# Geo-Cloudlet: Time and Power Efficient Geospatial Query Resolution using Cloudlet

Jaydeep Das*, Anwesha Mukherjee†, Soumya K. Ghosh‡, Rajkumar Buyya§

*Advanced Technology Development Centre
†‡Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, West Bengal, 721302, India
§Cloud Computing and Distributed Systems (CLOUDS) Lab
School of Computing and Information Systems
The University of Melbourne, VIC 3010, Australia
E-mail: *jaydeep@iitkgp.ac.in, †anweshamukherjee2011@gmail.com, ‡skg@cse.iitkgp.ac.in, §rbuyya@unimelb.edu.au

***Abstract*—Geospatial data analysis is an emerging area of research today due to the potential to enable varied location-aware services. The existing centralized cloud-based analysis becomes time and computing-intensive for huge amount of geospatial data processing. This paper addresses the challenge of time and power-efficiency in QoS-aware geospatial query resolution. We propose a cloudlet based hierarchical paradigm, namely Geo-Cloudlet, where the cloudlets contain the geospatial data of the districts. The state and national level geospatial data are stored inside the state cloud and country cloud respectively. The query resolution is performed by either the cloudlet or by the state cloud or country cloud depending upon the geographical region related to the query. The experimental analysis illustrates that the proposed architecture Geo-Cloudlet reduces the latency up to 61.3% and power consumption up to 61.1% over the use of only remote cloud servers for geospatial query resolution.**

***Index Terms*—Geospatial Query; Geo-Cloudlet; Quality of Service; Time-efficient; Power-efficient**

## I. Introduction

Geospatial data storage and analysis is a major challenge due to its large volume. With the advancement in location acquisition systems, sensor networks, and mobile computing, a huge volume of geospatial data is collected [1]. For resolving geospatial queries, the huge volume of data has to be processed, which is computationally intensive and requires ubiquitous network access. Cloud computing offers a platform for geospatial data processing because of its ability to provide ubiquitous network access, on-demand self-service, resource pooling, rapid elasticity, and measured services [2]–[4]. Geospatial cloud computing is a cloud computing paradigm that is driven by geospatial sciences and optimized by spatio-temporal principles for enabling geospatial science discoveries within a distributed computing environment [1]. However, storing and processing of data completely inside the long distant cloud servers increases latency in query resolution. As a solution towards this problem, fog computing has been used for geospatial data processing in [5]. However, the intermediate devices like switch, routers which act as fog devices, partially offload data or computation. In our work, we will use cloudlets for geospatial data processing. Cloudlet is a computer or cluster of computers, which stores the frequently accessed data and acts as an agent between the client device and cloud [6]. Cloudlet has reduced the energy and latency over remote cloud servers in computation offloading [7]. Our objective is to provide a time and power-efficient paradigm for geospatial query resolution. In this paper, we have introduced a cloudlet based hierarchical architecture for geospatial information storage and processing, and the mathematical model of latency and power consumption for the proposed paradigm is developed. In our paradigm, mobile devices are the clients, which generate geospatial queries.

### A. Related Work

Geospatial information refers to the data with respect to a geographical place, in terms of geographic coordinates. Geospatial data collection takes place by Geographic Information System (GIS) [8]. Using multi-dimensional data set, a method has been proposed for geospatial query resolution in [9]. The use of cloud data centres for geospatial data storage and analysis has been demonstrated in [1], [2], [3]. For massive geospatial data processing, a GIS querying framework has been proposed in [10]. In Geospatial Cloud computing, the application tier is used for geospatial services. There are various categories of geospatial services: Web Map Service (WMS) [11], Web Coverage Service (WCS), Web Feature Service (WFS) [12], Catalog Service for the Web (CSW) and Web Processing Service (WPS) [13]. Open Geospatial Consortium (OGC) constraints geospatial service chain based geospatial query resolution on the cloud has been discussed in [4]. For VM allocation with geospatial service chain learning a method has been discussed in [14]. However, access to long distant cloud servers for geospatial data processing increases the latency. As a solution, fog computing has been used for geospatial data analysis in [5], [15], [16]. Fog devices are the intermediate devices between the user and the cloud servers, which participate in data processing [17]. Edge devices are also nowadays used for offloading computation and storage [18]. Geospatial query processing in edge devices has been discussed in [16]. However, the processing of voluminous geospatial data requires high-end processing, which is beyond the capability of the edge or fog devices. Cloudlet being a computer or cluster of computers, has the ability to store and process a large volume of data [6]. In our work, we use cloudlets and propose a hierarchical paradigm for geospatial query resolution.

### *B. Motivation and Contribution*

Geospatial data are sensitive and of huge amounts. The fog computing based model for different applications such as health care uses fog devices for preliminary data processing. But in geospatial query resolution, a huge amount of geospatial data has to be processed. Hence the use of a traditional fog or cloud-based model is not enough for geospatial application. A distributed hierarchical model with good Quality of Service (QoS) is required for geospatial query resolution. Our objective is to propose a hierarchical paradigm for geospatial data storage and analysis, which will improve the QoS in terms of the power consumption of the user device and latency. The key contributions of this paper are:

- A geospatial data hierarchy is maintained based on country, state, and district. The national level geospatial data is maintained inside the country cloud, the state level geospatial data is maintained inside the state cloud and the district level geospatial data is maintained inside the cloudlets. The proposed cloudlet based architecture for geospatial query resolution is referred as Geo-Cloudlet.
- As instead of keeping all geospatial data inside the cloud servers, geospatial data are maintained inside the cloud and cloudlet in a distributed manner, the load on the cloud data centre is reduced.
- By storing geospatial data in such a hierarchical manner an extra layer of data security is provided. Only country cloud can solve the national level geospatial queries, only state cloud can solve the state level geospatial queries and only cloudlets can solve the district level geospatial queries.
- The latency in geospatial query resolution and power consumption of the user device during that period are calculated for the proposed architecture and experimental analysis for geospatial query resolution using the proposed model is performed. The experimental results demonstrate that the use of cloudlet provides better QoS in terms of power consumption and latency than the remote cloud-based geospatial query resolution system.

The rest of this article is organized as follows. Section 2 discusses the proposed hierarchical paradigm Geo-Cloudlet. Geospatial query analysis has been illustrated in section 3. In section 4, the latency in the proposed paradigm is calculated. Performance analysis has been done in section 5. In the last section, we conclude the paper.

## II. Hierarchical Architecture of Geo-Cloudlet

The cloudlet-based system architecture of Geo-Cloudlet is shown in Fig. 1. In a country like India, there are a number of states, and each state contains a number of districts. In our system, we are considering a hierarchical architecture (Fig. 2), where level 1 is the country i.e. national region. The national region is divided into the state regions in level 2. Each state region is decomposed into the district regions in level 3. Two categories of the geospatial cloud are considered- state cloud and country cloud. National level geospatial data is maintained inside the country cloud, and state level geospatial data is maintained by the state cloud. The district level geospatial data is maintained inside the cloudlets. As instead of keeping all geospatial data inside the cloud servers, geospatial data are maintained inside the cloud and cloudlets in a distributed manner, the load on the cloud data centre is reduced. Only country cloud can resolve the national level geospatial query, only state cloud can resolve the state level geospatial query and only cloudlets can resolve the district level geospatial query. By storing geospatial data in such a hierarchical manner an extra layer of data security is provided.

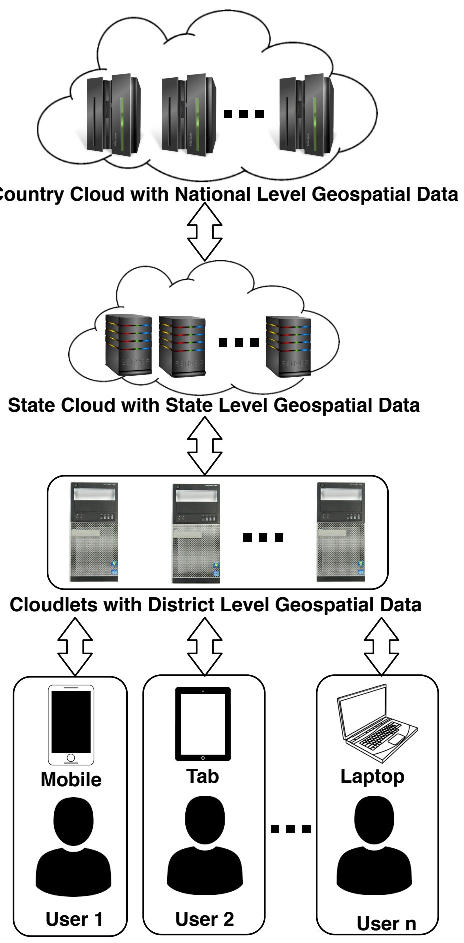

Country Cloud with National Level Geospatial Data
State Cloud with State Level Geospatial Data
Cloudlets with District Level Geospatial Data
Mobile
Tab
Laptop
User 1
User 2
User n

Fig. 1. Cloudlet based architecture for geospatial query resolution

In our system, cloudlet contains the geospatial information of the district and resolves geospatial queries related to the district level geospatial data. Otherwise, if the geospatial query is related to the state level geospatial data, the state cloud resolves the geospatial query. In other respect, if the geospatial query is related to the national level geospatial data,

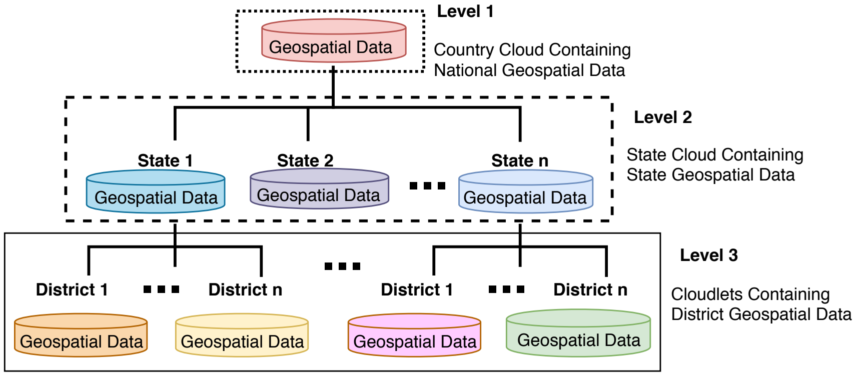


Fig. 2. Geospatial Data Hierarchy for Geo-Cloudlet

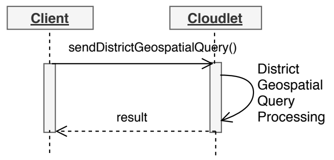

Client
Cloudlet
sendDistrictGeospatialQuery()
District Geospatial Query Processing
result

Fig. 3. Case 1: Sequence diagram of geospatial query resolution by cloudlet

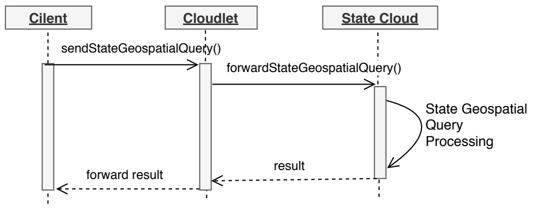

Cilent
Cloudlet
State Cloud
sendStateGeospatialQuery()
forwardStateGeospatialQuery()
State Geospatial Query Processing
result
forward result

Fig. 4. Case 2: Sequence diagram of geospatial query resolution using state cloud

the country cloud resolves the geospatial query. Thus, three following cases appear in our Geo-Cloudlet paradigm:

- **Case 1-** *Cloudlet resolves district level geospatial query:* When a mobile client generates a query related to the geospatial data of district level, the cloudlet with which the mobile device is connected, resolves the geospatial query. The sequence diagram of this type of geospatial query resolution is depicted in Fig. 3.
- **Case 2-** *State cloud resolves state level geospatial query:* When a mobile client generates a geospatial query related to the state level geospatial data, the cloudlet is unable to resolve because it is not containing that geospatial data. In that case, the cloudlet forwards the geospatial query to the state cloud, which after analyzing the state level geospatial data, returns back the result to the cloudlet. The cloudlet forwards the result to the mobile device. Fig. 4 depicts the sequence diagram of this kind of geospatial query resolution.
- **Case 3-** *Country cloud resolves national level geospatial query:* When a mobile client generates a geospatial query related to the national level geospatial data, the cloudlet and state cloud are unable to resolve because they are not containing that geospatial data. In that case, the cloudlet forwards the geospatial query to the state cloud, which further forwards the same to the country cloud. After analyzing the country level geospatial data, the country cloud sends back the result to the state cloud, which forwards the result to the cloudlet. The cloudlet forwards the result to the mobile device. This type of geospatial query resolution is depicted through a sequence diagram in Fig. 5.

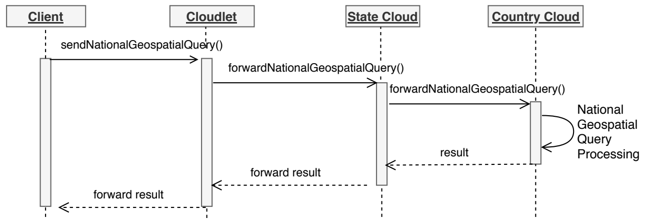

Client
Cloudlet
State Cloud
Country Cloud
sendNationalGeospatialQuery()
forwardNationalGeospatialQuery()
forwardNationalGeospatialQuery()
National Geospatial Query Processing
result
forward result
forward result

Fig. 5. Case 3: Sequence diagram of geospatial query resolution using country cloud

## III. Geospatial Query Resolution

Geospatial data analysis is performed for resolving different types of geospatial queries, generated by the users. In our system, the cloudlet, state cloud, and country cloud perform geospatial data analysis to resolve the queries related to the district, state, and national level geospatial data respectively. In this section, we have discussed different types of geospatial queries along with the geospatial operations [19] to be performed on the geospatial data to resolve them. *Geospatial Query 1 (District level query):* For preparing bridges over the drainage system where the roads are crossed over it in P district.

```
SELECT Rd.Lat, Rd.Long
FROM P_road Rd, P_drain Dr
WHERE Cross(Rd.Shape,Dr.Shape)=1
AND district_name='P';
```

To resolve the above geospatial query, find the latitude (Lat) and longitude (Long) of the crossing point of the road and drainage table. Here topological geospatial operation ‘cross’ is performed over spatial join query on the Road and Drainage table of P district.

*Geospatial Query 2 (State level query):* Find out the One-Way roads and create 2.74 meters buffer to make it Two-Way extension in Q state

```
SELECT road_name
FROM Q_road
WHERE road_type = ‘One_Way’
AND Buffer(area.shape, 2.74)
AND state_name=‘Q’;
```

For resolving this geospatial query, find out the one-way road names from the road table of Q state and perform geospatial analysis operation ‘Buffer’ of fixed length 2.74 meters.

*Geospatial Query 3 (National level query):* List out the towns of Country U where S River supplies water within 150 kilometer distance.

```
SELECT T.Name
FROM Town T, River R
WHERE Overlap(T.Shape,Buffer(R.Shape,
150))=1 AND R.Name =‘S’
AND country_name=‘U’;
```

To resolve this type of geospatial query, find out the town names where S river overlaps towns of country U. Here geospatial analysis operation ‘Buffer’ of fixed length 150 kilometers and topological geospatial operation ‘Overlap’ have been used.

## IV. Latency in Geospatial Query Resolution

The parameters used in latency calculation are defined in Table I. The sum of propagation, communication, processing, and queuing latency is the total latency consumption while resolving the geospatial query for a mobile device.
The propagation latency for case 1 is given as,

$$L_{p_1} = d_{ml}/S_p \tag{1}$$

The communication latency in case 1 is given as,

$$\begin{aligned} L_{com_1} = &(D_{ml}/Up_{ml}) * (1 + fu_{ml}) \\ &+ (D_{lm}/Dw_{ml}) * (1 + fd_{ml}) \end{aligned} \tag{2}$$

The processing latency in case 1 is given as,

$$L_{pr_1} = D_{q1}/S_l \tag{3}$$

The total latency for case 1 is given as,

$$L_{case_1} = L_{p_1} + L_{com_1} + L_{pr_1} \tag{4}$$

The power consumption of the user device (mobile device) during this period is given as,

$$P_{case_1} = L_{p_1} * P_i + L_{com_1} * P_c + L_{pr_1} * P_i \tag{5}$$

The propagation latency for case 2 is given as,

$$L_{p_2} = (d_{ml} + d_{ls})/S_p \tag{6}$$

TABLE I
Parameters used in latency calculation

| Parameter | Definition |
|---|---|
| $d_{ml}$ | Distance between mobile device and cloudlet |
| $d_{ls}$ | Distance between cloudlet and state cloud |
| $d_{sc}$ | Distance between state cloud and country cloud |
| $S_p$ | Propagation speed |
| $D_{q1}$ | Data amount processed for resolving query in case 1 |
| $S_l$ | Data processing speed of cloudlet |
| $D_{q2}$ | Data amount processed for resolving query in case 2 |
| $S_s$ | Data processing speed of state cloud |
| $D_{q3}$ | Data amount processed for resolving query in case 3 |
| $S_c$ | Data processing speed of country cloud |
| $D_{ml}$ | Data amount transmission from mobile device to cloudlet |
| $D_{lm}$ | Data amount transmission from cloudlet to mobile device |
| $D_{ls}$ | Data amount transmission from cloudlet to state cloud |
| $D_{sl}$ | Data amount transmission from state cloud to cloudlet |
| $D_{sc}$ | Data amount transmission from state cloud to country cloud |
| $D_{cs}$ | Data amount transmission from country cloud to state cloud |
| $Up_{ml}$ | Data transmission rate from mobile device to cloudlet |
| $Dw_{ml}$ | Data transmission rate from cloudlet to mobile device |
| $Up_{ls}$ | Data transmission rate from cloudlet to state cloud |
| $Dw_{ls}$ | Data transmission rate from state cloud to cloudlet |
| $Up_{sc}$ | Data transmission rate from state cloud to country cloud |
| $Dw_{sc}$ | Data transmission rate from country cloud to state cloud |
| $fu_{ml}$ | Uplink failure rate between mobile device and cloudlet |
| $fd_{ml}$ | Downlink failure rate between mobile device and cloudlet |
| $fu_{ls}$ | Uplink failure rate between cloudlet and state cloud |
| $fd_{ls}$ | Downlink failure rate between cloudlet and state cloud |
| $fu_{sc}$ | Uplink failure rate between state cloud and country cloud |
| $fd_{sc}$ | Downlink failure rate between state cloud and country cloud |
| $D_{w1}$ | Queuing latency for case 1 |
| $D_{w2}$ | Queuing latency for case 2 |
| $D_{w3}$ | Queuing latency for case 3 |
| $P_i$ | Power consumption of mobile device in idle mode |
| $P_c$ | Power consumption of mobile device in active mode |

The communication latency in case 2 is given as,

$$\begin{aligned} L_{com_2} = L_{com_1} &+ (D_{ls}/Up_{ls}) * (1 + fu_{ls}) \\ &+ (D_{sl}/Dw_{ls}) * (1 + fd_{ls}) \end{aligned} \tag{7}$$

The processing latency in case 2 is given as,

$$L_{pr_2} = D_{q2}/S_s \tag{8}$$

The total latency for case 2 is given as,

$$L_{case_2} = L_{p_2} + L_{com_2} + L_{pr_2} \tag{9}$$

The power consumption of the user device (mobile device) during this period is given as,

$$P_{case_2} = L_{p_2} * P_i + L_{com_2} * P_c + L_{pr_2} * P_i \tag{10}$$

The propagation latency for case 3 is given as,

$$L_{p_3} = (d_{ml} + d_{ls} + d_{sc})/S_p \tag{11}$$

The communication latency in case 3 is given as,

$$\begin{aligned} L_{com_3} = L_{com_2} &+ (D_{sc}/Up_{sc}) * (1 + fu_{sc}) \\ &+ (D_{cs}/Dw_{sc}) * (1 + fd_{sc}) \end{aligned} \tag{12}$$

The processing latency in case 3 is given as,

$$L_{pr_3} = D_{q3}/S_c \tag{13}$$

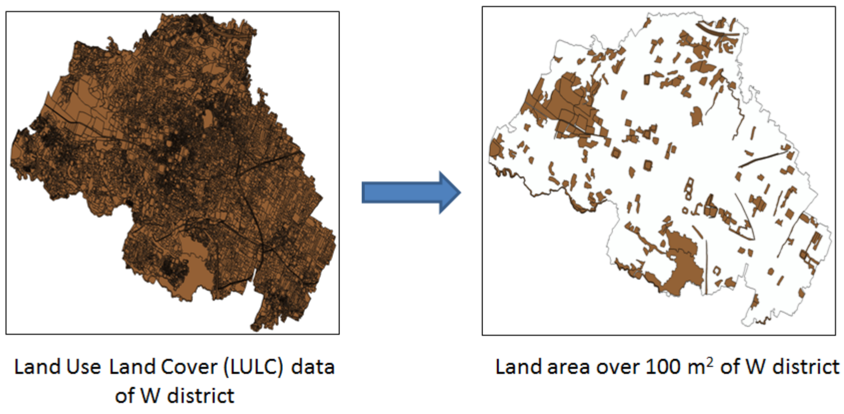

Land Use Land Cover (LULC) data of W district
Land area over 100 m² of W district

Fig. 6. Result of Geospatial Query 1

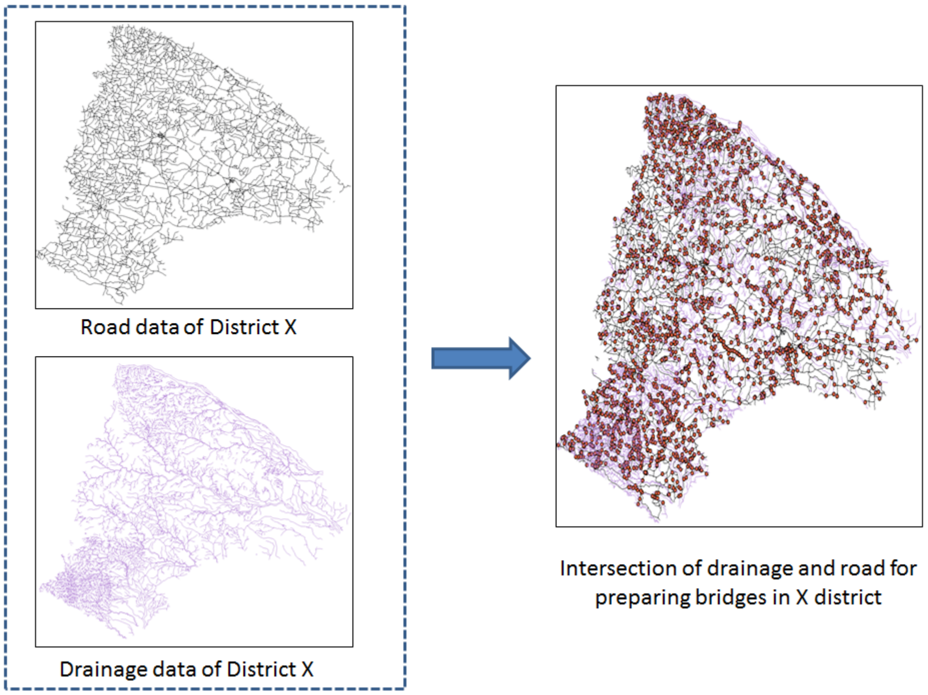

Road data of District X
Drainage data of District X
Intersection of drainage and road for preparing bridges in X district

Fig. 7. Result of Geospatial Query 2

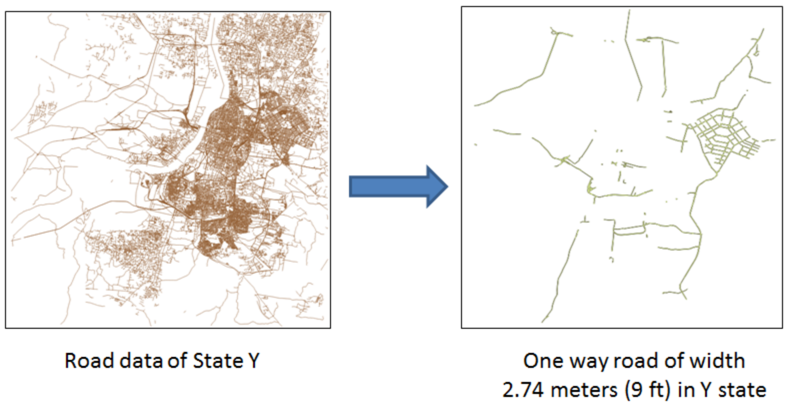


Fig. 8. Result of Geospatial Query 3

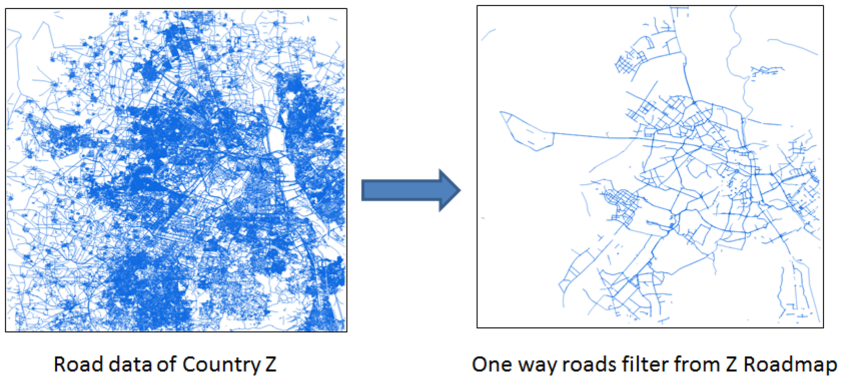


Fig. 9. Result of Geospatial Query 4

The total latency for case 3 is given as,

$$L_{case_3} = L_{p_3} + L_{com_3} + L_{pr_3} \tag{14}$$

The power consumption of the user device (mobile device) during this period is given as,

$$P_{case_3} = L_{p_3} * P_i + L_{com_3} * P_c + L_{pr_3} * P_i \tag{15}$$

In the next section, the performance of the proposed system will be evaluated through experimental analysis, and the latency and user device's power consumption during query resolution will be determined.

## V. Performance Evaluation

We have performed the experimental analysis using the private cloud of the institution, Google Cloud Platform (GCP), mobile devices and cloudlet. Geospatial data are of two types: vector data and raster data. In this experiment, we have used vector data. We have used the geospatial reference system, EPSG:32645 and the geospatial data related to the road, rail, and land area of W and X districts of Y state of country Z.

### A. *Devices used in experiment*

Two mobile phones are used in our experiment, which generate geospatial queries. We have used a laptop as a cloudlet, which is connected with mobile devices through the hotspot. The cloudlet is storing the geospatial data of the districts W and X. We have used the private OpenStack cloud of our institute, where a Virtual Machine (VM) instance is taken. In our experiment, the private cloud is used to analyse state level data. We have used Google Cloud Platform (GCP) where we have created one VM instance. In our experiment, the GCP is used to analyse country level data. The laptop which is used as a cloudlet is connected with the network through a Wi-Fi access point (AP). The AP is connected with Access Switch (AS), which is connected with Distribution Switch (DS). DS is connected with the Core Switch (CS), which is connected with the router. The used protocols are TCP, UDP, and ICMP. The data transmission rate is 40 Mbps. The configurations of the mobile phones, laptop (used as cloudlet), and VM instances are presented in Table II. QGIS (version 2.18.28) [20] is used for processing and analyzing geospatial queries.

### B. *Experimental results of query resolution*

Four geospatial queries are generated from the mobile devices related to districts W, X, state Y, and country Z:

- One query is related to district W.
- One query is related to district X.
- One query is related to state Y.
- One query is related to country Z.

The laptop (cloudlet) is holding the geospatial data related to districts W and X. Therefore the laptop resolves the first two geospatial queries using QGIS and sends the result to the mobile devices. The private OpenStack cloud VM is containing the data of state Y. Hence for the third query, the private cloud VM is accessed. The GCP VM is containing the data of country Z. Thus for the fourth query, the VM of GCP is accessed. The results of geospatial query resolution after geospatial data analysis are presented in Fig. 6, Fig. 7, Fig. 8, and Fig. 9. Total latency in geospatial query resolution considering the propagation, communication, and processing (as discussed in section IV), are presented in Table III along with the power consumption of the mobile device during that period. The latency and power consumption are measured in seconds (sec) and milli-Watt (mW) respectively.

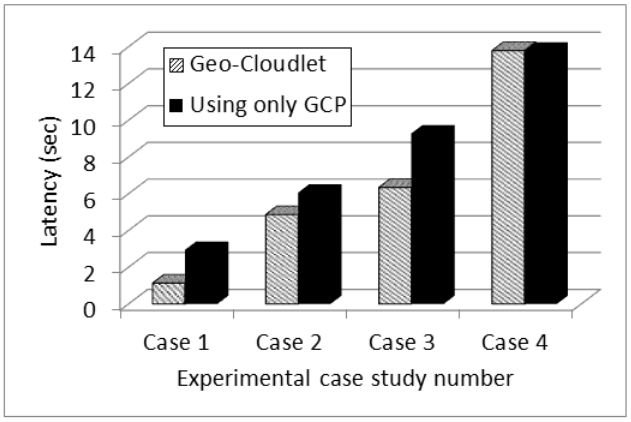
In Fig. 6, the result of geospatial query 1 is displayed. W district land use land cover (LULC) map image is considered. The user of mobile phone 1 queries for the land area over $100m^2$ in the W district. The geospatial data of W district is analyzed, and the 'filter' operation is used. The cloudlet (laptop) resolves the query and sends the result to the mobile phone 1. The latency for resolving this query and the power consumption of the mobile device during this period are presented in Table III and compared with the results if GCP VM resolves the query and sends the result to the mobile device. This is observed from case study 1 that using our system the latency and mobile device's power consumption has been reduced to 61.3% and 61.1% respectively.

TABLE II
CONFIGURATIONS OF DEVICES USED IN EXPERIMENT

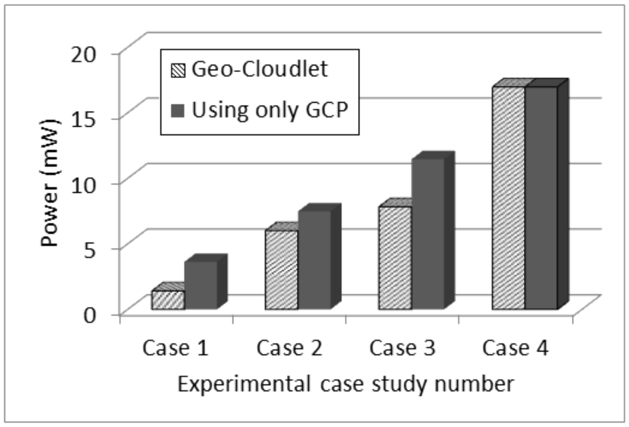
| Device | RAM | HDD | Processor |
|---|---|---|---|
| Mobile phone 1 | 3 GB | 32 GB | Qualcomm MSM8940 Octa Core |
| Mobile phone 2 | 2 GB | 8 GB | 64-bit 1.2 GHz Qualcomm Snapdragon 410 quad-core processor |
| Cloudlet | 4 GB | 250 GB | Intel Core i5 |
| GCP VM instance | 3.75 GB | 250 GB | 1 vCPU |
| OpenStack VM | 4 GB | 45 GB | 2 vCPU |

TABLE III
LATENCY AND POWER CONSUMPTION IN QUERY RESOLUTION

| Case Study | Geospatial Query | Transmitted data between consecutive nodes (MB) | Latency in Geo-Cloudlet (sec) | Latency in GCP VM (sec) | Power consumption in Geo-Cloudlet (mW) | Power consumption in GCP VM (mW) | Reduction in latency in Geo-Cloudlet than GCP | Reduction in power in Geo-Cloudlet than GCP |
|---|---|---|---|---|---|---|---|---|
| 1 | SELECT area_name FROM District_W WHERE $area > 100$; | 1.1 | 1.14 | 2.9464 | 1.41 | 3.63 | 61.3% | 61.1%. |
| 2 | SELECT Rd.Lat, Rd.Long FROM X.road Rd, X.drain Dr WHERE Cross(Rd.Shape,Dr.Shape)=1; | 1.6 | 4.87 | 6.068 | 6.004 | 7.48 | 19.74% | 19.78% |
| 3 | SELECT road_name FROM State_Y WHERE road_type = 'One_Way' and Buffer (area.shape, 2.74); | 3.75 | 6.35 | 9.286 | 7.83 | 11.45 | 31.61% | 31.6% |
| 4 | SELECT road_name FROM Country_Z WHERE road_type = 'One_Way'; | 5.7 | 13.8 | 13.8 | 16.97 | 16.97 | Nil | Nil |



(a) Latency (sec) in query resolution



(b) Power consumption of user device during query resolution period

Fig. 10. Latency and power consumption during query resolution using proposed Geo-Cloudlet and using only GCP

In Fig. 7, the result of geospatial query 2 is displayed. The user of mobile phone 2 queries for the crossing points of drainage and roads for preparing bridges in X district. The geospatial data of X district is analyzed, and 'cross' operation is used. The cloudlet (laptop) after resolving the geospatial query sends the result to the mobile phone 2. The latency for resolving this query and the power consumption of the mobile device during this period are presented in Table III and compared with the results if GCP VM resolves the query and sends the result to the mobile device. The experimental results of case study 2 show that using our system the latency and mobile device's power consumption has been reduced to 19.74% and 19.78% respectively.

In Fig. 8, the result of geospatial query 3 is displayed. The user of mobile phone 1 queries for one-way streets of Y state of width 2.74 meters (9 ft), to double the road width to make it two-way lane. For this purpose, the Y state geospatial data is analyzed, and a buffer of 2.74 meters over the one-way road is created. After resolving this geospatial query using private OpenStack cloud VM, the result is returned back to the mobile phone 1. The latency for resolving this query and the power consumption of the mobile device during this period are presented in Table III and compared with the results if GCP VM resolves the query and sends the result to the mobile device. From the experimental results of case study 3, it is observed that using our system the latency and mobile device's power consumption has been reduced to 31.61% and 31.6% respectively.

In Fig. 9, the result of geospatial query 4 is displayed. The user of mobile phone 2 queries for one-way roads from Z country road map. The geospatial data of country Z is analyzed, and 'filter' operation is used. After resolving the geospatial query using GCP VM, the result is sent back to the mobile phone 1. The latency for resolving this query and the power consumption of the mobile device during this period are presented in Table III. As GCP VM is used for resolving query 4, the latency and power consumption of the mobile device is the same for the proposed model and if GCP resolves the query.

In Fig.10 the latency in query resolution and power consumption of user devices during that period are presented with respect to four experimental studies. From the experimental results of the four case studies of query analysis (see Table III and Fig.10), we observe that using the cloudlet up to 61.3% reduction in latency and 61.1% reduction in power consumption of the mobile device can be achieved than only remote cloud-based system, and the QoS is improved.

## VI. Conclusions and Future Work

In this paper, a cloudlet based hierarchical paradigm for geospatial query resolution, namely Geo-Cloudlet, has been proposed. The cloudlets contain geospatial data of the district regions. For the state and national level geospatial data storage and analysis, state cloud and country cloud are used respectively. When a geospatial query is received from a mobile device regarding the district region, the cloudlet resolves the geospatial query after analyzing the geospatial data and responds to the mobile device. Otherwise, if the geospatial query is regarding the state or national level geospatial data, the cloudlet responds using the state cloud or the country cloud. The experimental results illustrate that our proposed system improves the QoS by reducing the latency up to 61.3% and power consumption of the user device up to 61.1% than only remote cloud-based query resolution. Thus we can conclude that the proposed framework, Geo-Cloudlet, is a time-efficient paradigm as well as provides low power consumption of the user device during the query resolution period. As we are using a multi-tier framework, partial data processing and dealing with the inter-dependency among different task segments during geospatial data processing is a challenging future scope.

## References

[1] C. Yang, M. Yu, F. Hu, Y. Jiang, and Y. Li, "Utilizing cloud computing to address big geospatial data challenges," *Computers, Environment and Urban Systems*, vol. 61, pp. 120–128, 2017.

[2] C. Yang, M. Goodchild, Q. Huang, D. Nebert, R. Raskin, Y. Xu, M. Bambacus, and D. Fay, "Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing?" *International Journal of Digital Earth*, vol. 4, no. 4, pp. 305–329, 2011.

[3] K. Evangelidis, K. Ntouros, S. Makridis, and C. Papatheodorou, "Geospatial services in the cloud," *Computers & Geosciences*, vol. 63, pp. 116–122, 2014.

[4] J. Das, A. Dasgupta, S. K. Ghosh, and R. Buyya, "A geospatial orchestration framework on cloud for processing user queries," in *Proceedings of the 2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. IEEE, 2016, pp. 1–8.

[5] R. Barik, H. Dubey, S. Sasane, C. Misra, N. Constant, and K. Mankodiya, "Fog2fog: augmenting scalability in fog computing for health GIS systems," in *Proceedings of the Second IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies*. IEEE Press, 2017, pp. 241–242.

[6] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, 2009.

[7] A. Mukherjee, D. De, and D. G. Roy, "A power and latency aware cloudlet selection strategy for multi-cloudlet environment," *IEEE Transactions on Cloud Computing*, 2016.

[8] C.-R. Shyu, M. Klaric, G. J. Scott, A. S. Barb, C. H. Davis, and K. Palaniappan, "GeoIRIS: Geospatial information retrieval and indexing system—content mining, semantics modeling, and complex queries," *IEEE Transactions on geoscience and remote sensing*, vol. 45, no. 4, pp. 839–852, 2007.

[9] M. Malensek, S. Pallickara, and S. Pallickara, "Fast, ad hoc query evaluations over multidimensional geospatial datasets," *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 28–42, 2017.

[10] K. M. Al Naami, S. Seker, and L. Khan, "GISQF: An efficient spatial query processing system," in *Proceedings of the 2014 IEEE International Conference on Cloud Computing*. IEEE, 2014, pp. 681–688.

[11] J. Wenjue, C. Yumin, and G. Jianya, "Implementation of OGC web map service based on web service," *Geo-spatial information Science*, vol. 7, no. 2, pp. 148–152, 2004.

[12] P. A. Vretanos, "Web feature service implementation specification," *Open Geospatial Consortium Specification*, vol. 1325, pp. 04–094, 2005.

[13] P. Schut and A. Whiteside, "Openggis web processing service, version 1.0. 0, ogc 05-007r7, open geospatial consortium," *Inc*, vol. 87, 2007.

[14] J. Das, A. Dasgupta, S. K. Ghosh, and R. Buyya, "A learning technique for vm allocation to resolve geospatial queries," in *Proceedings of the 2017 International Conference on Advanced Computing, Networking and Informatics (ICACNI)*. Springer, 2019, vol. 1, pp. 577–584.

[15] R. K. Barik, H. Dubey, A. B. Samaddar, R. D. Gupta, and P. K. Ray, "FogGIS: Fog computing for geospatial big data analytics," in *Proceedings of the 2016 IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON)*. IEEE, 2016, pp. 613–618.

[16] R. K. Barik, A. Tripathi, H. Dubey, R. K. Lenka, T. Pratik, S. Sharma, K. Mankodiya, V. Kumar, and H. Das, "Mistgis: Optimizing geospatial data analysis using mist computing," in *Progress in Computing, Analytics and Networking*. Springer, 2018, pp. 733–742.

[17] H.-J. Hong, "From cloud computing to fog computing: unleash the power of edge and end devices," in *Proceedings of the 2017 IEEE international conference on cloud computing technology and science (CloudCom)*. IEEE, 2017, pp. 331–334.

[18] R. Sosa, C. Kiraly, and J. D. P. Rodriguez, "Offloading execution from edge to cloud: a dynamic node-red based approach," in *Proceedings of the 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2018, pp. 149–152.

[19] S. Shekhar and S. Chawla, *Spatial databases: a tour*. prentice hall Upper Saddle River, NJ, 2003, vol. 2003.

[20] Q. D. Team *et al.*, "QGIS geographic information system," *Open Source Geospatial Foundation Project. Disponível em: http://www. qgis. org/. Acesso em*, vol. 27, 2015.