National College of Ireland

Cloudlet caching algorithm to improve quality of experience for users in vehicular edge computing

MSc Research Project
Cloud Computing

Nileshwari Vispute
Student ID: x19200960

School of Computing
National College of Ireland

Supervisor: Jitendra Kumar Sharma

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Nileshwari Chandrakant Vispute
Student ID:	x19200960
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Jitendra Kumar Sharma
Submission Due Date:	15/08/2022
Project Title:	.
Word Count:	6533
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Nileshwari Vispute.
Date:	15th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Cloudlet caching algorithm to improve quality of experience for users in vehicular edge computing

Nileshwari Chandrakant Vispute
x19200960

Abstract

Cloud computing has witnessed tremendous growth over the past few years due to its capacity to provide customers and organizations with mighty computational power for data processing and analysis in their data centers at reasonable prices. However, owing to the lack of cloud data centers worldwide, accessing faraway clouds results in increased response latency, hence reducing the efficiency of cloud computing. As a result, it may be able to provide users with robust computation resources and low turnaround time due to its location at the edge.However, because a cloudlet is a mini-scale cloud, it must choose which data should be cached in its data center. This project intends to provide cloudlet information caching.Utilizing k-means clustering, the technique identifies the diverse usage patterns and the most popularly requested content to cache.The selection rule is then utilized to get potentially accessible data. This method was tested on AWS lambda edge. The findings indicate that, compared to existing caching strategies, the proposed method boosts the cloudlet content cache hit ratio and reduces response latency.The recommended technique has already been deployed on AWS lambda edge. Therefore, this post will attempt to develop it utilizing the cost-efficient and potentially more effective open-source platform OpenStack.

1 Introduction

Cloud computing is a paradigm which is providing a resource-rich, dynamic, adaptable and economical service to a user located all over the world. Recently, this technique has become a dynamically developing technology as it is proving a stable, consistent, and effective method for data access. Whilst cloud computing enhances cost-efficiency, it still has some downsides. One of the greatest disadvantages is the lack of data centers generally available around the globe. Because of this, a user that accesses the data from a distance from the data center may have a prolonged response latency. Constructing new data centres is impossible since the process is too expensive and time extensive. In order to solve this problem, mobile edge computing, a small-scale form of cloud computing, is introduced.

ETSI and ISG, telecommunication standardization bodies, introduce mobile edge computing, also known as multi-access edge computing.In 2015, they issued a white paper (1) that describes MEC as a technology that enables IT software services and cloud computing possibilities at the network's edge, allowing users to contact the closest available server within Radio Access Network (RAN) range. MEC helps end-users to meet

their computing needs and performs offloading of computations when low-computing or time-sensitive tasks are required (2).

Vehicular edge computing extends to the booming mobile edge computing with a traditional vehicular network. VEC aims to bring connectivity, computing, and cache management services proximate to vehicular end users. In contrast to standard MEC, VEC prioritizes vehicle activity, resulting in more rapid and dramatic topology changes and complex communication features. However, the time and energy-intensive technique is severely performed in a vehicular network due to MEC's insufficient resources (3).

As MEC is referred as small-scale cloud computing, cloudlets can be referred to as small-scale cloud servers, as cloudlets are positioned close to the user. As with MEC, cloudlets provide the offloading of computational tasks from the mobile user to the network. The term cloudlet was originated by Satyanarayanan et al.(4). It refers to a mini server positioned close to the end user delivering cloud functionalities. It focuses primarily on legitimate and latency-sensitive activities. However, cloudlets do not require interaction with a centralized cloud server; thus, they can operate independently. Cloudlets employ virtualisation on the basis of virtual machines, while MEC may also consider other techniques like docker.

1.1 Motivation

Internet-based vehicular networks have arisen to provide a platform that connects all automobiles using the Internet of things paradigm. Multiple IT advancements have been made to connect all vehicles for intelligent roads and traffic management. Smart sensors and actuators are installed in vehicles and roadside units for data collection and analysis and are linked to the Internet via advanced, effective communication. Despite the remarkable development of technologies in vehicular networks, certain obstacles impede the expansion of vehicular networks due to limited storage. Therefore, it is difficult for each vehicle to provide active support for applications requiring complex analysis and vast storage. Bringing computational power nearer to the vehicles or sharing them among them is a potential solution to these problems. This inspires the paper to develop a vehicular network based on cloudlets.

1.2 Research Question

Can the quality of experience for users in vehicular edge computing be improved using the cloudlet cluster based caching algorithm on OpenStack?

Caching techniques for online caching, content distribution networking, and data-centric networking have received a lot of attention in wired networks. The basic idea behind caching is to pre-cache data in multiple locations in order to reduce network congestion caused by heavy traffic.

MEC is a revolutionary technology that relocates caching and compute assets to the network's edge so that users may easily get data according on their needs (5). A cloudlet can utilise its local memory to save cache data and transmit it to mobile devices on demand. A cloudlet design can predict future user requests based on prior behaviour and location by storing the content. Likewise, the cloudlet can operate as a remote server for the central cloud server. It can cache info asked by one device and serve that info to the same device in the upcoming request. In this application, cloudlets can assist users in locating nearby public services and other pertinent information. A Cloudlet can

save data for such services before their usage, based on geographical data, and offer the stored information to connected device (6). Storing the data in the cloudlets surrounding the device will enable faster retrieval of the required data due to low latency and high bandwidth (7).

Meanwhile, suppose the desired information is missing at the cloudlet. In that case, users must wait to get it because they are not directly linked to the central cloud, resulting in a degraded user experience (QoE). Consequently, it is essential to utilize cloudlet sources intelligently by selecting the precise data to cache. Zhang et al.(8) have suggested and tested a cluster-based cloudlet caching method in the AWS environment using lamda egde to address this issue. This study will implement the same methodology on OpenStack, an open-source technology.

1.3 Structure Of The Paper

The remaining paper is designed as follows. Section 2 describes the Related work of cloudlet and caching technique. Section 3 provides the methodology for obtaining the desired outcome . Section 4 gives the design specification of the proposed system. Section 5 will provide a comprehensive explanation of the implementation. and Section 6 and 7 give the performance evaluation along with discussion and conclusion respectively.

2 Related Work

2.1 Vehicular edge computing

Regarding the general vehicular network, Vehicular Edge Computing is a compliment to Mobile Edge Computing. Components for computation, connectivity, and caching will be moved closer to end users by VEC. This is the main factor contributing to the growing demand for VEC as it only concentrates on providing steadily growing low latency and high transmission rate for the edge devices. As the vehicle position is dynamic and constantly changing, which would require complex computations to establish the network, which would lead to rapid variations in the networking context over time, are the distinctive features of Vehicular Edge computing as compared to Mobile Edge Computing. VEC systems have a very efficient system named caching, which helps in reducing the workload that needs to be transmitted to enhance the Quality of Experience(QoE). Cache in each node is limited to a certain amount and cannot exceed because of the storage limitations (3).

Due to the increasing number of various data-consuming applications, a significant increase in data traffic for the vehicular network has happened as people have started shifting their focus toward smart vehicles. This created an issue for the backhaul network. To address this issue, Lien e. al. .(9) suggested a system built on edge with energy-efficient caching that uses Lyapunov optimization and optimal control theory.

The roadside unit can be utilized by transferring the most requested content near the user vicinity as the form of cached content in the vehicular network. However, storing data and connecting ability of RSUs are constrained, making it difficult to choose whether content should be considered good enough to be cached. The main issue here would be the size of files, as the files with audio or video content range in larger sizes. To enhance the QoE of the system, it is necessary to make effective decisions regarding the caching material.

Hu et al.(10) analysis of the situation took into account various views of the suppliers. A multi-object auction method was suggested as an alternative to cope with the competition of multiple contents caching their material while utilizing constrained resources. The whole system's throughput is managed using the segmentation method, which is the algorithm used to use RSU properly. Hence, the user should download the data which is remaining in a manual fashion or wait till RSU is connected. Rahim et al. (11) proposed the solution in which the RSU will store everything, and once the user connects with the RSU, it can download all its data directly. Ding et al. (12) have concentrated on distributing the data among the various RSU by considering the quantity of data and analysing the amount of RSU present and the cache storage.

2.2 State-of-the-art Cloudlet

In today's world, everyone is surrounded by mobile gadgets, and with the help of information services, they can converse with other people. These applications are becoming resource hungry and need to increase the device's battery life. Because there are limitations on the devices such as the processing power, ability to storage space and bandwidth for communication. To solve this problem, scientists have devised an idea to relocate the place where computation takes place, i.e. distant cloud servers, and use those servers as the extension for data processing, thanks to the notion of mobile cloud computing (13).

Cloud Servers used in MCC have assisted in the expansion of the processing and storing capability inside the mobile applications. The data calculations are done in MCC on distant cloud servers using a WAN. But because of the intricate structure of the WAN, the performance of some applications requiring low latency may get affected as intricate structure means high latency sometimes. This was the main reason for Satyanarayanan et al. (4) to propose cloudlet, a mobile cloud computing extension. A cloudlet is a dependable, powerful computing device or group of computing devices in constant contact with the internet and access to adjacent mobile devices. The reason cloudlet is so popular is that it is just one hop away the bandwidth for transmission is also very high. They can meet the real-time needs of mobile engagement. Offloading the resource-intensive jobs to cloudlets might save a lot of time for the device.

Cloudlet will operate as a small subset of the central cloud to carry out its fundamental functions in the event of a failure. After the job is completed, a completed task might need to be sent to the main cloud for validation. Cloudlets also offer another advantage, i.e. the maintenance of confidentiality and privacy. Users' data must be sent to a remote cloud server for computation on the cloud, raising concerns about the security of the data. However, employing cloudlets will keep all sensitive information beside the user, protecting security and privacy (14).

2.3 Data Caching Technique

The process of compiling data replicas using directories in a temporary area so users may quickly access the material is known as caching. Various caching methods are used in MCC to store frequently required data close to the user, thus reducing the backhaul and enhancing the user quality experience (15) (16).

2.3.1 Caching Techniques in Cloudlets

Caching the content in the gadget's cloudlets will aid in obtaining the necessary information faster since short-range, single-hop networks between cloudlets and edge devices offer low latency and high bandwidth. Before considering data caching in cloudlets, the following crucial factors must be taken into account (7):

- **Data needs for future use:** Data prediction is an excellent option for deciding which data should be considered for caching. The data forecast can be given by users viewing the data they accessed in the past. For instance, Lymberopoulos et al. (17) have used the machine learning method based on the stochastic gradient boosting technique to give the site of data web browsing for prediction. In contrast, Pitkow et al.(18) use the Markov model approach, which only considers the features that change with time but not a location for future access to a user's data on the web.

- **Prior to being sent to mobile apps, data can be processed** To prevent converting incorrect data to the gadget, prefetching can be considered on the cloudlet after the future data has been forecasted. Additionally, the cloudlet may forecast data from several endpoints, decreasing duplicate data by caching the data just once if they make the same request more than once. This same module establishes a connection to the closest cloudlet upon a phone and offers that the data must be stored if the latency of the file system is very high (19).

- **Deciding which cloudlet will perform pre-fetching:** The user mobility must be predicted to choose the cloudlet that will handle data prefetching. Users' movement can be forecasted based on their present position, status, movement direction or speed. Akyildi et al. (20) developed an algorithm to signify the future's role or state by examining the travelling principle. Nicholson et al.(21) developed a motion forecasting method that looks at the users' often travelled similar pathways. Then, using the information obtained, a forecast is created.

The customer profits by caching data on cloudlets because it reduces the period of time it takes to connect to and receive information from the server, which enhances the user QoE. Most of the time, cloudlets can cache the data provided by Content Delivery Network (CDN) servers (7).

Koukoumidis et al. (22) make use of the huge storage capability of smartphones to cut down on latencies and power use when accessing cloud-based services. They have put forth a micro cloudlet; a memory buffer architecture made up of the ROM of the gadget. This architecture incorporates a small environment and single-user access strategies to increase hit rate, cut overall service latency, and save energy.

2.4 Comparison of algorithm analysis

Reference	Title	Design /Algorithm	Outcome
(23) Yang et al.	AmazingStore: Available, Low-Cost Online Storage Service Using Cloudlets	A system that uses cloudlets to extend the cloud server storage.	Data accessibility is improved along with reducing storage cloud expenses
(24) Ando et al.	In-Network Cache Simulations Based on a YouTube Traffic Analysis at the Edge Network	A system for reducing redundant video data by caching material on an edge router	Data redundancy is reduced
(25) Guan et al.	A Cloudlet-based task-centric offloading to enable energy-efficient mobile applications	Task offloading using cluslet based model	Computational efficiency of task and total throughput achieved with energy conservation

Table 2: Comparison of algorithm analysis.

3 Methodology

This sections describes about the methodology which is used in this research. The process flow is explained in 3.1 and 3.2 section will give a overview of tools and technologies used.

This research aims to improvise the end user's quality of experience in vehicular edge computing by using cloudlets and caching techniques. A cloudlet caching algorithm is performed on the data from the central cloud to provide data using low latency and bandwidth to the user. Then by using the selection rule, the result data is stored in the cache memory of cloudlet.

In this research, the user firstly asks for the data from the cloudlet. Cloudlet will check its local cache memory for data availability. If data is available, it will provide the result to the end user. If data is unavailable on the cloudlet, then the cloudlet will go to the central cloud for data requests and fetch the data from the central cloud. Cloudlet will use the user clustering technique on the central cloud data and then, using the selection rule, will store the result data in the cache memory and return the requested data to the user.

3.1 Flow chart of research method

Below is the flow chart of complete process implementation. Firstly input is taken from the user and then searched it on cache storage OpenStack Cloudlet. If data is not available then it will access the storage of AWS cloud. It will perform then k-means clustering and selection rule respectively on the data and then stored in cloudlet cache storage. At the end it gives the required output to the user.

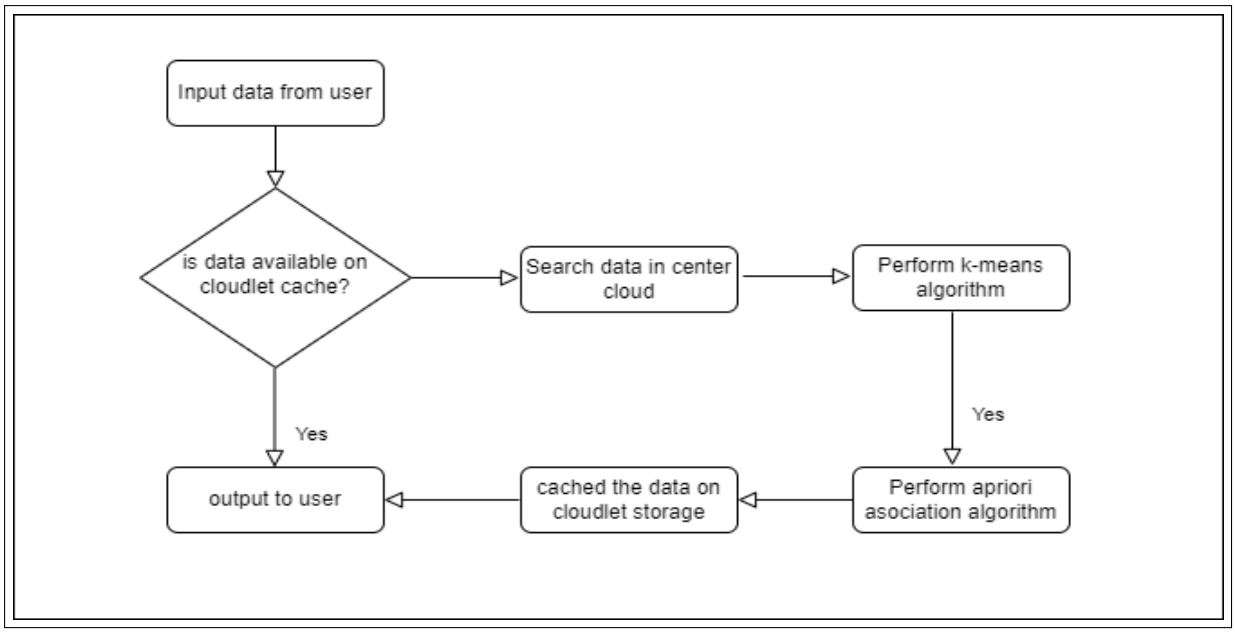

Figure 1: Flow chart of research method.

3.2 Tools and Technologies Used In Research

This project is performed on the OpenStack instance of m1.medium (Ubuntu). The user application and algorithms are developed in python language.

The following tools and technologies are used in the given project:

- **MySQL:** MySQL relational database management system which is used here for the data storage. The version of MySQL used is 8.0 (26).
- **AWS Relational Database Service(RDS):** Amazon cloud is used as central cloud in the research. AWS RDS Service is relational database service that simplifies the configuration, administration, and scalability of relational databases in the AWS Cloud. It provides scalable, cost-effective storage for an industry-standard relational database and performs common database administration activities. All the datasets which are been used in the project are configured with AWS RDS service (27).
- **Python3 :** Python is interpreted and object oriented programming language. The user application is built up on python language hence python3 is installed on the cloudlet instance.
- **scikit-learn :** The scikit-learn is a open source machine learning library. It is a simple yet efficient tool used for predictive data analysis. It is used to perform k-means algorithm on a data (28).
- **mlxtend :** mlxtend is a maching learning extension and is a python library which consist of various tools which are beneficiary for the daily data science operations. This library is used to perform apriori association rule (29).

3.3 Assessment Carried Out In This Research

A total of two experiments is performed to obtain evaluation of the system. First one has been to assessed the cache hit ratio by comparing the proposed algorithm with the least recently used (LRU) which is a traditional type of cache algorithm. In second experiment response time is measured for data access by comparing if data is available on cloudlet or it has been fetched from central cloud.Both experiments are examined using different volumes of dataset to obtain consistency in the result.

4 Design Specification

The further section is describes about the design specification of the project. The section includes 4.1proposed specification of the system, 4.2proposed architecture of cloudlet and 4.3proposed caching algorithm.

4.1 Proposed Specification Of The System

In this research, OpenStack virtual instance is used as a cloudlet. This instance is created using Ubuntu 20.04-x86_x64 as a machine image with flavour type m1.medium. The instance is created using keypair. A floating IP address is allocated to this instance to access it dynamically. The instance is connected using a key pair and floating IP address through the ssh client. AWS is taken as a central cloud in this project. The dataset is configured on the AWS RDS service using MYSQL. The hardware specification of the instance is.

Virtual Machine	
vCPU	2
Memory	40GiB
RAM	4 GB

Table 2: Required configuration for the Virtual machine.

4.2 Proposed Architecure Of The System

The suggested architecture is made up of three tire architecture (user-cloudlets-cloud). Cloudlets are nothing but mini data centres deployed on the edge of a network near the user. Cloudlet can be said as a communicator between the user and the central cloud. Some data is scattered on the cloudlet if a user application is built using this architecture. A user will ask for the data to its nearby cloudlet according to his requirement. As soon as a data request is received on cloudlet, it will check its storage or cache to see if the data is available or not. If it finds the data, it will be returned to the user. If it does not find any data in its data storage, then cloudlet will make a call to the central cloud and search for the data in it. As soon as the data is found on the central cloud, it is delivered to the user by cloudlet. It is essential to store frequently accessed data on the cloudlet to improve the quality of experience for the user by improving the efficiency and performance of the system. As cloudlets are located nearby the user, the user can readily retrieve the information with reduced latency if the cloudlet can supply the needed data without contacting the central cloud. If the caches in the cloudlet are missed, it will take a longer time to reply and retrieve the data from the central cloud and give it to the requested user. As a result, depending on how to achieve the cache hit ratio for cloudlet, the three-tier cloudlet framework can aid in improving user application speed.

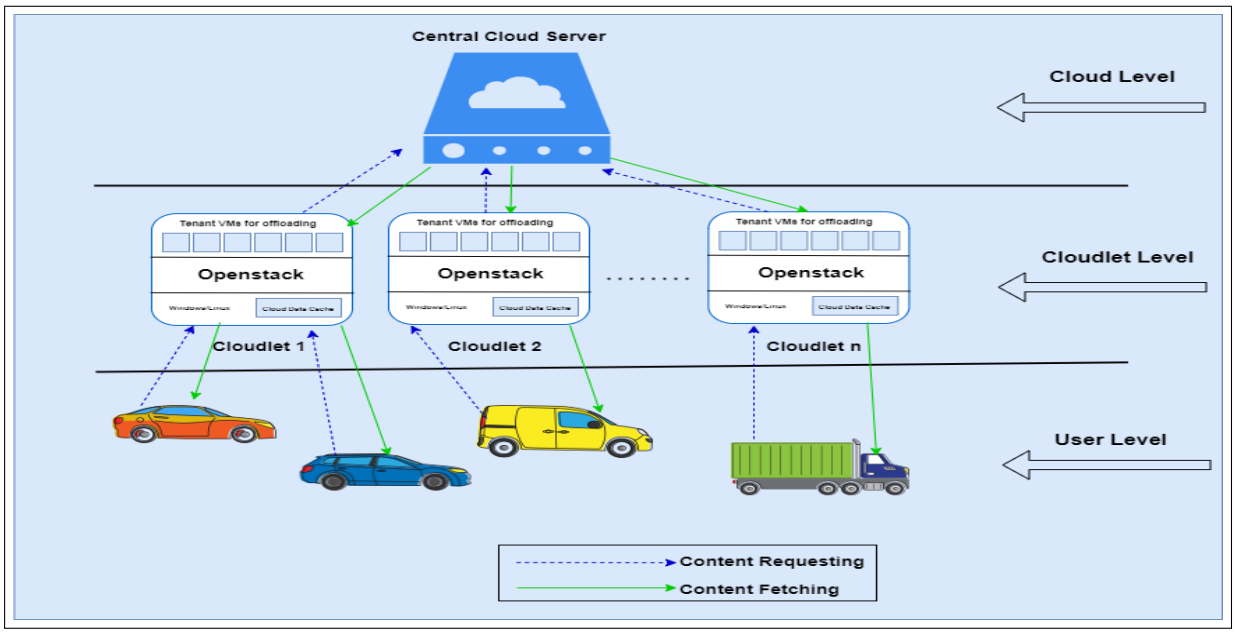

Figure 2: Proposed Architecture Of System

4.3 Proposed architecture of cloudlet

Cloud utilises a virtual machine or VM-based architecture to perform the offloading operations. The cloudlet has anticipated an active VM-Synthesis system. With dynamic VM synthesis, the interaction between the user device and VM is user-driven and on-demand; Therefore, edge devices can quickly and dynamically start the virtual machines and collaborate with cloudlet to fulfil demands. Considering the robustness of the Wide Area Network (WAN) connection, the VM can be dynamically created and terminated.Using VM synthesis, the end user module sends a small virtual machine layer to the cloudlet setup, which already possesses the parent virtual machine from where the overlay was generated.The architecture overlays the foundation to the launch VM, which continues execution in the same state as when it was halted.

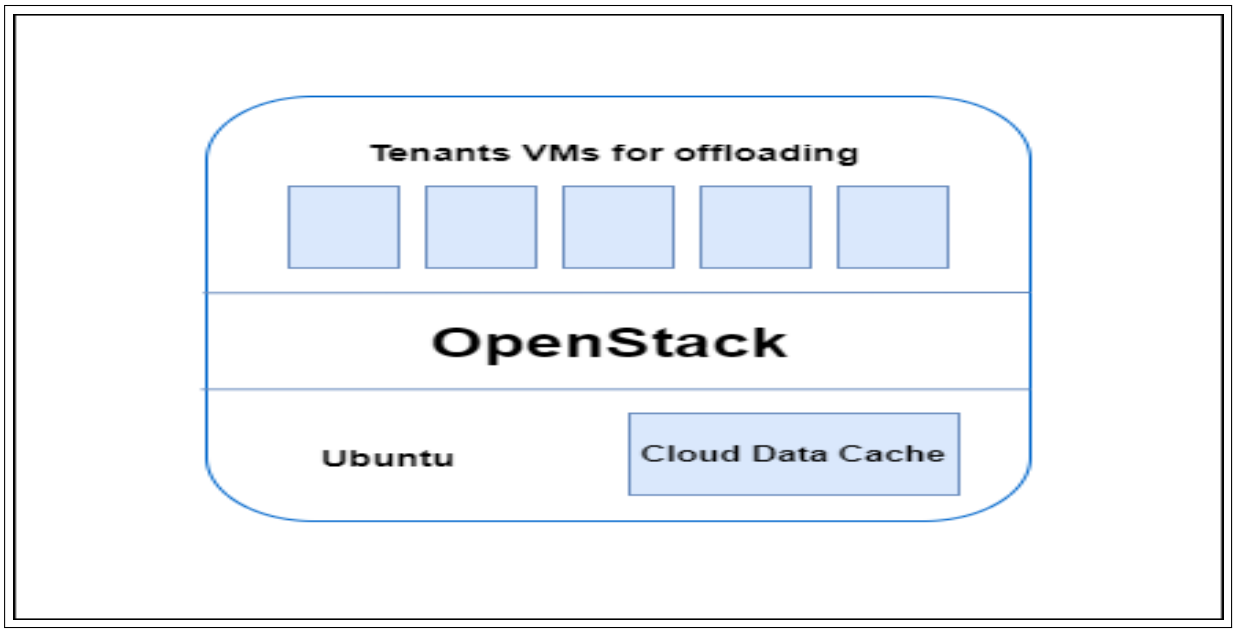

Figure 3: Proposed Architecture Of System

5 Implementation

This section explains the implementation of the algorithm of current research. The algorithm will help to improvise the quality of experience (QoE) for a user by boosting the ratio of cache hit and reducing the latency and time. The algorithm uses k-means clusterisation of data along with the association rule to accomplish the purpose. The clusterization will be doThis research used 3 to 6 road traffic data from the (30). These datasets are public, so we can use them without any cost. (30) is the site where every type of dataset is publicly available for education and research. The dataset is downloaded and configured with AWS RDS service for testing purposes as AWS acts as a central cloud in the study.ne by analysing the pattern, and the data will be saved on the cloudlet using the association rule.

5.1 Dataset and central database connection

This research is carried out using 3 to 6 road traffic data from the (30). These dataset are public so we can use it without any cost. (30) is the site where every type of dataset is available publically for education and research purpose. In the research, the dataset is downloaded and configured with AWS RDS service for testing purpose as AWS is acting as a central cloud.

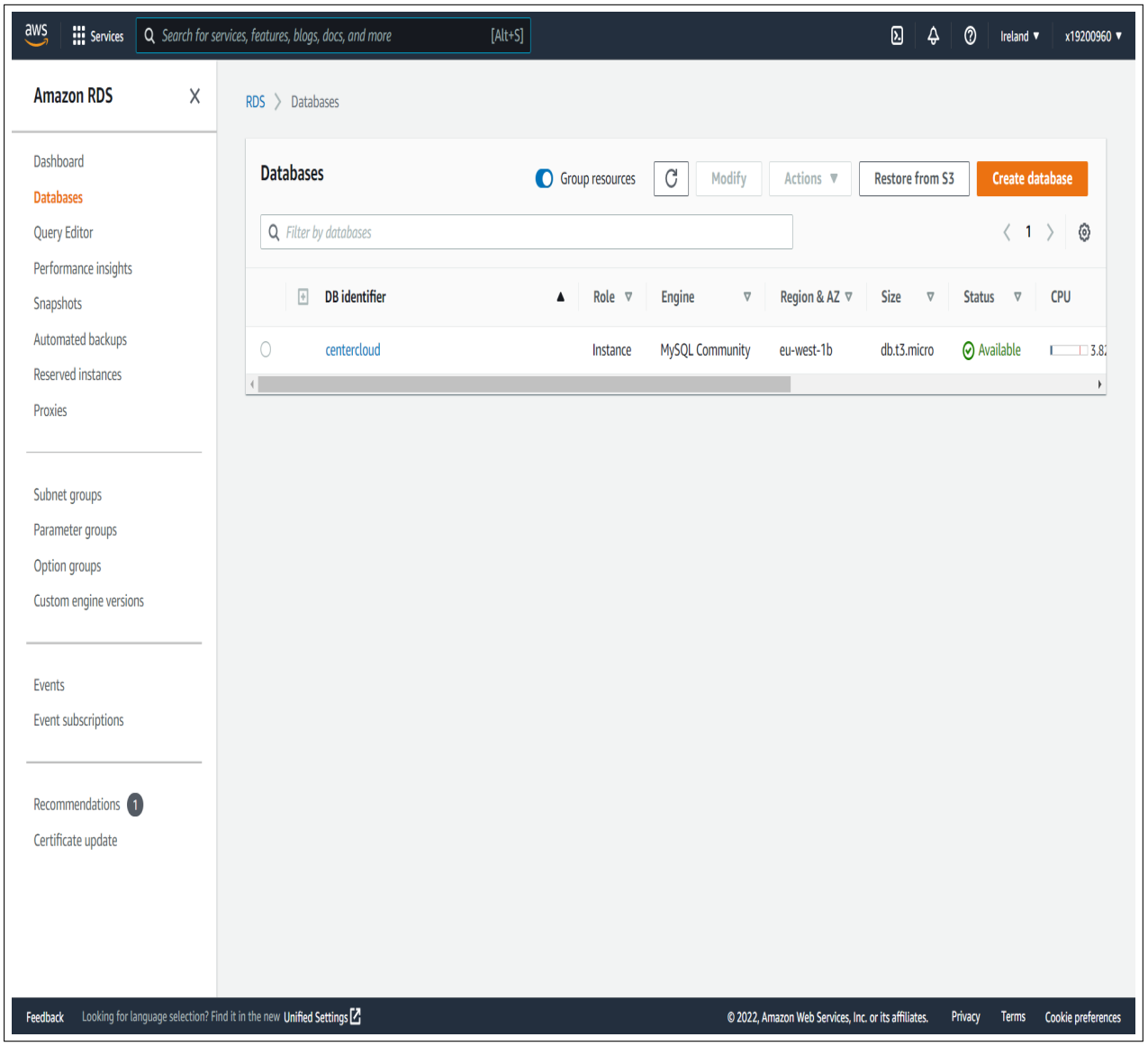

Figure 4: AWS RDS Service

5.2 Algorithm

The algorithm implemented in the research is divided in two parts. First is clusterization of data is done on the basis of accesing pattern and then this data is stored on cloudlet using selection rule. For clusterisation technique , k-means clustering algorithm is used and for selection rule, apriori association rule is used.

5.2.1 k-means algorithm

The proposed algorithm start with clusterising of data on the basis of several access pattern.K-means algorithm is based match for this task. It is defined as follow:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} dist(x, \mu_i),$$

where, S is all present clusters
x is d-dimentional vector
μ_i is mean of one cluster

Figure 5: k-means cluster equation.

The k-means clustering approach uses iterative refinement. When there isn't any modification in the task or when the pre-determined iterations value is fulfilled, the K-means clustering algorithm works in a similar way.As the concept of k-means clustering implies, a number of clusters must be entered. Hence the Elbow method is used to calculate the ideal number of cluster to perform k-means algorithm. This algorithm's time complexity is $O(n^2)$. The pseudocode for algorithm is given below.

Algorithm 1 pseudocode for k-means clustering

```
procedure cluster(data)
    variance = [];
    for k=1 to n do
        km = kmeans(data,cluster=k)
        variance.append(km.variance)
    for k = 1 to n do
        plot(k,variance[k])
    return elbow_n
```

Figure 6: Pseudocode for k-means algorithm

5.2.2 Selection algorithm

After cluster analysis, all clusters are reviewed to assess the group with the most people and most frequently accessed data. Once the algorithm starts, the highest key data is get selected. Then, starting with the connected list of data points, a data item with a high ranking is selected. If there is any memory in the cache, the algorithm selects the second essential pair in the same way, and so on. This algorithm's time complexity is $O(n^2)$. The pseudocode for an algorithm is given below.

```
Algorithm 1 pseudocode for selection algorithm
procedure preparation(data,datapoints)
      for each dataobj in data do
          p_f = access_frequency(dataobj)
          p_s= size(dataobj)
          p_rating = p_f/p_s
          for each dataitem in data.datapointdo
              d_f = access_frequency(dataobj)
              d_s= size(dataobj)
              d_rating = d_f/d_s
          sort(dataobj,datapoints,d_rating)
      sort(data,p_rating)
end procedure
procedure selection(data,datapoints,cache_size)
      for eachdataobj in data do
          S = S + size(dataobj)
         for each dataitem in data.datapointdo
              S = S + size(dataitem)
      y= S/ cache_size * mean(datapoints)
      i = 1; C_remaining = cache_size
      while C_remaining > 0 do
          select p(i)
          C_remaining = C_remaining + size(p(i))
          for j = 1 to length(p(i).d) do
              if p(i).d(j) > y do
                  select p(i).d(j)
              C_remaining = C_remaining + size(p(i).d(j))
              i++
      If there is memory remaining, select the data item with the highest rating value
until memory full.
end procedure
```

Figure 7: Pseudocode for selection algorithm

6 Evaluation

6.1 Performance Evaluation

The application went through some experiments to measure its performance. The results are calculated by comparing experiment cases with a traditional caching algorithm. Each experiment is performed at least five to six times for better results and efficiency. All the experiments are carried out on the OpenStack instance.

These experiments are tested on different volumes of a dataset, and the observation is noted after every experiment. In this experiment, the cache size of the instance is considered. Records are being cached by taking cache memory into account. The experiment proceeds by considering different cluster values in the K-means algorithm to get an efficient result and find the ideal number of a cluster for evaluation. Similarly, in association rules, multiple support and confidence values are considered for caching the record and finding the perfect support for evaluation. The cache hit ratio is calculated and compared between the traditional and proposed algorithms in the first experiment. The second experiment measures response time for the traditional and proposed algorithm. In the second experiment, the proposed algorithm is divided into two scenarios. The first scenario is fetching requested data from the cloudlet server if available. In the second scenario, the requested data is fetched from the center cloud server as it was unavailable on the cloudlet server.

6.2 Experiment 1

The first experiment is carried out by performing the LRU policy algorithm and proposed caching algorithm. In this experiment, user input is searched into the memory using these algorithms, respectively, and their results are observed. This experiment is performed to calculate the cache hit ratio of algorithms. A cache hit is calculated by using the following formula:

$$\frac{\text{Number of cache hits}}{(\text{Number of cache hits} + \text{Number of cache misses})} = \text{Cache hit ratio}$$

Figure 8: Cache hit Formula

The observed cache hit ratio for this experiment is given in Table 3

Cache hit ratio	
Traditional Algorithm	**Proposed Algorithm**
10	**5**

Table 3: Observations after experiment one

6.3 Experiment 2

In this experiment, proposed algorithm is divided into two scenarios. One is when user requested data is fetch from cloudlet server and other is when user requested data is fetch from center cloud server.This experiment is carried out to measure response time of a application with and without cloudlet. The experiment is done various time and the average response time is observed. Table 4 shows the observed value of this experiment.

Average Response Time(ms)	
Data fetched from cloudlet	**Data fetched from center cloud**
2500	**5300**

Table 4: Result comparison after experiment two

6.4 Discussion

The result of the experiment varied while performing as those were performed on different volumes of a dataset.

Traditional and proposed algorithms measure the cache hit ratio in the first experiment. The traditional algorithm uses the LRU(least recent used) policy, which puts recently used data at the top of the cache list. Whenever new data is requested to access, the LRU puts it at the top of the cache. Once the cache limit exceeds the least used data is removed from the list. Here least accessed data will always lie at the bottom of the list; hence the removal of data starts from the bottom. The proposed algorithm uses k-means clustering and association rule to cache the data on cloudlet. Using this algorithm most frequently used data with their associated data is stored on the cloudlet. As a result, the proposed algorithm performs better than the traditional one and has a greater cache hit ratio than the traditional one which ultimately results in improved quality of experience (QoE) for user as he gets a data within less time.

The below figure shows the comparison between cache hit ratio of both algorithms.

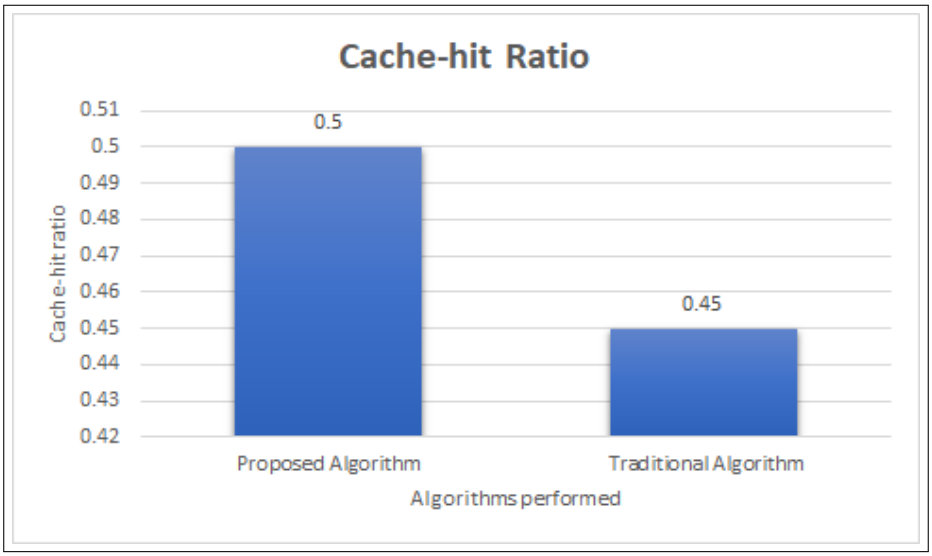

Figure 9: Cache hit ratio result

The second experiment measures average response time by splitting a proposed algorithm into two scenarios. The first scenario considers fetching requested data from the cloudlet server where the requested data has already been cached. In the second scenario, the requested data has not been cached on the cloudlet; hence, it will fetch from the center cloud and present to the user. As a result, the first scenario takes less time to provide data to a user than the second scenario. The user gets their requested data more quickly in the first scenario than in the second one, as data is already available on the cloudlet. Therefore the quality of experience is better in the first scenario than in the second one. The below graph shows the comparison between the average response time of both scenarios.

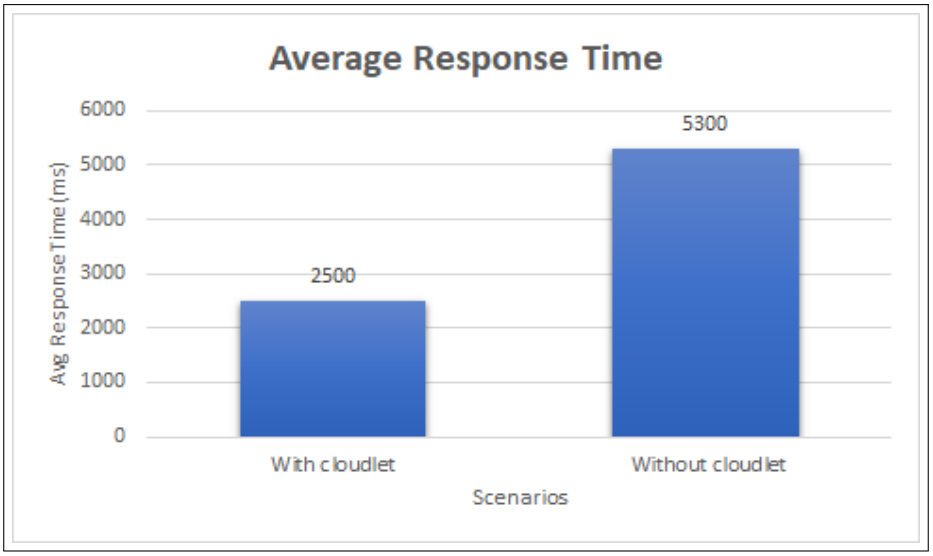

Figure 10: Average Response Time of proposed algorithm

7 Conclusion and Future Work

In this paper, we proposed cloudlet as a mid-layer in a three-tier architecture system to improvise the quality of experience in vehicular edge computing. Since the cloudlet is located on the edge server, it has limited memory to save the data. Hence we used the cloudlet caching algorithm to reduce the latency and response time of the data access by the user by achieving hi cache hit ratio.The algorithm firstly used a clustering technique to recognise the access pattern of a user on a data and then analyse them. After analysis, the selection rule is applied to the data to store it in cloudlet cache memory. The cache hit ratio of the proposed algorithm is compared with traditional and found that the proposed algorithm has a high cache hit ratio. After experimenting with response time evaluation, it is found that response time is less if the data is stored on the cloudlet than the data fetched from a central cloud. In conclusion, using the cloudlet caching algorithm, the wise utilisation of cache memory is achieved and the quality of experience for the user is improved.

References

[1] Y. Hu, M. Patel, D. Sabella, and V. Young, "Mobile edge computing a key technology towards 5g," 2015.

[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, pp. 450–465, 02 2018. JCR Impact Factor: 9.936.

[3] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile Networks and Applications*, 07 2020. JCR Impact Factor: 3.426.

[4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, 10 2009. JCR Impact Factor: 3.175.

[5] F. Zeng, Y. Chen, L. Yao, and J. Wu, "A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing," *Peer-to-Peer Networking and Applications*, vol. 14, pp. 467–481, 09 2020. JCR Impact Factor: 3.06.

[6] U. Shaukat, E. Ahmed, Z. Anwar, and F. Xia, "Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges," *Journal of Network and Computer Applications*, vol. 62, pp. 18–40, 02 2016. JCR Impact Factor: 7.09.

[7] Z. Pang, L. Sun, Z. Wang, E. Tian, and S. Yang, "A survey of cloudlet based mobile computing," 2015 International Conference on Cloud Computing and Big Data (CCBD), 11 2015. CORE Rank: Not available.

[8] Z. Zhang and W. Hao, "A new caching algorithm for boosting edge computing performance," p. 0168–0175, 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 10 2020.

[9] S.-Y. Lien, S.-C. Hung, H. Hsu, and D.-J. Deng, "Energy-optimal edge content cache and dissemination: Designs for practical network deployment," *IEEE Communications Magazine*, vol. 56, pp. 88–93, 05 2018. JCR Impact Factor: 9.619.

[10] Z. Hu, Z. Zheng, T. Wang, L. Song, and X. Li, "Roadside unit caching: Auction-based storage allocation for multiple content providers," *IEEE Transactions on Wireless Communications*, vol. 16, pp. 6321–6334, 10 2017. JCR Impact Factor: 10.76.

[11] M. Rahim, S. Ali, A. N. Alvi, M. A. Javed, M. Imran, M. A. Azad, and D. Chen, "An intelligent content caching protocol for connected vehicles," *Transactions on Emerging Telecommunications Technologies*, vol. 32, 02 2021. JCR Impact Factor: 2.638.

[12] R. Ding, T. Wang, L. Song, Z. Han, and J. Wu, "Roadside-unit caching in vehicular ad hoc networks for efficient popular content delivery," p. 1207–1212, IEEE Xplore, 03 2015. CORE Rank: B.

[13] Y. Wang, I.-R. Chen, and D.-C. Wang, "A survey of mobile cloud computing applications: Perspectives and challenges," *Wireless Personal Communications*, vol. 80, pp. 1607–1623, 10 2014. JCR Impact Factor: 1.95.

[14] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, pp. 30–39, 01 2017. JCR Impact Factor: 2.683.

[15] Y. Zhao, W. Zhang, L. Zhou, and W. Cao, "A survey on caching in mobile edge computing," *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–21, 11 2021. JCR Impact Factor: 2.96.

[16] L. Li, G. Zhao, and R. S. Blum, "A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies," *IEEE Communications Surveys Tutorials*, vol. 20, p. 1710–1732, 2018. JCR Impact Factor: 25.249.

[17] D. Lymberopoulos, O. Riva, K. Strauss, A. Mittal, and A. Ntoulas, "Pocketweb: instant web browsing for mobile devices," *ACM SIGARCH Computer Architecture News*, vol. 40, pp. 1–12, 04 2012. JCR Impact Factor: Not available.

[18] J. Pitkow and P. Pirolli, "Mining longest repeated subsequences to predict world wide web surfing," www.usenix.org, 1999. CORE Rank: Not available.

[19] J. Flinn, S. Sinnamohideen, N. Tolia, and M. Satyanaryanan, "Data staging on untrusted surrogates," FAST'03, USENIX Association, 03 2003. CORE Rank: A.

[20] I. Akyildiz and W. Wang, "The predictive user mobility profile framework for wireless multimedia networks," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 1021–1035, 12 2004. JCR Impact Factor: 3.56.

[21] A. J. Nicholson and B. D. Noble, "Breadcrumbs: Forecasting mobile connectivity," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, (New York, NY, USA), p. 46–57, Association for Computing Machinery, 2008. CORE Rank: A+.

[22] E. Koukoumidis, D. Lymberopoulos, K. Strauss, J. Liu, and D. Burger, "Pocket cloudlets," *ACM SIGPLAN Notices*, vol. 46, pp. 171–184, 03 2011. JCR Impact Factor: 3.25.

[23] Z. Yang, B. Y. Zhao, Y. Xing, S. Ding, F. Xiao, and Y. Dai, "Amazingstore: Available, low-cost online storage service using cloudlets," in *Proceedings of the 9th International Conference on Peer-to-Peer Systems*, IPTPS'10, (USA), p. 2, USENIX Association, 2010. CORE Rank: Not available.

[24] S. Ando and A. Nakao, "In-network cache simulations based on a youtube traffic analysis at the edge network," in *Proceedings of The Ninth International Conference on Future Internet Technologies*, CFI '14, (New York, NY, USA), Association for Computing Machinery, 2014. CORE Rank: Not available.

[25] S. Guan, R. E. De Grande, and A. Boukerche, "A cloudlet-based task-centric offloading to enable energy-efficient mobile applications," p. 564–569, IEEE Xplore, 07 2017. CORE Rank: B.

[26] Oracle, "Mysql :: Mysql 8.0 reference manual :: 1.3.1 what is mysql?," 2019.

[27] A. W. Services, "What is amazon relational database service (amazon rds)? - amazon relational database service," 2019.

[28] scikit learn, "scikit-learn: machine learning in python," 2019.

[29] S. Raschka, "mlxtend: Machine learning library extensions," 2014.

[30] G. of Ireland, "Traffic volumes from scats traffic management system jan-jun 2021 dcc - data.gov.ie."