudlets is based on these values. Brokers or owners who possess the index value or level value can easily determine the resource requirements, such as CPU, hard disk, memory, and bandwidth for each cloudlet. When a cloudlet receives a request from a broker or owner, it checks its available capacity and minimum resources. If the request aligns with the available resources, it is considered "USEFUL" and processed by the cloudlet. Otherwise, the broker forwards the request to another cloudlet that is already connected with edge devices.

Background

This section provides in-depth insights into the designs, models, and architectural aspects related to the proposed work. The specific calculation steps are highlighted that are essential for methodology and ensuring a comprehensive understanding of its implementation.

Collective Work

The proposed work provides solutions for resource allocation problems and ensures that minimum requests are sent to the remote Cloudlet host. "A cloudlet is a small-scale data center or cluster of computers designed to quickly provide cloud computing services to mobile devices, such as smartphones, tablets and wearable devices, within close geographical proximity". In Fig. 2, there are many cloudlets named PAN1, PAN2, and PAN3, each with different owners and cloud servers [18]. Every edge device has its own cloudlet, and that cloudlet will have previous references to the broker which will be available in the federation. Cloud federation is the process of linking the cloud computing environments of two or more service providers to load balanced traffic and accommodates services in demand.

Only the recommended resources are locked for sharing compared to the priority and percentage. This ensures that no resources may be over-provisioned and compromised the security of the cloudlet owner's performance of tasks and services on the cloudlet node [19]. A broker is someone who acts as an intermediary between two or more parties during negotiations. The broker keeps track of the resources and maintains the number of cloudlet states. When a request comes in (as shown in Fig. 3), the broker matches requests containing the necessary resources with all of its participant cloudlets. Similarly, when a virtual machine has to be transferred due to load or to a better location, engaged resources are released from the cloudlet server, and the current status of resources is maintained. The broker manages various tasks for the cloudlet federation, including cloudlet registration, maintaining the flow of resources, and choosing the optimal cloudlet. Additionally, the cloud of members handles resource information. Resources include CPU, memory, storage, and bandwidth, whereas optimum selection includes considerations for cloudlet and VM migration [20]. Virtual memory is utilized when a computer's existing cloudlet-based models provide resource sharing and load balancing based on the local knowledge of other nearby cloudlet nodes (fixed or mobile), preferably within the same LAN.

Resource Allocation

Since there are many applications located on a cloudlet and share resources [21]. Based on the time, increased wait time is observed for new applications, and ultimately, the availability of poor resources affects the performance of the overall system and the running applications. To control the load, the resource constraint not only impacts performance but also causes the cloudlet to transmit requests to the remote cloud [22]. In the present implementation, the main focus is on the challenges, such as distance, limited bandwidth, and latency with single or multiple groups of cloudlets in the same location [23]. There are two types of criteria for the design of an internal network: one called LAN and the other MAN or WAN. The current scenario is based on an internal network within the LAN but will be supplemented with WAN or MAN [24]. It refers to the relationship between the Cloudlet federation. The popularity of the scope of federation distance improves the connection of more cloud nodes to address stability or resource-sharing problems. Finally, we will discuss how cloudlet selection is made considering minimum latency compared to the remote cloud. Figure 3 represents an overview of how to communicate between cloudlets and brokers. The remote cloud is a conservative cloud that is used when the federation is unable to access the resources and is kept as a worst-case scenario where the outcomes of the suggested method are equal.

Calculation Part for Resource Allocation

When discussing resource operation, we acknowledge that the problem of choosing the optimum cloudlet selection is challenging due to reliance on several variables such as CPU, memory, hard disk, bandwidth, and storage [25]. This problem can be mitigated by assigning weights to each resource on an edge device to facilitate resource sharing. The owner assigns weights of 7, 6, 8, and 9 (respectively) to each resource [25]. When a cloudlet communicates with the host (edge device) or with other cloudlets via a VM, time is required to calculate the index value for resources and the level value for resource calculation [26, 27]. By utilizing these values, we can identify and access optimal cloudlets with minimal reliance on multiple variables [28].

Proposed Approach

When dealing with high-performance computing applications, even if we are availing services from applications like AWS, Microsoft Azure, Google Cloud Platform, etc., there exists a virtual connection between clients and the main server. Another aspect is the local side (edge) that performs computing using cloudlets in edge computing, which is more interactive with IoT [29–31] devices and offers low latency. In this case, there are brokers involved. Therefore, when new requests are made to establish connections, create new connections, or update existing connections by a client, the broker communicates with a cloudlet host and the broker then communicates with the cloud-service provider (CSP) or main server. Once the broker assigns a service, the connection is established through the embedded cloudlet in the local network. Subsequently, the cloudlet updates the resource index or resource level, allowing the broker to easily select and connect the new user based on their specific requirements. Therefore, how do you select the best cloudlet with the assistance of a broker? The broker can provide the best service. With this objective in mind, the initial calculation for a resource index is proposed, which identifies order of the resources based on availability. After that, the resource level is calculated that needs to be checked before the next service. By utilizing these two methods, ultimately optimum cloudlet is selected for VM settlement. The proposed algorithms are discussed in detail as follows.

Calculation of Resource Index

First, we calculate the resource index using Eq. 1. Let Wight = wt1, wt2, wt3,... wtn, which is a set of weights assigned by the owner to resources. Cloudlet = $c1, c2, c3, \ldots cn$ collection of cloudlets, $A = a1, a2, a3, \ldots an$ and collection of available, Tr = $t1, t2, \ldots tn$ total resource usage, by given details in Eq. 1

$$\text{Total Index Resource (TIR)} = \sum_{i=1}^{n} \left(\frac{A_i}{TR_i} \right) * Wt_i. \tag{1}$$

Algorithm 1 Calculation of Resource Index

Require: The bunch of cloudlets, resource weight and value allocated by the owner
Ensure: Total Index for Resource

```
Start
Index total resources = φ
for separately cloudlet Ci do
    for separately resource Tr do
        weight wt. assigned by the owner
        Available resources value Ai
        Total Index Resources Tr
        TIR = ∑_{i=1}^{n} (Ai / TRi) * Wti
        Ir=Accumulate all values of TIr
    end for
    return TIr
end for
End
```

Calculation of Resource Level

There are two phases: the first is "useful" and the second is "NOT useful", which are compared with an index value and a threshold value. The value of the threshold will be taken by the host. If the state index value is above the minimum threshold level, then it is considered "useful". If the state index is below the minimum threshold level, then it is considered "NOT useful".

Equation 2 calculates the total resource of segment cloudlet and Table 1 describes abbreviations

$$\text{TRS}[trij] = \text{ARS}[Aij] + \text{URS}[Yij] + \text{MRS}[Zij]. \tag{2}$$

Total resource of segment Cloudlet = available resource + utilized resource + minimum resource

Table 1 List of abbreviations used

Available resources of Segment Cloudlet	ARS_{Aij}
Total resources of Segment Cloudlet	TRS_{TRij}
Minimum required resources of Segment cloudlet for host	MRS_{Zij}
Utilize Resource of Segment Cloudlet	URS_{Yij}

Demand resource = available resource - minimum resource of a member of Cloudlet for the host.

In this case, we are using a technique to calculate resource level that we have combined with the OS concept of how much total resource remains in each segment cloudlet and what constitutes "demand resources". Equation 3 is representing the compression of available resources with minimum required resources of segment cloudlet for the host

$$F_x = \begin{cases} 1, & \text{if } \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{n} \left(\text{ARS}_{Aij} < \text{MRS}_{Zij} \right) \right) \right) \\ 0, & \text{Otherwise when it true,} \end{cases} \tag{3}$$

If the value of the available resource is less than the minimum resource required, it will get 1 and be represented as "not useful". If the value of the available resource is greater than the minimum resource required, it will get 0 and be represented as "useful" (See Algorithm 2).

Algorithm 2 Calculation of Resource Level

Require: Available resources at separate Cloudlet

```
Start
Index total resources = φ
for separately cloudlet Ci do
    for separately resource Tr do
        Calculated for Resource level R_L
        Total Index Resources Tr
        if Aij < Zij then
          True "NOT USEFUL"
        else
          FALSE "USEFUL"
        endif
    end for
end for
return R_L
End
```

Selection of Optimum Cloudlet for VMs' Settlement

Two phases are involved in the selection of the optimum cloudlet. First, the cloudlet federation is compared with another federation within the cloudlet where all hosts are interconnected. These hosts include edge devices, such as mobiles, smartphones, PCs, etc., which are connected and share resources virtually [32]. The minimum available resources are compared with the minimum required resources as represented by Eq. 4.

If the job's resource requirement falls within the available resources, then the cloudlet has sufficient resources to execute the job and it falls within the "safe" state. Whereas, if the jobs' resource requirements are either below or above the threshold, the cloudlet will not be able to execute the jobs, and it falls within the "unsafe" state

$$F_x = \begin{cases} \text{SAFE}, & \text{if } \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{n} \left(\text{RRS}_{Aij} \geq \text{ARS}_{Zij} \right) \right) \right) \\ \text{UNSAFE}, & \text{Otherwise when it true.} \end{cases} \tag{4}$$

After deciding the optimal selection for resources, index values of the resources are referred to the broker or cloudlet which keeps the maximum index value. Let Maximum Available Resources (MAR) are represented as Mr = Mr1, Mr2, Mr3...Mrn. The optimal selection used for cloudlet or broker will be specified that which work is required according to their requirement. Further, the cloudlet owner or broker has to authorize that which service is executed virtually to which cloudlets (See Algorithm 3).

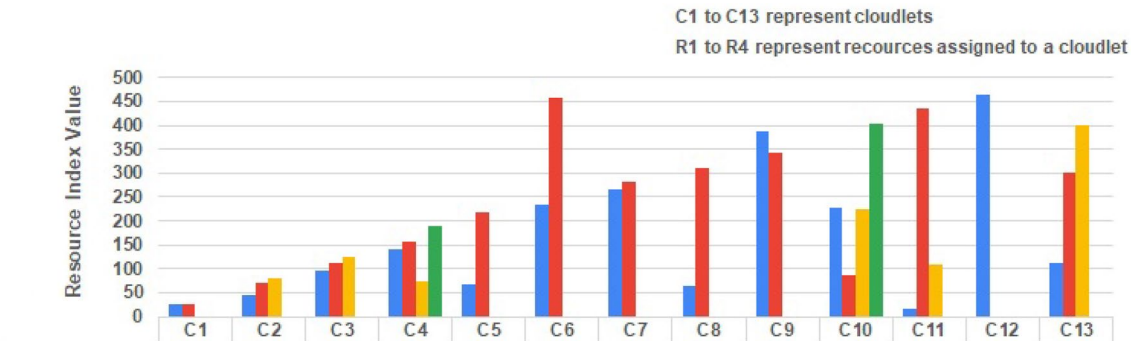
Algorithm 3 Optimal Selection of Cloudlet

Require: Available resources and Cloudlet have some resources for each request which is received by the Broker.

```
Start
Optimal Cloudlet selection C_lo = NULL
for separately cloudlet Ci do
    for separately request q at broker WR do
        if WR = Waiting Result then
        for each resource Available (Ai) and Resources Zi do
            if Aij ≥ Zij then
            Push value SAFE in the cloud list
            endif
        end for
        endif
    end for
end for
if SAFE is available in the list then
for Get resource-level RL do
    if MRL ≠ NOT USEFUL and TIR = Max value with min L than
     WRoptimal = WRsafe
    endif
end for
return WRoptimal value
End
```

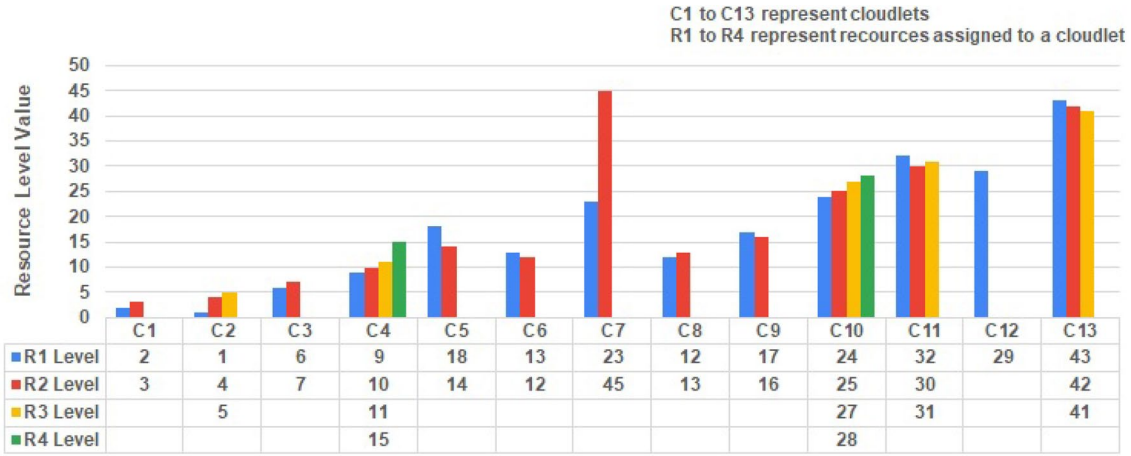
Implementation and Results

Specifically, this paper calculates the values of resource metrics: the index and level values for resources, and the optimal selection of resource cloudlets on the basis of these values. Brokers or owners who possess the index value or cloudlet resource-level value can easily determine the resource requirements for each cloudlet [33]. When a Cloudlet receives a request from a broker or owner, it checks its available capacity and requirement of minimum resources. If the request aligns with the available resources, it is considered "USEFUL" and is processed by Cloudlet. Otherwise, the broker forwards the request to another cloudlet that is already connected to edge devices. This request initiates data sharing and remote connectivity between the cloudlet,



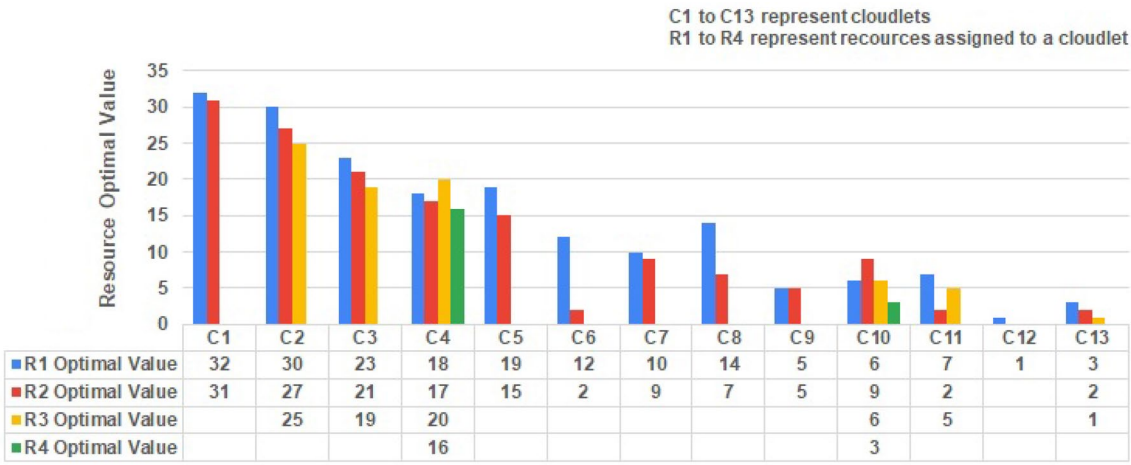
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
R1 Index Value	25	45	95.4	141.6	66	234	264.8	65	386	226	15	465	112
R2 Index Value	26	70	110.8	157	218.6	456	280.2	311	341.8	88	434.2		300
R3 Index Value		80	126.2	75						225	110		400
R4 Index Value				187.8						403.4			

Fig. 4 Resource Index Value



	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
R1 Level	2	1	6	9	18	13	23	12	17	24	32	29	43
R2 Level	3	4	7	10	14	12	45	13	16	25	30		42
R3 Level		5		11						27	31		41
R4 Level				15						28			

Fig. 5 Resource level value



	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
R1 Optimal Value	32	30	23	18	19	12	10	14	5	6	7	1	3
R2 Optimal Value	31	27	21	17	15	2	9	7	5	9	2		2
R3 Optimal Value		25	19	20						6	5		1
R4 Optimal Value				16						3			

Fig. 6 Resource optimal value

cloudlet host and broker, as well as enables efficient resource allocation.

Steps of the Procedure

1. Create the connection between the broker and cloudlets when VMs initiate it.
2. Sort the virtual machines in descending order based on the number of requests.
3. Calculate the index value as shown in Algorithm 1. This will help in deciding the number of cloudlets to serve a job, and the optimum cloudlet will be selected easily according to the index value.
4. Further, resource level and demand values have to be calculated as shown in Algorithm 2.
5. If less number of resources are available at the broker, then the cloud broker replicates the VM.
6. The broker will be kept with detailed resource level or demand values; using these values, it will send requests to the cloudlets on the basis of requirement. And finally, cloudlet has index values that can be used to provide service directly to the resources.

Simulation Results

The proposed approach is implemented and tested on the Cloud-Sim simulator, which includes modules like VMs, packages, data centers, hosts, and cloudlets. Each VM has a number of packages and a set of costs and tasks that are handled by the cloudlet paradigm. The mathematical calculations mentioned in Algorithms 1 and 2 are implemented

in Cloud-Sim using Java Eclipse to run the algorithms with the specified inputs. The resource index and resource-level values are calculated. Further, the notation of "useful" or "NOT useful" as well as "safe" or "unsafe" are used for the optimal cloudlet selection.

1. In Algorithm 3, for the calculation of optimal cloudlet selection, if a cloudlet has $Ai \geq Zi$, then the value "SAFE" is pushed into the cloud list.
2. The resource-level RL is checked, if MRL ≠ NOT USEFUL and TIR = the maximum value with the minimum L, then the optimal solution exists in a safe state.

An index value has been calculated for all specific resources (CPU, RAM, Storage, Cost, etc.) and assigned certain parameters (weight, Cloudlet, availability, and total resource usage), where the weight is assigned by the owner of the resources. The index value is a type of identification pointer that points to the resource usage by the user inside the cloudlet. Multiple resources may be available within a cloudlet and an index value will be calculated for each resource.

In Fig. 4, the *X*-axis represents the 13 cloudlets (from C1 to C13) considered in the simulation. The number of vertical bars is representing the number of resources available at each cloudlet. Minimum 1 and maximum 4 resources (from R1 to R4) are considered in the simulation. The Y-axis represents the calculated index value (as per Algorithm 1) of each resource available at the cloudlets. The higher index value represents the higher suitability of the resource for the job assignment.

Similarly, the *X*-axis of Figs. 5 and 6 represents cloudlets and their resources considered in the simulation. Whereas, the Y-axis in Fig. 5 represents the level value calculated for each resource as per Algorithm 2. The resource level value identifies the availability and demand of resources running inside the cloudlets. The Y-axis in Fig. 6 represents the optimal value of each resource calculated as per Algorithm 3. The lowest optimal value represents the most suitable resource for the job allocation.

Conclusion and Future Scope

The cloud refers to servers that are accessed over the Internet as well as the software and databases that run on these servers. The intention is to deliver efficient services to the cloud users by the cloud-service providers. However, Edge computing provides services more quickly near the requesting users. All resources are connected to edge devices, so remote or virtual access is preferred when data are shared online. Even the cloud shares its services through a username and password, enabling remote or virtual access. When the cloud federation shares its services, it connects with a broker or server, which receives new requests from users or clients. The broker receives many requests and sends them to the cloudlets, which provide the resources. However, when the brokers are unaware of the cloudlets' available resources and index values, it can consume much time or result in sub-optimal resource allocation. There are already many algorithms available that provide load balancing and resource optimization. In this paper, a simple solution is provided that allows the broker to accurately identify available resource index and resource-level values and provide services accordingly.

Further, an opportunistic network is an extension of a mobile ad-hoc network in which information is transferred between network nodes without the assumption of an existing path, using a store-carry-and-forward approach [34–38]. The opportunistic network is most suitable for various applications of vehicular ad-hoc networks [39–41]. When the number of vehicles and data to be transferred and processed are huge, cloud/edge computing is the only solution. Therefore, an emerging future research direction is an integration of vehicular opportunistic networks and cloud/edge computing.

Funding This research work is funded by 'Seed Grant to Faculty Members under IoE Scheme (under Dev. Scheme No. 6031)".

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Radouane B, Lyamine G, Ahmed K, Kamel B. Scalable mobile computing: from cloud computing to mobile edge computing. In: 2022 5th international conference on networking, information systems and security: envisage intelligent systems in 5g//6G-based interconnected digital worlds (NISS), 2022. pp. 1–6. https://doi.org/10.1109/NISS55057.2022.10085600.
2. Liu F, Tong J, Mao J, Bohn R, Messina J, Badger L, Leaf D, et al. Nist cloud computing reference architecture. NIST Spec Publ. 2011;500(2011):1–28.
3. Patidar S, Rane D, Jain P. A survey paper on cloud computing. In: 2012 second international conference on advanced computing & communication technologies, 2012. pp. 394–398. https://doi.org/10.1109/ACCT.2012.15.

4. Nayyer MZ, Raza I, Hussain SA. Revisiting vm performance and optimization challenges for big data. In: Advances in computers, vol. 114. Elsevier; 2019. pp. 71–112.
5. Nayyer MZ, Raza I, Hussain SA. A survey of cloudlet-based mobile augmentation approaches for resource optimization. ACM Comput Surv (CSUR). 2018;51(5):1–28.
6. Dolui K, Datta SK. Comparison of edge computing implementations: fog computing, cloudlet and mobile edge computing. In: 2017 global internet of things summit (GIoTS), 2017. IEEE. pp. 1–6.
7. Mahmoudi C, Mourlin F, Battou A. Formal definition of edge computing: an emphasis on mobile cloud and iot composition. In: 2018 third international conference on fog and mobile edge computing (FMEC), 2018. IEEE. pp. 34–42.
8. Voorsluys W, Broberg J, Venugopal S, Buyya R. Cost of virtual machine live migration in clouds: a performance evaluation. In: Cloud computing: first international conference, CloudCom 2009, Beijing, China, December 1–4, 2009. Proceedings 1, 2009. Springer. pp. 254–65.
9. Gritto D, Muthulakshmi P. Scheduling cloudlets in a cloud computing environment: a priority-based cloudlet scheduling algorithm (pbcsa). In: 2022 11th international conference on system modeling & advancement in research trends (SMART), 2022. pp. 80–86. https://doi.org/10.1109/SMART55829.2022.10047622.
10. Khan KA, Wang Q, Grecos C, Luo C, Wang X. Meshcloud: integrated cloudlet and wireless mesh network for real-time applications. In: 2013 IEEE 20th international conference on electronics, circuits, and systems (ICECS), 2013. IEEE. pp. 317–20.
11. Rawadi J, Artail H, Safa H. Providing local cloud services to mobile devices with inter-cloudlet communication. In: MELECON 2014–2014 17th IEEE Mediterranean electrotechnical conference, 2014. IEEE. pp. 134–38.
12. Baktir AC, Ozgovde A, Ersoy C. How can edge computing benefit from software-defined networking: a survey, use cases, and future directions. IEEE Commun Surv Tutor. 2017;19(4):2359–91. https://doi.org/10.1109/COMST.2017.2717482.
13. Jararweh Y, Ababneh F, Khreishah A, Dosari F, et al. Scalable cloudlet-based mobile computing model. Procedia Comput Sci. 2014;34:434–41.
14. Verma A, Pattanaik K. Failure detector of perfect p class for synchronous hierarchical distributed systems. Int J Distrib Syst Technol. 2016;7(2):57–74.
15. Chaurasia B, Verma A. A comprehensive study on failure detectors of distributed systems. J Sci Res. 2020;64(2):250–60.
16. Verma A, Singh M, Pattanaik KK. Failure detectors of strong s and perfect p classes for time synchronous hierarchical distributed systems. In: Applying integration techniques and methods in distributed systems and technologies, 2019. IGI Global. pp. 246–80.
17. Chaurasia B, Verma A, Verma P. Heartbeat-based failure detector of perfect p class for synchronous hierarchical distributed systems. In: Research advances in intelligent computing, 2023. CRC Press. pp. 293–310.
18. Krishna Nayak R, Srinivasarao G. A greedy load balancing strategy with optimal constraints for edge computing in industrial cloud environment. In: Innovations in computer science and engineering: Proceedings of the ninth ICICSE, 2021, 2022. Springer. pp. 31–8.
19. Moon S, Lim Y. Task migration with partitioning for load balancing in collaborative edge computing. Appl Sci. 2022;12(3):1168.
20. Satyanarayanan M, Bahl P, Caceres R, Davies N. The case for vm-based cloudlets in mobile computing. IEEE Pervas Comput. 2009;8(4):14–23.
21. Lin Z, Liu J, Xiao J, Zi S. A survey: resource allocation technology based on edge computing in iiot. In: 2020 international conference on communications, computing, cybersecurity, and informatics (CCCI), 2020. pp. 1–5. https://doi.org/10.1109/CCCI49893.2020.9256663.
22. Agarwal DA, Jain S. Efficient optimal algorithm of task scheduling in cloud computing environment. 2014. arXiv:1404.2076.
23. Mukherjee A, De D, Roy DG. A power and latency aware cloudlet selection strategy for multi-cloudlet environment. IEEE Trans Cloud Comput. 2019;7(1):141–54. https://doi.org/10.1109/TCC.2016.2586061.
24. Kaur S, Verma A. An efficient approach to genetic algorithm for task scheduling in cloud computing environment. Int J Inf Technol Comput Sci. 2012;4(10):74–9.
25. Sadiku MN, Musa SM, Momoh OD. Cloud computing: opportunities and challenges. IEEE Potent. 2014;33(1):34–6.
26. Mahajan K, Dahiya D. Cloudanalyzer: a cloud based deployment framework for service broker and vm load balancing policies
27. Mishra M, Das A, Kulkarni P, Sahoo A. Dynamic resource management using virtual machine migrations. IEEE Commun Mag. 2012;50(9):34–40.
28. Parmar D, Kumar AS, Nivangune A, Joshi P, Rao UP. Discovery and selection mechanism of cloudlets in a decentralized mcc environment. In: 2016 IEEE/ACM international conference on mobile software engineering and systems (MOBILESoft), 2016. pp. 15–6. https://doi.org/10.1145/2897073.2897114.
29. Rudra B, Verma A, Verma S, Shrestha B. Futuristic research trends and applications of internet of things. Boca Raton: CRC Press; 2022.
30. Srivastava S, Verma A, Verma P. Fundamentals of internet of things. In: Futuristic research trends and applications of internet of things. CRC Press; 2022. pp. 1–30.
31. Verma A, Verma P, Farhaoui Y, Lv Z. Emerging real-world applications of internet of things. Boca Raton: CRC Press; 2022.
32. Liu H, Li S, Sun W. Resource allocation for edge computing without using cloud center in the smart home environment: a pricing approach. Sensors. 2020;20(22):6545.
33. Zhu A, Wen Y. An efficient resource management optimization scheme for internet of vehicles in edge computing environment. Comput Intell Neurosci. 2022;2022.
34. Verma A, Srivastava DA. Integrated routing protocol for opportunistic networks. 2012. arXiv:1204.1658.
35. Verma A, Pattanaik K, Ingavale A. Context-based routing protocols for oppnets. Routing in opportunistic networks. New York: Springer; 2013. p. 69–97.
36. Verma A, Verma P, Dhurandher SK, Woungang I. Opportunistic networks: fundamentals, applications and emerging trends. New York: CRC Press; 2021.
37. Verma A, Singh M, Pattanaik K, Singh B. Future networks inspired by opportunistic networks. In: Opportunistic networks. Chapman and Hall; 2018. pp. 230–46.
38. Verma A, Pattanaik K. Routing protocols in opportunistic networks. In: Opportunistic networking. Boca Raton: CRC Press; 2017. pp. 123–66.
39. Singh M, Verma P, Verma A. Security in opportunistic networks. In: Opportunistic networks. Boca Raton: CRC Press; 2021. pp. 299–312.

40. Singh M, Verma A, Verma P. Empirical analysis of the performance of routing protocols in opportunistic networks. In: Research advances in network technologies. Boca Raton: CRC Press; 2023. pp. 257–72.
41. Gond MK, Singh M, Verma A, Verma P. Average time based prophet routing protocol for opportunistic networks. In: International conference on advanced network technologies and intelligent computing. Berlin: Springer; 2022. pp. 400–12.