ENERGY AND TIMELINESS-AWARE GEOSPATIAL QUERY PROCESSING IN INTEGRATED CLOUD-FOG-EDGE COMPUTING ENVIRONMENTS

Jaydeep Das

Energy and Timeliness-Aware Geospatial Query Processing in Integrated Cloud-Fog-Edge Computing Environments

Thesis submitted for the Award of the Degree

of

Doctor of Philosophy

by

Jaydeep Das

Under the guidance of

Prof. Soumya Kanti Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, India
and
Prof. Rajkumar Buyya
School of Computing and Information Systems
The University of Melbourne, Australia

ADVANCED TECHNOLOGY DEVELOPMENT CENTRE
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
OCTOBER 2021

Dedicated to -

My Respected **Grandfathers** (*Late Sasadhar Das* and *Late Kalipada Dutta*)
and **Grandmothers** (*Late Subodhbala Das* and *Late Shishubala Dutta*)

APPROVAL OF THE VIVA-VOCE BOARD

Date: 26 / 10 / 2021

Certified that the thesis entitled **Energy and Timeliness-Aware Geospatial Query Processing in Integrated Cloud-Fog-Edge Computing Environments** submitted by **JAYDEEP DAS** to the Indian Institute of Technology Kharagpur, for the award of the degree Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

Prof. Indranil Sengupta
(Member of DSC)

Prof. Shamik Sural
(Member of DSC)

Prof. Santanu Chattopadhyay
(Member of DSC)

Prof. Soumya Kanti Ghosh
(Supervisor)

Prof. Rajkumar Buyya
(Supervisor)

Prof. Phalguni Gupta
(External Examiner)

Prof. Tarun Kanti Bhattacharyya
(Chairman)

CERTIFICATE

This is to certify that the thesis entitled "**Energy and Timeliness-Aware Geospatial Query Processing in Integrated Cloud-Fog-Edge Computing Environments**" submitted by **Jaydeep Das** to Indian Institute of Technology Kharagpur, is a record of bonafide research work under our supervision and we consider it worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

Dr. Soumya Kanti Ghosh
Professor
Department of Computer Science and Engineering
Indian Institute of Technology
Kharagpur 721302, India

Dr. Rajkumar Buyya
Professor
School of Computing and Information Systems
The University of Melbourne
VIC 3010, Australia

Date: 26 / 10 / 2021

DECLARATION

I certify that

a. The work contained in the thesis is original and has been done by myself under the general supervision of my supervisors.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in writing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

_____

Jaydeep Das

ACKNOWLEDGEMENTS

I am delighted to write this part of my dissertation. This is immense blessings of almighty on me that I am able to reach the final destination of my Ph.D.

I would like to express my deepest gratitude and thanks my supervisors, Prof. Soumya Kanti Ghosh and Prof. Rajkumar Buyya, for giving me the opportunity to undertake this Ph.D. Their continuous support, guidance and motivation throughout my Ph.D. tenure, makes this thesis materialized. I feel privileged to get this opportunity to work with them.

I would also like to express my gratitude to the chairman of my doctoral scrutiny committee (DSC), Prof. Tarun Kanti Bhattacharyya, and all DSC members, Prof. Indranil Sengupta, Prof. Shamik Sural, and Prof. Santanu Chattopadhyay, for their valuable comments and suggestions to enrich my research thoughts.

I would like to give special thanks to Dr. Arindam Dasgupta, Dr. Anwesha Mukherjee, and Dr. Sourav Kanti Addya for giving their valuable advice and continuous support in various part of my research. I have learnt many things from them. I would also like to thank all the past and current members of the GIS and SDS Laboratory, at IIT Kharagpur. In particular I thank Chandan Misra, Shreya Ghosh, Shubha Brata Nath, Arpita Chaudhuri, Manoj Kumar Sharma for their support and encouragement during my Ph.D. journey. The source of ideas and energy to have friends like Rajesh, Debopriyo, Saptarshi, Sreeja, Anushri, Tauheed, Shabnam, Tapabrata, Amartya, Saikat, Utpal, Satendra, Ayan, Tamoghna, Ujjal, and Utpal Da for making the stay at Kharagpur ever memorable.

I am grateful to my parents, Dr. Nitya Ranjan Das and Mrs. Alaka Das, for their endless love, appreciations, sacrifice and support. I must specially acknowledge my beloved brother Dr. Pradyot Das, and sister-in-law Mrs. Pubali Das for constantly keeping me motivated through all their encouraging phone calls. My two nieces, Aaishani and Aaishiki, always keep me charged with their childish and bustling nature. Last but not least, I would like to thank all members of my family for their constant encouragement, and understandings.

Jaydeep Das
IIT Kharagpur, October 2021

ABSTRACT

In recent times, there is a growing need to analyse Spatio-temporal datasets to extract meaningful information and provide location-aware services, such as trip planning, weather forecasting, and even health management. From its inception, Spatio-temporal data mining has shown a significant impact on varied aspects of our lives. However, analysing a huge volume of Spatio-temporal datasets is challenging since it requires enormous computing and storage power and several spatial operations to perform analytics efficiently. In this regard, Cloud computing is undoubtedly the most feasible solution as it provides unlimited computing resources and data storage facilities. However, frequent communications with distant cloud servers increase the delay and may affect the Quality of Service (QoS) of any framework. Here, the fog or edge nodes can be made intelligent enough to analyse and adapt timely measures to reduce the intervention of cloud servers at each time. While fog or edge computing is not a replacement for cloud computing, the magnificent integration of these two booming technologies can efficiently facilitate delay, energy awareness, and real-time applications.

To be specific, our research focuses on exploring the spatial cloud computing domain to facilitate several real-life applications in less delay and energy consumption. The most critical aspect of facilitating any real-life application by analysing a large volume of data is to develop an efficient query processing module. The demand for computing resources to process the geospatial queries has been increased drastically. The query helps the users to get a variety of information to serve their needs. A huge number of heterogeneous data sources and different computing services are involved in resolving the geospatial queries. Extracting appropriate results within a specific time bound and orchestration among those data sources and web services are essential. These services are available on the web and require different resource specifications in order to resolve a geospatial query.

The major contributions of this thesis are (1) Development of a taxonomy for geospatial cloud-fog-edge computing environments. (2) Resolution of geospatial queries in the cloud with heterogeneous data sources. (3) Various geospatial queries resolve in cloud platform after generating geospatial service chaining. Preestimating the cloud resources helps provide resources to geospatial queries within

user-defined budget and time deadline using a game theory-based approach. (4) Geospatial query resolution within region-specific fog devices. It is an energy-efficient and delay-aware geospatial query resolution framework. (5) Finally, a real-time healthcare service provisioning with geospatial queries in a cloud-fog-edge integrated platform. It is an energy and latency-aware framework, leading to a green geospatial query resolution platform. We have also performed an extensive comparative study with the benchmark and state-of-the-art spatial cloud computing systems, demonstrating our proposed spatial cloud-based methods' efficacy and superiority. The overall study comprises collection, orchestration, analysis, and visualization of Spatio-temporal data sources, such as road-networks, land-use information, location-based movement patterns, and user health profiles at different spatial locations at various time-scales.

Keywords: Spatial Cloud Computing, Geospatial Query, Geospatial Services, Fog Computing, Edge Computing.

Contents

List of Figures

List of Tables

Glossary

ANN	: All Nearest Neighbour
AR	: Augmented Reality
ATM	: Automated Teller Machine
CCTV	: Closed-circuit Television
CloudSim	: Cloud Simulator
CO	: Carbon Monoxide
CPU	: Central Processing Unit
CSW	: Catalog Service for the Web
CQI	: Concurrent Query Intensity
DDoS	: Distributed Denial-of-Service
EFC	: Edge and Fog Computing
EGIS	: Enterprise Geographical Information Systems
EPSG	: European Petroleum Survey Group
GCP	: Google Cloud Platform
GI	: Geographical Information
GIS	: Geographical Information Systems
GML	: Geography Markup Language
GQ	: Geospatial Query
GRASS	: Geographic Resources Analysis Support System
GSI	: Geological Survey of India
IaaS	: Infrastructure as a Service
IDW	: Inverse Distance Weighting
IMD	: Indian Meteorological Department
I/O	: Input/Output
IoHT	: Internet of Health Things
IoT	: Internet of Things
ISRO	: Indian Space Research Organisation
JPEG	: Joint Photographic Experts Group
LiDAR	: Light Detection and Ranging
LSTM	: Long Short Term Memory
LULC	: Land Use Land Cover
LYRIC	: deadLine and budget aware spatio-temporal querY pRocessing In Cloud
MMS	: Mobile Mapping System

NRDF : National Disaster Response Force
NFV : Network Function Virtualization
NN : Nearest Neighbour
OE : Orchestration Engine
OGC : Open Geospatial Consortium
OWS : OGC Web Services
PaaS : Platform as a Service
PCA : Principal Component Analysis
PNG : Portable Network Graphics
POI : Point Of Interest
QGIS : Quantum Geographical Information System
QoS : Quality of Service
QP : Query Plan
QS : Query Sensitivity
RAM : Random Access Memory
RF : Random Forest
RNNQ : Reverse Nearest Neighbour Query
RSU : Road Side Unit
SaaS : Software as a Service
SCC : Spatial Cloud Computing
SDN : Software Defined Network
SLA : Service Level Agreement
SOA : Service Oriented Architecture
SOI : Survey of India
SQL : Structured Query Language
TCO : Telephone Central Offices
TIFF : Tagged Image File Format
UML : Unified Modeling Language
VCPU : Virtual Central Processing Unit
VM : Virtual Machine
WCS : Web Coverage Service
WFS : Web Feature Service
WMS : Web Map Service
WPS : Web Processing Service
WRIS : Water Resources Information System
WSN : Wireless Sensor Network
XML : eXtensible Markup Language

Chapter 1

Introduction

The advances in various location acquisition systems, sensor networks, and mobile computing methods have generated substantial volume of geospatial data. Various applications, namely, navigation systems, traffic analysis, sensing, disaster management, etc., have been proposed with this myriad of geospatial data. To resolve the geospatial query, a huge amount of geospatial data need to be accessed and processed. The processing of such data is often computationally intensive and requires incessant access. Cloud computing provides ubiquitous network access, on-demand self-service, resource pooling, rapid elasticity, and measured services [1]. The rapid advances of computer hardware, software, and fast network speed have fostered enough opportunities in research in cloud-based geographical information systems (GIS).

Spatial Cloud Computing (SCC) [2] refers to the cloud computing paradigm driven by geospatial sciences (see figure 1.2). It also employs Spatio-temporal principles for enabling geospatial science discoveries and cloud computing within a distributed computing environment. Spatio-temporal principles [3] are essential for their abilities to enable the discoverability, accessibility, and usability of the distributed, heterogeneous, and massive data. SCC optimizes cloud computing infrastructure by helping arrange, select, and utilize high-end computing for computing-intensive problems, along with enabling timely response to world-wide distributed and locally clustered users. SCC also enables the timely response to world-wide distributed and locally clustered users through geospatial optimization. SCC assists the design of Spatio-temporal data structure and algorithms to optimize the information workflow for solving complex problems. Cloud orchestration is required while dealing with services and operational concerns such as servicing large numbers of simultaneous user queries, enforcing policies that reflect on services and engineering rules,

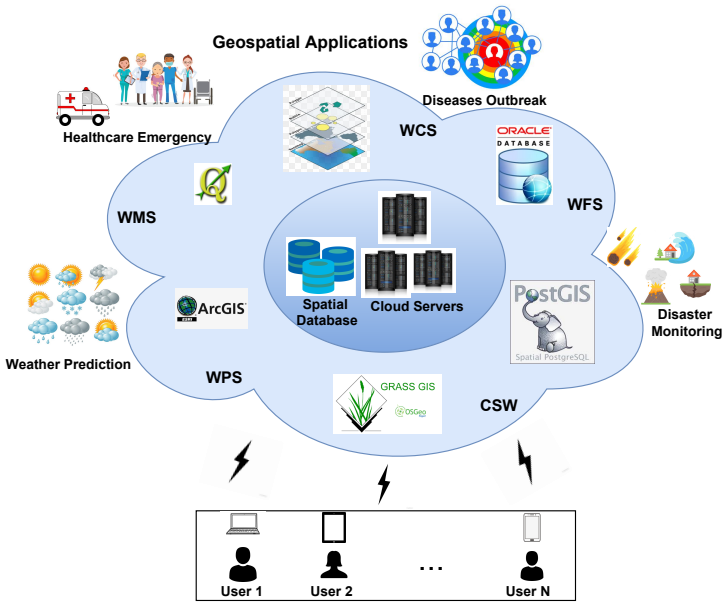


Figure 1.1: A typical Spatial Cloud Computing environment

and performing fault and error handling in a highly dynamic environment.
Beyond geospatial services, the key success of these technologies is the interoperability because "no single organization produces all the data (so it's inconsistent) and no single vendor provides all the systems" (OGC, 2008a) [4]. So, the different types of geospatial services are naturally published via different interfaces. For achieving the geo-processing solution, it requires the mediate component to integrate those heterogeneous services. Orchestration Engine(OE) plays such intermediate component role. It selects and coordinates required intelligent geospatial services [5] applying business logic. Using catalog services OE architecture describes the way of choosing required data services. It also synchronises web services as per requirement. For examples, Web Feature Service (WFS) itself cannot coordinate with Web Map service (WMS), Web Coverage Service (WCS) itself also cannot coordinate to WFS; therefore, orchestration engine, aggregate service, or workflow managed service is necessary. Geospatial cloud as a system-based-on cloud computing has several advantages over traditional GIS [9] , namely, easy setup, low cost, easy publishing, modular component, interoperable collaboration. With traditional GIS, collaborating among stakeholders located at various places in the world is a persistent problem. With the advent of spatial cloud, this problem can be addressed, because multiple stakeholders at different locations, can create accounts on the geospatial cloud and collaborate among one another. Service Oriented Architecture (SOA) helps seamless integration of distributed data sources. Enterprise Geographical Information Systems (EGIS)

framework for sharing and utilizing the heterogeneous data sources. Each stakeholder can access the data which is centralized, make modifications, and build any project collaboratively.
Moreover, storing and processing a massive amount of data on cloud data centers results in high energy consumption as well as the use of long distant cloud servers compromise with Quality of Service (QoS) in terms of delay and energy, from the perspective of the client mobile devices [6, 7]. In order to improve the QoS, fog computing has been introduced where the intermediate devices like a switch, or a router are used to perform data and computation offloading [8] in between the end node and cloud servers. The intermediate and end devices that monitor, store and process data are called fog and edge devices [9, 10] respectively. The use of fog and edge devices in geospatial data storage and processing can reduce the delay and energy consumption over remote cloud servers [11]. Various fog-edge based real-life applications are increasing in healthcare [12], agriculture [13], environment monitoring [14], disaster monitoring [15] etc.
The brief descriptions of the few terms/ topics, used in this thesis, are presented here.

- **Geospatial Query:** These queries return all documents whose locations are within the query-specified area. To specify holes within the area, so that one or more subsets of returned documents can be omitted from the final results, boolean queries should be applied to the set of documents returned by the geospatial query. The types of Geospatial Queries (GQs) [16] are mentioned below.
  Filter Query- This type of query[17][18] filters a particular geometry (say, geometry1) which presents in the another geometry (say, geometry2). The format of filter query is as follow-
  SDO_FILTER (*geometry*1, *geometry*2, *parameters*)
  Primary and Secondary Filter- This type of query filters one subset geometry (say, geometry1) or an object from the superset geometry (say, geometry2). For this kind of operations, *SDO_FILTER* and *SDO_RELATE* operators are used. The format of filter query is as follow-
  SDO_FILTER (*geometry*1, *geometry*2, *parameters*)
  SDO_RELATE (*geometry*1, *geometry*2, *parameters*)
  Within Distance- It measures whether one geometry or object (say, aGeom) is present within a particular euclidean distance of another geometry or not. The format of within distance query is as follow-
  SDO_WITHIN_DISTANCE (*T.column*, *aGeom*, *parameters*)

To proceed the within distance query, we can create a buffer of the radius of specified distance (say, d). It is also called Buffer query [19].
Nearest Neighbour (NN)- It measures whether geometries(say, geometry2) is the nearest neighbor of a particular geometry (say, geometry1) or not. The format of nearest neighbour query is as follow-
SDO_NN (*geometry*1, *geometry*2, *parameter*)
Reverse Nearest Neighbour Query (RNNQ) [20] and All-Nearest Neighbour (ANN) [21] [22] query are the two variants of the NN query.
Geospatial Join Query- This type of join query [23][24] is the same as the relational join of queries, but the predicates are attached with geospatial operators. It compares one layer of a geometry (say, geometry1) with the layers of the other geometries (say, geometry2). Geospatial index type (that is, R-tree or Quadtree) must be the same on the geometry column of all the tables involved in the join operation.
Example: How many states are crossed by a river?

```
SELECT R.GId, S.GId
FROM rivers R, states S
WHERE sdo_filter(R.shape, S.shape,'querytype = join') = 'TRUE';
```

- **Geospatial Services:** Geospatial web services, used in this thesis, are complaint with Open Geospatial Consortium (OGC) standards[1]. Geospatial web services emphasis on three main purposes - data discovery, data visualization and data access. Some of the OGC complaint geospatial services are as follows-
  Web Features Services (WFS)[25] is interfaces for defining data handling operations like create, update, delete a geographic feature instance. There are different operations, i.e., GetFeature, GetCapabilities, GetPropertyValue available on WFS.
  Web Processing Service (WPS)[26] provides a platform with different geo-processing operations like buffering, intersection, overlaying on a point, polyline or polygon. It could access across the network to utilize a preprogrammed computation model that operates on spatially referenced data.
  Web Map Service (WMS) returns one or more geo-registered map images from distributed geospatial databases in JPEG, PNG, TIFF etc. format on the response of WMS request.
  Catalog Services for the Web (CSW) is needed to publish the geospatial characteristics and search group of metadata for data, services, and related information

[1] https://www.ogc.org

objects provided by different sources. So, the client system can automatically bind with the required geospatial services. CSW interface helps a client to query on catalogs to discover resources. A CSW has different requirements for its three main types of users - Resource User, Resource Provider and Registry Manager.

- **Computing Paradigms:**
  Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1].
  Fog Computing extends the traditional cloud computing and services to the edge of network. It provides the computation, communication, controlling, storage and services capabilities at the edge of network. The decentralized platform is different from other conventional computational model in architecture [27].
  Edge Computing also extends cloud service to the edge devices. It refers to the enabling technologies which allow computation and storage to be performed on edge devices [28]. That is to say, computing and storing happens near things and data sources [29]. The edge nodes and devices with computing capacity perform a large number of computing tasks (e.g., data processing, temporarily storing, devices management, decision making, and privacy protection) to reduce the network latency and traffic between end devices and cloud [30]. These edge nodes can be composed of smart sensors, smart phones, and smart vehicles, even a special edge servers. They can interconnect and intercommunicate in the local to form an edge network. Moreover, edge devices connect with cloud data center by core network. Edge computing provides edge intelligence services nearby to meet the critical demands of the digital industry in agile connection, real-time services, data optimization, application intelligence, security and privacy protection.
  Edge computing and fog computing, their architectures are hierarchical, decentralized, and distributed, which is different from centralized cloud computing architecture. Their service locations are the proximity to end users. Edge computing is located in edge devices, while fog computing is located in network edge devices, which is single network hop or few network hops away from the edge. Their resources (e.g., computing, communication and storage resources) and computation and storage capabilities are limited by comparing with cloud

computing, and edge computing is more limited than fog computing. Because the resources and service capabilities of network edge devices are relatively stronger than edge devices. Moreover, these two computing paradigm have mobility support for end users. Because most services are provided locally, it is essential to take the existence of mobile devices into consideration. They also support the scalability of the whole ecosystem. The reason is that a large number of wide-spread and geo-distributed nodes are available if the situation requires them, including the nodes located at a certain site, neighboring nodes, or even the nodes situated at more remote geographical locations [31].

1.1 Motivation and Objectives

Geospatial query processing can be used in various applications in different aspects to solve real-life challenges. In order to resolve the crisis and effectively come up with an appropriate decision, several departments (such as electricity, transportation) need to collect huge data and work seamlessly. Extracting the Spatio-temporal information from various sources need a proper query execution framework and orchestration of geospatial services in an integrated cloud-fog-edge computing framework. Even though the cloud-only framework provides the capability of storing, managing, and analysing the massive volume of data, frequent communication with the cloud servers adds more delay and generates more energy consumption. Therefore, to provide energy-efficient and timeliness-aware services, it is necessary to utilize the computational capabilities of fog and edge nodes.

The broad objectives of this research works are summarized as follows:

- *Resolution of geospatial queries in the cloud with heterogeneous data sources.*
  Geospatial web services facilitate fetching of the geospatial data from heterogeneous data sources. A geospatial query can be represented as query tree. The leaf nodes of the query tree link to the geospatial data sources. The registry service helps to identify the data sources. The orchestration of geospatial services for fetching and displaying the data as per the user query is done in the cloud platform.

- *User-defined deadline and budget aware Spatio-temporal query processing in the cloud platform.*
  A game theory-based approach has been proposed to allocate cloud resources so that the query resolution can be done within user-defined deadline and

budget. The geospatial service chain plays a vital role in allocating the cloud resources for a Spatio-temporal query.

- *Framework for geospatial query resolution in a cloud-fog environment.*
  A fog layer has been incorporated into the existing cloud framework. Fog devices store local or regional data. Geospatial queries of the local region are resolved by the fog devices. It reduces the data load on a cloud server by distributing data over fog nodes. Consequently, the energy consumption of the user device during query resolution and delay are reduced.

- *A delay-aware and energy-efficient framework to assist users in a healthcare emergency.* Edge nodes are added within the cloud-fog architecture where the edge nodes are placed near the affected people or patient. This enables continuous data monitoring and triggering alerts to the administrator in case of any abnormalities. The cloud paradigm provides the capability of storing, managing, and analysing a massive volume of data. Since frequent communication with the cloud servers adds more delay and requires more energy consumption, it is important to address and efficiently resolve the crisis to provide adequate humanitarian relief and a sustainable environment.

1.2 Thesis Contribution

The contributions of the thesis are as follows:

1. Proposing the taxonomy on geospatial Cloud, Fog, Edge, and other computing environments. Categorizing existing works into geospatial computing, geospatial data, geospatial analysis methods, and geospatial applications.

2. Investigated different types of geospatial services, and orchestration of the services.

3. Geospatial service chaining for faster processing of geospatial query. Cloud resource allocation for queries with predicting user budget and deadline.

4. Geospatial query resolution in fog devices to facilitate energy efficiency and reduction of delay in query processing.

5. Location-based geospatial query resolution in healthcare system with Cloud-Fog-Edge-IoT architecture.

1.3 Thesis Organization

The overall organization of the thesis is presented in Fig.1.2.

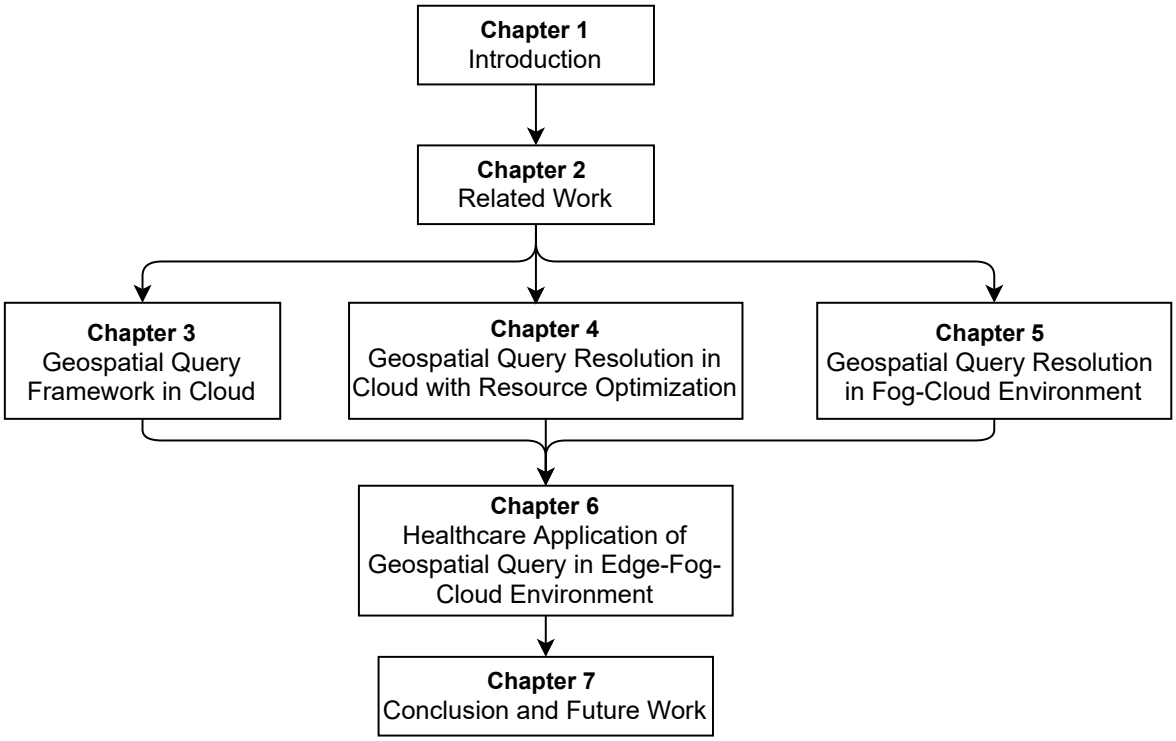


Figure 1.2: Structure of thesis

- **Chapter 2** presents the related works of Geospatial applications in different computing paradigm.
  Related Publication:
  - **Jaydeep Das**, Soumya K. Ghosh, Rajkumar Buyya, "Geospatial Edge-Fog Computing: A Systematic Review, Taxonomy, and Future Directions". In book titled Mobile Edge Computing (MEC), Springer, USA, 2020.

- **Chapter 3** elaborates geospatial query framework in Cloud platform. It discusses about the geospatial service orchestration to resolve the geospatial queries. A cloud infrastructure has been utilized for scalable resource allocation. An orchestration engine has been developed to access the geospatial resources according to query requirement.
  Related Publication:
  - **Jaydeep Das**, Arindam Dasgupta, Soumya K. Ghosh, Rajkumar Buyya, "A

Geospatial Orchestration Framework on Cloud for Processing User Queries", In Proceedings of IEEE International Conference on Cloud Computing in Emerging Markets, Pages: 1-8, IEEE, 2016.

- **Chapter 4** presents the work on the provisioning of cloud resources so that the geospatial query is resolved within user's budget and time deadline. In this regard, an efficient query resolution system can be deployed if we predict the infrastructure requirement of the user query apriori along with the identification of the geospatial service chain. The proposed framework attempts to resolve queries efficiently considering user-defined deadline and budget constraint.
  Related Publications:
  - **Jaydeep Das**, Shreya Ghosh, Soumya K. Ghosh, Rajkumar Buyya, "LYRIC: Deadline and Budget Aware Spatio-Temporal Query Processing in Cloud", in IEEE Transaction on Services Computing (TSC), April 2020, DOI:10.1109/TSC.2021.3073006.
  - **Jaydeep Das**, Sourav Kanti Addya, Soumya K. Ghosh, and Rajkumar Buyya, "Optimal Geospatial Query Placement in Cloud", In Proceedings of International Conference on Intelligent and Cloud Computing (ICICC), Pages: 335-344, Springer, Singapore, 2021.
  - **Jaydeep Das**, Arindam Dasgupta, Soumya K. Ghosh, Rajkumar Buyya, "A Learning Technique for VM Allocation to Resolve Geospatial Queries", In Proceedings of the 5^{th} International Conference on Advanced Computing, Networking, and Informatics (ICACNI), Pages: 577-584. Springer, Singapore, 2019.

- **Chapter 5** proposes a fog computing framework namely Spatio-Fog, where the fog devices contain the geospatial data of their current region and process geospatial queries using resources in the proximity. The geospatial query resolution is performed by the fog device either itself or using cloud servers or fog device of other region depending on the geographical region related to the geospatial query.
  Related Publication:
  - **Jaydeep Das**, Anwesha Mukherjee, Soumya K. Ghosh, and Rajkumar Buyya, "Spatio-Fog: A Green and Timeliness-oriented Fog Computing Model for Geospatial Query Resolution", Simulation Modelling Practice and Theory (SIMPAT), Elsevier, Volume 100, Article 102043, ISSN: 1569-190X, April 2020.

- **Jaydeep Das**, Anwesha Mukherjee, Soumya K. Ghosh, and Rajkumar Buyya, "Geo-Cloudlet: A Time-Efficient Geospatial Query Resolution Paradigm using Cloudlet", In Proceedings of IEEE 11th International Conference on Advanced Computing (ICoAC), Pages: 180-187, 2019.

- **Chapter 6** discusses an end-to-end framework which has four layers, namely, cloud, fog, edge and IoT. This framework has an efficient spatio-temporal data analytics module for information sharing, spatio-temporal data analysis to predict path for users to reach the destination (say, healthcare center or relief camps) with minimum delay in the time of exigency (say, natural disaster). This module analyzes the collected information through crowd-sourcing and assists the user by extracting optimal path post-disaster when many regions are non-reachable. The framework is deployed and evaluated using real-life datasets. The experimental and simulation results outperform the baselines to a significant margin in terms of accuracy, delay, and power consumption, and green service provisioning is achieved.
  Related Publication:
  - Shreya Ghosh, **Jaydeep Das**, Soumya K. Ghosh, Rajkumar Buyya, "CLAWER: Context-aware Cloud-Fog based Workflow Management Framework for Health Emergency Services", In Proceedings of 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid 2020), Pages: 810-817.

- **Chapter 7** summarises the overall thesis and indicates some directions of future research.

Chapter 2

A Review of Geospatial Cloud-Fog-Edge Computing Environments

Real-time geospatial applications are ever-increasing with modern Information and Communication Technology. Latency and Quality of Service-aware applications are required to process at the edge of the networks, not at the central cloud servers. Edge and fog nodes of the networks are capable enough for caching the frequently accessed small volume geospatial data, processing with lightweight tools and libraries. Several research works are carried out on edge and fog computing, especially in the geospatial domain. Health monitoring, weather prediction, emergency communication, disaster management, disease expansion are some of the examples of geospatial real-time applications. In this chapter, we have investigated the existing work of the edge and fog computing in the geospatial paradigm. We propose a taxonomy of the related works. At the end of this chapter, we discuss the limitations and future direction of the geospatial cloud-fog-edge computing.

2.1 Introduction

With the enormous usage of smartphone and IoT devices, generating, accessing, and analyzing geospatial data have increased manifolds. To access and analyze these geospatial data, substantial computing and processing resources are required [2]. The provisioning of resources varies with the applications. For the large computation, a huge infrastructure is needed for processing a large amount of geospatial data. In such cases, the central cloud computing infrastructure is the preferred solution. However, for the small amount of geospatial data processing, analyzing, and

decision making, the edge, and fog computing is a promising technology [11].

A pictorial view of the cloud, fog, and edge computing with geospatial applications is presented in Figure 2.1. Cloud is the core layer where high-end computing servers and databases are present. Users receive virtualized computing instances with different configurations for their geospatial applications. Moreover, the cloud is usually present at a multi-hop distance from the geospatial applications.

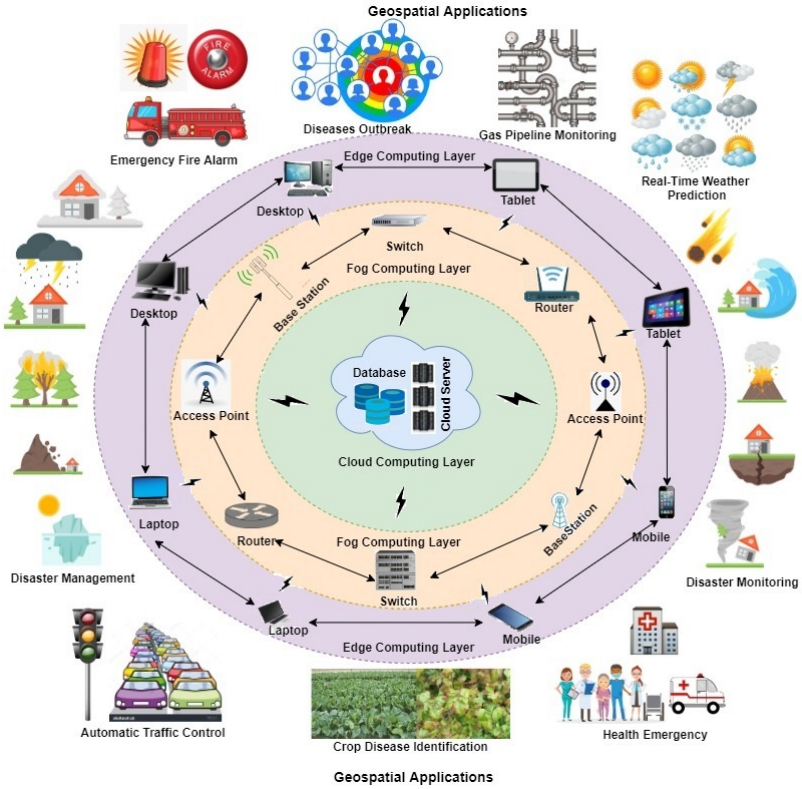


Figure 2.1: Geospatial Cloud-Fog-Edge computing layers

In fog computing layer, the computation is done in the fog nodes,i.e., switches, routers, gateways, access points, base stations [8]. These fog nodes are present in between the edge devices (mobile phone, laptop, tab) and the central cloud server. These fog nodes are capable to compute and analyze the small amount of geospatial data. After processing and analysis of the geospatial data, these fog nodes may communicate to the edge devices. Fog computing is effective in terms of service delay, energy efficiency, network congestion, etc.

Edge computing layer is constructed by the inter-connectivity among nearby edge devices like mobile phones. As edge computing is very near to the edge devices, it facilitates high network bandwidth, ultra-low latency, and real-time response [32, 33] to the geospatial applications like sending alert to the fire station, change the color of

traffic signal lights and its timespan, sending a message to the medical person about his/her patient's condition, etc.

Edge and Fog Computing (EFC), enriches the computing paradigm for real-time geospatial applications like health monitoring [34–36] systems, short-term weather prediction, disaster recovery [15, 37], crop diseases monitoring[38]. In all these cases, a quick decision has to be taken depending upon the analysis of captured geospatial data by edge nodes[39]. The response time is a major concern in all of the above situations. Fast decisions can be obtained from a geospatial EFC system than a centralised geospatial cloud system. Geospatial fog computing helps in the computation and analysis of the geospatial data. A layered architecture has been proposed in [40]. EFC system has an inner, middle, and outer edge layer. Different edge and fog devices are present in these three layers.

In summary, motivations move towards the Edge-Fog than cloud-centric computing paradigm are low latency or response-time, less network bandwidth utilization, uninterrupted service due to minimum distance from edge devices, resource-constraint at the individual edge devices affects cloud performance, and security of the edge devices is not controllable by cloud from distance [41].

In this chapter, we present a taxonomy based on a survey of cloud-fog-edge computing for geospatial domain. There are several surveys exist in edge and fog computing domain [27, 30, 31, 42–60], but none of them address geospatial aspects. In Section 2.2, we have discussed the geospatial related researches in Cloud, Cloudlet, Mist computing environment. A taxonomy on existing research work in geospatial cloud, fog, and edge computing has been structured in Section 2.3 and Section 2.4 makes a summary of these works in a tabular form for better understanding. Section 2.5 expresses the limitations in the geospatial cloud-fog-edge computing domain. Future scopes of geospatial cloud, fog, and edge computing is explored in Section 2.6. The conclusion of this chapter has been done in the last section.

2.2 Existing Computing Paradigms for Geospatial Applications

In this section, we focus on ongoing researches on cloud computing, cloudlet, mist computing with geospatial features.

2.2.1 Geospatial Cloud Computing

Currently, there are many computing strategies are available. Cloud computing[61] is the core of all these computing, where a large number of servers, databases are available. While huge computing is required for a geospatial application, then cloud is the only option for processing it. As the cloud servers reside multi-hop distance from the geospatial application nodes, it increases the overall communication delay which is sometimes critical for real-time geospatial applications like methane gas leakage monitoring, fire alarming, health monitoring [36]. The characteristics of the Cloud-GIS has been mentioned in [62], which are the extensible geospatial version of the cloud characteristics. These are - (i) elasticity of geospatial resources, (ii)on-demand geospatial service, (iii) measurable and pay-as-you-go for geospatial resources, i.e. geospatial data, geospatial tools, (iv) accessing diversity, (v) transparency, (vi) service based geospatial applications, and (vii) hardware and resource extendable. The geospatial based Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) are discussed in [63]. Along with these geospatial Data as a Service(DaaS) is also a major concern. Some geospatial services on the cloud are also mentioned in [64]. Cloud-based GIS architecture models have been discussed in [65–67]. Geospatial data indexing [68, 69] is performed for better data management in the cloud. Geospatial data interpolation [63, 70] is performed in the cloud for determining the missing geospatial data in the public dataset. Geospatial data mining [71, 72] and data processing [73–75] are performed for the getting results of the geospatial data query [76, 77]. All these geospatial data mechanisms have been done for getting the results from the geospatial applications running over the cloud computing platform.

2.2.2 Geospatial Cloudlet

Cloudlet is introduced to improve the latency of the cloud by caching the copies of data while users access the mobile applications[78]. It brings the performance of the cloud closer to mobile users. Cloudlets are computationally less powerful than the central cloud system [79]. Mobile phone, Laptop, an Access point can be used as a cloudlet. If many cloudlets are connected with each other, then the single point of failure can be avoided. Cloudlet supports mobility. The mobile device offloads the codes to the cloudlet and the code is migrated to another nearby cloudlet. While the mobile device reaches under the coverage of the second cloudlet, it starts getting the executed results from the second cloudlet [80]. Location-based service discovery is

done by the distributed cloudlets [81] and it generates less traffic in the network than a cloud-based approach. Geospatial query resolution using cloudlet reduces delay and power consumption than remote cloud access for geospatial data analysis.

2.2.3 Geospatial Mist Computing

According to [82], Mist computing is a computing layer between fog and cloudlets. Sensor and actuator devices are involved in the processing of data, which pushed the computing towards the edge node of the network [83] where edge devices are present. This reduces the communication latency within edge devices in milliseconds. Mist computing enhances the self-awareness among the edge devices in such a way that edge devices perform their operations with unstable Internet connections [40]. A Mist-GIS framework has been developed for clustering and overlying the geospatial data of the Ganga river basin [84] and malaria disease spread in the state of Maharastra, India [85].

2.2.4 Discussion

The changes of different parameters like distance from applications, computational capacity, cost, energy savings, real-time responses, etc. with respect to computing paradigms are represented in figure 2.2. However, communication delay, computational capacity, the infrastructural cost are more in a cloud environment than the other computing paradigms. Moreover, energy efficiency, closeness to the applications, and real-time response are promising in the edge, fog, and mist computing paradigm.

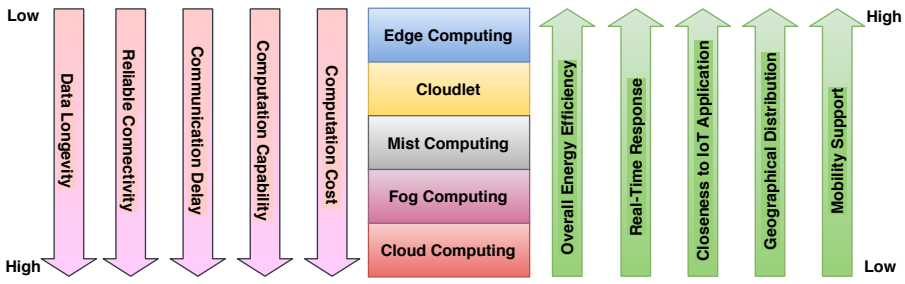


Figure 2.2: Different computing layers with parameters

2.3 Taxonomy

We have represented a taxonomy on geospatial Cloud-Fog-Edge computing in figure 2.3. This taxonomy is based on the existing works in the geospatial domain where the computation has been done in Cloud, Fog, and Edge computing environment. We have categories the works into four parts. These are-

- *Geospatial Computing:* We focus on service and resource management in edge-fog environments. Resource management is sub-categories in power, delay, cost, and geospatial data management. Whereas, service management is broken into four parts, i.e., network, application, geospatial data service, and quality of service management.
- *Geospatial Data:* The geospatial data which used for the applications running on the Edge-Fog computing are mentioned.
- *Geospatial Analysis Procedures:* The methods or procedures applied to the geospatial data, which help to identify the emergency or severity of the situations through the geospatial applications.
- *Geospatial Applications:* Different types of geospatial applications which run on the edge and fog computing environment.

In the following subsections (2.3.1-2.3.4), we elaborate existing related works that fall into the four categories mentioned above.

2.3.1 Geospatial Computing

In this section, we discuss about the overall cloud, fog, and edge computing management. It includes resource management, and service management.

Resource Management

Resource provisioning has been done depending upon the power, delay, cost by the cloud server, fog, and edge nodes. Also, keep in mind about the amount of geospatial data can be processed and stored by cloud server, fog or edge nodes [45].

Power Management: Edge and Fog computing paradigm are introduced to efficient power management of the overall network system. In [11, 86, 87], the processing of geospatial data is done at the edge and fog devices of local region.

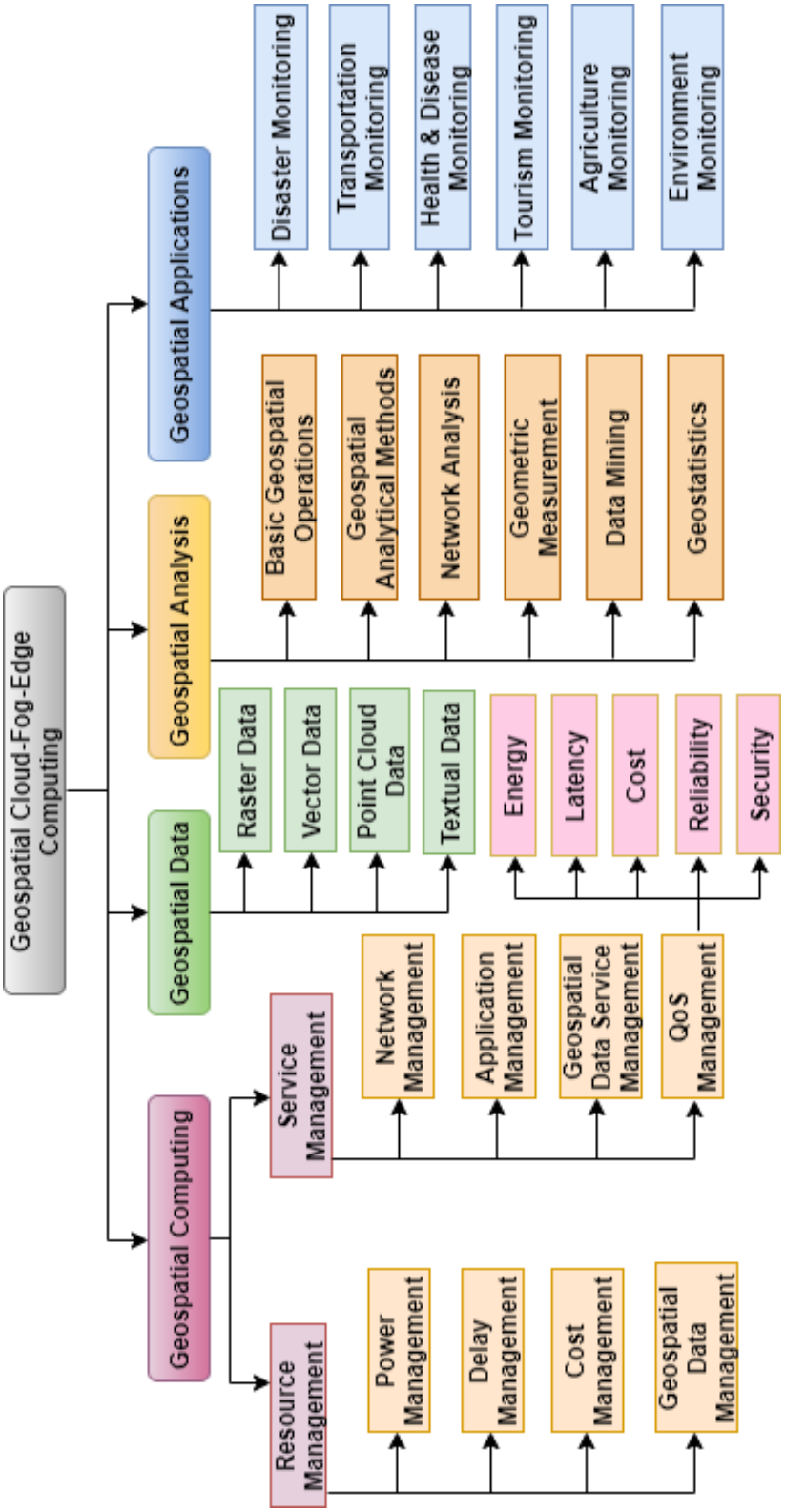


Figure 2.3: Taxonomy of geospatial Cloud-Fog-Edge Computing

Data processing at local devices reduces the data transfer to the remote cloud server. This leads to low power consumption in the overall system.

Delay Management: Delay in communication or in service is crucial for applications. Sometimes, an application loses its relevancy due to the delay. This is one of the major concerns that introduce Edge and Fog computing instead of Cloud computing. Geospatial queries are resolved within nearby Fog devices if concern data is available that fog devices. Otherwise, fog devices communicate to the cloud server for processing. They achieved 47–83% improvement in delay than the only-cloud environment. The shortest path within the critical zone has been determined in case of emergency situation [87] within nearby fog devices. They come by 9-11% better in average delay than the cloud platform. In time-critical applications [86], achieve improvement in delay on user devices as the processing of information done in nearby fog devices.

Cost Management: The cost management includes infrastructure deployment cost, networking, or communication cost, and application execution cost [49]. Data offloading cost, process migration cost are also considered for this category.

Geospatial Data Management: GIS applications are running based on geospatial data. These data are large in volume[88]. Only pre-processing of data can be done in edge and fog nodes because the infrastructure like memory, processor, storage capacity is small. Pre-processed data forward to the cloud for further processing. Sometimes, frequency used data are only cached in the edge and fog nodes, which helps to reply quickly to the user query. Various methods for matching geospatial vector data are mention in [89].

Service Management

We discuss network management, application management, geospatial data service management, and quality of service(QoS) management as overall service management of the Cloud-Fog-Edge computing environment.

Network Management: Networks are managed in the EFC paradigm through congestion control, seamless connectivity, and network virtualization. Congestion in the network can be avoided by minimizing the communication with the cloud server from the EFC network. Geospatial application requests are coming from any edge devices, and its resolution performed nearby edge or fog nodes. It leads to minimizing network traffic. Seamless connectivity helps to connect edge devices with cloud or fog servers without any latency. Seamless connectivity is possible with handover technology in future vehicular networks[90, 91]. Network virtualization

has been done by the software-defined network (SDN). Network function virtualization(NVF) helps to virtualize the traditional network functions. SDN based work in fog computing done in [92, 93]

Application Management: Real-time geospatial applications are road traffic monitoring, weather prediction, a spatial query against any point of interest(POI), emergency health monitoring. In all these cases, a cluster of reliable edge-fog nodes, low latency, and dedicated computing resources are required. Augmented reality(AR), real-time video streaming, content caching technique, bigdata analysis discussed in [94]. Using *offloading* technique[95], one nearby edge/fog nodes can forward computational tasks to its adjacent edge/fog node which has better computing resources. *Scaling* is another aspect that helps to run the application smoothly. Always the processing of geospatial data amounts is not the same. When it increases, the computation power needs to increase. This leads to a challenge for edge/fog nodes. In the case of scalability, cloud is still a promising technology.

Geospatial Data Service: Geospatial data are integrated from various sources through OGC compliant web services [96]. There are five types of web services available. These are Web Feature Service(WFS), Web Processing Service(WPS), Web Coverage Service(WCS), Web Map Service(WMS), and Catalogue Service for Web(CSW). WFS helps to extract the features according to queries. WPS applies different spatial operations over geospatial data. WMS displays the maps according to user demands. CSW prepares the registry of the available data sources.

QoS Management: Best quality of service is achieved in EFC through energy-efficient computation, low latency in communication, overall minimal cost, reliable, and secure connection.

- *Energy:* In the EFC paradigm, energy is consume minimize through energy-aware computation offloading, mobility management federation of constrained devices [59]. In [46], the overall edge computing system will be energy efficient through edge hardware design, computing architecture, operating system, and middleware.

- *Latency:* Computation latency and communication latency are considered for overall service latency management. Computation latency depends upon the configuration (Processor, RAM) of the edge and fog nodes. Whereas, communication latency relies on network bandwidth. It can be considered as within edge nodes, edge node to Fog node, and within fog nodes connectivity.

- *Cost:* It is the summation of the computational cost, deployment cost, and net-

working cost. Network bandwidth is responsible for the networking cost [97]. Whereas, computing devices like processing unit, RAM, virtual machine cost are considered as computational cost. Deployment of cloud server, edge-fog nodes and their communication elements expenses come under the deployment cost.

- *Reliability:* It is also the main concern while an application is running on reliable edge or fog nodes. The availability of such computing nodes should be guaranteed. In [59], mentioned to make a fog service reliable the replication of required functions is required, but it may not possible due to the limited computing resources available to the fog devices. So, it is a challenge to make a service reliable and available which is running in edge and fog devices.

- *Security:* Heterogeneous and geographically distributed edge and fog nodes have a major concern about the security. Rogue fog node identification, authentication, strengthen the network, and data storage security are ways to constitute a security in the cloud-fog-edge environment [98]. There are various security attacks, like Man-in-the-middle, Distributed Denial-of-Service (DDoS), ripple effects, injection attacks [31, 99] can be done through unauthorized access of user [100, 101]. Before deployment of any geospatial applications in the EFC system, the four basic security requirements, i.e., availability, authenticity, confidentiality, and data integrity should be verified.

2.3.2 Geospatial Data

Geospatial data has its geographic location (lat/lon) attached to it. These data are captured from different types of sensors. It is also captured by the high-resolution cameras from the satellites. Raster and vector data are primary data format [16], but in [88] types of geospatial data are extended with Point Cloud data and Textual data along with prior two categories.

Raster Data: It is made up of a grid of pixels and each pixel has an individual value. All kind of aerial photography and satellite imagery comes into this category. It includes thematic cartographic maps, topographical maps, orthophotos, time series of satellite images.

Vector Data: It is made up of the point, polyline, polygon. It has a shape feature, which contains the (x, y) coordinates. The shape contains latitude, polyline longitude information instead of (x,y) while the representation is done on earth surface with 2D view.

Point Cloud Data: This kind of data helps to visualize the 3D model of the terrain. Terrestrial Mobile Mapping System (MMS) data [102], LiDAR data are examples of point cloud data [103].

Textual Data: Text data are generated from several applications with location-tagged [104]. Social media data like Twitter, Facebook data, online blogs are coming into this category. These help to generate data-driven geospatial semantics.

2.3.3 Geospatial Analysis Procedures

Geospatial analysis[105, 106] is required for visualization of the geospatial data by using software and tools. The geospatial analysis methods are described below.

Basic Geospatial Operations: Buffer creation, nearest neighbor searching, overlay analysis are the basic GIS analysis tools. Overlay of the several geospatial layers has been done based on user queries. It reduces the overload of the computer memory displaying selected data layers instead of all layers. The clip, Intersect, Union are the basic overlay tools. Whereas, the buffering technique is used to identify the affected areas in flood, forest fire[107], earthquakes[108], tsunami[109], or disease outbreak like malaria, dengue fever [110], corona etc.

Geospatial Analytical Methods: It includes the clustering of the similar point patterns, generation of the heat map, analysis of points density. These methods help to identifying city traffic flow [111], air quality determination[112], monitoring of greenhouse gas emissions from factories, households, livestock agriculture [113].

Network Analysis: This type of geospatial analysis is based on graph analysis, where the connection between edges and nodes are defined. Transportation problems can be solved by finding the shortest path between two cities connected by a road network, or rail network, or a combination of both networks. This shortest-path generation helps in healthcare facility [114], tourism facility [115]. Human movement pattern identification after analyzing the trajectories in the road network has been done in [116, 117].

Geometric Measurement: Distance and proximity between one point to another point is the basic geometric measurement which is vastly used in the GIS applications. This measurement helps in tourism facility recommendations [118] like nearby hotels, restaurants, visiting places, ATM. It also helps to find nearby hospitals, medical shops in heath-care applications[86, 119]. In disaster management, transfer the victims to the nearby shelters, or reach to the victims with relief[120, 121].

Data Mining: A large number of geo-tagged data generate from sensor nodes, drone images, mobile devices, crowdsourcing, etc. Data mining is a technique to

generate information after analyzing such unstructured geospatial data. It helps to identify human movement pattern [117], urban growth over a time period [122], smarter traffic light control during time zones [123], wildlife monitoring[124].

Geo-statistics: Spatial interpolation is a geo-statistics technique[125] to analyse the surface. This technique estimates the value of an unknown point with the knowledge of nearby known point's value. Kriging [126], Inverse Distance Weighting (IDW), Regression are well known geospatial interpolation techniques. Using these techniques, many geospatial related work like malaria-prone zone identification [127], heavy metal, i.e. zinc, soil contamination [128], recognize area of irrigation water[129] for agriculture had been done.

2.3.4 Geospatial Applications

Here, we have discussed some geospatial applications which are run on the edge-fog environment or run on the cloud environment with the support of EFC.

Disaster Monitoring: Disaster prediction data are stored in telephone central offices (TCOs). These data are important for disaster monitoring. To prevent data loss, a data distribution technique among nearby edge devices has been proposed in [15]. They have used Japan Tsunami prediction data. In [130], identify the missing people in the disaster recognizing by face. To save the energy and network bandwidth only significant facial images are sent to the cloud server. Identifying the disaster-prone area after analyzing geospatial videos and satellite images in fog-cloud environment [37].

Transportation Monitoring: A traffic management system [131] is developed where RSU and vehicles (both parked and moving) act as fog nodes according to the queueing theory. They scheduled traffic flow among fog nodes and tried to minimize the response time to make it real-time traffic management.
A mobility pattern of moving agents predicted after applying a machine learning algorithm on spatio-temporal mobility data [86]. It helps to predict the next location of the moving agents, which added advantage for Time-Critical Applications.
A prediction model[111] is generated after analyzing of Bing Maps traffic jam information, and manage traffic flow in the Chicago city.
A smart traffic lighting system is proposed in [123], which is to optimize the management process. The lighting time changes according to the traffic conditions of the roads. It reduces human errors in signaling.

Health & Diseases Monitoring: Indoor, outdoor patient's continuous health monitoring is necessary. Mukherjee et al.[132] proposed a cloud-Fog based solution

for health monitoring with mobility data of patients while he/she is an outdoor location. Any small health data analysis has been done by fog devices, but any critical data analysis and mobility data analysis has been done in the cloud server.
A heart disease identifying, HealthFog [12], architecture has been developed with deep learning technology. They used FogBus for real-time data analysis by integrating the IoT-Edge-Cloud environment with delay and energy efficiency. Malaria [85, 127], dengue fever [110] prone zone identification with geospatial map and taking action accordingly are some aspects in this category.

Tourism Monitoring: Geo-tagged Flickr images are mining to detect the accurate tourist destination in [133]. RHadoop platform helps to organize such big spatial tourism data in the Cloud platform. A mobile-based tourist recommendation system has been developed in [118]. A tourist guide application for Cyprus is discussed in [115].

Agriculture Monitoring: Vatsavai et al. [13] synthetically generates images of crop fields. With the anomaly detection, feature extraction, and unsupervised technique, they identified the Weeds and crop diseases. Omran et al.[129] proposed an irrigation water quality evaluation method for agriculture in the Darb El-Arbaein area. They classified water quality depending on the salinity of the water. The computed index value determines the quality of the water. High index (above 70) is good for irrigation, where the lower index (below 40) is bad for irrigation. A livestock agriculture analysis has been done by [113]. They analyze the dataset of biodiversity, climate, water, land, people, farms, and animals using the cloud server.

Environment Monitoring: The presence of excessive Carbon Monoxide (CO) gas in the air is a cause of environmental pollution. Monitoring of CO level increment in pollution-prone areas is developed an application of Fog computing [14]. They used krigging methods to identify the distance among CO emission areas, calculated and plotted on Google map using lat/lon information. Air quality also have been checked at low concentration levels in [112] using AirSensEUR.
Various mineral resources of India are determined after data mining of spatial big data and displayed resources using overlay analysis in the QGIS tool [134]. They also have done Ganga river management using mist Computing.

Table 2.1: Existing works in Geospatial Cloud-Fog-Edge computing

Work	Edge/Fog Nodes	Associated Computing	Considered Data	Applications
Armstrong et al.[135]	Clusters of sensors	IoT Sensors, Cloud	Safecast data	Ionizing radiation risk detecting
Barik et al.[136]	Intel Edison	GIS Cloud	Global Map data	Different Compression techniques over GIS data
Barik et al.[134]	Raspberry Pi	Cloud	Mineral resources data	Mineral Resources Information Management
Cao et al.[137]	Simulated Edge nodes	Fog Server	Taxi-trajectory data	Trajectory data collection for IoT applications
Chemodanov et al.[37]	Not Mentioned	Cloud	Video and Satellite Image data	Disaster Situational Awareness
Dautov et al.[138]	Raspberry Pi 3	Cloud	CCTV image data	Metropolitan intelligent surveillance system
Denby et al.[139]	Jetson TX2	Image Sensor	Satellite image data	Nanosatellite Constellations
Ghosh et al.[86]	Mobile device	Cloud, IoT	Mobility data	Time-Critical application
Higashino et al.[140]	Cyber physical systems	IoT, Laser Range Scanner	Not Mentioned	safety management, and vehicle speeds prediction
Klein et al. [141]	Raspberry Pi	WSN, IoT	Sensor data	Methane gas leaks monitoring
Liu et al.[130]	Edge Server	Cloud, IoT Device	Face image data	Missing People Search
Liu et al.[142]	Performance Oriented Edge Computing	IoT	Not Mentioned	Multi-scale 3D scenery processing
Mishra et al.[87]	Simulation Node	WSN, Cloud	Simulated Data	Mission critical applications
Mukherjee et al.[132]	Raspberry Pi	Cloud, IoHT	Student health data	Personalized Health Care
Nugroho et al.[14]	Mikrokontroller ESP 8266, Access Point, MiFi	Gas Sensor, Cloud Server	CO gas sensors data	CO Gas Level Monitiring
Richardson et al.[143]	Raspberry Pi-2B, Pi Camera	Single board computer	Raster data	Solar Forecasting
Tsubaki et al.[15]	Telephone central offices(TCO)	Not Mentioned	Japan Tsunami prediction data	Data loss prevention in natural disasters.
Tuli et al.[12]	FogBus	Cloud, IoT	Heart patient data	Heart Diseases Monitoring
Vatsavai et al.[13]	Lenovo ThinkStation P320 with GPU	Not Mentioned	synthetically generated image	Weeds and crop diseases identification
Wang et al.[131]	RSU	Cloud, Cloudlet	taxi-trajectory datasets	Traffic Management System

2.4 Existing Work on Geospatial Cloud-Fog-Edge Computing: A Glance

We have prepared a table 2.1 for summarising the existing geospatial applications on the cloud, fog, and edge computing domain. Here, we pointed out the existing papers in the first column. In the second column said about the edge and/or for nodes used in their work. Associated with other computing paradigm, devices applied in different work are presented in the third column. In the last column, the geospatial applications which they have used in their work.

2.5 Limitations in Geospatial Cloud-Fog-Edge Computing

Every domain has its own limitations. We will discuss here the limitations of geospatial cloud-fog-edge computing.

- Geospatial data are large in volume. It is difficult to store and process it in small computing infrastructure, i.e., EFC. Whereas, the cloud has the advantage of a large data store.
- Large computation is required for geospatial prediction and analysis. Sometimes this cannot be fulfilled by EFC. Cloud servers can done large computation and support EFC.
- Small number of simulation tool, like iFogSim [144, 145], FogBus [146] for EFC are available.
- Long-distant cloud server increases the communication delay which can be reduced by local EFC.

2.6 Future Directions

In this section of the chapter, we discuss the future directions of the geospatial Cloud-Fog-Edge computing research work. Though many explorations have been done in the cloud, fog, and edge computing, very little progress happened with the geospatial EFC domain. The following aspects of geospatial Edge-Fog Computing are challenging tasks.

- Investigation of pricing policies is required individually for geospatial data providers and Edge-Fog computing service providers.
- Geospatial data management in the EFC environment is a challenge. Keeping a small amount of data within the edge and fog nodes of a distributed manner and synchronize them.
- Geospatial application management, EFC resource provisioning, with artificial intelligence and machine learning technique can be a future trend.

- Every geospatial application, i.e., weather prediction, health-care, crop analysis, etc. has its own requirements that are different from each other. Application relevant policies are required for proper management in the EFC environment.
- Automatic orchestration of different geospatial web services to resolve any geospatial query in the EFC domain can be future aspects.

2.7 Summary

In this chapter, we have discussed the existing works on the Geospatial Cloud-Fog-Edge computing domain in detail. We provide a taxonomy over Geospatial Cloud-Fog-Edge computing which considered about the different types of geospatial computing management, geospatial data types, geospatial analysis methods, and geospatial applications. We provide a brief of geospatial Cloud-Fog-Edge computing existing work in a tabular form. After that, we have discussed the limitations of the geospatial cloud, fog, edge computing. We ended our discussion with future possibilities of geospatial EFC.

Chapter 3

Geospatial Query Framework in Cloud

The demand for computing resources to process the geographical information (GI) queries has been increased drastically. The query helps the users to get the variety of information to serve their needs. Resolving the spatial queries, huge number of heterogeneous data sources along with different computing services are involved. Getting appropriate results within a specific time bound, orchestration among those data sources and web services are required. These services are available on the web and require different resource specifications in order to resolve a geospatial query. A cloud infrastructure has been utilized for scalable resource allocation. An orchestration engine (OE) has been developed to access the geospatial resources according to query requirement. In this chapter, we have proposed and developed geographical data query processing framework which orchestrates spatial services according to user query in cloud environment. The empirical experimentation shows the efficiency of the proposed framework to resolve spatial queries in timely manner.

3.1 Introduction

The demand for geospatial data has been increased with the development of data acquisition systems and various geographical information system (GIS) software. With the availability of such software people expect variety of geographical information (GI) instantly for their daily use. However, responding to these queries in timely manner require not only a large memory space, but also huge computing power. The need of such computing resources depends on the user query, both in terms of memory and processing power. This is more challenging when the GIS information are fetched in mobile devices. Again, the processing power and memory space requirements vary with the type of incoming query from the user. In this situation, the demand of processing power and memory requirements depend

on the amount of data resources and the number of computing functions involve with the user query. Hence, the demand of computing power and memory varies extremely. A framework is needed to provide computing resources on demand. In such situation, cloud computing environment provides the most efficient trade-off between the GIS query and system resources.

Spatial cloud computing [2] refers to the cloud computing paradigm that is driven by geospatial sciences, and optimized by spatiotemporal principles for enabling geospatial science discoveries and cloud computing within distributed computing environment. Spatiotemporal principles [3] are critical to enable the discover-ability, accessibility and usability of the distributed, heterogeneous and massive data. For computation intensive problem, cloud computing select the resources such that it optimize the utilization of high end computation resources. It enables the timely response either worldwide or local users through geospatial optimization. Further, it also assists the design of spatiotemporal data structure and algorithms to optimize the information workflow in order to solve complex problems. Again, it needs multiple processing of geospatial data from multiple heterogenious data sources. The geoprocessing functions in cloud environment can bring scalable, on-demand, and cost-effective services. Yue et al. [147] compared different geoprocessing services in different public cloud computing platforms. More complexities are involved in the case of heterogeneous data resources. Web feature services (WFS) accumulate these heterogeneous data into a single platform to resolve the user queries. Further, it is processed by web processing services (WPS) module as per the user query requirements and demands high amount of RAM. Scaling up or scaling down of RAM requirement is very much needed for processing service. It can be done by assigning virtual machine (VM) in cloud environment.

A spatial query may have different query execution trees and can be further divided into small query trees. From these trees, the OE chooses the optimal one using business logic [148]. Selected query tree can consider as a complex spatial data analysis task. Workflow management helps in realization of parallel implementation of the spatial analysis tasks.

In this work, the complex spatial queries are considered. The resolution of the query involves multiple heterogeneous geospatial data resources. In order to resolve such complex spatial queries, the following issues are considered in this work.

- Geospatial data are voluminous and distributed over multiple data centers at different locations.
- Query execution is complex because data are coming from multiple sources

with different data format.

To resolve user query, a sequence of geospatial web services is needed. Sequencing of existing services is achieved by a complex information service framework mentioned in [149]. Bernard et al. [150] have developed a Web service framework for heterogeneous environmental information systems.

The rest of the chapter is organized as follows. Section 3.2 presents the background of the work. Section 3.3 gives an overview of our proposed system architecture. Section 3.4 presents the methodology of our system. Section 3.5 elaborates a case study and results. Finally, Section 3.6 summarises the chapter.

3.2 Background

In this section, we present some related terminologies and technologies which are utilized in our work.

3.2.1 Geospatial Web Services

We have discussed about different OGC compliant geospatial web services, i.e. WFS, WPS, WMS, CSW in chapter 1.

3.2.2 Workflow

Workflow is a sequence of computational and data processing tasks. Workflow technologies are frequently used for complex analysis of engineering, business, or scientific processes. For on-demand complex data analytics in the cloud environment, a service-oriented workflow architecture [151] is needed.

3.2.3 Orchestration Engine

Orchestration is the description of communications, and messages flow between services in the context of a business process [152]. Goal of orchestration is to make participation of web services across the enterprise boundary to access large information. A geospatial OE, embedded with a rule repository, has been proposed in chapter [149]. This OE composes different geospatial web services across the enterprise boundary.

3.2.4 Spatial Query Parsing

Parsing is a stage in the processing of a query statement. Issuing a spatial query statement, an application makes a parse call to the spatial database. Parsing should be done in such a manner that accuracy of the result is high within a short time span considering I/O cost more than CPU cost. Unlike relational database query, spatial query deals with extremely large volumes of complex objects with spatial extension [16].

3.3 System Architecture

A multilayer client-server geospatial cloud system architecture, as shown in Fig. 3.1, is customized to serve our purpose such as scalable computation and suitable VM assignment. It illustrates an implementation scenario with selected open source software components, such as GeoServer, ArcGIS, GRASS GIS, etc., supporting OGC web services standards interface in cloud environment. There are three layers in this architecture- Client, Application and Data layer where Application and Data layers are in cloud [64]. Clients(thick and thin) in Client layer can access applications and data using request-response method from cloud. Clients are able to view resultant maps according to their spatial query. The Application layer helps communicating between clients and data providers. On top of this layer, a web server presenting web services(catalog/process/map) and serving requests to and response from application servers considered as the access point of the system. Application services implement CSW, WFS, WCS and WPS.

- Catalog server applications keep track of metadata information about data and processes, received from different sources. This step becomes quite essential due to the large amount of spatial data in the Cloud. Furthermore, a well-defined approach following the publish-find-bind service framework is defined in the OGC Web Services architecture.

- Data server applications which are used to provide spatial data to the users, categorized in standardized service forms, like WMS for map images, WFS for vector data and web coverage services (WCS) for grid data.

- Processing server applications offer a repository of geospatial processes and allow users to apply them over spatial data by implementing WPS standard. Users may query about details of every process, provide the processing service

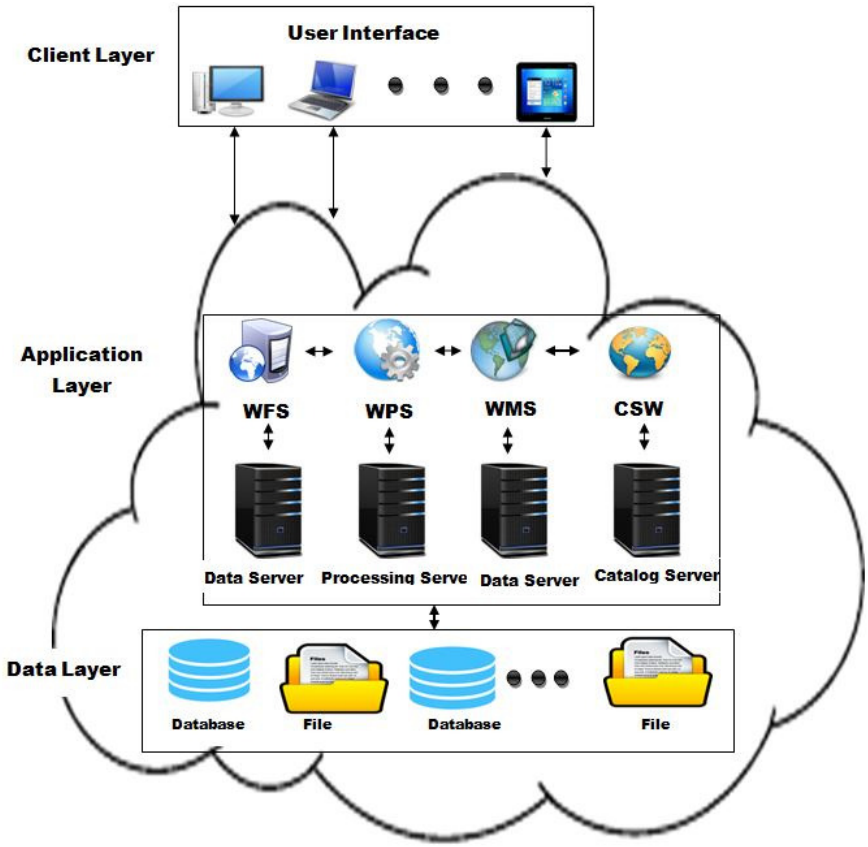


Figure 3.1: Spatial Cloud system architecture

along with these parameters, define a certain bounding box, and provide data having complex values such as binary data and XML structures. Input data given by the user will further be modified by processing units. These units can either be newly developed ones or existing GIS software tools (e.g., Grass GIS). An internal communication interface between the processing server, and the processing unit is required. This is introduced in terms of a unified modeling language (UML) sequence diagram which implements the WPS request handler component.

All the spatial data and information are available in the Data layer. This layer is used to retrieve spatial data and provide various standardized services for further computations. File systems, database management systems of different national and international organizations are accessible from this layer.

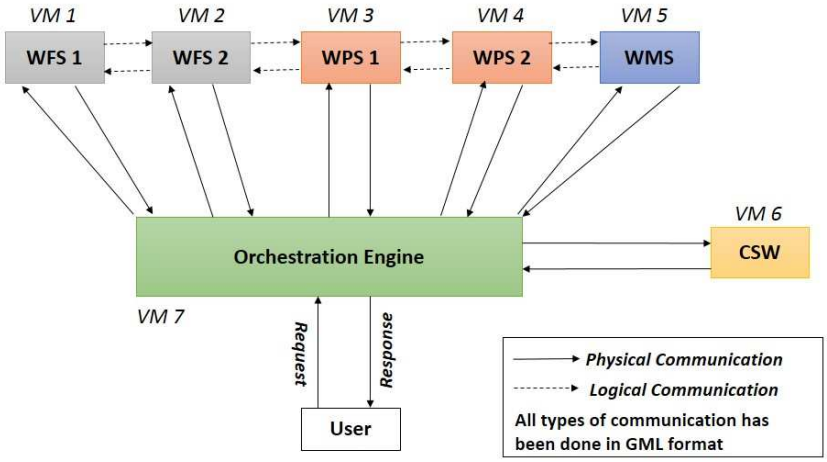


Figure 3.2: Workflow in Spatial Cloud system

3.4 Methodology

A workflow model has been developed based on OGC standard geospatial services. This workflow model (see Fig. 3.2) is utilized to generate derived information by accessing different geospatial services according to user query. The user query is interpreted by business logic in OE. After parsing the user query, a query tree will be generated by considering the essential geospatial services. It is the responsibility of OE to map the query tree into the workflow model. According to this workflow model, different web services will be required to access and list of data sources and processing sources will generate. To achieve this, OE will communicate with registry service. After getting the information about different web services and sources, OE communicates(binds) with virtual machines, which provide such web services and data sources. Maintaining the sequence (or parallel) of accessing web services and data sources, according to workflow, is a big challenge. It needs a cloud environment which can perceive, reason, learn GI services, and apply these services intelligently to construct the workflow of user query [153].

Web services are not available in a single virtual machine, rather these services are distributed in many virtual machines in a cloud platform. So, in cloud, parallel web service access can be possible. After processing of web services and data sources, results get back to OE and produce the query response to the user. Algorithm 1

describes the generation of workflow based on user query.

Algorithm 1 Workflow generation of user query

Input: User geospatial query
Output: Result of geospatial query
1: start
2: user query is placed to OE
3: the geospatial parse tree is generated from geospatial query
4: identify the essential geospatial services from parse tree
5: identify the geospatial data sources from the leaf nodes of parse tree
6: OE gathers information about VMs where geospatial services and data are available from registry service
7: OE binds with VMs according to sequence generated from parse tree
8: OE received results from VM
9: merge results in OE
10: OE sends query result to the user
11: end

User query can break into some specific service pattern

$$select\ S_F\ from\ S_D\ where\ S_C$$

- Let S_F be a collection of feature services available in the cloud in form of WFS, denoted as
  $S_F =< S_{F_1}, S_{F_2}, \cdots, S_{F_n} >$.
- Let S_D be a collection of data services available, denoted as $S_D =< S_{D_1}, S_{D_2}, \cdots, S_{D_n} >$.
- S_C is the query predicate which depends on the business logic of OE and based on the logic different WPS services are called and let S_P be a collection of processing services available in the cloud in the form of WPS, denoted as $S_P = < S_{P_1}, S_{P_2}, \cdots, S_{P_n} >$.

We have represented different service flows in a sequence diagram which is presented in a Fig. 3.3.

3.5 Case study

In this work, the spatial data set (Land Use Land Cover and Road) of Purulia and Hatasuria (Bankura district), West Bengal, India are considered for generating

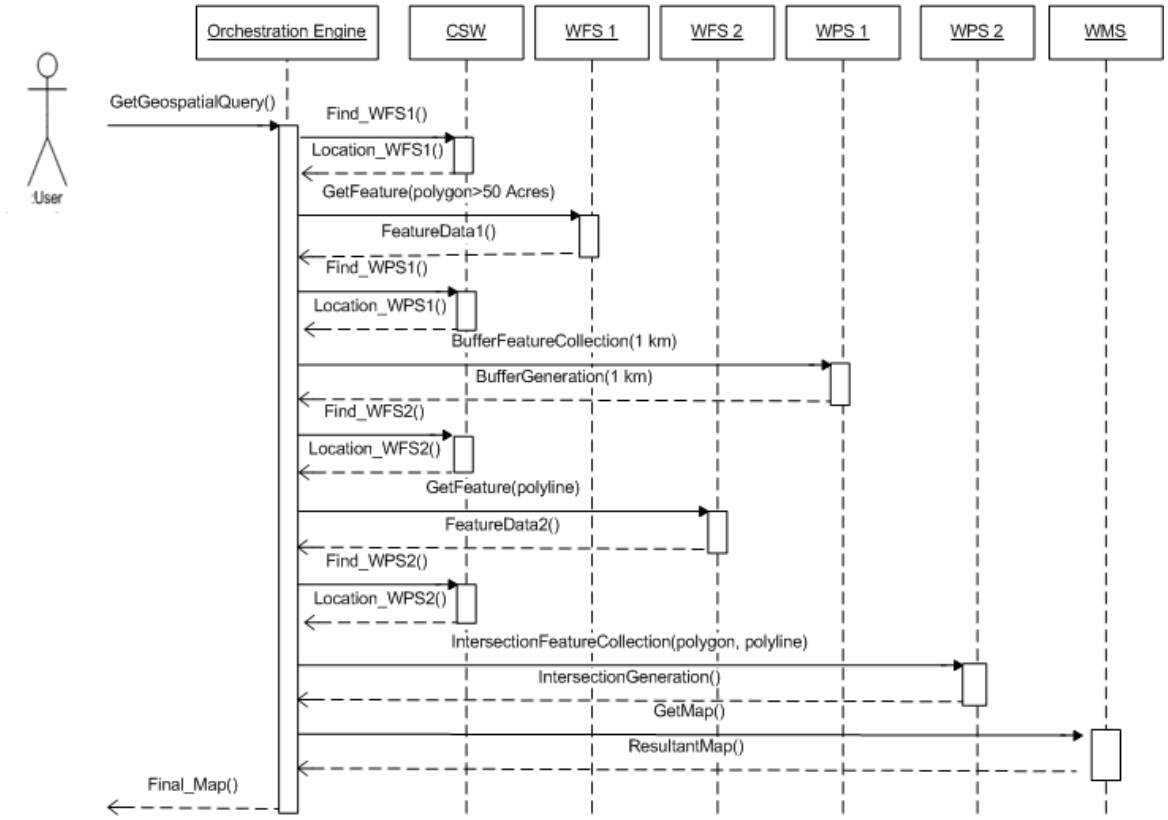


Figure 3.3: Sequential accessing of OGC web services for geospatial user query

workflow according to user query. The spatial reference system, EPSG:32645[1], is used for displaying various maps.

3.5.1 Query resolution

Query from the user: *Find the suitable(top 6) places within Purulia and Hatasuria (Bankura), West Bengal, which has at least 50 acres industrial lands and the distance from the high road less than 1 kilometer.*
Query resolutions are as follows:

```
SELECT area_name
FROM Purulia
WHERE area ≥ 50 and road = 'High Road' and Overlap (road.shape, Buffer
(area.shape, 1))
ORDER BY area desc;
```

[1] https://epsg.io/32645

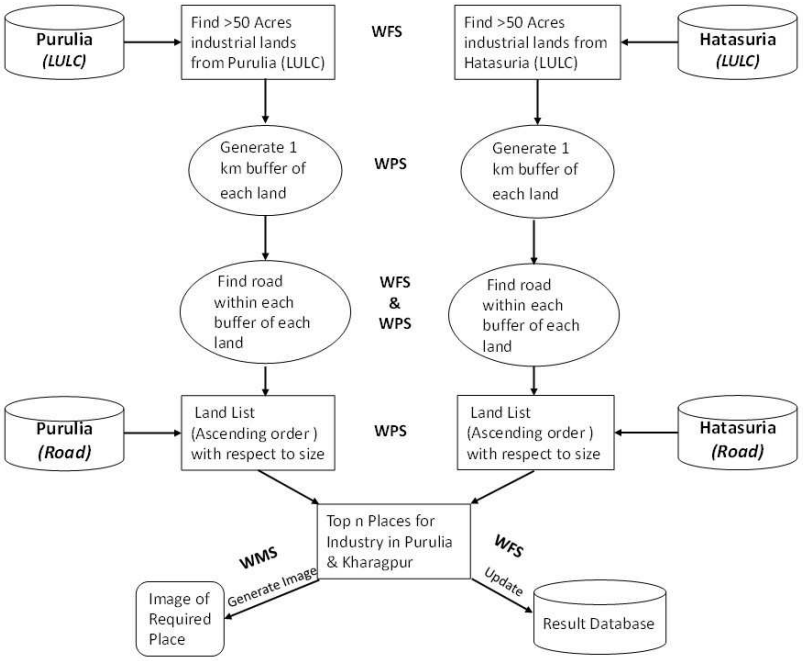


Figure 3.4: Workflow of case study

```
SELECT area_name
FROM Hatasuria
WHERE area ≥ 50 and road = 'High Road' and Overlap (road.shape, Buffer
(area.shape, 1))
ORDER BY area desc;
```

In order to resolve these queries, a predefined workflow is needed for OE. In this work, the workflow to solve such type of queries has been developed. After getting the user request, the parser interprets the query string and identifies the relevant geospatial services to solve the query. The predefined workflow model is mapped with the related services and produces a service chain. Then the service will be executed by OE to produce the result.

The steps for generating workflow are as follows:

1. Filter out the lands which have land area at least 50 acres using WFS *getFeature* service.

2. Create 1 km buffer of each filtered area using WPS *BufferFeatureCollection* service.

3. Filter out the specific roads from the road database using WFS *getFeature* service.

4. Make intersection lands buffer with filtered roads using WPS *IntersectionFeatureCollection* service.

5. Filter intersected lands using WPS *IntersectionFeatureCollection* service.

6. Generate geography markup language(GML) of the ascending order lists of resultant lands with *getFeature* service.

From both ascending order lists, user can choose suitable places meeting his requirements (land and road) from Purulia and Hatasuria. From fig. 3.4, we can observe that two parallel flow execute in a distributed system. Cloud computing is appropriate environment for executing these kinds of parallel operations in timely manner. Web services i.e., WFS, WPS and WMS are called several times. If these services are available in different virtual machines, then executions of jobs are done in short time span. However problem may occur to synchronize results.

3.5.2 Experimentation

To illustrate operational flow, we have taken snapshots of each step of the workflow. These are shown step wise in the figure Fig. 3.5. Fig. 3.5a shows all the areas of Purulia. Next Fig. 3.5b shows the filtration result of the industrial areas of Purulia. After creation of buffer of 1 km, the industrial areas look like the one shown in Fig. 3.5c. Similarly, road network of Purulia is shown in Fig. 3.6a. From this data high roads are filtered and is shown in Fig. 3.6b. The resultant intersection of Fig. 3.5c and Fig. 3.6b is shown in Fig. 3.7a. After that, the non-intersected high roads and industrial areas have been eliminated, which is shown in Fig. 3.7b. Final resultant industrial areas are shown in Fig. 3.7c.

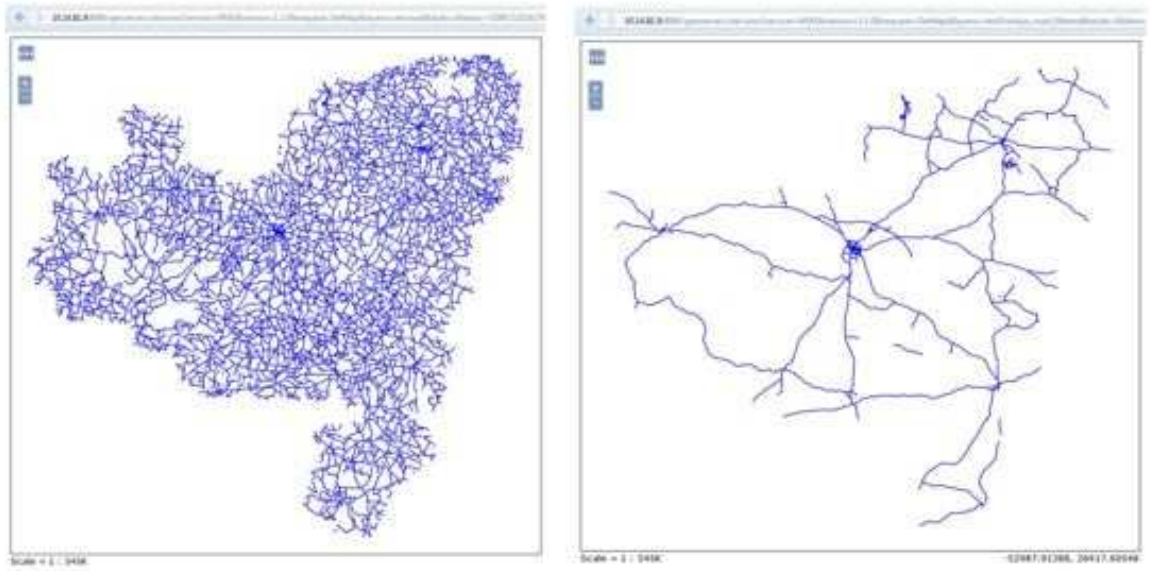
Prerequisites: Purulia and Hatasuria spatial databases with land use/land cover(LULC) and road informations.

System Configuration: The private cloud of IIT Kharagpur, *Meghamala* [1] has been used for this experimentation. Seven distinct VMs are used for different services. Web processing service has been assigned VM with 8 GB RAM. The web catalog service needs more space to store data or service registry and thus has been assigned 32GB persistent storage from Meghadata data service. Web map service is launched

[1] `http://www.sit.iitkgp.ernet.in/Meghamala/`

a. Purulia with areas b. Industrial areas of Purulia c. Industrial areas with 1 k.m. buffer

Figure 3.5: Spatial query outputs (Study area: Purulia district, West Bengal)



a. Purulia road network b. High roads of Purulia

Figure 3.6: Spatial query outputs for road network (Study area: Purulia district, West Bengal)

in the VMs with 4GB RAM and 2 VCPUs. OE assigns tasks using logic and accumulates all results. It has been instantiated in the VMs with 4 VCPUs and 8GB RAM. The details about the assignment of virtual machines are given in TABLE 3.1.

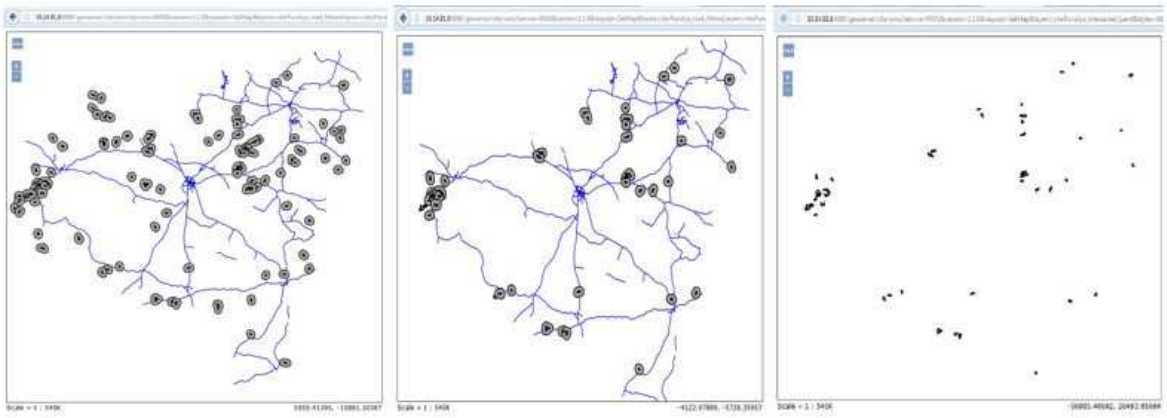


Figure 3.7: Intersection of high road and industrial area

Table 3.1: Virtual Machine configuration

VM No.	Service	VM Type	VCPUs	RAM (GB)	Ephemeral Storage (GB)	Persistent Storage (GB)
VM1	WFS1	IITKGP_regular	2	4	45	0
VM2	WFS2	IITKGP_regular	2	4	45	0
VM3	WPS1	IITKGP_large	4	8	45	0
VM4	WPS2	IITKGP_large	4	8	45	0
VM5	WMS	IITKGP_regular	2	4	45	0
VM6	WCS	IITKGP_regular	2	4	45	32
VM7	OE	IITKGP_large	4	8	45	0

3.6 Summary

In this chaper, we presents a geospatial query resolution framework using OE. The operations like filtration, buffer creation, intersection, display of data are realized which help in efficient resolution of spatial queries. OE abstracts the user query done by feature service, processing service and map service respectively. All the available services are published with metadata in the service catalog. Sequence of services are automated by OE getting information from catalog service. According to the need of spatial query, synchronization of such services, executing in several virtual machines, is a challenging task. The parallel execution of some services in the cloud, may decrease the spatial query execution time.

Chapter 4

Geospatial Query Resolution in Cloud with Resource Optimization

With the enormous growth of wireless technology, improved networking, and location acquisition techniques, a huge amount of spatio-temporal traces are being accumulated. These dataset facilitates varied location-aware services and helps to take real-life decisions. The analysis and extraction of meaningful information from these massive volumes of the spatio-temporal dataset is a challenging task. Efficiently handling and processing spatio-temporal queries are necessary to respond in real-time. Processing the vast geospatial data requires scalable computing infrastructure. In this regard, an efficient query resolution system can be deployed if we predict the infrastructure requirement of the user query apriori along with the identification of the geospatial service chain. In this work, we propose a framework, namely LYRIC (deadLine and budget aware spatio-temporal querY pROcessing In Cloud), where the geospatial queries are resolved efficiently considering user-defined deadline and budget constraint.

4.1 Introduction

The huge volume of spatio-temporal data instances has motivated the data science community to analyse and utilize the underneath knowledge. Spatio-temporal data-set consists of *objects* and *events* in spatial (location) and temporal (timestamp) context. These spatio-temporal data sources open up unprecedented opportunities to extract and leverage the usable knowledge and utilize it for a smart living such as route planning, trip recommendation, weather prediction, etc. However, managing this huge volume of spatio-temporal data and obtaining optimized query performance is inevitably challenging tasks. There are several challenges in spatio-

temporal query processing. Firstly, unlike conventional database, the attributes of spatio-temporal database have different structure (*geometry*), such as *polygon*, *polyline* [16] etc. The *processing cost* of accessing a record in the spatio-temporal database depends on the *spatial* and *temporal* extent of the query itself. Therefore, an effective query processing framework is necessary to retrieve information from these huge spatio-temporal datasets. Moreover, Cloud paradigm is suitable to leverage the *pay-as-you-go* model based on the resources used in query processing.

In this regard, the primary objective of this work is to propose an effective spatio-temporal query processing framework, which is capable of providing query response within the user's deadline and budget. When an organization, or an user submits a task containing bulk queries, the processing needs to be resolved within the *user-deadline* and *budget*. This *user-deadline* is the time frame provided by the user to get the query result from the time of the submission, and *budget* is the total price (example: Price of Google Cloud Platform services[1]) incurred for utilizing the compute, storage, and/or software services of the cloud servers. The query processing techniques must be optimized to store, search, and query the records defined in geographical space and time interval. The traditional query processing tools do not work well with spatio-temporal databases due to the complex geometric structures and computations. On the other side, spatio-temporal query processing requires varied OGC[2] compliant geospatial services (Feature, Processing, Map service) to respond. Each of those geospatial services has separate processing cost and execution time. For instance, say a user submits bulk-query with 10 *mins* deadline, and $20 budget threshold. The task requires 3 feature services[3] and map services. The price of each of the services (deployed in the cloud) is $2 for 10 *mins* timespan. However, it is observed that the task can not be completed within the deadline utilizing the present configurations of the services. In such a scenario, a proper query processing plan, such as adding more compute resources for the feature service to reduce the time, needs to be adapted. However, selecting an appropriate query plan considering both deadline and budget is difficult when several geospatial services are required to resolve the query. To address the issue, LYRIC implements *cooperative game theory* where the objective is to complete the task within the deadline and reducing the overall user budget. In other words, using the minimal resources[154] for the query processing in cloud servers within the user's deadline.

[1] `https://cloud.google.com/pricing/list`
[2] `https://www.ogc.org`
[3] `https://www.ogc.org/standards/wfs`

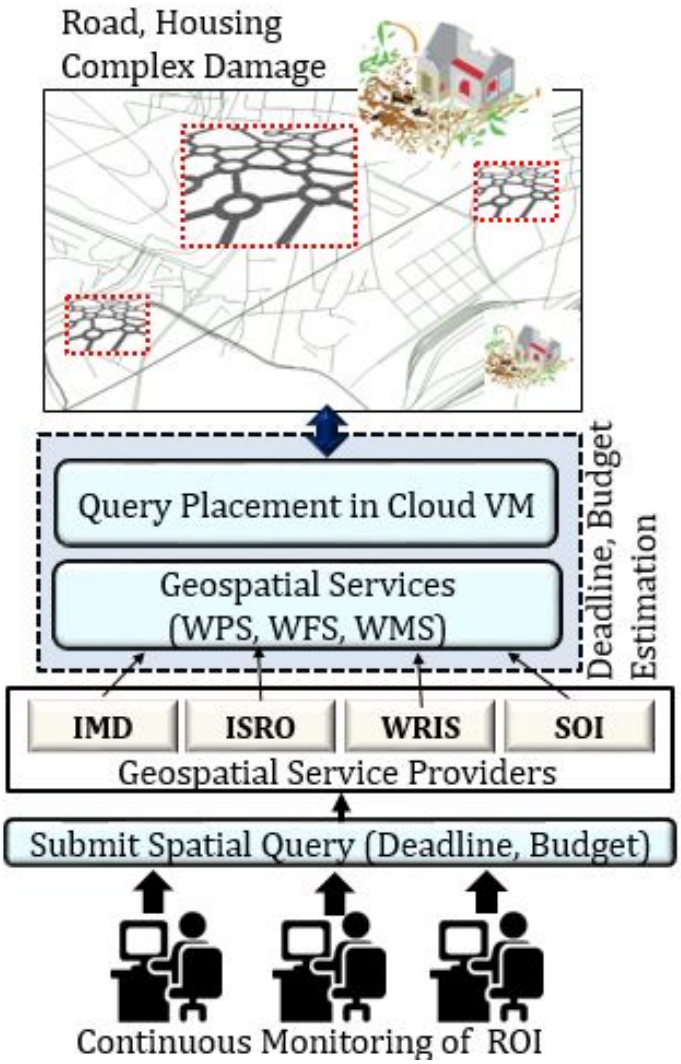


Figure 4.1: Motivating scenario

Motivating Example:

Fig. 4.1 illustrates a motivating scenario. In the time of exigency (say, super-cyclone *Amphan*[1]), the normal lives are disrupted due to power cut, shortage of water supply, road blockage or even residential place collapse. In such situation, several departments of state/ central govt. (such as electricity, communication, railway, highway, transportation, water resources, etc.) help seamlessly to *National Disaster Response Force (NDRF)* to continuously monitor the situation and taking appropriate steps to get back the normalcy. Extracting spatio-temporal information seamlessly from

[1]Super cyclone Amphan caused huge damage in eastern India. `https://en.wikipedia.org/wiki/Cyclone_Amphan`

various data sources need proper geospatial query.[1] execution and orchestration of geospatial services. For instance, national agencies of Indian government, such as GSI[2] (Geological Survey of India), India-WRIS[3] (Water Resources Information System), SOI[4] (Survey of India), IMD[5] (Indian Meteorological Department), or ISRO[6] (Indian Space Research Organisation) provide varied real-time feature services, data services, and map services to retrieve the present situation of the affected region. LYRIC provides a query execution plan considering the user's deadline and budget. This resolves the query seamlessly utilizing geospatial services in the cloud. Here, we provide an example scenario of an exigency situation in the Indian context. Our proposed framework is also suitable for any spatio-temporal query processing task with user budget and deadline constraints[155].

Our Contributions:

The key contributions of this chapter are as follows:

1. We propose an end-to-end framework, named *LYRIC*, to resolve the geospatial query within the deadline and budget provided by the user. The framework analyses the query and orchestrates several geospatial services required to resolve the query.

2. The framework is conducive to decompose the query into several components automatically. It generates the query parse tree and identifies the geospatial services, and build *geospatial service chains* for the processing. Further, it predicts the resource requirements for resolving the spatio-temporal query efficiently.

3. LYRIC proposes a novel method of choosing an appropriate query execution plan using *cooperative game theory*. The query execution plan provides the configuration of the VMs in the cloud to run the geospatial services and generates the query result considering the deadline and the user's budget.

4. The framework has been implemented and tested using spatio-temporal traces in the laboratory test-bed. The experimental observations yield encouraging results in terms of accuracy of task (bulk query processing) completion within

[1] In this chapter, we have used geospatial query and spatio-temporal query interchangeably.
[2] `https://www.gsi.gov.in`
[3] `https://indiawris.gov.in/wris/`
[4] `http://www.surveyofindia.gov.in/`
[5] `https://mausam.imd.gov.in/`
[6] `https://www.isro.gov.in/`

the deadline, reduced delay in the query response, and reduced memory and CPU usages.

The rest of this chapter is organized as follows. Section 4.2 discusses the related existing works. Section 4.3 presents the system model of our work, where we discussed geospatial query types, geospatial service chaining, and its utility. We also define the cost model of spatio-temporal queries. Section 4.4 elaborates on the performance evaluation with experimental setup and results. Summary of the chapter is discussed at the end.

4.2 Related Work

This section discusses the existing works about query processing, geospatial web services, and resource management in the cloud. Marcus et al. [156] used supervised learning techniques for batch processing and reinforcement learning techniques for online processing of user queries. This learning achieved a query scheduling with proper cost and performance management, which met the service level agreement(SLA) of user and service provider. Many researchers have also analysed the performance and latency of analytical queries through machine learning techniques [157–160].

A graph-based temporal relationship between entities like edge, vertices, properties has been proposed in [161]. Their approach is for path queries over dynamic temporal graphs. They used Granite distributed engine over Graphite ICM platform for experiments. Another graph embedded query performance prediction for concurrent queries has been proposed by [162]. They also used the graph update and compaction algorithm to determine the query workload. Chu et al.[163] predicts the query execution time using LSTM in graph database. Encoding the query plan tree, they used a post-order traversal algorithm. RF and PCA help to do feature engineering.

A resource modeling approach to measuring concurrent query performance is proposed by Duggan et al. [164], and prediction under concurrency is made in [165–167]. Concurrent Query Intensity (CQI) and Query Sensitivity (QS) are two matrices that determine the latency of concurrent queries. CQI helps to know how the resources are shared among concurrent queries, and QS defines how the query functions are changed in case of resource shortage. Popescu et al. [168] proposed to predict the runtime performance for a set of queries with a different dataset. They segmented the queries and measured the performance using the machine learning

Table 4.1: Comparisons with existing works and LYRIC

Feature	Related Works [157],[158]	[165]	[169],[170], [171]	[172]	**LYRIC** [Proposed Work]
Geospatial query execution	✓	✓	✘	✓	✓
Spatial service chain consideration	✘	✘	✓	✓	✓
Query placement to VM	✘	✘	✘	✘	✓
Priority-basis query resolution	✘	✘	✘	✘	✓
Apriori estimation of resource requirements	✘	✓	✘	✘	✓

model. Later they tried to predict the overall query runtime. They considered only tuple size and cardinality of the different dataset for estimating execution time.

Geospatial semantics and service-oriented architecture (SOA) based automatic compositions of geospatial services have been made in [169]. DataType, ServiceType, and Association type ontologies had been used as a semantic schema in SOA. They used geospatial services for ontology design, composition building, and semantic analysis. Geospatial services are used to knowledge transformation [170–172] using geospatial modeling, model instantiating, and model execution. Geospatial service orchestration in the cloud platform is described in [173]. A cloud and agent-based geospatial service chain are proposed by [174], where geospatial tasks are executed with agents movement in a single cloud environment. Agents act as part of the chain and interact with individual geospatial services. It prevents huge volumes of data transfer and service chain failure. Web service composition related literates have been done in [175].

Although several research works in this domain, all of these existing literature has some limitations. First of all, there is no clear indication of how performance characteristics (execution time and resource usages) of spatio-temporal queries can be predicted. As discussed earlier, the prediction of the performance of spatio-temporal queries is not straightforward. Most of the works need execution-time statistics of the queries and the count of the tuples processed. While this method adds more overhead, the simple count of tuples does not work in spatio-temporal queries. In brief, the contributions of LYRIC are manifolds. Firstly, it is capable of decomposing the queries into different segments and identifies several spatial-services. Our framework deploys a novel performance characteristics prediction technique and provides a query-plan to complete the query at minimal cost within the deadline.

4.3 System Model

This chapter has taken up the spatio-temporal query processing in cloud considering the user given deadline and budget to resolve the query. Spatio-temporal queries are generated by the user in bulk and submitted to the framework through a user interface. The query parser module breaks the query into a query tree with spatial and temporal information. Next, geospatial services are identified from the query tree, and a service chain is generated for processing. On the other side, decomposing the query helps identify the resource (RAM, CPU cores, storage) requirement for processing geospatial queries and predicting the execution time. Our framework, LYRIC, also considers the users' priority in resolving the query. The priority is determined by two parameters provided by the user: (i) deadline and (ii) budget to resolve it. LYRIC analyses all of these factors and offers a query execution plan resolving the task within the deadline incurring minimal budget. Geospatial queries are places into Cloud VM, and finally, the results of queries are sent back to the user through the query interface. The overall activities of our approach are illustrated in a block diagram (fig 4.2).

The query parser generates the query tree for the incoming geospatial query. The query tree nodes determine the types and number of geospatial services are required. After identifying the geospatial services, LYRIC generates the service chain. Here, we have provided different queries based on whether it requires sequential processing of the services or parallel service processing. After generating the geospatial service chain for the particular query, LYRIC provides the query execution plan based on the users' priority. This geospatial service chain formation is one of the key modules that help allocate the virtual machine to resolve the query effectively.

We also determine the resources like RAM, CPU cores, the storage requirement for a geospatial query from the query tree, and predict the query execution time. Next, we use game theory to find a proper query processing plan to resolve the geospatial query within the user-defined deadline and budget. We have shown the sequence diagram of the overall activity in fig 4.3.

4.3.1 Geospatial Query Types

Geospatial query type identification is essential to estimate the resource(RAM, CPU cores, storage) requirement for the geospatial query or batch of queries efficiently. We need to know the amount of geospatial data that has to be processed to resolve

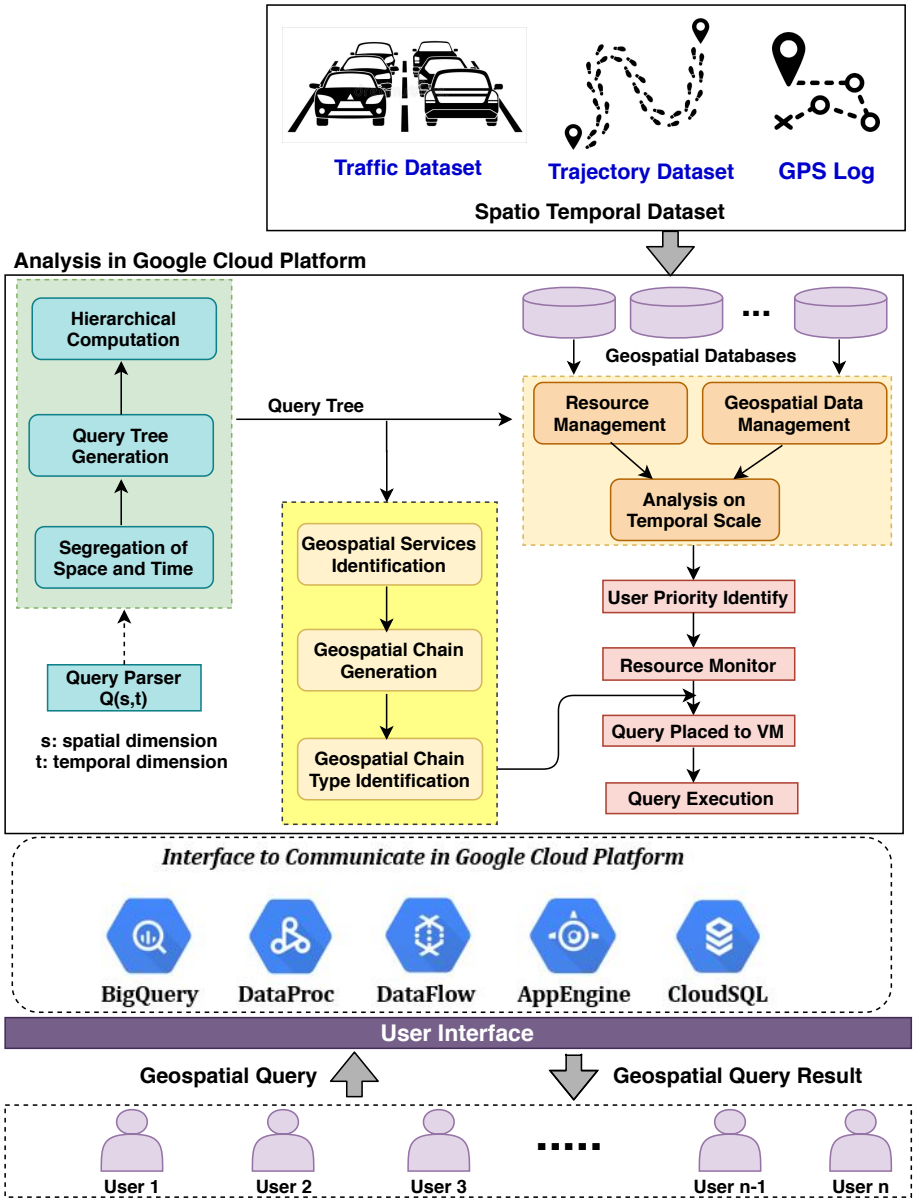


Figure 4.2: Block diagram of LYRIC framework

the geospatial query. The geospatial data amount depends upon the number of tables selected for the geospatial query. According to the number of table selection, we categorize the geospatial queries into the following two types.

- *Single Clause Query* These types of geospatial queries are with one clause. It considers only one table for extracting the result from the database.
  Example: `Select <A> from Table <B> where <C>;`
  Here, $A = \{A_1, A_2, ...A_n\}$ is name of features, $B = \{B_1, B_2, ...B_n\}$ is the set of relations or tables, and $C = \{C_1, C_2, ...C_n\}$ is set of clause.

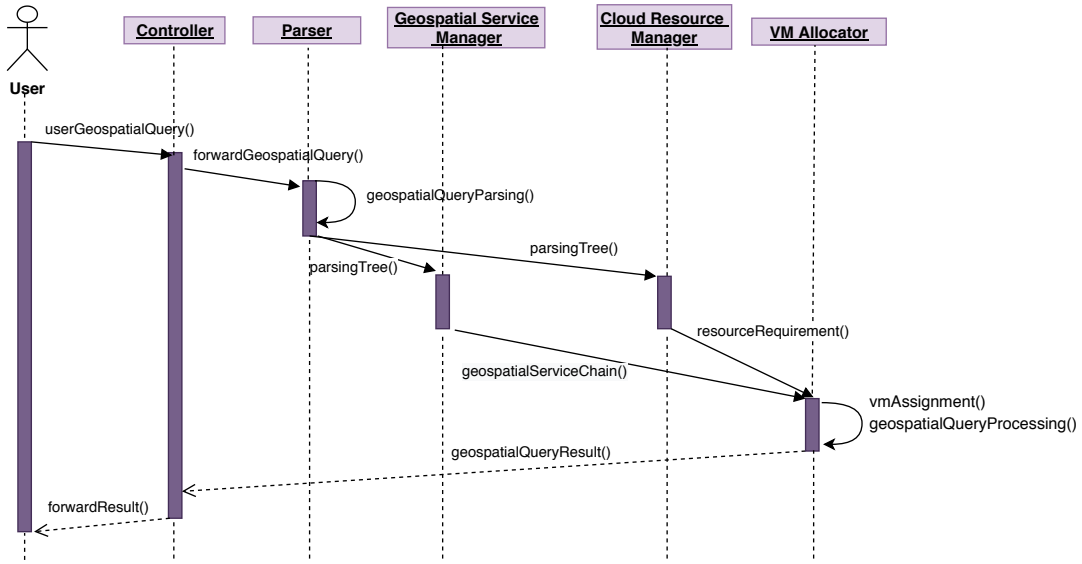


Figure 4.3: Sequence diagram of overall architecture

- *Nested Loop Query* These types of queries are associated with multiple clauses, which require either joining two or more relations or iteratively executing them. In general, the Cartesian product of the multiple tables are involved with these queries.
  Example: `Select <A> from Table <B> where <C> <conditions> (Select <A1> from Table <B1> where <C1>);`

The geospatial query parse tree has been generated from the geospatial query. It may be noted that geospatial query processing is both CPU and I/O intensive, and we need to minimize the number of disk accesses to reduce the I/O cost. We have adapted *spatio-temporal indexing* which helps to cluster the temporal information and store the spatial objects efficiently such that the number of disk accesses is minimized. Typically, LYRIC solves a spatio-temporal query in three steps (see Fig. 4.4): (i) temporal filtering, (ii) spatial filtering, and (iii) refinement step. In the temporal filtering, given any of the two types of the query (time-interval and time-instance), we search the temporal index (buckets) and find the appropriate buckets satisfying the temporal clause. The selected candidates/ tuples are returned. Next, the spatial objects (returned tuples) are represented by simple approximation. Here, we have used MBR (minimum bounding region), and in this step, the spatial operations are carried out. In the final step (refinement), the exact geometry information of the approximate candidate set is examined. It may be noted, we estimate the cardinality of each of these stages and combine them to get the exact resource requirement. For instance, the "scan" procedure is I/O intensive (requires disk accesses), and

"verify" (or, refinement) is CPU-intensive. For query parse tree generation, we follow the same steps. As depicted in the Fig. 4.5, for the following query : "Find all movement history having length greater than 1km and within 50m of Commercial building and within time-interval 09:00-0.9:15", the parse tree has been generated. For optimization purpose, it may be noted that "refinement" step needs to be pushed down. The selection of building type can be done earlier to reduce the cardinality (see the arrow sign in Fig. 4.5).

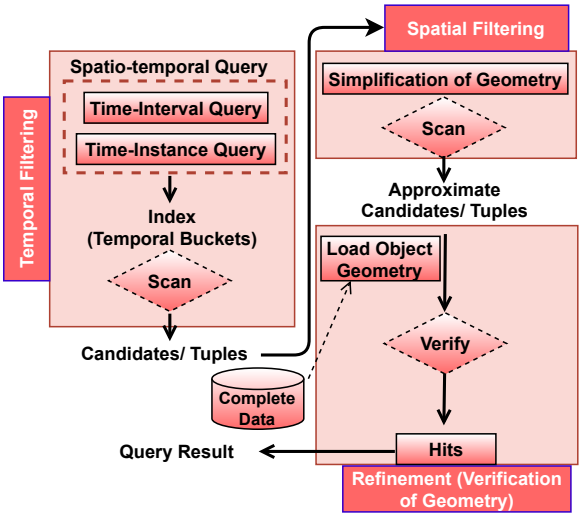


Figure 4.4: Spatio-temporal query processing stages

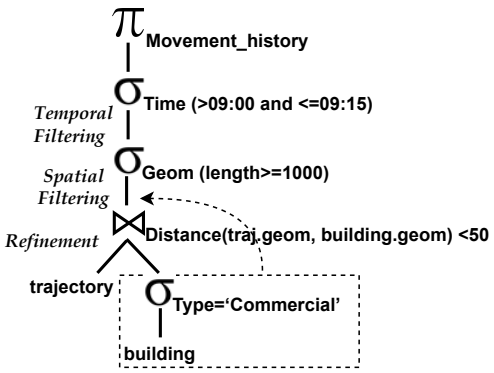


Figure 4.5: Spatio-temporal query parse tree generation

From the nodes of the tree, we can get the required geospatial services. We also get the geospatial service chain if we follow the tree's path from leaf nodes to the root node. The query parse tree generation has two major phases: (i) in the initial phase, the framework processes the *SQL-text*, and identifies the set of *A*, *B* and *C*. It also identifies whether the query processing needs a nested loop for resolving it. Based

on the conventional query parsing technique, it generates the parse tree. (ii) In the next phase, it analyzes the temporal extension of the query. Based on the temporal extension, new levels are generated at the leaf node. For each level, a specific amount of spatial data is processed. This hierarchical segmentation of spatio-temporal data in the query parse tree's leaf node helps to understand the amount of data that needs to be processed at each level of the execution effectively.

$$select\ S_{fchr}\ from\ S_{data}\ where\ S_c$$

- Let S_{fchr} be a collection of feature services available in the cloud in form of Web Feature Service(WFS), denoted as $S_{fchr} =< S_{fchr_1}, S_{fchr_2}, \cdots, S_{fchr_n} >$.
- Let S_{data} be a collection of data services available, denoted as $S_{data} = < S_{data_1}, S_{data_2}, \cdots, S_{data_n} >$.
- S_c is the query predicate which depends on the business logic of orchestration engine and based on the logic different Web Processing Services(WPS) are called and let S_{proc} be a collection of processing services available in the cloud in the form of WPS, denoted as $S_{proc} = < S_{proc_1}, S_{proc_2}, \cdots, S_{proc_n} >$.

4.3.2 Geospatial Service Chaining

Several OGC compliant geospatial services, i.e., Web Feature Service (WFS), Web Processing Service (WPS), Web Map Service (WMS), are available. The brief descriptions of the geospatial services are given below:

- *Web Feature Service*[1]*:* This service allows us to retrieve featured data from stored data. The user specifies these features. There are different operations, i.e., GetFeature, GetCapabilities, GetPropertyValue available on WFS.
- *Web Processing Service*[2]*:* This geospatial service allows us to perform different types of geospatial operations like buffering, intersection, overlaying on a point, polyline or polygon. These operations are depending upon the user's geospatial query.
- *Web Map Service*[3]*:* This service allows us to integrate multiple map layers from one or more distributed geospatial databases and displays one merged map

[1] `https://www.ogc.org/standards/wfs`
[2] `https://www.ogc.org/standards/wps`
[3] `https://www.ogc.org/standards/wms`

according to the geospatial query. The map images are in JPEG, PNG, TIFF format, which is displayed in a browser application.

These geospatial services are executing a sequential operation to generate the final result. Though multiple services are made in this chain, it still appears to be aggregated one chain to the query-user. We categories the geospatial service chain according to the number of geospatial services, i.e., WFS, WPS, WMS, involvement. We represent the six types of geospatial service chains in Fig.4.6.

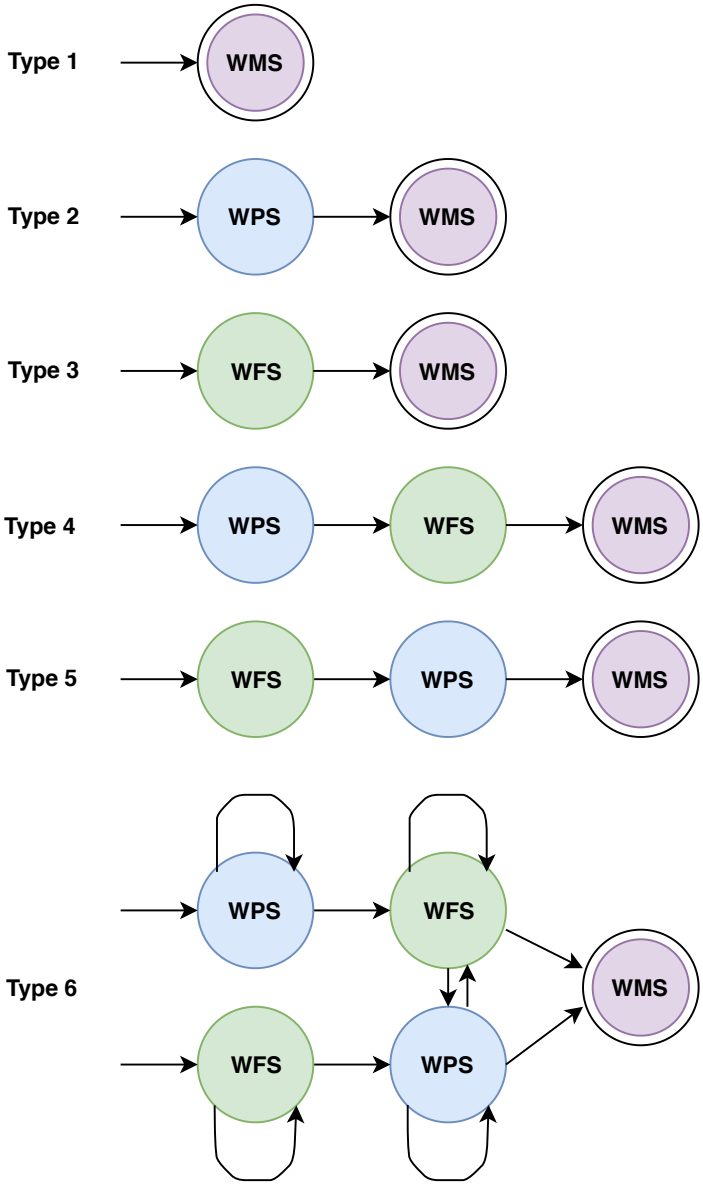


Figure 4.6: Different geospatial service chains in a state diagram form

- *Type 1: Only View* This type of geospatial queries is only for visualizing a map.

There is no such filtration or specification present here. Only web map service is responsible for this kind of geospatial queries. Already existing maps are displayed here.
Considering our motivating example, Land Use Land Cover (LULC), Transportation(Road, Rail), Drainage map of super-cyclone affected areas (here, Area_X)

```
SELECT LULC FROM Area_X;
SELECT Road Network FROM Area_X;
```

The above geospatial queries retrieve data of the individual layers (LULC/ Road Networks) of Area_X and a *getMap* WMS display these layers individually or combine in any one of the png, jpeg, and tiff format.

- *Type 2: Process and View* These types of geospatial queries are the combination of process and view. Processing is done over already existing maps. One WPS and WMS are responsible for resolving this kind of geospatial queries.
  The following example helps to identify the locations of rail and road crossing bridges of Area_X. NDRF monitors these bridges, which are affected by super-cyclone or not, and proceed accordingly.
  Example: Select crossing points of rail and road of an Area_X

```
SELECT point.geom
FROM X.rail ra, X.road ro
WHERE Cross(ra.Shape, ro.Shape)=1;
```

  In this example, first, it will take the rail network layer and road network layer of area X. *LineIntersectionService* WPS is used to obtains the intersection points with lat/lon and *getMap* WMS displays the crossing points.

- *Type 3: Filter and View* A particular area or parameter is considered for this type of query. A specific feature of a map is visualized here. One WFS and WMS will resolve this kind of geospatial queries.
  The geospatial query identifies the high roads of Area_X from the road network. It helps NDRF to clear the blockage on the high road due to super-cyclone.
  Example: High roads from Road Network

```
SELECT *
FROM X.Road_Network
WHERE road_type = 'High_Road';
```

  This query filters the high roads from Area_X road network. *getFeature* WFS identify the high roads and *getMap* WMS display the roads.

- *Type 4: Filter over the Processed area and View* A WFS, WPS, and WMS together resolve this type of geospatial queries. Filtration can be done after the processing of an existing map.
  NDRF team wants to check the conditions of the narrow bridges over rivers after super-cyclone. The following geospatial query helps to identify the same.
  Example: Identify the narrow bridges from all the intersections of road and water networks.

```
SELECT *
FROM Bridge B
WHERE B.type = 'narrow'
AND B.geom =(
SELECT point.geom
FROM Water_Network W, Road_Network R
WHERE Cross(W.shape, R.shape)= 1));
```

  This geospatial query obtains two layers, i.e., a water network and road network, to process it to determine the junction points with lat/lon. It will use *LineIntersectionService* WPS for that. Now, in the bridge layers, it filters the narrow-type bridges with the same lat/lon as junction points by using *getFeature* WFS. Finally, *getMap* WMS displays the narrow bridges in a map.

- *Type 5: Process over Filtered area and View* Here processing is done after the filtration from the existing map. WFS comes before the WPS. At last, WMS visualize the resultant map.
  Matla river is situated in the super-cyclone affected area. NDRF teams identify the damage of both the banks of *Matla* river that spread 1 kilometer of each side.
  Example: 1km buffer zones of *Matla* River

```
SELECT *
FROM Water_Network
WHERE W_Net_type = 'River' AND river_name = 'Matla'
AND Buffer(area.shape, 1);
```

  The above geospatial query first, filter the rivers with name 'Matla' using *getFeature* WFS, and then it creates buffers of 1 km over filtered river with *BufferFeatureCollection* WPS. Buffer of *Matla* will display as a result by using *getMap* WMS.

- *Type 6: Multiple Filter, Processed area and View* In this type of query, multiple

filtration and processing can be done one after another. There should not be a specific chain of WFS and WPS. The sequence can be any combination of WFS and WPS.
Suppose NDRF teams start rescue operations according to the population of the area. Densely populated areas, which are near to the highway, gets the highest priority and so on.
Example: Finding the fifty towns which have above one thousand population nearest to the national highway NH36.

```
SELECT t.name, t.population, sdo_nn_distance
FROM interstates i, town t
WHERE i.highway_name = 'NH36'
AND sdo_nn(t.location, i.geom)='TRUE'
AND t.population > 1000
AND rownum < 51
ORDER BY sdo_nn_distance;
```

The above geospatial query has many WFS like population counts, specific highway names. WPS helps to determine the nearest towns to the highway NH36 by using the Nearest Neighbour algorithm.

The process of determining the types of the geospatial service chain has been mentioned in Algorithm 2. The execution of the service-chain depends on the number and variety of geospatial services.

4.3.3 Cost Model of Spatio-Temporal Queries

Query Plan (*QP*): Query plan consists of the segments $(S_a, S_b, ..., S_z)$ of the execution of the query along with a probable time-deadline of each such segment.
$QP_q := \{S_a[t_a, f_a] \ldots S_z[t_z, f_z]\}$, where query plan ($QP$) of the query $q(T)$ is given, where T is the user-deadline. t_a and f_a is start and finish timestamp of segment S_a. t_z and f_z is start and finish timestamp of segment S_z. For each execution segment, $[t_a, f_a]$ is given, and f_z-$t_a \leq T$. After determining the cost-effective execution strategy, a query plan is produced, and the execution is carried out based on this QP.

For instance, an user u_i submits a spatio-temporal query (Q) with a deadline t_i at time-instance T_a. Further, the user also provides a budget (Bt) for resolving the query. For efficiently resolving the query considering the user-defined timeline and budget, the framework needs to compute the probable execution time. We have

Algorithm 2 Determine the types of geospatial query from parse tree

Input: Geospatial query parse tree
Output: Type of the geospatial query

```
start
the geospatial parse tree is generated from geospatial query
identify the leaf nodes of the tree, which are data nodes
the spatial operation is held on the parent node of the leaf nodes
identify the spatial operations WFS, WPS, WMS
if only WMS required then
    Geospatial Query Type 1
else
    if first WFS and then WMS required then
        Geospatial Query Type 2
    else
        if first WPS and then WMS required then
            Geospatial Query Type 3
        else
            if first WPS, then WFS, lastly WMS required then
                Geospatial Query Type 4
            else
                if first WFS, then WPS, lastly WMS required then
                    Geospatial Query Type 5
                else
                    multiple time WFS, WPS, and WMS required
                    Geospatial Query Type 6
                end if
            end if
        end if
    end if
end if
end
```

also predicted the execution time of bulk queries submitted by the users. A trade-off is required to resolve the queries within the user-deadline at minimum cost. It also helps in the capacity planning of cloud VMs, i.e., whether the VMs should be upgraded or downgraded based on the workloads. The main objective here is to *predict resource usages, i.e., memory, CPU usages, and disk accesses. Also, LYRIC analyses the query's probable run-time and user-deadline.* Based on the predicted resource usages, LYRIC computes the cost and compares it with the user's budget. Based on these two factors, it produces an appropriate query plan and configures the VMs.

The cost model of our framework follows four steps: (i) estimation of input and output cardinality; (ii) computing the CPU cost based on the cardinality estimation; (iii) computing the I/O cost based on the estimated number of accessed pages; and finally (iv) combining CPU cost, I/O cost along with WMS (map service cost for visualizing the results on map). We discuss the cardinality estimation procedure (including analyzing temporal extension) in the next section. The query parse tree and sequential-sampling approach help to get the estimation of cardinality, and the indexing and storage method is utilized to estimate the I/O cost.

4.3.4 Cost Estimation and Prediction of Run-time of Spatio-temporal Queries

In the context of the Spatio-temporal query, we define our problem space into two broad aspects: (i) static spatial objects (these are "fixed location assets", such as buildings, road-segments, lakes, mountains, etc.) where location information does not change with time; and (ii) moving objects in the two-dimensional space (moving agents, say, trajectory traces of people, vehicles, etc.). It may be noted, that temporal information is crucial in the latter case, since the location changes with time-instances, along with the data size. For the spatial query, the cost is determined by the cardinality of the relational tables. And for the spatio-temporal queries, we take the temporal extension and multiply it with the time required to process one unit of spatio-temporal operations. Let us explain the cost incurred for resolving spatio-temporal queries and how our framework, LYRIC, predicts the queries' execution time. Table 4.2 shows the notations used in the cost model. The cost of a query (Q) can be written as:

$$Cost(Q) = cs \times np + cr \times nr + cc \times nt + ci \times nti + co \times nc \tag{4.1}$$

We have used these five parameters of PostgreSQL's model in our cost model. It

Table 4.2: Notations used in cost model

Notation	Meaning
cs	I/O cost to access a page sequentially
cr	I/O cost to access a page randomly
cc	CPU processing cost of a tuple
ci	CPU processing cost of a tuple (using index)
co	CPU processing cost to carry out an operation
np	Number of sequentially scanned pages
nr	Number of randomly accessed pages
nt	Number of tuples processed/ accessed
nti	Number of tuples processed/ accessed (Indexing)
nc	Number of CPU operations
qt	Query execution time

may be noted that an accurate query time predictor requires an accurate estimation of these variables. The CPU operations mean the spatial operations such as *buffer*, *intersection* etc. along with common SQL operations (count, aggregate etc.). We consider the following query as an example:

Q_a:
```
SELECT count(*)
FROM Road_Network
WHERE road_type = 'High Road'
AND road_id = 'N'
AND Buffer(area.shape, 1);
```

Here, the relation *Road_Network* is memory resident, and two CPU operations, *count*(∗) and BUFFER() are present. Two conditions need to be satisfied. Suppose road_id has clustered index, and road_type is an attribute with a non-clustered index. Therefore, the query plan consists all five parameters (np_{Q_a},nr_{Q_a},nt_{Q_a},nti_{Q_a},nc_{Q_a}). In general, we generate such query plans where varied combinations of cost variables are required and solve the following equation:

$$qt = NC \tag{4.2}$$

where N is the cardinality of tuples or operations, and C is the CPU or I/O cost. By solving the system of linear equations, we can find out the accurate value of C. It may be noted that PostgreSQL uses the default values for I/O and CPU cost, which does not capture the real-life computational cost.

In order to estimate the run-time, two important facts need to be considered: (i) calibration of cost units and (ii) estimation of input and output cardinality of

the query. Calibration of cost units means to get the exact value of cost (say, CPU operation time) to execute a single unit of task. It may be noted that PostgreSQL uses default values for this purpose, however those are not accurate and it definitely varies with the hardware and software of the VMs where query will be executed. Here, we come up with the fundamental idea to design particular query template such that it isolates specific cost parameters from others. As we presented six types of geospatial service chain in Section 3.2, those types help to calibrate the unit costs. For example: *SELECT LULC FROM Area_X*. In this (type 1: Only View) query template, there is no I/O cost (we assume, Area_X is memory resident), and only *cpu_tuple_cost* and WMS cost (getMap) are involved. We execute the query without the WMS service and take the execution time (say, t_1). Next, we call WMS service, and note the execution time (say, t_2). Therefore, the WMS cost is $t_2 - t_1$. In our framework, we divide the study area (geographical extent) in uniform grids and temporal extent in buckets of 15 minutes. Say, the query processes n_g grids, then:

$$\begin{aligned} t_1 &= T_g \times n_g \\ t_2 &= n_g \times (T_g + WMS_g) \end{aligned} \tag{4.3}$$

where WMS_g is the time taken to call getMap service and visualize for one grid, and T_g is the time for CPU operation for one grid. From these equations, we get the values for T_g and WMS_g. In the similar way, we utilize all six types of service chains with different query sets to get the CPU cost and geospatial service cost of different spatio-temporal service chains. Also, to make the procedure more robust, we use multiple queries for each of the service chain template, and get the best-fitting of the costs. This is one-time phenomenon for any computing platform or VMs, therefore, the time taken to carry out this task does not effect the overall time-complexity of LYRIC.

Next, we need to estimate the cardinality of the input and output tuples of the query plan. Here, LYRIC utilizes a variant of *sequential-sampling* method[176]. We follow the similar steps of [157] for each *WPS* and *WFS* of the query plan. It makes the estimator more efficient and suitable for spatio-temporal queries. After this cardinality estimation and refinement stage, LYRIC effectively predicts the query execution time (qt).

For estimating the cost, we segregate into: start-cost: initial cost (c_1) before operator produce the first output tuple; and total-cost (c_2) : the total cost when all output tuples are generated. Thus, the execution cost can be represented as $ec = c_2 - c_1$. Let us illustrate with two broad categories of queries as depicted in

Section 3.1, i.e, *single clause query* and *nested loop query*. For example, if single clause query aims to select an object's location in a particular time-instance, then:

$$\begin{aligned} c_1 &= 2 \times c_o \times in_t \times \log in_t \\ ec &= c_t \times in_t \end{aligned} \tag{4.4}$$

where in_t is the number of input tuples for the operator. Here, we have discretized the temporal information into buckets, and those temporal buckets are in sorted order. Hence, we can deploy any sorting algorithm for the data-instances, therefore log factor is present in the equation. Next, for the nested loop query:

$$\begin{aligned} c_1 &= c_1 of innerclause + c_1 of outerclause \\ ec &= c_t \times in_t^{inner} \times in_t^{outer} + in_t^{outer} \times (eco finnerclause) \end{aligned} \tag{4.5}$$

where the number of input tuples from inner and outer clauses are presented by in_t^{inner} and in_t^{outer} respectively.

Let us illustrate, the cardinality estimator process here. Prior to that, let us define two terms, namely, *rank (μ)* and *selectivity (ω)*.

$$\begin{aligned} \mu &= \frac{\omega - 1}{pertuplecosto f aspatialoperation} \\ \omega(q) &= \frac{cardinality(output(q))}{cardinality(input(q))} \end{aligned} \tag{4.6}$$

Here, the *per tuple cost of spatial operation* is computed by the calibration of cost units as described before. While creating the query parse tree, we put the predicates or services based on the ascending order of μ. Thus, we optimize the query processing in terms of operations. The calculation of cardinality of input is straightforward which is the product of the cardinalities of the spatio-temporal relations (tables) that are input to the operator. We obtain this information from the metadata or system catalogue.

Say, in the spatio-temporal database SD, there are M relations or tables $\{ST_1, ST_2, \ldots, ST_M\}$. Each of the relations (say, ST_i) are partitioned into p_i blocks, and each block stores g spatial grids and bu temporal buckets. Therefore, $|ST_i| = p_i \times (g, bu)$. Now, consider two operators: *select (σ)* and *cross-product ($\times$)*. The selectivity of the operations are

as follows:

$$\omega_{\sigma(ST)} = \frac{|\sigma_{clause}(g, bu)|}{|(g, bu)|} \qquad \omega_{\times(ST)} = 1 \tag{4.7}$$

The pair (g, bu) is the spatial and temporal extents of the input relation. Now, we can say, if a relation ST_i is segmented into p grids, then $\omega_{p_i} = \frac{|\sigma_{clause}(p_i)|}{|p_i|} (1 \leq i \leq n)$, then $E[\omega_{p_i}] = \omega_{ST}$, where, we have taken n random samples of grids from ST relation. In this way, when we get the value of *selectivity*, it is straightforward to get the output cardinality.

Now, we present the estimation process of output cardinality for different spatio-temporal queries. For *spatial intersection query*, let us assume $P_i(O_1)$ is the probability that object O_1 intersects the grid G_i. Again, the probability of intersecting exactly p grids is: $P(O_1, p)$. Then, the expected number of spatial grids that O_1 will intersect is given as (g is the total number of grids present in a particular relation/ table):

$$E(O_1) = \sum_{p=0}^{g} p \times P(O_1, p) \qquad = P_i(O_1) \tag{4.8}$$

Now, for point query, an object is likely to be present at any point of the grid. Hence, we get:

$$E(O_1) = area of the grids present in the relation \qquad = \sum_{i=1}^{g} a_1 \times a_2 \tag{4.9}$$

where a_1 and a_2 are the length of the sides of the grids. For range-query $O_1(a_1 \times a_2)$, we have:

$$E(O_1) = \sum_{i=1}^{height-1} g_i \times \prod_{j=1}^{2} (n\_ext_{i,j} + a_j) \tag{4.10}$$

where *height* is the height of the spatial indexing tree (we have used R* tree [177]), and the average node extent in j dimension and i level is $n\_ext_{i,j}$. Thus, following these approaches, we estimate the cost and predict the run-time of the spatio-temporal queries.

4.3.5 Query Plan Generation using Game Theory

At this stage, we have the predicted query execution time (qt) and user-deadline (T) along with the budget (Bt) of the query. LYRIC's objective is to provide a query execution plan such that the total cost is minimized and $qt \leq T$. To obtain such a query execution plan, we deployed a cooperative game theory, where joint actions taken by the group of players provide collective payoffs. In general, $P(p_1, p_2, ...p_n)$ number of players are present in cooperative game theory, and for each subset of players, a vector of payoff (R) is defined. The tuple (P, r) is defined as a characteristic function. In our problem set-up, we define varied services of the query plan, such as, *WFS-controller*, *WPS-controller*, and *WMS-controller* as group of players. Each of the players may choose a strategy $s = \{s_1, s_2, \ldots s_j\}$ to adapt such that the aggregate payoff is maximized. In other words, all the players cooperate in the game, taking varied strategies such that the outcome of the game is the agreement condition from all the players. When we consider two influencing factors, *budget* (Bt) and user-deadline (T) of the queries, the *bargaining* method can resolve the problem.

As discussed, there are P players participating in the game, where players are the spatial-services required for the query. Each non-empty set of P is termed as a coalition. For each coalition (say, A), we denote a set $R(A) \in \mathbb{R}^{|A|}$ - which is the payoff vector and feasible for coalition A. The collection (Co) of the coalition is denoted to be balanced if the following statement is satisfied.

$$\begin{gathered} \sum_{A \in Co,\, j \subseteq A} we(A) = 1 \\ when\ we(A) \in [0,1]\ for each\ j \in P \end{gathered} \tag{4.11}$$

where $we(A)$ is the set of weights for each $A \in Co$. We have considered (P, R) as a transferable utility case of game theory in our approach. Since one of the service (or player) can losslessly transfer its utility to another service (or player). Typically, for each $A \subseteq P$, there exists a real number $r(A)$

$$R(A) = \{g \in \mathbb{R}^{|A|} : \sum_{i \in A} g_i \leq r(A)\} \tag{4.12}$$

where g is the set of vectors. In our problem, each of the strategy consists of

(*resource*,*executiontime*). The set of payoffs define the set of allocations of games (P,r)

$$G(P,r) = \{g \in \mathbb{R}^n : \sum_{i \in P} g_i = g(P) \leq r(P)\} \tag{4.13}$$

Thus, the core of the game is deployed using the following equations:

$$Core(P,r) = \{g \in G(P,r) : \forall A \subseteq P, g(A) \geq r(A)\} \tag{4.14}$$

Furthermore, it is already well-known that if the game is balanced, the core is non-empty. The bargaining solution is the function of:

$$f(U_g) : \sum^{|P|} \rightarrow U_g \tag{4.15}$$

where $\sum^{|P|}$ is the class of all bargaining problems. Based on the cooperative game theory, there is a unique solution $s(U_g)$ of bargaining problem. It can be achieved by:

$$s(U_g) = \arg\max_{y \in U_g}(y_1 - d_1) \times (y_2 - d_2) \times \ldots (y_p - d_p) \tag{4.16}$$

where d is the disagreement point. After each step, the payoff is the percentage of the completion of the task, and the utility (U_g) of the game is defined as:

$$U_g = \begin{cases} |T - qt| \times |Bt - acost| & if((T > qt) and (Bt > acost)) \\ (-1) \times |T - qt| \times |Bt - acost| & otherwise \end{cases} \tag{4.17}$$

where *acost* is the actual cost of the query execution. The user-deadline, budget, and the query execution time are represented by T, Bt, and qt respectively. It is obvious that maximizing the utility function both benefits in terms of budget and reduces the response time.

After resolving the budget and deadline trade-off using the game theory approach, we deploy a query scheduling module to assign the query into appropriate VMs and the time-stamp. The basic steps of the process are represented in Algorithm 3.

4.3.6 Example Scenario

This section presents a sample query and corresponding service chains and execution process. **Geospatial Query:** *Determine the housing complex of Area A with more than 10*

Algorithm 3 Scheduling algorithm of geospatial query placement to virtual machine)

Input: Geospatial query
Output: VM allocated to geospatial query

1: start
2: Receive geospatial query along with response deadline from user
3: Search and select available VM by the VM manager according to the type of the geospatial query
4: Calculate weights for each available VM
5: Sort VMs in increasing weights
6: Assign VM to a geospatial query according to the weights of the VM
7: If the requirement of the geospatial query is not satisfied, go to step 6
8: If the requirement of the geospatial query is not satisfied, and VMs are not available. Notification send to VM manager
9: Receives the suspended VM list from the VM manager. If there is no suspended VMs, the assignment is a failure. Go to step 11
10: Add suspended VMs to the available VM list. Go to the step 4
11: Send the assignment result to the VM manager
12: end

acres situated within 500 meters of state highway R

```
SELECT area_name FROM Area_A WHERE area ≥ 10 AND area_type='Housing_Complex'
AND road = 'State_Highway' AND road_name = 'R' AND Overlap (road.shape,
Buffer(area.shape, 0.5)) ORDER BY area desc;
```

To resolve the above geospatial query, the following geospatial services are required. (1) WFS getFeature service for *area ≥ 10* WFS getFeature service for *area_type='Housing_Complex'* (2) WPS BufferFeatureCollection service for *Buffer(area.shape, 0.5)* (3) WFS getFeature service for *road = 'State_Highway'* (4) WFS getFeature service for *road name = 'R'* (5) WPS IntersectionFeatureCollection service for *Overlap* operation (6) WFS getFeature service for *list preparation* (7) WMS getMap service for *display* final result. Hence, from the above lists of geospatial services, we can get the following geospatial service chain, which belongs to Geospatial Chaining Type 6 (refer section 4.3.2).
WFS → WFS → WPS → WFS → WFS → WPS → WFS → WMS In the next step, we generate the query parser tree, where the leaf nodes store the spatial and temporal extents of the query. We also already know the cost units of several services and CPU and I/O cost from calibration module. Then, we compute the cardinality estimation and refinement from the process described in Section 3.4. Now, given the user deadline and budget, LYRIC forms the pay-off matrix with players: five WFS, 2 WPS, and 1 WMS. All these players participate in the cooperative game and find out the coalition, where each of them gets a specific amount of time for execution and

budget or resources to utilize. Finally, the game objective is to minimize the difference between (deadline, total execution time of all players) and (budget allocated, total resource consumption of all players). Based on the game coalition, the query plan is generated and executed.

4.4 Performance Evaluation

To illustrate the efficacy of the proposed architecture, we have designed and executed a large set of experiments on mobility datasets. The intuition behind taking the mobility dataset is that the movement data is dynamic and accumulates on a large scale. Furthermore, most of our real-life spatio-temporal queries are associated with the movement, traffic, and road datasets. Therefore, we consider the use case of mobility-related queries. However, the framework is generic to handle any types of spatio-temporal datasets and resolve queries efficiently.

4.4.1 Experimental Test-bed

We have used real-life mobility traces of three geographical regions, *Kharagpur* (22.346010, 87.231972), *Durgapur* (23.5204, 87.3119), and *Waranangle* (17.9689, 79.5941) in India. The study area of these regions are $12.6 km^2$, $5.23 km^2$, and $4.08 km^2$ respectively. The number of participants in this study is 204, 42, and 76 from three regions, respectively, for a timespan of 12 months. We developed a web interface to extract their movement path, and utilized the *Google Map Timeline*, *Google Map Services* to extract the underlying road networks. It was observed that the road networks of these three regions have more than $50,000$ edges, and the cardinality of the road network makes the information extraction and resolving mobility queries more challenging factor. On the other side, the *reverse geocoding* technique was used to extract the POIs (point-of-interests) (such as commercial places like shopping malls, banks, market, or academic and residential areas, etc.) from the map database. The mobility traces are collected in $60 secs$ to $180 secs$ time interval. The total size of the dataset is $64 GB$, $36 GB$, and $49 GB$, respectively.

The experiment is conducted in the VMs of *Google Cloud Platform*, where we have used several computational and storage features of Google Cloud. Our test-bed consists of several types of compute engine instances. For instance, we have created 5 general-purpose VM instances (Ubuntu-16.04 LTS) ranging from $4 vCPU$, and $15 GB memory$ to $32 vCPUs$, and $120 GB memory$. Each such instance's approxi-

mate cost is $0.134 per hour to $1.065 per hour. These VMs are used for common workloads and having low cost and more flexibility. Along with these general-purpose VMs, we have also created 2 memory-optimized and 3 compute-optimized instances, which are used for memory and compute-intensive workloads. The initial configuration that we have selected is $40\,vCPUs$, $961\,GB\,memory$, and $30\,vCPUs$, $120\,GB\,memory$ respectively. The approximate cost is $1.253 per hour. Next, we create an instance group with these compute engine instances for auto-scaling and load-balancing based on the user requirement and predefined budget. We have also deployed spatial tools and databases, namely, *QGIS*, and *PostgreSQL with PostGIS extension*.

To store and retrieve spatio-temporal data effectively, we have utilized a novel spatio-temporal storage structure [178] for indexing temporal information and R* tree for storing and managing spatial objects. We have adapted the K-level spatio-temporal hashing technique for storing the spatio-temporal datasets. Here, the data instances are segmented and stored in different temporal buckets, and the storage technique is implemented using a hashing scheme that considers the spatial-proximity feature (nearby locations are stored in nearby buckets). Therefore, in the initial level, the spatial features of the data instances are stored, and from the next level, the data is stored into different temporal buckets.

4.4.2 Performance Results

It is quite obvious that spatial query resolution requires analysing a large number of spatial data[179]. Moreover, the performance is dependent on I/O and computational efficacy. To consider these, we have used both I/O and spatial query metrics. The I/O performance is measured by sequential and random reads along with bulk loading. The efficacy of spatial query resolution is measured by *r-query (range)*, *p-query (point)* and *t-query (trajectory)*. The range or r-query ($RangeQ(\mathcal{S},\mathcal{T})$) finds all trajectory segments which intersects the given spatial ($\mathcal{S}$) and temporal ($\mathcal{T}$) extent.

$$RangeQ(\mathcal{S},\mathcal{T}) \rightarrow Traj$$

where $Traj$ is the set of trajectory segments within spatial($\mathcal{S}$) and temporal($\mathcal{T}$) extent. The t-Query or trajectory-based query finds all trajectory segments of a moving agent (a) within the time interval ($\mathcal{T}$).

$$TrajectoryQ(a,\mathcal{T}) \rightarrow Traj$$

Here, *Traj* is the output trajectory of the query. We have used both the r-query and t-query in our evaluation set-up.

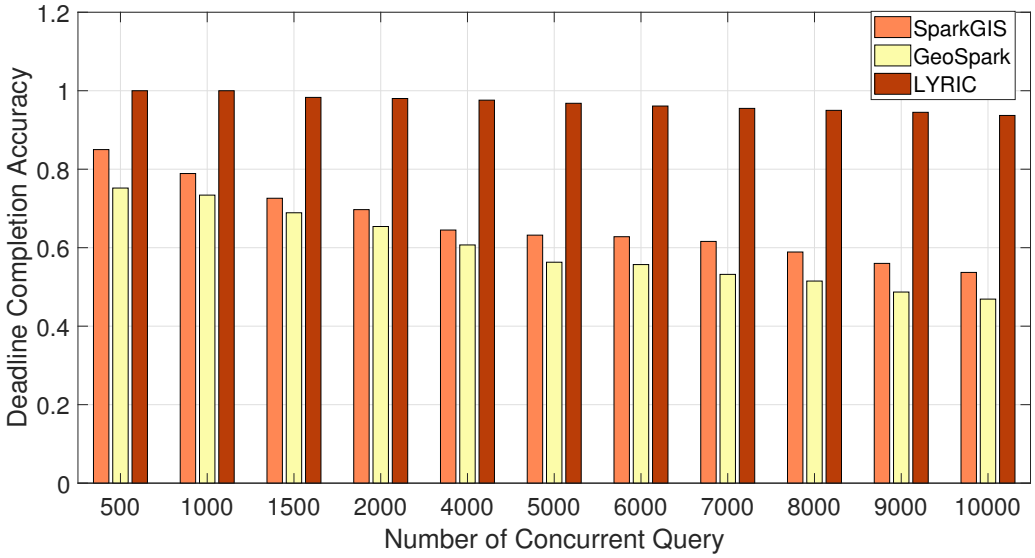


Figure 4.7: Task Completion accuracy within deadline

From Fig. 4.7, the accuracy based on the specific deadline is shown with a varied number of concurrent queries ranging from 500 to 10000. We have compared the accuracy of the result with two well-known baselines, *SparkGIS*, *geoSpark*, *GeoMesa*[1] and *JUST* [180]. The accuracy is computed based on the percentage of completion of the task in the user-deadline. It is observed that with more numbers of concurrent queries, LYRIC, significantly outperforms than others. It shows high deadline completion accuracy in the range of $1.0 - 0.937$. Whereas, in the same set-up, SparkGIS, GeoSpark, GeoMesa and JUST provide $0.85 - 0.537$, $0.752 - 0.469$, $0.928 - 0.703$, $0.921 - 0.689$ respectively. Fig. 4.8 illustrates the memory footprints for concurrent query processing compared to the other four popular methods. The memory footprints denote the amount of main memory segment accessed or referred for the query processing during execution. It is observed that the savings of memory footprints are significantly better than the existing methods. The key reason behind this result is that LYRIC computes the resource requirements apriori and assigns the required resources effectively. It also reduces the percentage of under-utilization of resources accordingly.

We evaluate the prediction accuracy of the execution time of the spatio-temporal queries. To depict the efficacy, we experiment in two set-ups: (i) prediction module

[1]http://www.geomesa.org/

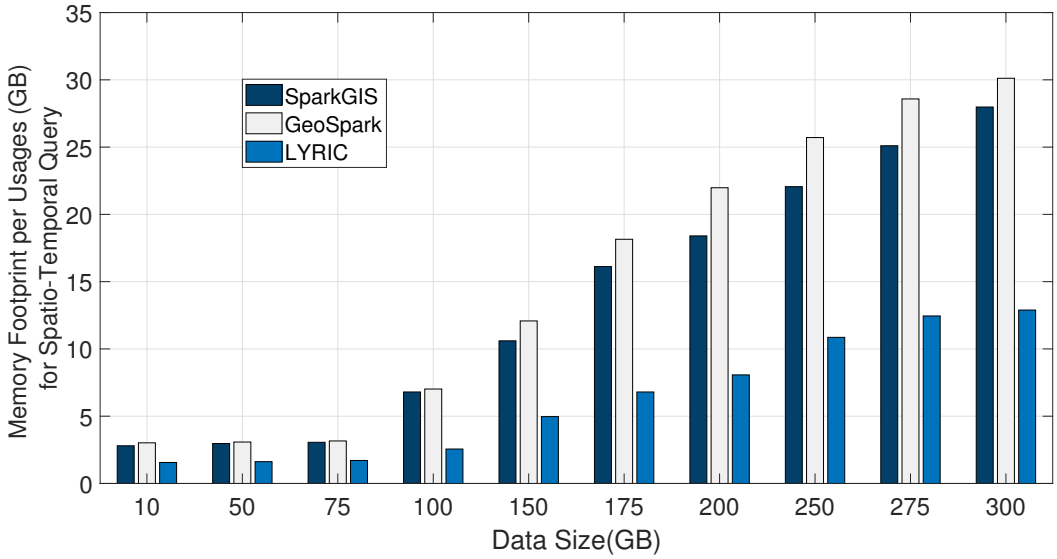


Figure 4.8: Comparison of memory footprints for concurrent query processing

without geospatial service chain and (ii) considering the geospatial service chain. The later method shows better accuracy in Fig. 4.9.

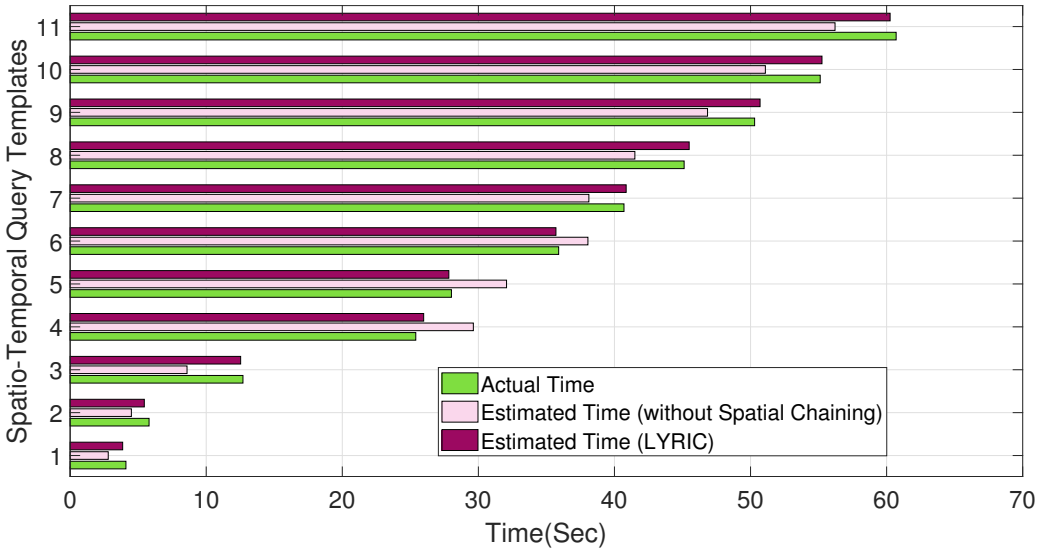


Figure 4.9: Prediction of query execution time

The reason is that the geospatial chain is one of the integral parts of providing the query result to the end-users. For instance, a few segments of query processing can be carried out in parallel. In contrast, few segments depend on the previous one, thus require sequential processing. Since LYRIC explores and identifies such geospatial service chain automatically and predicts the execution time, it achieves a more

accurate result. It is observed that the prediction module without the geospatial service chain provides 14.50% error (differences in actual execution time and prediction time), while LYRIC achieves only 2.68% prediction error. Therefore, LYRIC can reduce the prediction error by 11% incorporating the geospatial service chaining method.

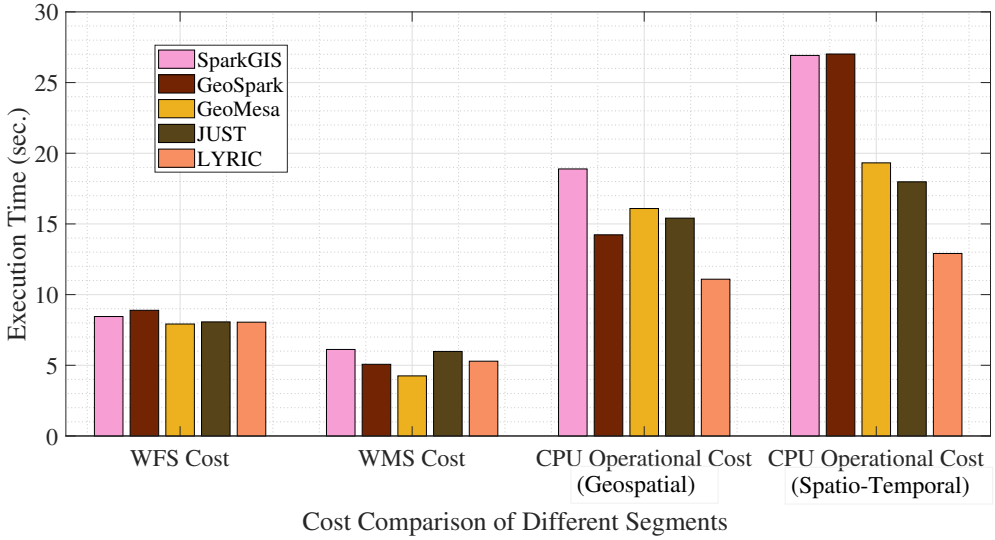


Figure 4.10: Cost comparisons with baselines

To demonstrate the effectiveness of LYRIC, we have evaluated our proposed framework using three cost metrics, namely, *Web-Feature Service Cost*, *CPU Cost* and *Web Map Service Cost*. The *CPU cost* is categorized in two classes, namely, cost related to static or spatial query operations and cost related to spatio-temporal query tasks. It is observed from Fig. 4.10 that there are marginal differences between *Web-Feature Service Cost* and *Web Map Service Cost*, however, *CPU cost* is significantly less in LYRIC compared to other methods. The major reason is that LYRIC efficiently generates the query plan and subsequently executes it for better execution time and less budget to complete the tasks. In this work, we have not used any new method for WMS or WFS cost reduction, except that we have augmented spatio-temporal indexing method [**?**] for less feature extraction delay. Therefore, the WFS cost and WMS cost are marginally different from other baselines, however, LYRIC outperforms in a significant margin in CPU cost, thus, facilitating better query execution time and budget requirements.

To evaluate the framework's effectiveness, we have used the CloudSim [181] toolkit, where we simulated the same environment as in GCP. We simulate the query

arrival scenario. As discussed, we define a set of six types of spatio-temporal queries in the list. For each such type, we initially write 120 queries manually, each having a spatial and temporal variable. From the list of such 120 queries, we generate 120×10^5 queries in the list varying the spatial and temporal domain. It may be noted that in the simulation process, we provide a bounding box for the spatial variable. Otherwise, any random pick of spatial variables may lead to a region outside our datasets' study-region. For each query, there are two other parameters, *user-deadline (T)* and *budget (Bt)*. The query arrival rate is determined by well-known *Gaussian distribution*. To evaluate LYRIC performance under varied workloads, we experiment with different arrival rates.

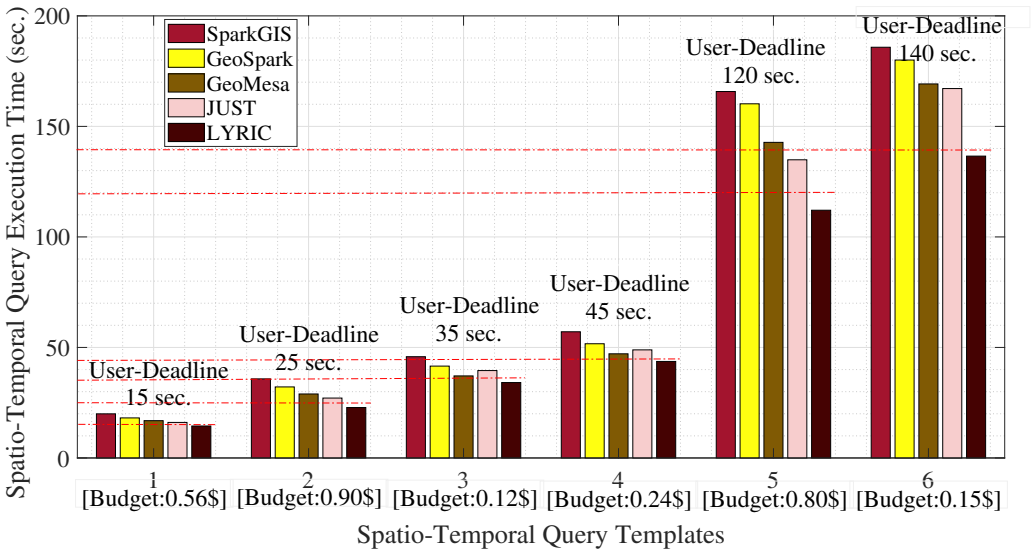


Figure 4.11: Comparison of actual execution time and deadlines

In the experimental set-up, we have considered several samples of deadlines and budgets (which are set-up based on real-life knowledge of a GIS domain expert). Fig. 4.11 and Fig. 4.12 illustrate the specific values of the deadline, allocated budget, and actual execution time and budget required to resolve the task. We have shown results for six spatio-temporal query templates and queries are randomly selected using CloudSim toolkit. In Fig. 4.11, in the experimental set-up, strict budget allocation estimation and the deadline (query task completion time) have been provided. It has been observed that for most of the cases (baselines), query execution time has exceeded the deadline provided. Again, in Fig. 4.12, we have specified that the deadline should be strictly maintained, and it is observed that the allocated budget has been overshoot in a significant margin for the baselines. The

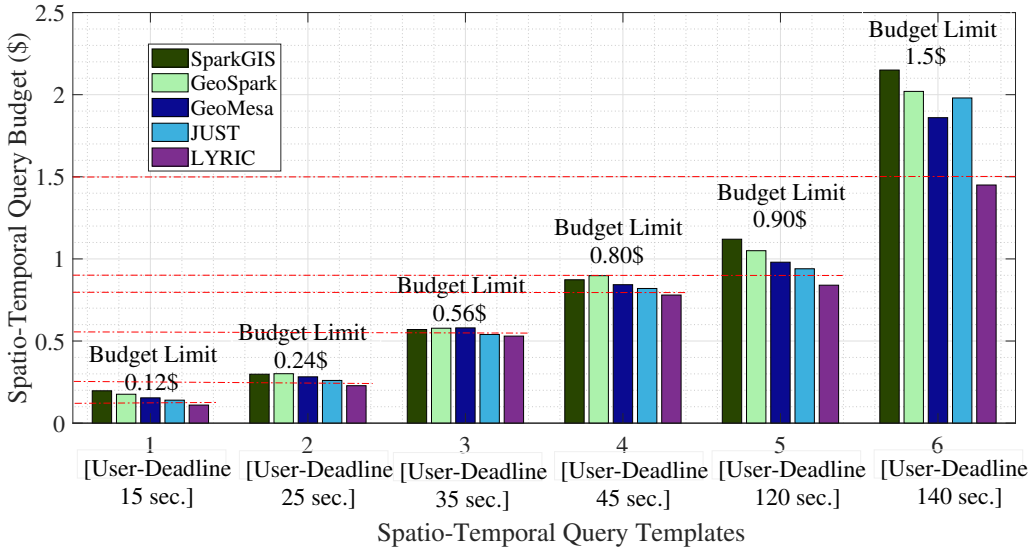


Figure 4.12: Comparison of budget requirements with baselines

key reason is that for all other distributed platforms for spatial and spatio-temporal query processing, no method/ algorithm is present to optimize both deadline and budget allocation. Moreover, the baseline methods do not consider the service chain and provision concurrent processing based on service chain execution in the query processing. However, in our case, LYRIC identifies the service chains, generates the query plan, and concurrently processes the query using a game theory approach to satisfy both deadline and budget optimally.

4.5 Summary

Accurate estimation of query execution time, as well as computational resources, is a challenging task. It helps in query scheduling based on the user-deadline and budget of the query processing. Furthermore, we can monitor the queries' progress, and for bulk query processing, particular queries taking unreasonably long time can be identified and eliminated apriori. Also, it helps in system sizing or obtaining the approximate estimation of total budget or resource utilization. Our proposed framework, LYRIC, has three main components. First, it models the cost of an incoming spatio-temporal query based on the known PostgreSQL's cost model. However, instead of the default parameters used by the PostgreSQL, LYRIC extracts the accurate CPU and disk-access cost and effectively predicts the query execution time. Next, it identifies several spatio-temporal services required to complete the

query processing task and further decomposes it in a query tree. LYRIC is capable of considering the user-defined timeline and given budget for each query. The framework utilizes the concept of cooperative game theory to obtain the trade-off between more resources and budget or cost. LYRIC is deployed in GCP, and real-life experiments with mobility datasets and simulations yield encouraging results.

Chapter 5

Geospatial Query Resolution Cloud-Fog Environment

Geospatial data analysis is an emerging area of research today. Systems need to respond to user requests in a timely manner. We propose a fog computing framework namely Spatio-Fog, where the fog devices contain the geospatial data of their current region and process geospatial queries using resources in the proximity. The geospatial query resolution is performed by the fog device either itself or using cloud servers or fog device of other region depending on the geographical region related to the geospatial query. We carried out both theoretical and experimental analysis to demonstrate feasibility of our proposed approach. The theoretical analysis illustrates that the proposed architecture Spatio-Fog reduces the power consumption and delay by approximately 43-47% and 47-83% respectively over the use of existing geospatial query resolution system. The experimental analysis demonstrates that the proposed framework reduces the power consumption and delay by 30-60% approximately than the existing geospatial query resolution system.

5.1 Introduction

Geospatial information storage, processing, and query resolution is a promising research area [182]. Usually, cloud servers are used to store and process the geospatial information. However, as geospatial information is related to geographical regions, storing and processing large volume geospatial data inside the remote cloud servers can suffer from delay and energy consumption. Mobile devices while requests for any information related to geospatial, then accessing remote cloud servers may degrade the quality of user experience by enhancing delay and energy. Moreover, storing and processing a massive amount of data on cloud data centres results in

high energy consumption as well as the use of long distant cloud servers compromise with Quality of Service (QoS) in terms of delay and energy, from the perspective of the client mobile devices [6, 7]. Fog computing has been introduced to improve the QoS, where the intermediate devices between the end node and cloud servers, e.g., switch, router, perform data and computation offloading [8]. The intermediate devices which store and process data are called fog devices [9, 10]. Use of fog devices in geospatial data storage and processing can reduce the delay and energy consumption over remote cloud servers.

5.1.1 Motivation and Contribution

The use of fog based framework in different applications such as health-care, home monitoring, has reduced the delay and energy consumption over the only cloud-based system [146]. In these types of applications, the intermediate devices perform preliminary data processing. However, for large scale data storage and processing, cloud is still the only option. The major challenges of geospatial application are [3, 182–185]:

- For geospatial data analysis, a large volume of multidimensional data has to be processed.
- For exploration, multiple approximate queries are involved in rapid succession.
- The approximate queries involve ranges specified along the spatio-temporal dimensions. The query evaluations require other concurrent rigorous and approximate queries.

The traditional fog computing based model unable to provide satisfactory QoS in case of geospatial query resolution. In [84, 136], fog device is used for partial computation on spatial data. However, the geospatial data storage and large scale processing are performed inside the cloud, and the load on cloud servers remains high, whereas resourceful fog devices remain underutilized. Hence, there should be a method for distributing the voluminous data among the fog devices and cloud, so that the resources of the fog nodes will be utilized as well as the load on the cloud will be reduced. Our objective is to introduce a new hierarchical fog based framework for the geospatial application, which will distribute the voluminous data among the fog devices and cloud servers in such a way that the load on cloud data centre will be reduced as well as the query processing time will be minimal. The contributions of this chapter are:

- A fog computing based architecture is proposed where fog devices store the regional spatial data based on the pre-calculated volume of data of the regions. The proposed system is referred as Spatio-Fog.
- The geospatial data processing and query resolution are performed inside the fog device. When a mobile user queries for information related to the geographical region, where the user is currently present, the fog device resolves the query and sends the result to the requesting device without loading the cloud servers.
- The communication and computation costs in terms of delay and power consumption for the proposed architecture are measured in terms of theoretical and experimental analysis to show that the proposed framework is delay-sensitive and energy-efficient than the existing model for geospatial query resolution.

The rest of this chapter is organized as follows. Section 5.2 discusses on the related existing strategies. Section 5.3 discusses on the geospatial data analysis. Section 5.4 elaborates the proposed fog based architecture for geospatial query resolution. Section 5.5 presents the performance analysis. Section 5.6 summaries the chapter.

5.2 Related Work

Geospatial data analysis is an emerging research challenge due to its large volume and computational intensity of processing the data. Various researches have focused on the challenges of geospatial applications [3, 182–185] and the use of cloud computing to address them. However, the use of only cloud-based system may not be efficient in terms of accessing the data at low latency, low energy consumption etc. Moreover, the load on the cloud data centre is also very high. Use of fog computing for distribution of the voluminous data has not been explored much. In this section, we have discussed on the geospatial information and the query resolution along with the existing approaches on geospatial cloud computing and fog computing.

Geospatial information refers to the data with respect to a geographical place in terms of geographic coordinates, collected using Geographic Information System (GIS) [186]. Remote sensing images are recognized as important geospatial data. Geospatial data are of two types: Raster data and Vector data. Indexing, scoring, and ranking of remote sensing images have been discussed in [187]. Query resolution for spatiotemporal database has been discussed in [188]. Query processing for

the spatial database has been discussed in [189]. Usually Quadtree [190] and R-tree [191] are popular for spatial query processing [189]. However, R^+-tree [192], R^*-tree [177] and PMR Quadtree [193] are also used for spatial query processing [189]. The authors in [189] have used XBR^+-tree for spatial query processing, which is a dynamic, disk-resident and balanced Quadtree-based index structure. Multiple query optimization for relational database has been highlighted in [194]. For geospatial query evaluation, a method has been discussed in [182], where multi-dimensional data set has been considered. A predictive and exploratory analysis over high volume multi-dimensional data set in a distributed environment has been performed in [195]. For land cover classification a geospatial web service has been designed in [196]. For storage and analysis of high volume geospatial data, cloud data centres have been used [3, 64, 183]. For high-resolution geospatial interpolation, cloud computing has been used in [63]. In the Geospatial Cloud framework, the application tier is used for geospatial services, such as Web Map Service (WMS) [197], Web Coverage Service (WCS), Web Feature Service (WFS) [25], Catalog Service for the Web (CSW) and Web Processing Service (WPS) [198]. Though cloud computing has been largely used for geospatial data analysis, use of long distant cloud servers for query resolution increases delay and energy. Use of fog computing for geospatial data analysis has been highlighted in [34, 84, 136]. Geospatial query processing in edge devices has been discussed in [84]. However, geospatial data is of large amount. Hence, the processing of voluminous data requires high-end processing, which the edge or fog devices may not be able to deal with. However, if this huge volume of data is stored in a distributed manner in fog devices and cloud servers, so that the high-end processing will be executed by the cloud, there a trade-off will be maintained between communication and computation costs.

In fog based frameworks, the fog devices are used for preliminary processing of data. Fog devices usually offload the computational tasks, rather than permanently storing massive volume of data. In the existing methods, the fog devices perform preliminary processing before forwarding the data to the cloud, for example, overlay analysis or compression on the spatial data is performed inside the fog devices, and then the data is forwarded to the cloud [34, 84, 136]. However, from the perspective of geospatial data analysis, the use of fog computing is promising due to the data intensity as well as the computational intensity of the spatial data. Hence, there should be a framework which will deal with decision making regarding the distribution of spatial data, load balancing, and query resolution with energy-efficiency and delay optimization. The users will generate queries related to relational regional spatial

Table 5.1: Comparison with existing schemes on spatial query resolution

Feature	**Existing Strategies**				**Proposed Strategy**
	Spatial cloud computing [3]	Geospatial services using cloud [64]	Geospatial data analysis using cloud [183]	FogGIS [136]	Proposed Spatio-Fog
Contribution	The use of cloud computing for spatial data search, access and utilization has been discussed.	An architecture for geospatial data processing and data acquisition services has been proposed.	The use of cloud computing for big geospatial data storage and analysis has been discussed.	Fog computing is used for preliminary processing on geospatial data.	Region-specific fog computing paradigm for geospatial query resolution by data analysis has been proposed.
Fog computing is used	X	X	X	✓	✓
Delay is calculated	X	X	X	X	✓
Power is calculated	X	X	X	X	✓
Response time is calculated	✓	X	X	X	✓

data, and the system has to respond to it. The objective of using fog computing in our work is to process a user-generated regional spatial query at an optimum delay and power consumption of the client device. Towards the solution of this, we propose a cloud-fog based distributed framework in this chapter. For query processing in the geospatial relational database, parse tree [199] is used in our work. The communication and computation costs in terms of delay and power consumption of the user device in our framework are measured and compared with the existing systems on geospatial data analysis and services in cloud and fog environment.

5.2.1 Comparison with existing schemes

As our work is based on geospatial query processing in cloud-fog framework, we have compared the proposed model with the existing strategies on geospatial data analysis and services in cloud and fog environment. Table 5.1 compares the contributions of the proposed work with the existing schemes on geospatial data analysis and services. In the existing cloud-based systems, the geospatial data storage and analysis take place inside the cloud servers [3, 64, 183]. In the existing fog based system, the fog devices are used to perform only the preliminary processing such as compression, on the geospatial data [136]. In our system, the data to be contained by the fog devices are categorized regionally, with an objective to optimize the delay and power consumption of the client device during query resolution, as well as to reduce the load on cloud data centres. This is the uniqueness of the proposed model with respect to the existing frameworks [3, 64, 136, 183]. From Table 5.1 it is also observed that in our system, we have not only analysed the data for query resolution but also calculated the delay, power consumption, and system response time.

It is observed that most of the existing approaches on geospatial data analysis and query resolution are based on cloud computing, and the fog computing has been used only for partial computation on the spatial data before forwarding to the cloud. However, none of them has focused on the use of fog nodes for distributing the storage and processing of the high volume spatial data in order to reduce the load on the cloud as well as for better utilization of the fog devices' resources. We have focused on this, and as a solution towards this challenge, we have proposed a fog-cloud based collaborative framework in section 5.4. But before that, we have discussed on the geospatial data analysis and different types of services required for query processing in section 5.3. The use of the proposed architecture for resolving these types of queries has been discussed in section 5.4.

5.3 Geospatial data analysis

Geospatial data analysis [200] is a collection of techniques and models which are used to determine the spatial relationships among geospatial data. In cartographic modelling, a map is represented by processed geospatial data. In mathematical modelling, results depend upon the spatial interaction between the objects of the model. Geospatial data analysis improves the development and application of statistical techniques. Differences between relational database query and geospatial query [201] are there are no fixed set of operators for geospatial query evaluation, and databases deal with a large volume of geospatial data, computationally expensive due to spatial predicates. Open Geospatial Consortium (OGC) compliant geospatial services help to process and publish the results of the geospatial queries. They offer the following capabilities:

- Web Feature Service(WFS) helps to extract the featured data from the dump of geospatial data.
- Web Processing Service(WPS) helps to process like buffer, intersection operations over the geospatial data.
- Web Map Service(WMS) helps to generate a map from the geospatial data according to its coordinates.

Two examples of the geospatial query are provided as follows.
Geospatial Query 1 (GQ1): List of areas which are present within 10 kilometers of each hospital in Bankura district.

```
SELECT B.area_name
FROM Bankura B
WHERE Overlap (area.shape, Buffer (hospital.shape, 10))=1
GROUP BY hospital_name
ORDER BY area desc;
```

Here,'.shape' stands for shapefile. A shapefile[1] is a simple, nontopological format for storing the geometric location and attribute information of geographic features. Geographic features in a shapefile can be represented by points, lines, or polygons (areas). The workspace containing shapefiles may also contain database tables, which can store additional attributes that can be joined to a shapefile's features.

[1]http://desktop.arcgis.com/en/arcmap/10.3/manage-data/shapefiles/what-is-a-shapefile.htm

Query parsing tree for GQ1 is presented in Figure 5.1. In GQ1, first, select the hospitals of Bankura and create a 10-kilometer buffer of each Hospital. Now, make an 'overlap' spatial operation between buffered hospital shapefile and area shapefile of Bankura. From there, we obtain area names of Bankura which are within 10 kilometers of hospitals. For resolving GQ1, the following geospatial services are

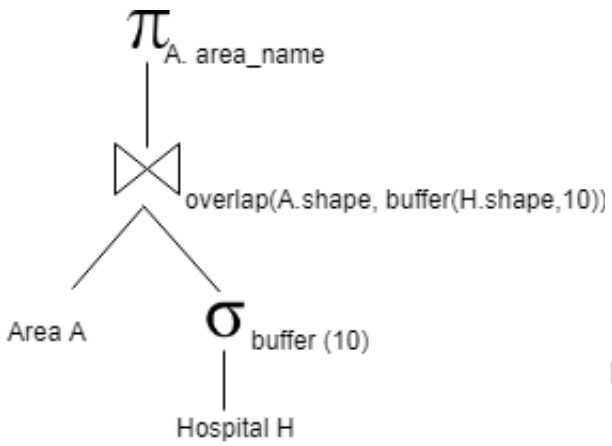


Figure 5.1: Query parsing tree of GQ1

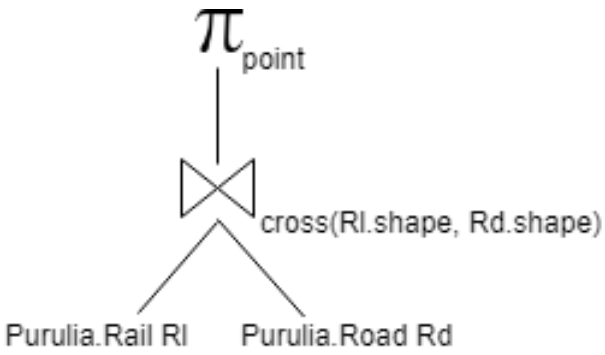


Figure 5.2: Query parsing tree of GQ2

required.

- WPS BufferFeatureCollection service is used for the creation of 10 km buffer of the hospitals.
- WPS IntersectionFeatureCollection service is used for the overlap operation.
- WMS GetMap service is used to display the final area map of Bankura.

Geospatial Query 2 (GQ2): List the 'one_Way' roads of Bengaluru, India.

```
SELECT B.road_name
FROM Bengaluru_Road B
WHERE B.road_type = 'One_Way';
```

Query parsing tree for GQ2 is presented in Figure 5.2. A 'filter' spatial operation is done over the road network shapefile of Bengaluru.

For resolving GQ2, the following geospatial services are required.

- WFS getFeature service is used for filtering the 'One_Way' roads from Bengaluru road network.
- WMS GetMap service is used to display the 'One_Way' road map of Bengaluru.

In this section, we have discussed on the geospatial data analysis and query processing. In the next section, we will propose the Spatio-Fog architecture for geospatial data storage and query processing. These types of queries will be processed by the fog devices or cloud based on the geospatial area for which the query has been made.

5.4 Proposed Spatio-Fog Architecture

The fog based system architecture for geospatial query processing is presented in Figure 5.3. The total geographical area is divided into sub-areas, e.g., countries. Each geographical sub-area has coverage which is decomposed into several zones, e.g., in a country, there are a number of states. Each zone is decomposed into several regions, e.g., each state contains a number of districts. In our system, we are considering a hierarchical architecture (see Figure 5.4), where level 1 is a sub-area, e.g., country, which is then decomposed into several zones, e.g., states, in level 2. Each zone, e.g., the state is then decomposed into several regions, e.g., districts, in level 3.

Let a is a sub-area and z_a is a zone under sub-area a, then $z_a \in a$. Let r_{z_a} is a region under zone z_a, then $r_{z_a} \in z_a$. The spatial data of region r_{z_a} is denoted by $S_{r_{z_a}}$. If f is a fog device located in r_{z_a}, then f will contain $S_{r_{z_a}}$.

In our system, fog devices located in each region contains the geospatial information of that region. Usually, switch, gateway, an indoor base station with storage and computation ability [202] are used as fog devices. In our system, the intermediate devices between the end node and cloud servers, possessing high storage and computation ability are considered as fog devices, because they will store the spatial data of the region as well as process the data for query resolution. When a user generates one or multiple queries to retrieve information related to a region where he/she is currently present, then instead of accessing the cloud servers the fog device accesses its geospatial data and responds accordingly, e.g., the geospatial query resolution is performed by the fog device instead of the cloud servers. The advantages of this framework are:

- As the fog device stores the geospatial data of the region instead of the cloud data centres, the propagation and communication delays are reduced, which in turn helps to reduce the total delay.

- As the fog device is containing the region-specific geospatial data instead of the cloud, the load on cloud data centre will be reduced. Moreover, an extra layer for regional data privacy will be maintained. However, the spatial data

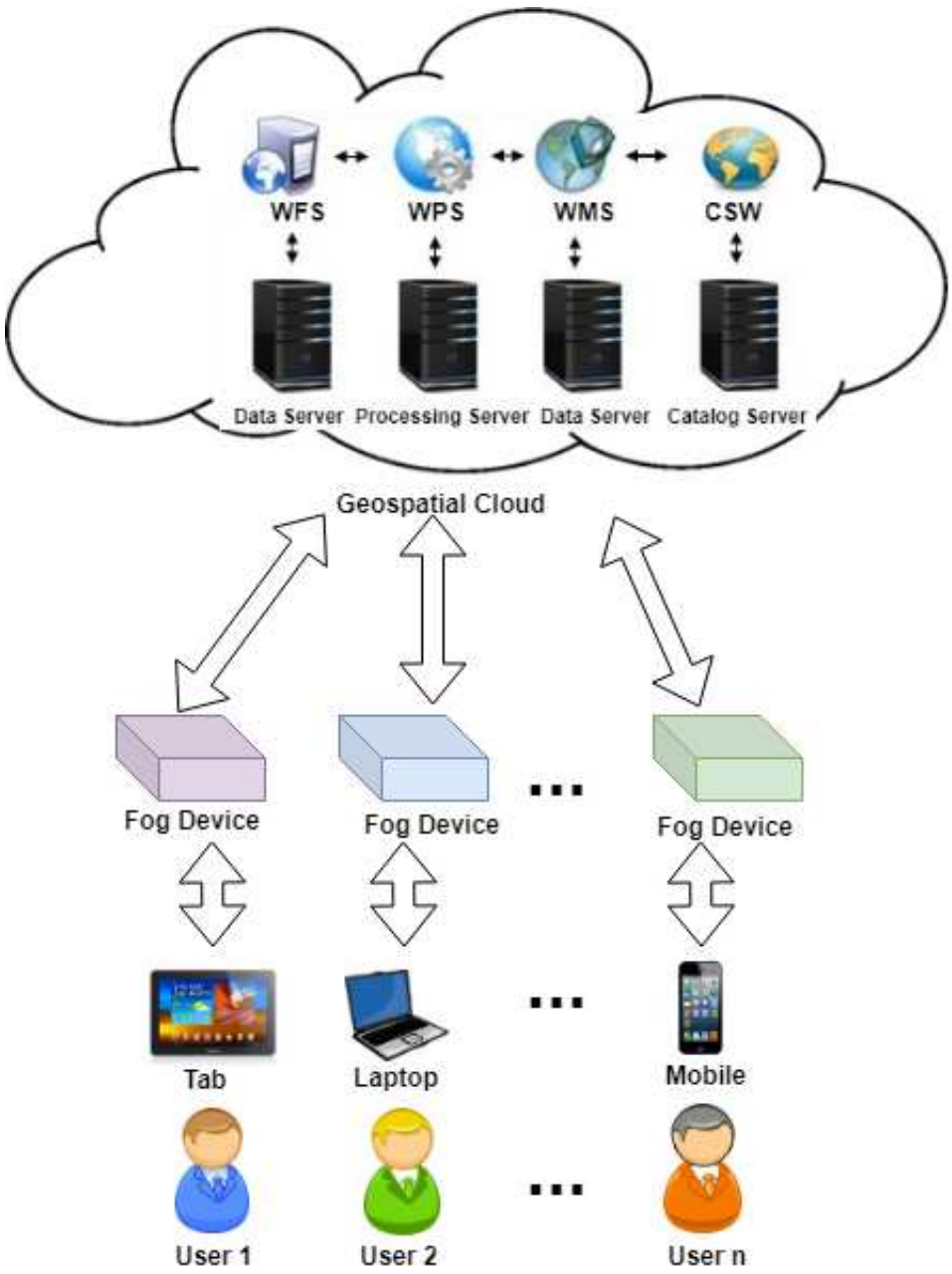


Figure 5.3: Architecture of Fog-based system for geospatial query resolution

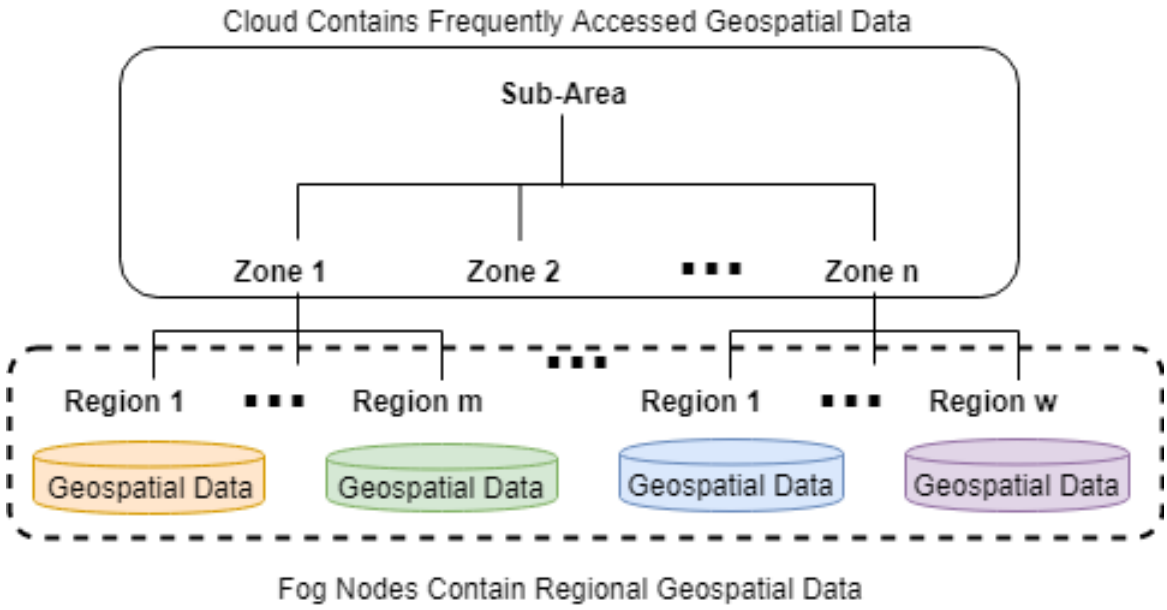


Figure 5.4: Hierarchy of proposed Spatio-Fog framework

of the regions to be frequently accessed, are stored in the cloud, which will not compromise with the delay over remote cloud if a query is generated regarding such regions other than the current region.

5.4.1 Query Generation from Mobile User

A mobile device located in a region is connected with a fog device. The mobile device generates a query and sends to the fog device.

Let a mobile device M currently located in region r_{z_a} is connected with a fog device f of the region and the geospatial data stored by f is $S_{r_{z_a}}$. M generates a query Q and sends to f.

5.4.2 Query Resolution by Fog Device

The fog device of a region contains the geospatial data of that region. The geospatial data storage and processing are performed by the fog device. When the fog device receives a query from the mobile device, first it verifies whether the query is related to the current region where both the mobile device and fog device are located. If the query is related to the current region, the fog device analyses its geospatial data and responds accordingly. Otherwise, the fog device forwards the query to the cloud servers. If the query Q received by the fog device f is related to $S_{r_{z_a}}$, f responds after

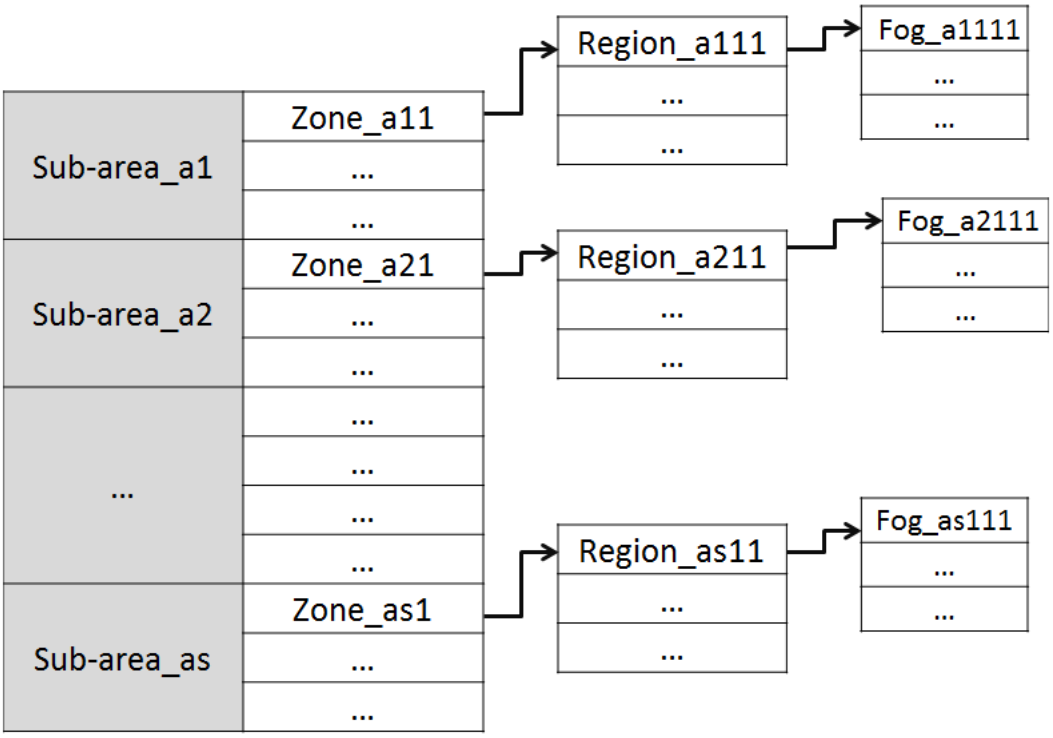


Figure 5.5: Area registry maintained inside the Cloud

analysing the data. Otherwise, f forwards Q to the cloud.

5.4.3 Cloud Servers

The fog devices are connected with the cloud servers. The cloud servers contain the spatial data of the regions for which the users frequently generate queries. When the cloud receives a geospatial query, it checks whether it has the intended spatial data to solve the query. If so, it analyses the data and sends the result. Otherwise, the cloud servers identify the region related to the spatial data and forward the request to the fog device of that region.

If the query Q received by the cloud is related to the spatial data which it contains, it resolves the query and sends the result to the fog device, which further forwards it to the mobile device. Otherwise, the cloud forwards Q to a fog device g which contains spatial data related to Q. Here, fog devices f and g belong to two different regions. To find out the sub-area, zone, and region of a query and then finding out the respective fog device, area registry is maintained inside the cloud servers. The registry is containing the sub-area IDs under the area and the zone IDs under the sub-areas. The zone IDs are referring to the region IDs of the respective zones.

The region IDs are referring to the fog device IDs containing the spatial data of the respective regions. The registry for area *a* is presented in Figure 5.5. There are *as* sub-areas present under the area *a*. Each sub-area has several zones, which are pointing to the regions under them. The regions are pointing to the fog devices containing the spatial data of those regions. By accessing the area registry, the cloud can find out the fog device containing the geospatial data related to a particular region.

5.4.4 Case Study

In section 5.3 we have discussed on the queries and the services for processing those queries. Now, the mobile user when generates such queries, then the proposed framework resolves them based on the location for which the query has been made. For resolving these queries, the WPS, WFS, and WMS services are used according to the type of operation required. Based on the location for which the query is made, the following three cases come into the scenario:

- Geospatial query resolution by fog device of the current region
- Geospatial query resolution using cloud servers
- Geospatial query resolution using fog device of another region

These three cases are discussed as follows.

Case 1: Geospatial query resolution by fog device of current region

In this case, the mobile device sends a query to the connected fog device of its region. The query is related to the spatial data of the current region. Thus the fog device analyses its spatial data and sends the result to the mobile device. The corresponding sequence diagram is presented in Figure 5.6.

Case 2: Geospatial query resolution using cloud servers

In this case, the mobile device sends a query to the connected fog device of its region. The query is related to the spatial data of the other region, whose data the cloud servers contain. The fog device forwards the query to the cloud servers. The cloud servers analyse the intended spatial data to resolve the query and send the result to the fog device. The fog device then sends the result to the mobile device. The corresponding sequence diagram is presented in Figure 5.7.

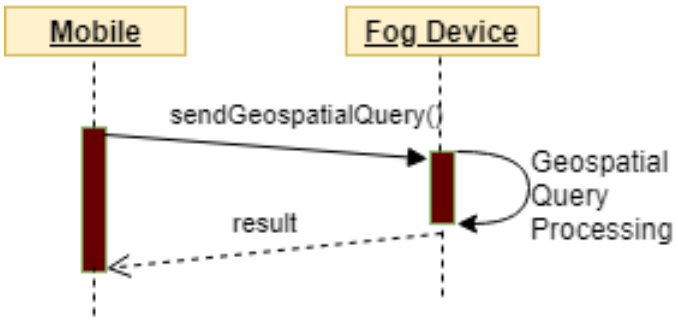


Figure 5.6: Case 1- sequence diagram of geospatial query resolution by Fog device of current region

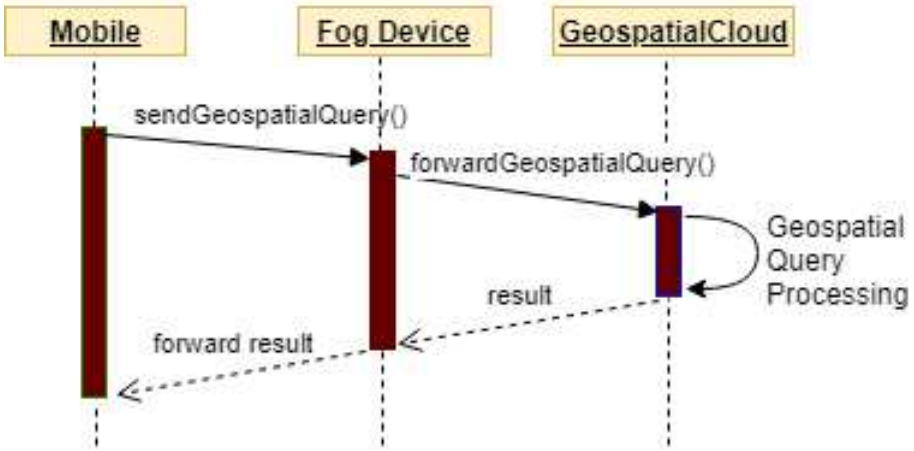


Figure 5.7: Case 2- sequence diagram of geospatial query resolution using Cloud servers

Case 3: Geospatial query resolution using fog device of another region

In this case, the mobile device sends a query to the connected fog device of its region. The query is related to the spatial data of the other region, whose data is not available inside the cloud servers. The fog device forwards the query to the cloud servers. The cloud accesses the area registry and finds out the fog device containing the corresponding spatial data. After that, the cloud forwards the request to that fog device along with the ID of the requesting fog device and the query ID. The corresponding fog device analyses its spatial data to resolve the query and sends the result to the requesting fog device directly. The fog device then sends the result to the mobile device. The corresponding sequence diagram is presented in Figure 5.8.

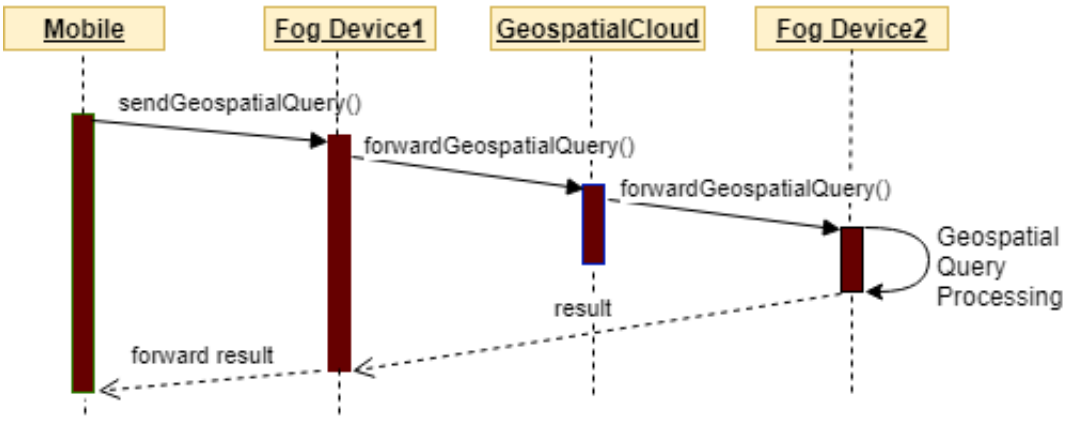


Figure 5.8: Case 3- sequence diagram of geospatial query resolution using Fog device of other region

5.4.5 Delay and Power Consumption in Query Resolution

The parameters used in delay and power calculation are defined in Table 5.2.

The sum of propagation delay, communication delay, processing delay and queuing delay is the total delay while resolving query generated from a mobile device.

In the first case, the propagation delay is (d_{prop1}/S_p) as the fog device of current region resolves the query. However, in the second case, the cloud resolves the query. Therefore, the propagation delay is (d_{prop2}/S_p). In the third case, the fog device of another region resolves the query. Hence, the propagation delay is (d_{prop3}/S_p). If all the three cases are considered with their probability of occurrences, then the propagation delay is given as,

$$D_p = (p_1 * d_{prop1} + p_2 * d_{prop2} + p_3 * d_{prop3})/S_p \tag{5.1}$$

In the first case, the fog device of current region resolves the query. Hence, for the first case, the uplink and downlink communication delays between mobile device and fog device are considered. The communication delay from mobile device and fog device is given as,

$$D_{up_1} = (D_{mf}/Up_{mf}) * (1 + fu_{mf}) \tag{5.2}$$

The communication delay from fog device to mobile device is given as,

$$D_{dw_1} = (D_{fm}/Dw_{mf}) * (1 + fd_{mf}) \tag{5.3}$$

In the second case, the cloud resolves the query. Hence, for the second case, the uplink and downlink communication delays between fog device and cloud are taken

Table 5.2: Notations used in delay and power calculation

Parameter	Definition
d_{prop1}	Distance to be covered during propagation while fog device of current region resolves query
d_{prop2}	Distance to be covered during propagation while cloud resolves query
d_{prop3}	Distance to be covered during propagation while fog device of other region resolves query
S_p	Propagation speed
D_f	Data amount processed for resolving query
S_f	Data processing speed of fog device
S_{cl}	Data processing speed of cloud
D_{mf}	Data amount transmission from mobile device to fog device while sending query
D_{fm}	Data amount transmission from fog device to mobile device while sending result
D_{fc}	Data amount transmission from fog device to cloud while sending query
D_{cfq}	Data amount transmission from cloud to fog device while sending query
D_{cf}	Data amount transmission from cloud to fog device while sending result
D_{ff}	Data amount transmission from serving fog device to requesting fog device while sending result
Up_{mf}	Data transmission rate from mobile device to fog device
Dw_{mf}	Data transmission rate from fog device to mobile device
Up_{fc}	Data transmission rate from fog device to cloud
Dw_{fc}	Data transmission rate from cloud to fog device
Dw_{ff}	Data transmission rate from serving to requesting fog device
fu_{mf}	Link failure rate from mobile device to fog device
fd_{mf}	Link failure rate from fog device to mobile device
fu_{fc}	Link failure rate from fog device to cloud
fd_{fc}	Link failure rate from cloud to fog device
fd_{ff}	Link failure rate from serving fog device to requesting fog device
D_w	Queuing delay
P_t	Power consumption of mobile device for data transmission per unit time
P_r	Power consumption of mobile device for data reception per unit time
P_m	Power consumption of mobile device in idle mode per unit time
p_1	Probability of case 1
p_2	Probability of case 2
p_3	Probability of case 3

into account along with the delays in the first case. The communication delay from fog device to cloud is given as,

$$D_{up_2} = (D_{fc}/Up_{fc}) * (1 + fu_{fc}) \tag{5.4}$$

The communication delay from cloud to fog device is given as,

$$D_{dw_2} = (D_{cf}/Dw_{fc}) * (1 + fd_{fc}) \tag{5.5}$$

In the third case, the fog device of another region resolves the query. Hence, for the third case, the uplink and downlink communication delays between two fog devices are taken into account along with the delays in the first case. As the fog device responds to the query using fog device of another region, the communication delay from requesting to serving fog device is given as,

$$D_{up_3} = ((D_{fc}/Up_{fc}) * (1 + fu_{fc})) + ((D_{cfq}/Dw_{fc}) * (1 + fd_{fc})) \tag{5.6}$$

In this case, the query is sent from the requesting fog device to the serving fog device through the cloud. However, after resolving the query the serving fog device directly delivers the result to the requesting fog device. Therefore, the communication delay from serving to requesting fog device is given as,

$$D_{dw_3} = (D_{ff}/Dw_{ff}) * (1 + fd_{ff}) \tag{5.7}$$

After considering the probabilities of occurrences of all the three cases, the uplink communication delay is therefore given as,

$$\begin{aligned} D_{up} &= p_1 * D_{up_1} + p_2 * (D_{up_1} + D_{up_2}) + p_3 * (D_{up_1} + D_{up_3}) \\ &= D_{up_1} + p_2 * D_{up_2} + p_3 * D_{up_3} \end{aligned} \tag{5.8}$$

where $p_1 + p_2 + p_3 = 1$. Accordingly, The downlink communication delay is given as,

$$\begin{aligned} D_{dw} &= p_1 * D_{dw_1} + p_2 * (D_{dw_1} + D_{dw_2}) + p_3 * (D_{dw_1} + D_{dw_3}) \\ &= D_{dw_1} + p_2 * D_{dw_2} + p_3 * D_{dw_3} \end{aligned} \tag{5.9}$$

The communication delay as a sum of the uplink and downlink communication delays, is therefore given as,

$$D_{com} = D_{up} + D_{dw} \tag{5.10}$$

In the first and third cases, fog device processes the query. The data processing delay of fog device is given as,

$$D_{procf} = D_f / S_f \tag{5.11}$$

In the second case, cloud processes the query. The data processing delay of cloud is given as,

$$D_{proccl} = D_f / S_{cl} \tag{5.12}$$

After considering the probabilties of occurrences of all the three cases, the processing delay is given as,

$$D_{proc} = p_1 * D_{procf} + p_2 * D_{proccl} + p_3 * D_{procf} \tag{5.13}$$

The total delay as the sum of the propagation, communication, processing and queuing delays is given as,

$$D_{tot} = D_p + D_{com} + D_{proc} + D_w \tag{5.14}$$

The power consumption of mobile device during propagation is given as,

$$P_p = P_m * D_p \tag{5.15}$$

The power consumption of mobile device during uplink communication is given as,

$$P_{mu} = P_t * D_{up_1} + p_2 * P_m * D_{up_2} + p_3 * P_m * D_{up_3} \tag{5.16}$$

The power consumption of mobile device during downlink communication is given as,

$$P_{md} = P_r * D_{dw_1} + p_2 * P_m * D_{dw_2} + p_3 * P_m * D_{dw_3} \tag{5.17}$$

Therefore, the power consumption of mobile device during communication is given as,

$$P_{mcom} = P_{mu} + P_{md} \tag{5.18}$$

The power consumption of mobile device during data processing by fog device or cloud, is given as,

$$P_{proc} = P_m * D_{proc} \tag{5.19}$$

The power consumption of mobile device during queuing period is given as,

$$P_{mqu} = P_m * D_w \tag{5.20}$$

Therefore the total power consumption of the mobile device as a sum of the power consumption during propagation, communication, query processing (inside the fog device/cloud) and queuing periods, is given as,

$$P_{tot} = P_p + P_{mcom} + P_{proc} + P_{mqu} \tag{5.21}$$

In the next section, using this mathematical model we have determined the delay in resolving a mobile user-generated query using the proposed Spatio-Fog framework, and the power consumption of the user device during query resolution period, and compared with the existing cloud-based query resolution framework. After that we have validated the theoretical results with the experimental results obtained by carrying out different types of query resolution using the proposed framework in the laboratory.

5.5 Performance Evaluation

In this section, we have performed theoretical analysis using MATLAB and experimental analysis using OpenStack Cloud and Google Cloud Platform. We have compared our framework with the existing query processing system in terms of delay and power consumption of the mobile device. We have also performed a comparative study between the theoretical and experimental analysis in this section.

5.5.1 Theoretical Analysis

In this section, the delay and power consumption during geospatial query resolution using proposed fog based system and existing cloud-based system are compared [3, 64, 183]. MATLAB 2015 is used for theoretical analysis. For theoretical analysis, the total amount of geospatial data is considered 1 to 10 TB. It is assumed that $p_1 < 1$, $p_2 < 1$, $p_3 < 1$ and $p_1 + p_2 + p_3 = 1$. For theoretical analysis the data amount transmitted in uplink and downlink is assumed 1.25 to 2.50 MB and 15 to 27.5 MB respectively, the uplink and downlink data transmission rate are assumed 5 to 10 Mbps and 10 to 30 Mbps respectively, and the power consumption of mobile device per unit time in transmit, receive and idle modes are assumed 0.01 to 0.1 W, 0.005 to 0.05 W, and 0.005 W respectively. Figure 5.9 shows the delay in query resolution using proposed fog based system Spatio-Fog, calculated using equation (14) and using existing cloud-based system [3, 64, 183]. The simulation results show that Spatio-Fog reduces the delay by approximately 47-83% than the existing system.

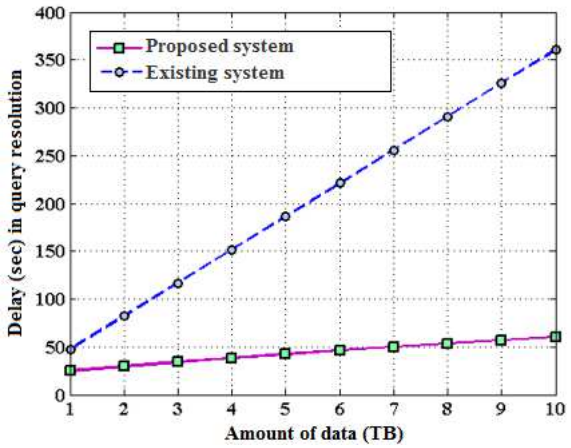


Figure 5.9: Delay in geospatial query resolution using proposed Spatio-Fog and existing system

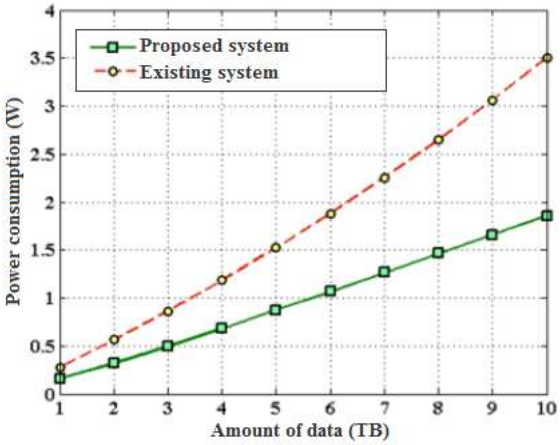


Figure 5.10: Power consumption by mobile device during geospatial query resolution using proposed Spatio-Fog and existing system

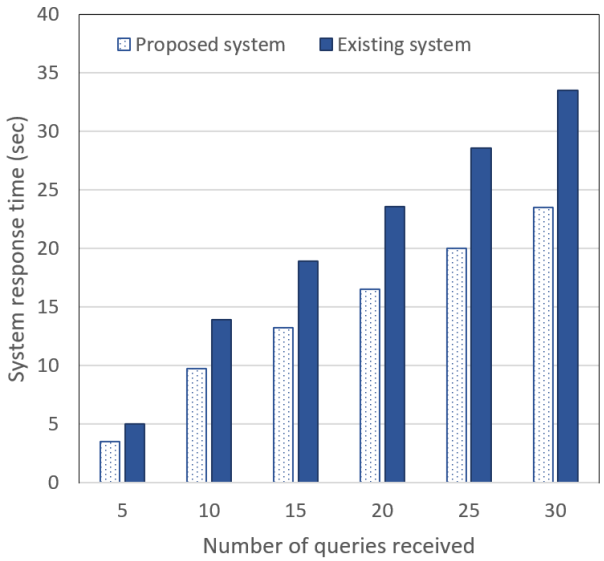


Figure 5.11: System response time during geospatial query resolution using proposed Spatio-Fog and existing system

Figure 5.10 shows the power consumption of mobile device during query resolution using proposed fog-based system Spatio-Fog, calculated using equation (21) and using existing cloud-based system [3, 64, 183]. The simulation results show that Spatio-Fog reduces the power by approximately 43-47% than the existing system.

Figure 5.11 shows system response time during query resolution using proposed fog-based system Spatio-Fog and using existing cloud-based system [3, 64, 183]. The simulation results show that Spatio-Fog reduces system response time by approximately 29-41% than the existing system.

If the query size and the size of the result to be sent to the device are large, then communication delay will be high. The processing delay depends on the volume of data has to be processed to resolve the query. Hence, the reduction in delay and power consumption while using the proposed architecture will differ for large, medium, or small data size.

5.5.2 Experiment Analysis

The configurations of the devices used in the experimental analysis are presented in Table 5.3. Two mobile phones are used, which generate queries. The first mobile phone has 3 GB RAM, 32 GB HDD, and Qualcomm MSM8940 Octa Core processor. The second mobile phone has 2 GB RAM, 8 GB HDD and 64-bit 1.2 GHz Qualcomm Snapdragon 410 Quad Core processor. The operating system of both the mobile phones is Android 6.0.1. Two laptops with 4 GB RAM, 250 GB HDD, and Intel Core i5 processor have been used as fog devices. The operating system of these two devices is Windows 7 Professional and Ubuntu. For local cloud environment, we have used OpenStack Cloud Platform of IIT Kharagpur, referred as Meghamala (MGL), where we have taken a regular VM (2 vCPU, 4 GB RAM). For remote cloud environment, we have used Google Cloud Platform (GCP), where we have created one Virtual Machine (VM) instance (zone: asia-south1-c). The machine type is n1-standard-1 (1 vCPU, 3.75 GB RAM). The protocols used are TCP, UDP, ICMP. The mobile devices are connected through Wi-Fi Access points. The data transmission rate is 10-50 Mbps.

Mobile phone 1 and mobile phone 2 are connected with fog device 2 and fog device 1 respectively. In this work, we have used the spatial reference system, EPSG:32645. We have considered the spatial data related to road, land area and railway track of Purulia, India, road network of Mumbai, India, road network of Delhi, India, and forest and road network of Raipur, India. Geospatial data are of two types: vector data and raster data. In this experiment, we have considered

Table 5.3: Configurations of devices used in experiment

Device	RAM	HDD	Processor	Operating system
Mobile phone 1	3 GB	32 GB	Qualcomm MSM8940 Octa Core	Android 6.0.1
Mobile phone 2	2 GB	8 GB	64-bit 1.2 GHz Qualcomm Snapdragon 410 Quad Core	Android 6.0.1
Fog device 1	4 GB	250 GB	Intel Core i5	Windows 7 Professional
Fog device 2	4 GB	250 GB	Intel Core i5	Ubuntu
VM instance in Google Cloud Platform (Remote Cloud referred as GCP)	3.75 GB	250 GB	1vCPU	Windows Server 2012 R2 Datacenter
IITKGP_regular VM OpenStack (Local Cloud referred as MGL)	4 GB	45 GB	2 vCPU	Ubuntu 14.04

vector data. Two mobile phones send queries related to geospatial data of Purulia, Delhi, Mumbai and Raipur. Fog device 1 and fog device 2 are containing the data of Purulia and Raipur respectively. The data of Mumbai and Delhi are stored in the cloud. The two mobile devices generate total six queries which come under the three case studies, discussed in section 5.4. Concerning the three cases, the analysis of generated queries from two mobile devices is discussed in the subsections 5.5.2, 5.5.2, and 5.5.2.

We have used QGIS (version 2.18.28) for analysing the spatial queries. The time complexity for buffer creation is $O(N)$, for insertion and retrieval of event points is $O(logN)$, for intersection is $O(N^2)$, where N is the number of edges for a geometric buffer. The results of data analysis are presented in Figures 5.12, 5.13, 5.14, 5.15, 5.16 and 5.17. The total delay in query resolution are presented in Table 5.4. In this case we have measured the round trip delay (difference between the time stamp of sending request and receiving response). The power consumption of the mobile phones during this period is presented in Table 5.4. To determine the power consumption we have multiplied the measured delay with the power consumption of mobile device per unit time. The results are compared with query resolution using the existing remote cloud-based system [64, 183], and local cloud-based system.

Experimental results of query analysis by fog device of current region

The user of mobile phone 1 generates the first query, where he asks for an information related to Raipur, and the query is sent to Fog device 2, with which it is connected. As fog device 2 is containing the geospatial data of Raipur, it responds. The user of mobile phone 2 generates the second query, where he asks for an information related to Purulia, and the query is sent to Fog device 1, with which it is connected. As fog device 1 is containing the geospatial data of Purulia, it responds.

Result of query 1 resolution: In Figure 5.12, the result of query 1 is displayed. The query is to find the road adjacent to the forest of Raipur. The forest area data and road network data of Raipur are analysed by fog device 2 to resolve the query. The following three geospatial services used to resolve this geospatial query:

- WPS BufferFeatureCollection service is for 1 km buffer creation around Raipur forest shape file.
- WPS IntersectionFeatureCollection service is for the overlap of Raipur road and buffered forest shape file.
- WMS GetMap service is used to display the resultant map of query 1.

Table 5.4: Delay and power consumption results obtained from experiment

No.	Geospatial Query	Transmitted data between consecutive nodes	Delay	Power consumption of mobile device	Reduction using proposed framework
1	SELECT road_id FROM Raipur WHERE Overlap(road.shape, Buffer (forest.Shape,1));	1.98 MB	1.72 sec using Spatio-Fog, 4.71 sec using GCP, 3.31 sec using MGL	0.19 W using Spatio-Fog, 0.52 W using GCP, 0.365 W using MGL	Delay: 63% than GCP, 48% than MGL, Power: 63% than GCP, 48% than MGL
2	SELECT road_name FROM Purulia WHERE road = 'High Road' ORDER BY area desc;	1.56 MB	1.13 sec using Spatio-Fog, 3.61 sec using GCP, 2.21 sec using MGL	0.124 W using Spatio-Fog, 0.397 W using GCP, 0.243 W using MGL	Delay: 71% than GCP, 49% than MGL, Power: 68% than GCP, 49% than MGL
3	SELECT road_name FROM Delhi WHERE road_type = 'One_Way';	2.28 MB	2.85 sec using Spatio-Fog, 4.91 sec using GCP, 4.45 sec using MGL,	0.31 W using Spatio-Fog, 0.54 W using GCP, 0.49 W using MGL	Delay: 41% than GCP, 35% than MGL, Power: 42% than GCP, 36% than MGL
4	SELECT road_name FROM Mumbai WHERE road_type = 'One_Way';	2.32 MB	2.91 sec using Spatio-Fog, 5.22 sec using GCP, 4.45 sec using MGL,	0.32 W using Spatio-Fog, 0.57 W using GCP, 0.49 W using MGL	Delay: 44% than GCP, 35% than MGL, Power: 43% than GCP, 34% than MGL
5	SELECT point FROM Purulia.rail Rl, Purulia.road Rd WHERE Cross(Rl.Shape, Rd.Shape)=1 ORDER BY area desc;	2.12 MB	2.76 sec using Spatio-Fog, 3.94 sec using GCP, 3.32 sec using MGL	0.31 W using Spatio-Fog, 0.45 W using GCP, 0.38 W using MGL	Delay: 30% than GCP, 18% than MGL, Power: 31% than GCP, 17% than MGL
6	SELECT area_name FROM Purulia WHERE *area* > 50 and road = 'High Road' and Overlap (road.shape, Buffer(area.shape,1)) ORDER BY area desc;	3.15 MB	5.25 sec using Spatio-Fog, 7.66 sec using GCP, 6.35 sec using MGL	0.577 W using Spatio-Fog, 0.842 W using GCP, 0.69 W using MGL	Delay: 31% than GCP, 17% than MGL, Power: 31% than GCP, 16% than MGL

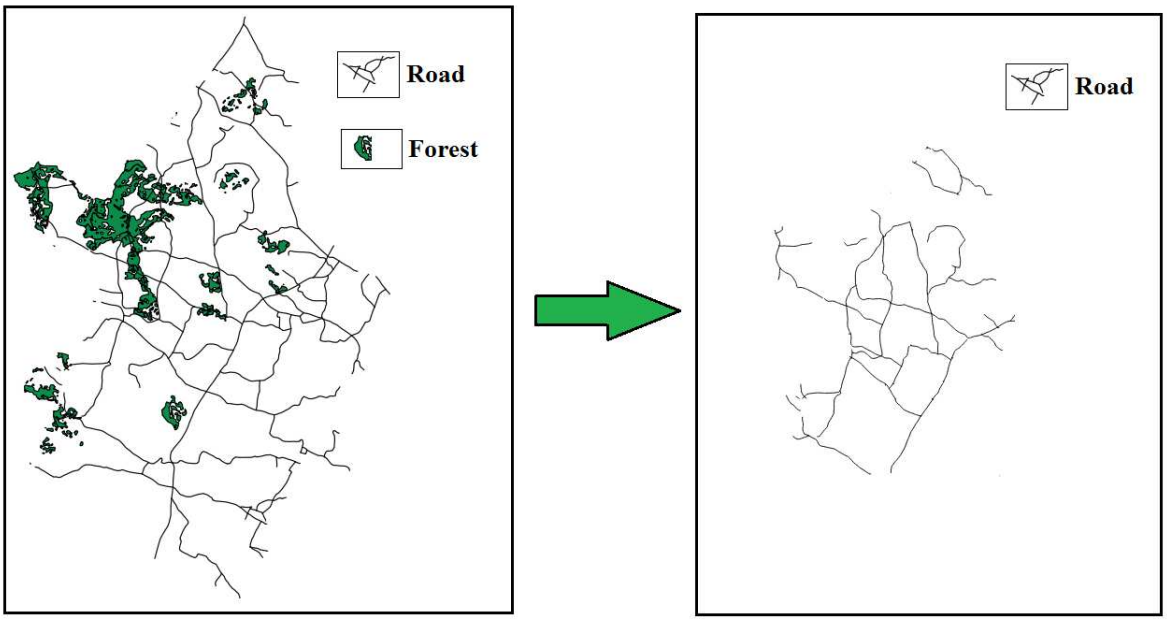


Figure 5.12: Result of geospatial query 1 (connecting roads to forest in Raipur)

The amount of data transmission, delay and power consumption of the user device are presented in Table 5.4. The amount of data transmission between consecutive nodes, in this case, is 1.98 MB. The delay in query resolution using the proposed model, only GCP (remote cloud VM) and only MGL (local cloud VM) are 1.72 sec, 4.71 sec, and 3.31 sec respectively. The power consumptions of the user device during this period are 0.19 W, 0.52 W, and 0.365 W respectively for query resolution using the proposed model, only GCP, and only MGL. The experimental results show that the proposed model reduces the delay in query resolution by 63% and 48% approximately than only GCP and only MGL respectively. The experimental results also show that using the proposed model the power consumption of the user device is reduced by 63% and 48% approximately than only GCP and only MGL respectively.

Result of query 2 resolution: In Figure 5.13, the result of query 2 is displayed. The query is to find the high roads in Purulia. The spatial data of road of Purulia is analysed by fog device 1 to resolve the query. The following two geospatial services are used to resolve this geospatial query:

- WFS getFeature service is used for filtering the 'High Roads' from Purulia road network.
- WMS GetMap service is used to display the 'High Roads' map of Purulia.

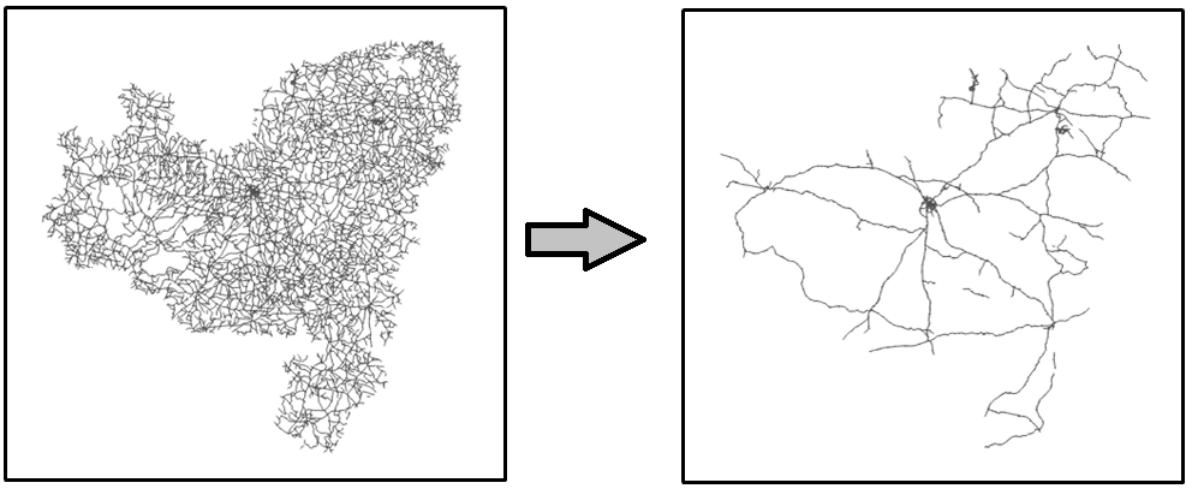


Figure 5.13: Result of geospatial query 2 (high roads in Purulia)

The amount of data transmission, delay, and power consumption of the user device are presented in Table 5.4. The amount of data transmission between consecutive nodes, in this case, is 1.56 MB. The delay in query resolution using the proposed model, only GCP (remote cloud VM) and only MGL (local cloud VM) are 1.13 sec, 3.61 sec, and 2.21 sec respectively. The power consumption of the user device during this period are 0.124 W, 0.397 W, and 0.243 W respectively for query resolution using the proposed model, only GCP, and only MGL. The experimental results show that the use of the proposed model reduces the delay in query resolution by 71% and 49% approximately than only GCP and only MGL respectively. The experimental results also show that using the proposed model, the power consumption of the user device is reduced by 68% and 49% approximately than only GCP, and only MGL respectively. Hence, it is observed that the proposed framework has ~ 60% less power consumption of user device and ~ 60% less delay than the existing remote cloud only system [64, 183].

Experimental results of query analysis by cloud

The user of mobile phone 1 generates the third query, where he asks for an information related to Delhi, and the query is sent to Fog device 2. Nevertheless, fog device 2 does not have the geospatial data of Delhi. Hence, it forwards the request to the cloud. The cloud has the geospatial data of Delhi. Thus, it processes the query and then sends the result to fog device 2. Fog device 2 forwards the result to mobile phone 1. The user of mobile phone 2 generates the fourth query, where he asks for

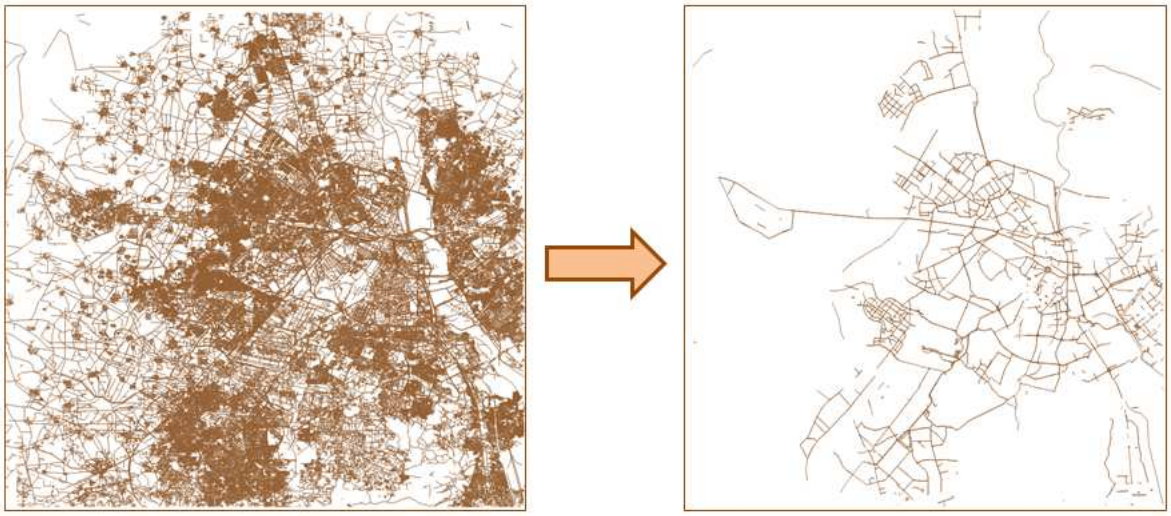


Figure 5.14: Result of geospatial query 3 (one-way roads in Delhi)

an information related to Mumbai, and the query is sent to Fog device 1. However, fog device 1 is not containing the geospatial data of Mumbai. Hence, it forwards the request to the cloud. The cloud has the geospatial data of Mumbai, and it sends the result to fog device 1 after processing the query. Fog device 1 forwards the result to mobile phone 2.

Result of query 3 resolution: In Figure 5.14, the result of query 3 is displayed. The query is to find one-way roads in Delhi. The road network data of Delhi is analysed by the cloud to resolve the query. The following two geospatial services are used to resolve this geospatial query:

- WFS getFeature service is used to filter out the 'One_Way' roads from the Delhi road network.
- WMS GetMap service is used to exhibit the query 3 result.

The amount of data transmission, delay, and power consumption of the user device are presented in Table 5.4. The amount of data transmission between consecutive nodes, in this case, is 2.28 MB. The delay in query resolution using the proposed model, only GCP (remote cloud VM), and only MGL (local cloud VM) are 2.85 sec, 4.91 sec, and 4.45 sec respectively. The power consumption of the user device during this period are 0.31 W, 0.54 W, and 0.49 W respectively for query resolution using the proposed model, only GCP, and only MGL. The experimental results show that the use of the proposed model reduces the delay in query resolution by 41% and

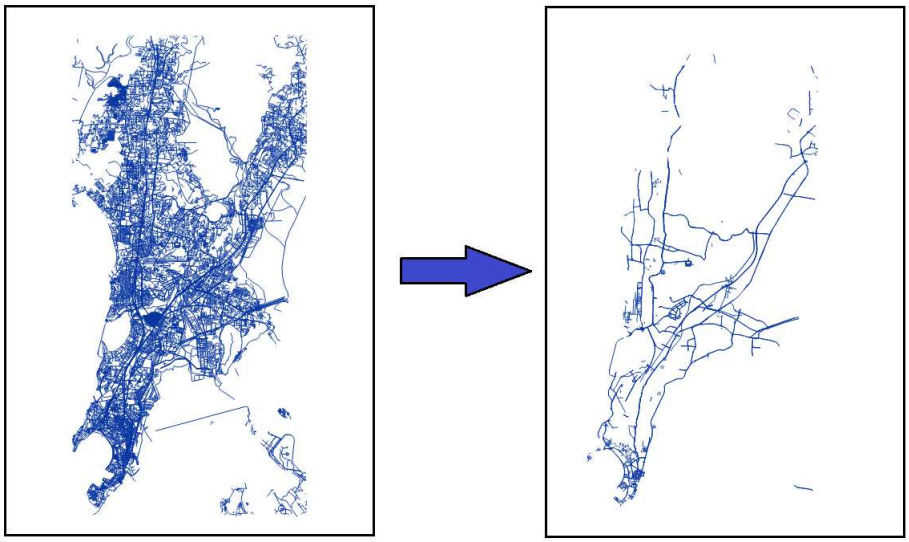


Figure 5.15: Result of geospatial query 4 (one-way roads in Mumbai)

35% approximately than only GCP and only MGL respectively. The experimental results also show that using the proposed model, the power consumption of the user device is reduced by 42% and 36% approximately than only GCP, and only MGL respectively.

Result of query 4 resolution: In Figure 5.15, the result of query 4 is displayed. The query is to find one-way roads in Mumbai. The road network data of Mumbai is analysed by cloud to resolve the query. The following two geospatial services are used to resolve this geospatial query:

- WFS getFeature service is used to filter the 'One_Way' roads from the Mumbai road network.
- For displaying the result of query 4, WMS GetMap service is used.

The amount of data transmission, delay, and power consumption of the user device are presented in Table 5.4. The amount of data transmission between consecutive nodes, in this case, is 2.32 MB. The delay in query resolution using the proposed model, only GCP (remote cloud VM) and only MGL (local cloud VM) are 2.91 sec, 5.22 sec, and 4.45 sec respectively. The power consumptions of the user device during this period are 0.32 W, 0.57 W, and 0.49 W respectively for query resolution using the proposed model, only GCP, and only MGL. The experimental results show that the use of the proposed model reduces the delay in query resolution by 44% and 35% approximately than only GCP and only MGL respectively. The experimental

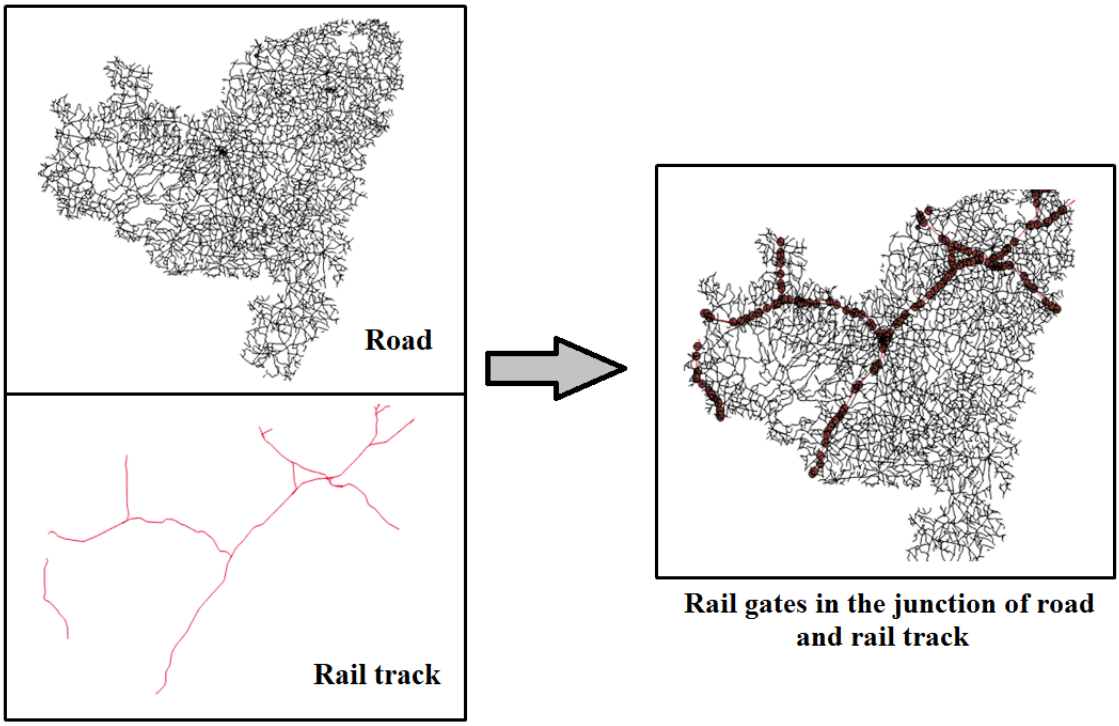


Figure 5.16: Result of geospatial query 5 (rail gates in the junction of road and rail track in Purulia)

results also show that using the proposed model, the power consumption of the user device is reduced by 43% and 34% approximately than only GCP, and only MGL respectively. Hence, the proposed framework has ~ 40% less power consumption of user device and ~ 40% less delay than the existing remote cloud only system [64, 183].

Experimental results of query analysis by fog device of another region

The user of mobile phone 1 generates the last two queries, where he asks for information related to Purulia, and the query is sent to Fog device 2. However, fog device 2 is not containing the geospatial data of Purulia. Hence, it forwards the fifth and sixth queries to the cloud. But the cloud is also not containing the data of Purulia, hence forwards it to fog device 1 that is containing the data of Purulia. Fog device 1 then processes the fifth and sixth queries and sends the result directly to fog device 2. Fog device 2 forwards the result to mobile phone 1.

Result of query 5 resolution: In Figure 5.16, the result of query 5 is displayed. The query is to find the rail gates in the junction of road and rail track in Purulia. The

spatial data of rail track and road of Purulia are analysed by fog device 1 to resolve the query. The following two geospatial services are used to resolve this geospatial query:

- WPS LineIntersectionFeature service is used for performing the cross operation on the road network and rail tracks of Purulia.
- WMS GetMap service is used to display the final map of query 5.

The amount of data transmission, delay, and power consumption of the user device are presented in Table 5.4. The amount of data transmission between consecutive nodes, in this case, is 2.12 MB. The delay in query resolution using the proposed model, only GCP (remote cloud VM) and only MGL (local cloud VM) are 2.76 sec, 3.94 sec, and 3.32 sec respectively. The power consumption of the user device during this period are 0.31 W, 0.45 W, and 0.38 W respectively for query resolution using the proposed model, only GCP, and only MGL. The experimental results show that the use of the proposed model reduces the delay in query resolution by 30% and 18% approximately than only GCP and only MGL respectively. The experimental results also show that using the proposed model, the power consumption of the user device is reduced by 31% and 17% approximately than only GCP and only MGL respectively.

Result of query 6 resolution: In Figure 5.17, the result of query 6 is displayed. The query is to find wastelands with an area greater than 50 acres within the distance of 1 km from the high road in Purulia. The spatial data of road and land use land cover (LULC) of Purulia are analysed. The amount of data transmission, delay and power consumption of the user device are presented in Table 5.4. The following four geospatial services are used to resolve this geospatial query:

- WFS getFeature service is used twice to filter the 'High Road' roads from Purulia road network and to filter areas which are greater than 50 acres from Purulia LULC.
- To create 1 km buffer of each filtered area using WPS BufferFeatureCollection service.
- WPS IntersectionFeatureCollection service is used to make intersection Land buffer with filtered Roads.
- Finally, WMS GetMap service is used to display the resultant map of the query 6.

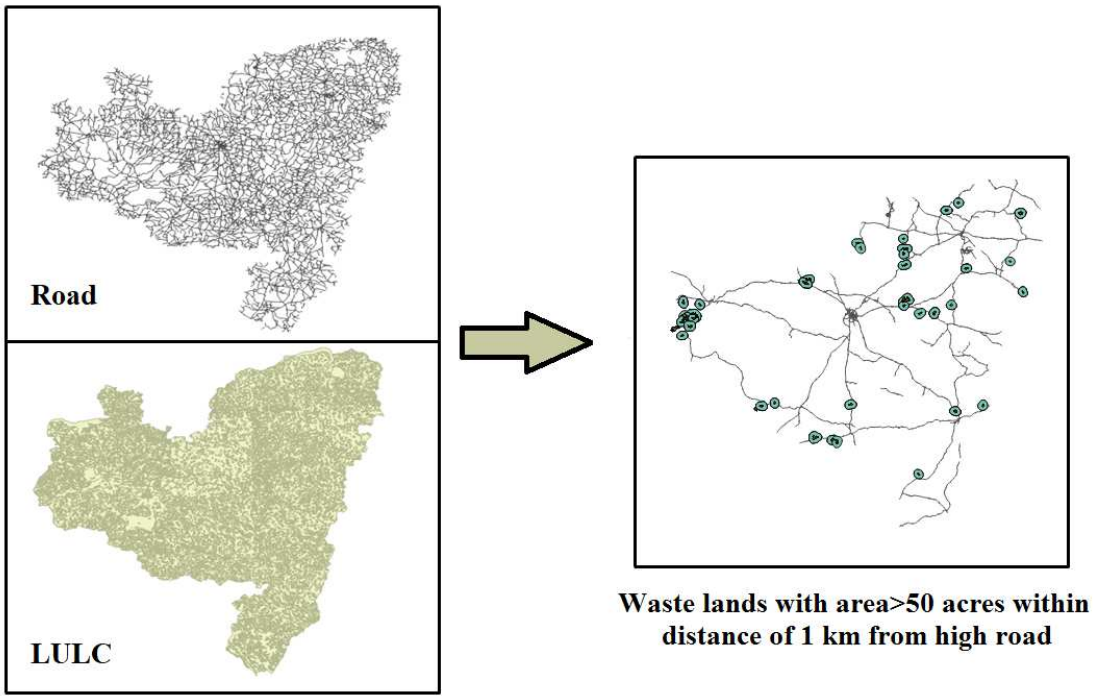


Figure 5.17: Result of geospatial query 6 (waste lands with area of greater than 50 acres within the distance of 1 km from high road in Purulia)

The amount of data transmission between consecutive nodes in this case is 3.15 MB. The delay in query resolution using the proposed model, only GCP (remote cloud VM) and only MGL (local cloud VM) are 5.25 sec, 7.66 sec and 6.35 sec respectively. The power consumption of the user device during this period are 0.577 W, 0.842 W and 0.69 W respectively for query resolution using the proposed model, only GCP and only MGL. The experimental results show that use of the proposed model reduces the delay in query resolution by 31% and 17% approximately than using only GCP and only MGL respectively. The experimental results also show that using fog device the power consumption of the user device is reduced by 31% and 16% approximately than using only GCP and only MGL respectively.

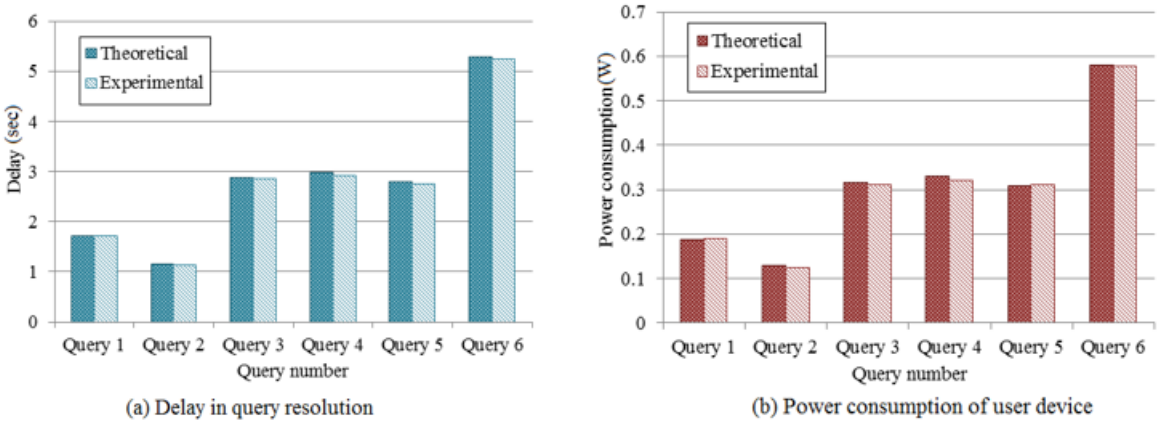
Hence, from this two experimental studies it is observed, the proposed framework has ~ 30% less power consumption and ~ 30% less delay than the existing remote cloud only system [64, 183].

The experimental results in Table 5.4 illustrate that the use of the proposed model reduces delay and power than existing cloud-based system. In the first two experimental studies the fog device connected with the requesting mobile phone is containing data of the region for which the queries are made. Hence, the reduction

in delay and power consumption of user device is ~ 60% than the existing remote cloud-based system [64, 183]. In the next two experimental studies the fog device connected with the requesting mobile phone is not containing data of the region for which the queries are made, whereas the cloud is containing the data. Hence, the communication cost in terms of delay and power consumption is same as in the existing cloud-based system. However, in the existing system the cloud has to access a huge volume of data to respond to the query, as the spatial data of all the regions is contained by the cloud. As a result the computational cost is higher than the proposed system where cloud is containing only the frequently accessed spatial data. Consequently the total cost (sum of communication and computation) is higher in the existing system. It is observed that use of cloud for query resolution in case of the proposed Spatio-Fog architecture achieves ~ 40% less delay and power consumption of user device than the existing remote cloud-based system [64, 183]. In the last two experimental studies the third case is considered i.e. the use of another fog device for query resolution. In this case the proposed model has higher communication cost than the existing cloud-based system. Nevertheless, the computational cost in case of the remote cloud-based system is much higher (as discussed earlier) than the proposed model. As a result the total cost becomes less in the proposed framework. It is observed that use of fog device of another region for query resolution in case of the proposed Spatio-Fog architecture achieves ~ 30% less delay and power consumption of user device than the existing remote cloud-based system [64, 183]. Hence, from the six experimental studies, it is observed that the proposed system provides 30% – 60% less power consumption of user device and 30% – 60% less delay than the existing system.

5.5.3 Comparison study between theoretical and experimental analysis

In the theoretical analysis, we have considered large data size, whereas in the experimental analysis, we have a comparatively small amount of data. Hence the results differ. However, we have performed the theoretical analysis also for the six queries discussed above. Using the mathematical model, the delay in query resolution and power consumption of user device during that period have been calculated and compared with the experimental results, which is graphically presented in Fig.5.18. It is observed from Fig.5.18 that the theoretical and corresponding experimental results are approximately the same. Experimentally it is already noted that the proposed



(a) Delay in query resolution

(b) Power consumption of user device

Figure 5.18: Comparison of theoretical and experimental results of query resolution

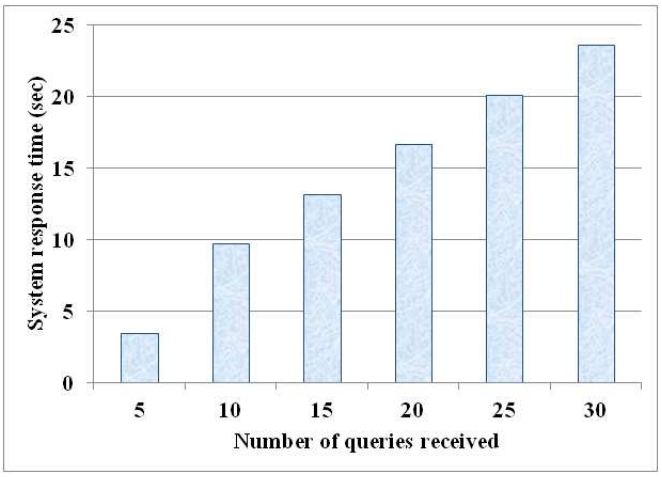


Figure 5.19: System response time in proposed Spatio-Fog system

framework reduces the power consumption and delay than the existing frameworks. Hence, it is concluded that the proposed fog computing system Spatio-Fog is a low power, i.e., green and delay-sensitive system.

5.5.4 System Response Time

System response time refers to the time difference between receiving a request and sending the corresponding response by a system i.e. the sum of waiting time and service time is the response time [3]. If a system receives N queries and the response time of a query Q is T_Q, then the average system response time is given as, $RT_N = \frac{\sum_{Q=1}^{N} T_Q}{N}$. The average system response time with respect to the number of queries received are presented in Figure 5.19. From the figure it is observed that the average system response time for the proposed Spatio-Fog system is < 25 sec for the number

of received queries 5-30.

5.6 Summary

In this chapter, a fog computing based architecture namely Spatio-Fog has been proposed for geospatial query resolution. The fog devices of different regions contain geospatial data of the respective regions. When a query is received from a mobile device regarding the current region, the fog device resolves the query after analysing the data and responds to the mobile device. Otherwise, if the query is regarding other regions, the fog device responds using cloud servers or fog device of the corresponding region. Theoretical analysis shows that the proposed framework reduces the power consumption and delay by approximately 43-47% and 47-83% respectively than the existing system. The experimental results illustrate that the proposed framework reduces the power consumption and delay by 30-60% approximately than the existing query resolution system. Thus, we conclude that the proposed framework is a green and delay aware framework.

Chapter 6

Healthcare Application of Geospatial Query in Edge-Fog-Cloud Environment

Internet of things (IoT) has a pivotal role in developing intelligent and computational solutions to facilitate varied real-life applications. To execute high-end computations and data analytics, IoT and cloud-based solutions play the most significant role. However, frequent communication with long distant cloud servers is not a delay-aware and energy-efficient solution while providing time-critical applications such as healthcare. The hierarchical and collaborative architecture with cloud, fog, edge, IoT, computing overcomes the issues by providing computing and storage services at the edge of the network. Further, the use of a cloud-only setup increases the overall energy consumption of the cloud datacenters and emits a huge volume of greenhouse gases. This chapter explores the possibilities and opportunities of integrating cloud technology with fog and edge-based computing to provide healthcare services to users in exigency. Here, we propose an end-to-end framework, named RESCUE, which has four layers, namely, cloud, fog, edge and IoT. This framework has an efficient spatio-temporal data analytics module for efficient information sharing, spatio-temporal data analysis to predict path for users to reach the destination (say, healthcare center or relief camps) with minimum delay in the time of exigency (say, natural disaster). This module analyzes the collected information through crowd-sourcing and assists the user by extracting optimal path post-disaster when many regions are non-reachable. The framework is deployed and evaluated using real-life datasets. The experimental and simulation results outperform the baselines to a significant margin in terms of accuracy, delay, and power consumption, and green service provisioning is achieved.

6.1 Introduction

Internet of Things (IoT) has manifested a dramatic revolution in all spheres of human lives by connecting billions of devices. It is estimated that by the end of 2025, more than 75 billion IoT devices will be connected to the web[1]. The proliferation of IoT paradigm has a significant impact in different industries [203] and facilitated varied applications, such as time-critical applications, like healthcare, smart city, smart home, agriculture[204] etc. With the rapid development of IoT and other sensor technologies, this IoT paradigm has created new domains of research, namely, *Internet of Health Things (IoHT)* [132, 205], *Internet of Spatial Things (IoST)* [206], *Industrial Internet of Things (IIoT)* [207, 208], *Internet of Military Things (IoMT)* [209, 210]. The IoT devices need to send data to cloud servers frequently for processing and analysing the accumulated data. However, this increases the delay, therefore affects the Quality of Service (QoS). Here, edge or fog nodes [56] extend the functionality of cloud computing by processing, analysing, and storing the information at the edge of the network.

In the present decade, several IoT devices such as Raspberry Pi[2], SmartThings Hub[3] facilitate temporary storage, limited computation capability, and memory resources along with the conventional end-to-end connectivity *anywhere and everywhere*. These promising features have a significant impact on any large scale IoT deployment[211], like smart city, smart healthcare, etc. Further, the integration of edge or fog[4] nodes helps in taking an adaptive and dynamic decision based on the sudden changes of the environment, and improves the efficacy of the IoT system. However, the computational power of these IoT devices is not sufficient for large-scale and compute-intensive analytics. Cloud computing is the only feasible solution where the processing is carried out in the cloud data centers. Nevertheless, the enormous amount of interconnected IoT devices generate a massive volume of data to be handled in the cloud server itself, and the cloud datacenters emit an enormous amount of greenhouse gases (due to the high energy consumption) taking a deep toll on the surroundings. On the other side, it is also observed that time-critical applications, such as healthcare, evacuation system, smart traffic monitoring, or defense applications, need real-time and latency-aware decision modules. Frequent communications with distant cloud servers increase the delay and may be fatal for

[1] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/
[2] https://www.raspberrypi.org/
[3] http://www.smartthings.com/
[4] Fog (From cOre to edGe) term was coined in 2012 by CISCO

many cases. Here, the fog or edge nodes can be made *intelligent* enough to analyse and adapt timely measures to reduce the intervention of cloud servers at each time. While fog or edge computing is not a replacement for cloud computing, the magnificent integration of these two booming technologies can efficiently facilitate delay, energy-awareness, and real-time applications. In this work, we leverage the functionality of both cloud and fog/ edge computing to provide timely assistance to users in the time of emergency. It is achieved by analysing heterogeneous data sources, namely, health parameters, contextual information (mobility, environment temperature, air pressure, etc.), and real-time information (crowd-sourcing data). The word 'Green' refers to low power; a system with low power or low energy consumption can be referred to as a green system. Due to the enormous volume of data analysis and transmission, accessing various applications through a mobile device has caused a tremendous amount of energy consumption not only by the cloud servers, network but also by mobile devices. Thus, low power, i.e., green service provisioning, has become a significant challenge. The work also mathematically formulates the power consumption, latency, and compares with the baseline methods to prove the eco-friendliness of the proposed framework in terms of low power and low latency.

In recent times, there is a growing need to analyse spatio-temporal datasets to extract meaningful information and provide location-aware services, such as trip-planning, weather forecasting, and even health management. From its inception, *spatio-temporal data mining* has shown a significant impact on varied aspects of our lives. For instance, it was found in a spatio-temporal data analytics [212] that the source of Cholera was public pumps and transmitted through contaminated drinking water. The finding was immensely helpful in combating the spread of Cholera. To this end, the Internet of Spatial Things (IoST) combines IoT with spatial context [206], where the location information of the objects plays an important role. Our framework aims to provide proper assistance to users when emergencies occur, such as disaster or health emergencies. RESCUE assists users by finding a path to reach healthcare centers or other places post-disaster situations or when the patient's health status is deteriorating. In the latter case, the cloud sends the alert to nearby fog nodes, and the fog nodes inform the ambulance service and nearest healthcare center as a preventive measure. The ambulance's route or path is predicted by the cloud server, such that the ambulance can reach in minimum delay avoiding any fatal condition. Therefore, an efficient module for analysing a massive amount of real-time mobility and road-information is required to assist the users in the time of

emergency.

6.1.1 Motivations and Challenges

There are varied real-life applications which can be facilitated from this framework. Here, we have considered an exigency situation (say, a natural disaster like super-cyclone) when substantial losses occur to human lives and public infrastructures like housing complex, roads, electric poles, etc. While the normal lives of people are disrupted, getting proper healthcare facility becomes a challenging issue. For instance, healthcare centers can not be reached due to the inundation of the roads. Moreover, any emergency or disaster planning requires seamless information exchange and updating information about the affected regions and demands of the people (like healthcare or food facilities). *In this work, the proposed framework, RESCUE, aims to provide a better management framework in terms of providing preliminary health checkups, information collection, and sharing mechanisms and finding paths to nearby healthcare centers while several regions are not-reachable due to the damage of the disaster*. It helps in proper post-disaster planning and improves overall urban sustainability and resilience. There are few challenges to provide such facilities in the time of emergency, such as,

1. How proper information about the affected regions can be accumulated quickly to take the recovery steps?

2. How these massive amounts of information can be stored, managed and analysed?

3. How to deploy a delay-aware system to assist users in the time of healthcare emergencies, when most of the roads and regions are affected and non-reachable?

4. How can we deploy an energy-efficient framework which provides all of these services?

It may be noted that the cloud paradigm provides the capability of storing, managing, and analysing a massive volume of data. Still, frequent communication with the cloud servers adds more delay and requires more energy consumption. All of these issues need to be addressed and efficiently resolved to provide adequate humanitarian relief and a sustainable environment.

6.1.2 Contributions

In this direction, the key contributions of this chapter are summarized as follows:

- We develop a hierarchical model that captures and accumulates data from heterogeneous IoT devices and performs preliminary data analysis in the edge of the network (i.e., in Fog nodes) to reduce the communication with the distant cloud servers. Further, the local data is stored in the fog nodes' temporary storage, and only the required information is sent to the cloud servers. The framework leverages the VGI (volunteered geographical information) to accumulate information about the affected regions or any events in the surroundings to take countermeasures.

- We consider healthcare service as a prototype for this model, where mobility is an important aspect. RESCUE is conducive of analysing the users' mobility pattern, present road-conditions, and finds an appropriate, less time-consuming path to reach the destination. The model utilizes the *autoencoder* and *markov decision process* to model and predict path to users in less time. It may be noted that the framework is flexible enough to find out the paths from source to destination, which requires less fuel consumption by computing the distance of all possible routes efficiently, leading to fewer carbon footprints.

- We also provide a geospatial query processing service, where the user can get the required emergency service information in less execution time as all local geospatial data are stored nearby different fog nodes in a distributed fashion.

- The chapter performs extensive simulation-based analysis using iFogsim and real-time data analysis in Google Cloud Platform (GCP). The experiment results show that our proposed model decreases up to 81% for the indoor user device's and up to 80% for the outdoor user device's power consumption and reduces the carbon footprints of the IoT, Fog devices, and Google cloud server, which moves a step towards green environment. The latency in healthcare service provisioning and finding routes to assist users is reduced up to 55% for the indoor user and up to 51% for the outdoor user in our proposed framework compared to cloud-only approaches.

The rest of this chapter is arranged as follows. Section 6.2 represents the related works of the topic. Section 6.3 elaborates on our proposed RESCUE architecture with mobility and geospatial health query analysis. The latency and power calculations

of user devices have been discussed in Section 6.4. In Section 6.5, the performance of our proposed architecture is measured with experimental setup details. The last section summarises the chapter.

6.2 Related Work

Over a decade, several techniques have been adopted to reduce the carbon footprint and make the computing paradigm green. We discuss a few existing research works in this domain of interest.

Albreem et al.[213] surveyed the existing green IoT research. They categorise the available green initiative into three parts. First part is working over Radio-frequency Identification (RFID) tags [214–216]. Second is making the sensor network energy efficient with different types of routing algorithms [217–219], clustering schemes [220, 221]. The third is green Internet technology with hardware and software solutions for different types of services. A deployment scheme has been proposed by [222]. Two types of nodes are considered sensing nodes and relay nodes. Traffic loads are distributed from the sensing node to the relay node as it has direct communications among each other. It reduces the battery power consumption of nodes and increases the overall network lifetime.

Bharti et al. [223] present a framework to recognize and classify complex activities at home using wearable devices. In order to reduce power consumption due to the continuous tracking of mobile devices, a method named *HARKE* has been proposed in [224]. Barik et al. [225] have proposed an ontology-based solution combined with statistical inferencing to recognize complex activities. On the other side, volunteered geographical information (VGI) provides a new opportunity for a better healthcare system by collecting a time-sensitive dataset from a huge number of subjects. Initially, [226] proposed the concept of VGI, where several examples are presented to illustrate the strength of VGI. A systematic overview of public healthcare research using VGI is presented in [227]. Various quality measures and indicators for VGI are mentioned in [228]. [229] propose a *Fog-based SDI* framework (*GeoFog4Health*) for analysing big geospatial health data. The potentials of VGI in pervasive healthcare computing applications are presented in [230], where the authors illustrate varied data sources using OpenStreetMap (OSM) in their case-study. Another work [231] presents the challenges and solutions regarding computationally intensive spatial analytics.

There are also varied research works on the Internet of Health Things (IoHT).

Mukerjee et al. [132] presents a framework for personalized health care in the IoT system. The techniques in the optimization of resources in the fog environment are presented in [232]. Spatial service orchestration in cloud has been proposed for geospatial query execution. *Spatio-fog* framework proposes an energy efficient and delay-aware fog computing model for processing geospatial query. However, this work does not consider the mobility aspect of the users, which is an important factor in provisioning QoS-aware services. In this regard, Ghosh et al. [86] presents a mobility-aware framework (*Mobi-IoST*) which considers the mobility information of the users in a region and assists them in the time of emergency. However, this frameworks cannot handle a disaster scenario, which affects the underlying road network.

Another recent work [233] extracts the correlations of spatio-temporal events by proposing mobility-association rules. However, it falls short in reducing energy consumption as it only relies on cloud servers to process the data. To the best of our knowledge, the proposed framework, RESCUE, overcomes all of these limitations as mentioned above of the existing works and provides green and delay-aware systems to assist users in the time of exigency.

Table 6.1: Comparison of existing works with related features

Existing Works	Integration of Cloud-Fog -Edge-IoT	Mobility Awareness	Crowd-Sourcing Approach (VGI)	Geospatial Query Processing	Delay Awareness	Low Power Consumption	Time -Critical Application
Kaur et al.[234]	✘	✘	✘	✘	✘	✘	✓
GeoFog4Health[229]	✘	✘	✘	✘	✘	✓	✓
HARKE[224]	✘	✘	✘	✘	✘	✓	✘
Goranson et al.[227]	✘	✘	✓	✘	✘	✘	✓
Mooney et al.[230]	✘	✘	✓	✘	✘	✘	✓
IoHT[132]	✓	✓	✘	✘	✓	✓	✓
RESCUE	✓	✓	✓	✓	✓	✓	✓

6.3 RESCUE: Proposed Architecture

This section describes our proposed framework, namely *RESCUE*, which facilitates efficient health-management and post-disaster recovery mechanisms efficiently in

minimum time and power consumption. Fig. 6.1 depicts the overall architecture of RESCUE. It is shown that there are two major modules: (i) public health awareness and (ii) home-health monitoring of users and assisting them. For the former case, RESCUE depends on the crowd-sourcing information or VGI. It accumulates the information and finds out the correlation of such reported events with the spatial information. It helps in enhancing public health awareness and taking preventive measures. In the next case, the health of a user is being monitored using Body Area Network (BAN), and in case any abnormality is detected, proper measures are taken. In both cases, IoT devices, fog/edge nodes, and cloud servers communicate seamlessly to facilitate users' services. The major working modules are also shown in the figure, and the modules are latency-aware and require less power consumption. To ease the readability of the workflow of the modules, we have presented the sequence diagram in Fig. 6.2.

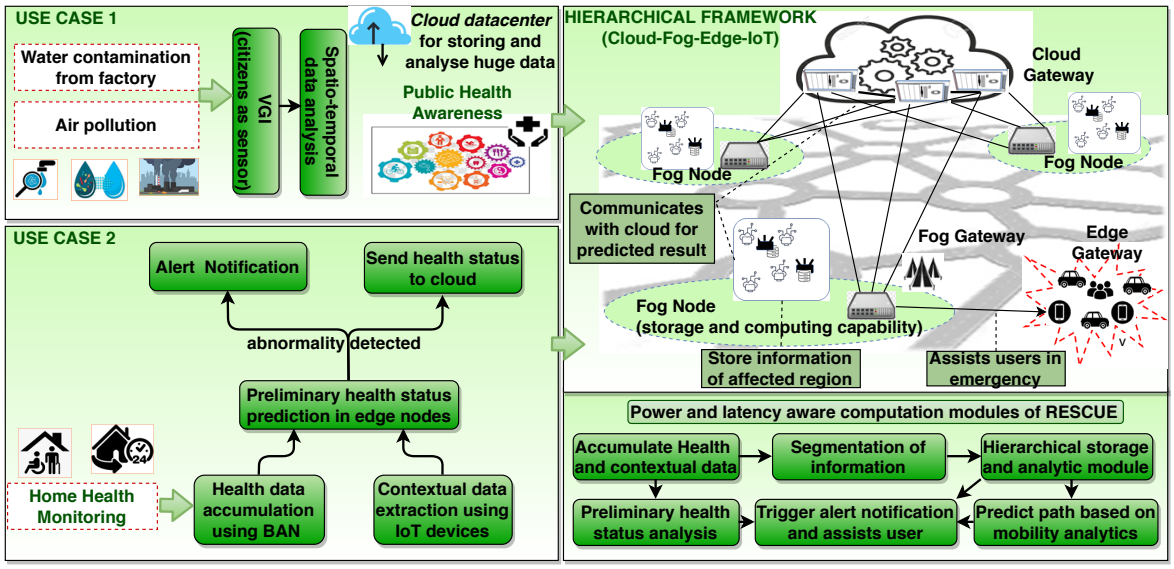


Figure 6.1: RESCUE framework

6.3.1 Public health management: volunteered geographic information (VGI) approach

In the era of sensor network development, VGI is termed as *humans as sensor* or *citizen as sensor*, since a number of independent individuals (human/ citizen) can provide information about their surroundings. VGI has a massive impact on public health monitoring. For instance, VGI could assist emergencies like the recent outbreak of COVID-19. The users can log information about the suspected cases (such as

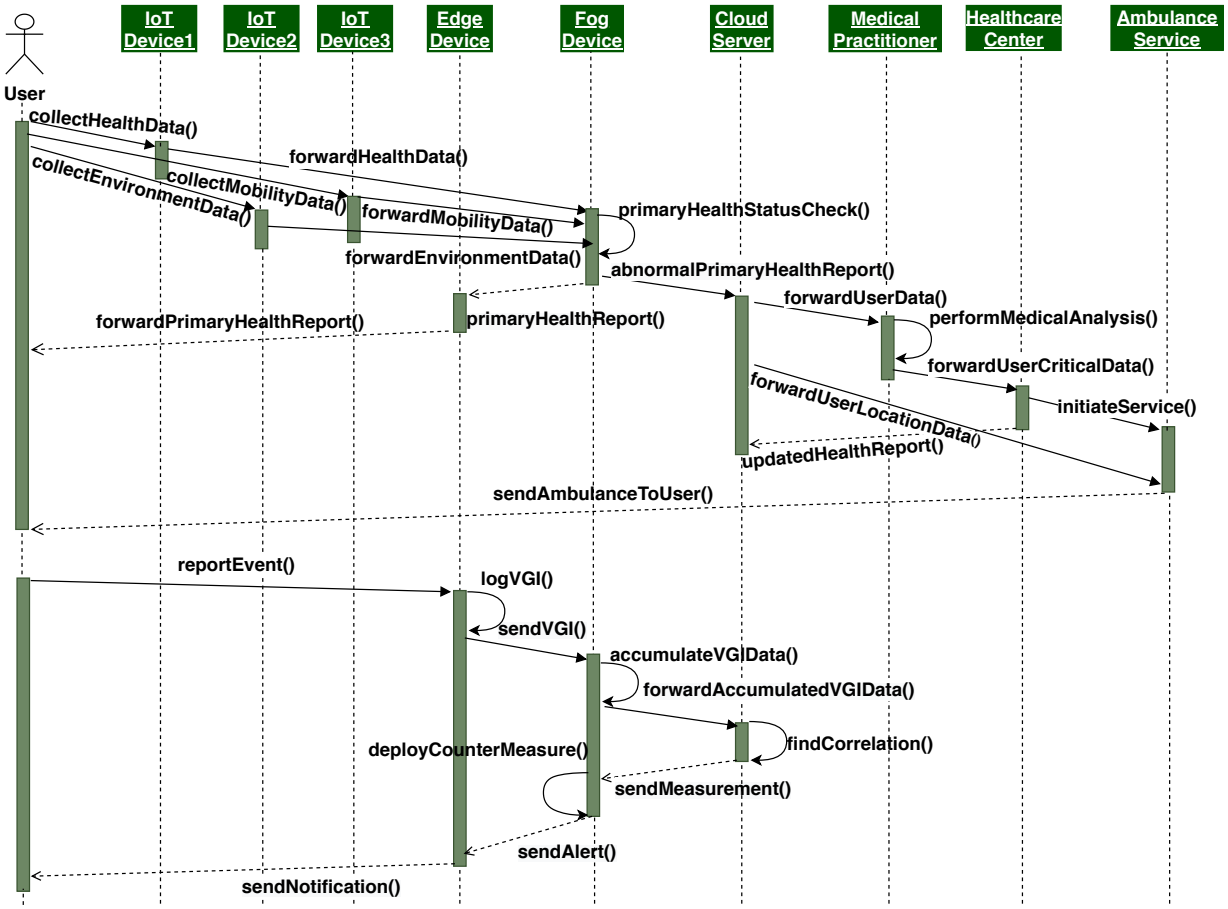


Figure 6.2: RESCUE sequence diagram

having symptoms or having a recent travel history from the affected regions in the world). The countries with high population density, such as India, can also benefit by collecting information like a shortage of essential ingredients or medicines in a lockdown situation.

In short, VGI refers to the use of smart-devices to assemble, modify, and share geographical information provided by the users voluntarily. Now, the volunteers can give a large amount of data at different spatio-temporal resolution. However, there are few challenges: (a) since the data is collected from different people, the accumulated data can be heterogeneous, (b) the reliability of data quality needs to be maintained. In this work, we mainly focus on public health status. To restrict the heterogeneous nature of the collected data, we provide a list of events (*ev*) such as accident, water contamination, excess household wastage, or a sudden outbreak of a disease. The volunteers can select any option from the list and mark the severity (Se_{ev}) of the event. Furthermore, they also log the location information for such events. For maintaining the data quality, we have deployed a *hierarchical approach*, where RESCUE relies on a reduced group of trusted individuals (who act as moderators). Furthermore, when a large amount of data is collected from a region and such moderators are not present, in that case, we follow *Crowdsourced approach*, where the convergence on the reliability of the data is fully dependent on the crowd (or volunteers) by identifying and correcting errors collectively. Once the dataset is collected, we form different spatial clusters with the collected dataset. Each cluster consists of the information $< ev_i, Se_{ev_i}, cardinality >$. Here, ev_i is an event i with severity Se_{ev_i}. The *cardinality* is computed by aggregating the number of data entries with the same value. Next, we generate a *heatmap* using this information from all of the places and find out the correlation with other contextual parameters.

RESCUE also can analyse such historical records, if available. For instance, in some villages of India, in *rainy season*, few infectious diseases (e.g., Cholera or Dengue) occur on a large scale. Likewise, several diseases depend on temperature, humidity, and rainfall patterns. If these relations can be found apriori, early preventive measures can be adapted to prevent the widespread of the diseases. Thus, we can find out the hotspots of such diseases in different spatial and temporal scales.

6.3.2 Mobility analysis to find the routes

In this section, we describe the process of finding a path to reach the destination in minimal time, avoiding the regions with risk (or affected areas). First, we model the study-region in a *graphical structure* where the nodes are the POIs, and the edges

Algorithm 4 : Extracting routes from source to destination in time of exigency

Input: VGI, mobility and POI data of N connected regions (R) of graph $G(V, E)$
Output: Route $< Route(S, D, \Upsilon) >$ ▷ Route from source S to destination D

$V, E, \Upsilon \leftarrow NULL$;
for *each regions* $l_j \in N$ **do** ▷ Check all regions for accumulated VGI data of events
  for *each GPS point* $p_i \in R$ **do**
    for *each accumulated VGI* $v_k \in V$ **do**
      $flag \leftarrow checkAuthen(v_k, p_i)$
      if $flag == 1$ **then**
        $S_j \leftarrow$ ComputeCardinality (v_k) ▷ Compute the number of times an event is reported in a particular location and store using hierarchical indexing
        $S_j \leftarrow geotagg()$ ▷ Associate location information with the reported event
        $h_j \leftarrow heatmapGen(S_j)$ ▷ Generate a heatmap and analyse the correlation value
        $S.insert(h_j)$ ▷ Append the event in the datastore
      end if
    end for
  end for
end for
for *each mobility trace* $tr \in T$ **do** ▷ Analyse the mobility information in the region
  for *each event information* $h \in S$ **do**
    $GE.append(h)$ ▷ A new node in the affected region is generated
    $GE.buffer(50)$ ▷ Buffer with 50 meter for each affected region is generated
    $path \leftarrow autoencoder(GE, POI, M)$ ▷ Learn representation of data and context using autoencoder
    $path \leftarrow refinement(GE, v_{risk}, path)$ ▷ Extract routes using MDP
    $\Upsilon : route \leftarrow$ Add nodes from $path$
    $t \leftarrow extractTemporal(path)$ ▷ Extract path with less commute time
    $\Upsilon.append(Edge(G, t))$ ▷ Append edges between the nodes to complete the route from source to destination
    Print Υ ▷ Print the path
  end for
end for

are the road-segments. The region is segmented into uniform grids, and the fog nodes store the road-network information as well as the POI information of the region under its coverage area. The fog nodes are capable of communicating with each other and sends the information to the cloud gateway, which forwards it to the cloud. The route finding method requires a large set of data analysis. Hence the cloud will perform the data analysis and extract the optimal path. We begin the discussion by defining the preliminary concepts as follows:

1. Road network ($R(V, E)$): The underlying road network is defined by a directed graph, where the edges $e_i \in |E|$ are the road-segments and the intersections of the edges are represented by nodes $v_i \in |V|$.

2. POI (P): The POI or Point-of-interest depicts the landmarks of a region, such as, residential area, commercial area etc. We build a tree-based structure to store all of these POI information.

3. Route (S, D, Υ): The route represents a sub-graph of $R(V, E)$ where the two end-points are source (S) and destination (D) nodes and the intermediate edges and nodes represent the optimal path to reach the destination.

It may be noted that the volume of data including road-network structure, POIs and movement information is huge, and increases along with the spatial range of the study region. It is not possible to store all such information in a centralized server. To this end, we build an efficient indexing scheme and deploy it. Here, we have used the large cell base stations (Road Side Unit or RSU) as the fog nodes. The RSUs have temporary storage and computing capabilities. These are divided into two categories (i) macro RSU and (ii) micro RSU. The macro RSU has a coverage area of 1-20km and micro RSU has coverage area of 200m-1km. All the POI information and other mobility related information are stored in these RSUs. We segregate the study regions into hexagonal grids of uniform area. Each of the centre points of the grids are extracted and geo-hash code is generated. The list of the geo-hash codes are stored in the cloud server. The fog nodes store the POIs within its coverage using layered hashing scheme. The base idea is taken from [233]. RESCUE uses three layers of hashing, where each layer stores more granular spatial information.

RESCUE uses *Markov Decision Process (MDP)* to recommend the routes from given source to destination. Typically, MDP is used as it has a fast convergence speed and it provides global optimal route instead of only considering the local benefits. The *reward function* ($path_{re}$) of the process is defined as:

$$Path_{re}(POI_a, POI_b, Seg) = \frac{Risk_d(POI_a, POI_b)}{NW_{dis}(POI_a, POI_b)^{\frac{Risk_d(POI_a, POI_b)}{|Risk_d(POI_a, POI_b)|}}} \tag{6.1}$$

This is the reward function when a particular grid (Seg) is selected in the path. It may be noted that when the user is in a grid, it is only possible to visit in any of its neighboring grids in the next step. The NW_{dis} is the *Haversine distance* between POI_a and POI_b. It is assumed that the user wants to visit POI_b starting from POI_a. The difference of risk in moving from POI_a to POI_b is represented by $Risk_d(POI_a, POI_b)$. These risk values are computed from the available datasources like crowd-sourcing data. The heatmap is used to compute the risk value in a range of [0,1] for any particular location. Algo. 4 represents is the basic steps of modelling such networks

along with the movement information. Here, we have used the auto-encoder and Markov Decision Process to model the road network and related information and subsequently finding the path to reach the destination from the source. A Road network is defined by a directional graph with road-segments as edges and the intersection points of the road-segments as nodes. In the time of disaster, information from heterogeneous sources need to be augmented which makes it high-dimensional dataset and the process becomes time-intensive. For this reason, here RESCUE uses an *autoencoder* to learn the representation of data related to emergency situation by dimensionality reduction.

Here, we deploy two phase processing: offline and online. In offline process, we compute and store the optimal paths between different pairs of location, which are accessed frequently. The extraction of *optimal path* can depend on various aspects, like, minimum commute time, shortest distance or minimum fuel consumption. In the *online* phase, when a path-query is processed, the fog nodes take the computed route from the given source to destination, and checks the feasibility of the solution. Next, a feedback is sent to the cloud server, in case the route is not optimal.

6.3.3 Geospatial query and services

Geospatial service helps to retrieve the geospatial data from the different databases seamlessly. There are many types of Open Geospatial Consortium (OGC[1]) standardise spatial services available. Web feature service(WFS[2]) extracts the featured geospatial data from the database. Web processing service(WPS[3]) processes the geometrical operations, i.e., overlap, buffer, intersection, cross, etc. over existing geospatial data. Web coverage service(WCS[4]) accesses the multi-dimensional geospatial data from the server database. Web map service(WMS[5]) displays the map with the user's point of interest(POI). Catalog service(CSW[6]) keeps the information about the available geospatial data along with its source.
Some examples of healthcare system related geospatial query where the geospatial services are applied.
Geospatial Query 1(GQ_1): *List out the hospitals where MRI facility available of city C.*
SQL syntax of GQ_1 is as follow:

[1] `https://www.ogc.org/`
[2] `https://www.ogc.org/standards/wfs`
[3] `https://www.ogc.org/standards/wps`
[4] `https://www.ogc.org/standards/wcs`
[5] `https://www.ogc.org/standards/wms`
[6] `https://www.ogc.org/standards/cat`

```
SELECT H.name, H.time, H.price
FROM Hospital H
WHERE H.facility='MRI' AND City='C';
```

In this geospatial query, one layer(hospital facility) is involved. A geospatial filter operation (facility='MRI') is applied over the layer. The query parse tree of GQ_1 is presented in Fig. 6.3.

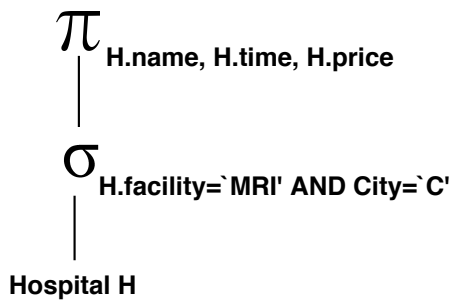


Figure 6.3: Query parse tree of GQ_1

To retrieve hospital data from multiple hospital data sources, the essential geospatial web services for GQ_1 are as follows.

1. getFeature feature service is needed to retrieve hospital data with *MRI* facility availability in city C.

2. getMap service displays the result of GQ_1 on a map.

Geospatial Query 2(GQ_2): *Find out the hospital details which are available within r kilometers radius of a point P(x,y) in ascending order by distance. x and y are latitude and longitude of point P.*
SQL syntax of GQ_2 is as follow:

```
SELECT H.name, H.address, H.contact, H.distance
FROM Hospital H
WHERE Overlap (H.shape, Buffer('P(x,y)', r))=1
ORDER BY H.distance ASC;
```

In this geospatial query, two thematic layers, Land use land cover(LULC) and Hospital, are involved. Distance calculation between the user location (x,y) and different hospitals is possible only after the integration or overlying of these two layers. In general, the euclidean distance is calculated between two points. But, in a real

scenario, there may not have a path between these two points. The shortest euclidean distant hospital may have no path to reach there. Whereas, the euclidean distance-wise comparatively far hospital has good communication and reach there quickly. So, only one or two thematic layers are not capable enough to consider all real-life factors. Many thematic layers' involvement is required. But, the increment of layers also increases the computational complexities and requires more resources and energy to compute these. The query parse tree of GQ_2 is presented in Fig. 6.4.

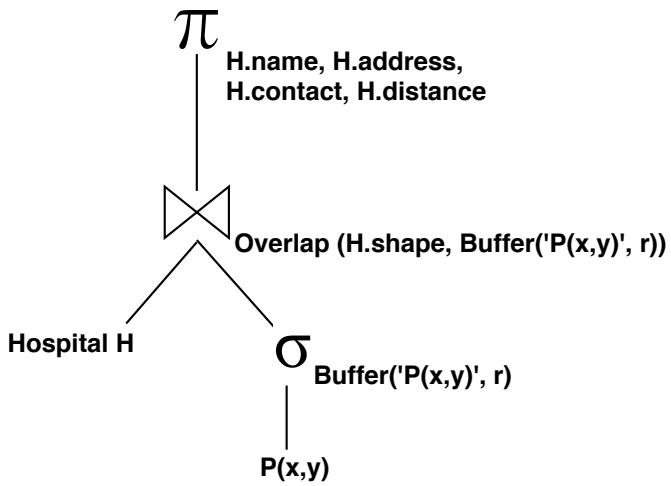


Figure 6.4: Query parse tree of GQ_2

To retrieve geospatial data from multiple data sources, the essential geospatial web services for GQ_2 are as follows.

1. BufferFeatureCollection processing service is needed for '*r*' k.m. radius buffer.
2. IntersectionFeatureCollection processing service is needed for overlapping of point P(x,y) and hospital layer.
3. getMap service displays the result of GQ_2 on a map.

6.4 Latency and Power Consumption of user device during health status detection

In the proposed framework, it is observed that the sensor nodes do the data collection, and the accumulation is done inside the smartphone, which then forwards the data

to the fog device. In case of abnormal health conditions, the fog device forwards the data to the cloud. Further data analysis is performed inside the cloud, and notification is sent in case of an emergency. Here, to calculate the latency and power consumption of the user's smartphone in this entire process, two scenarios are considered: indoor and outdoor. In case of the indoor region, either small cell cloud enhanced enodeB (SCceNB), which is a small cell base station equipped with storage and computational resources (if the user is registered under a cellular network), or the switch, router working as fog device (if the user is connected with Wi-Fi) is used for connecting the smartphone with the network. In the case of the outdoor region, the user is usually connected with the network through the Road side unit (RSU). Now, for these two scenarios, the latency and power consumption of the user device during the entire process of health status detection will be determined. The parameters used in latency and power consumption calculations are defined in Table 6.2.

6.4.1 Latency in case of indoor region

The latency in data collection by the sensor nodes is given as,

$$L_{cs} = max((D_{s1}/S_{s1}),(D_{s2}/S_{s2}),..,(D_{s_{N_s}}/S_{s_{N_s}})) \tag{6.2}$$

The latency in sending data from sensor nodes to smart phone is given as,

$$L_{sm} = max(((D_{s1}/R_{sm})\cdot(1+F_s)),((D_{s2}/R_{sm})\cdot(1+F_s)),..,((D_{s_{N_s}}/R_{sm})\cdot(1+F_s))) \tag{6.3}$$

As there are multiple sensor nodes, the maximum latency is considered. The latency in data accumulation inside the smart phone is given as,

$$L_{ac} = (D_m/S_m) \tag{6.4}$$

The latency in sending data from smart phone to SCceNB/ fog device is given as,

$$L_{mfi} = (D_m/R_{mf})\cdot(1+F_{mi}) \tag{6.5}$$

The latency in data processing inside the fog device/ SCceNB is given as,

$$L_{fpi} = (D_m/S_{fi}) \tag{6.6}$$

Table 6.2: Parameters used in latency and power calculation

Parameter	Definition
D_{sj}	Data amount collected by sensor j
D_m	Data amount accumulated inside the smart phone and transmitted to the fog device
D_c	Data amount transmitted by the fog device after processing to the cloud through Gateway
F_s	Link failure rate from sensor to smart phone
F_{mi}	Link failure rate from smart phone to SCceNB/ fog device
F_{mo}	Link failure rate from smart phone to RSU
F_{fi}	Link failure rate from SCceNB/ fog device to cloud
F_{fo}	Link failure rate from RSU to cloud
L_{qpi}	Latency in processing query if the device is at indoor region
L_{qpo}	Latency in processing query if the device is at outdoor region
N_s	Number of sensor nodes collecting health, movement, environmental data
P_t	Power consumption of the smart phone per unit time in data transmission mode
P_r	Power consumption of the smart phone per unit time in data reception mode
P_a	Power consumption of the smart phone per unit time during data accumulation
P_i	Power consumption of the smart phone per unit time in idle mode
R_{sm}	Data amount transmitted per unit time from sensor node to smart phone
R_{mf}	Data amount transmitted per unit time from smart phone to SCceNB/ fog device/ RSU
R_{fc}	Data amount transmitted per unit time from fog device/ SCceNB/ RSU to cloud through Gateway
S_{sj}	Data collection speed of sensor node j
S_m	Data accumulation speed of smart phone
S_{fi}	Data processing speed of fog device/ SCceNB
S_{fo}	Data processing speed of RSU
S_c	Data processing speed of cloud

The latency in sending data from SCceNB/ fog device to the cloud through Gateway is given as,

$$L_{fci} = (D_c/R_{fc}) \cdot (1 + F_{fi}) \tag{6.7}$$

The latency in data analysis inside the cloud is given as,

$$L_{cp} = (D_c/S_c) \tag{6.8}$$

Therefore the total latency for health status detection while the user is at indoor region is given as,

$$L_{toti} = L_{cs} + L_{sm} + L_{ac} + L_{mfi} + L_{fpi} + L_{fci} + L_{cp} + L_{qpi} \tag{6.9}$$

6.4.2 Latency in case of outdoor region

The latency in data collection by the sensor nodes is determined using equation (6.2). The latency in sending data from sensor nodes to a smartphone is determined using equation (6.3). The latency in data accumulation inside the smartphone is determined using equation (6.4). The latency in sending data from the smartphone to RSU is given as,

$$L_{mfo} = (D_m/R_{mf}) \cdot (1 + F_{mo}) \tag{6.10}$$

The latency in data processing inside the RSU is given as,

$$L_{fpo} = (D_m/S_{fo}) \tag{6.11}$$

The latency in sending data from RSU to the cloud through Gateway is given as,

$$L_{fco} = (D_c/R_{fc}) \cdot (1 + F_{fo}) \tag{6.12}$$

The latency in data analysis inside the cloud is determined using equation (6.8). The total latency for health status detection while the user is at outdoor region is given as,

$$L_{toto} = L_{cs} + L_{sm} + L_{ac} + L_{mfo} + L_{fpo} + L_{fco} + L_{cp} + L_{qpo} \tag{6.13}$$

6.4.3 Power consumption of user device in case of indoor region

The power consumption of the smartphone, i.e., user device while data collection take place by sensor nodes is given as,

$$P_{cs} = P_i \cdot L_{cs} \tag{6.14}$$

The power consumption of the user device while receiving data from sensor nodes is given as,

$$P_{sm} = P_r \cdot L_{sm} \tag{6.15}$$

The power consumption of the user device during data accumulation is given as,

$$P_{ac} = P_a \cdot L_{ac} \tag{6.16}$$

The power consumption of the user device while transmitting data to the SCceNB/ fog device is given as,

$$P_{mfi} = P_t \cdot L_{mfi} \tag{6.17}$$

The power consumption of the user device while data processing takes place inside the SCceNB/ fog device is given as,

$$P_{fpi} = P_i \cdot L_{fpi} \tag{6.18}$$

The power consumption of the user device while fog device/ SCceNB transmits data to the cloud through the Gateway given as,

$$P_{fci} = P_i \cdot L_{fci} \tag{6.19}$$

The power consumption of the user device while data analysis takes place inside the cloud is given as,

$$P_{cp} = P_i \cdot L_{cp} \tag{6.20}$$

The total power consumption of the user device during the entire process of health status detection while the user is at indoor region is therefore given as,

$$P_{toti} = P_{cs} + P_{sm} + P_{ac} + P_{mfi} + P_{fpi} + P_{fci} + P_{cp} + +(L_{qpi} \cdot P_i) \tag{6.21}$$

6.4.4 Power consumption of user device in case of outdoor region

The power consumption of the smartphone, i.e., user device while data collection takes place by sensor nodes, is determined using equation (6.14). The power consumption of the user device while receiving data from sensor nodes is determined using equation (6.15). The power consumption of the user device during data accumulation is determined using equation (6.16). The power consumption of the user device while transmitting data to the RSU is given as,

$$P_{mfo} = P_t \cdot L_{mfo} \tag{6.22}$$

The power consumption of the user device while data processing takes place inside the RSU is given as,

$$P_{fpo} = P_i \cdot L_{fpo} \tag{6.23}$$

The power consumption of the user device while RSU transmits data to the cloud through the Gateway given as,

$$P_{fco} = P_i \cdot L_{fco} \tag{6.24}$$

The power consumption of the user device, while data analysis takes place inside the cloud, is determined using equation (6.20). The total power consumption of the user device during the entire process of health status detection, while the user is at the outdoor region, is therefore given as,

$$P_{toto} = P_{cs} + P_{sm} + P_{ac} + P_{mfo} + P_{fpo} + P_{fco} + P_{cp} + (L_{qpo} \cdot P_i) \tag{6.25}$$

The latency and power consumption of the user device, i.e., smartphone during the entire process, will be compared with the cloud-only framework in the next section. In that case, the smartphone will send the data to the cloud through the intermediate nodes like a base station, switch, router, gateway, but the intermediate nodes will not perform data processing. The entire data processing is performed inside the cloud servers for detecting the health status. This scenario will be compared with the proposed framework in the next section concerning the latency and power consumption of a user device. Here, it has to be noted that in the latency and power calculation model, the latency in path prediction and alert message sending to the smartphone has not been considered. The total latency in path prediction, alert message, and path information transmission to the smartphone is given as:

For indoor region:

$$L_{pathalin} = ((D_{al} + D_{path})/R_{cf}) \cdot (1 + F_{cf}) + ((D_{al} + D_{path})/R_{fm}) \cdot (1 + F_{fm}) \quad (6.26)$$

For outdoor region:

$$L_{pathalout} = ((D_{al} + D_{path})/R_{cr}) \cdot (1 + F_{cr}) + ((D_{al} + D_{path})/R_{rm}) \cdot (1 + F_{rm}) \quad (6.27)$$

The power consumption of the user device during this period will be:
For indoor region:

$$P_{pathalin} = (((D_{al} + D_{path})/R_{cf}) \cdot (1 + F_{cf})) \cdot P_i + (((D_{al} + D_{path})/R_{fm}) \cdot (1 + F_{fm})) \cdot P_r \quad (6.28)$$

For outdoor region:

$$P_{pathalout} = (((D_{al} + D_{path})/R_{cr}) \cdot (1 + F_{cr})) \cdot P_i + (((D_{al} + D_{path})/R_{rm}) \cdot (1 + F_{rm})) \cdot P_r \quad (6.29)$$

where D_{al}, D_{path} are the data amount transmitted for alert message and path information respectively, R_{cr}, R_{rm}, R_{cf}, R_{fm} are the data amount transmitted per unit time from cloud to RSU, RSU to user device, cloud to SCceNB/ fog device, SCceNB/ fog device to user device respectively, F_{cr}, F_{rm}, F_{cf}, F_{fm} are the link failure rate from cloud to RSU, RSU to user device, cloud to SCceNB/ fog device, SCceNB/ fog device to user device respectively.
In that case the total latency for indoor region will be $(L_{toti} + L_{pathalin})$ and for outdoor region the total latency will be $(L_{toto} + L_{pathalout})$. In that case, the total power consumption of the user device for the indoor user will be $(P_{toti} + P_{pathalin})$, and for the outdoor user, the total power consumption of the user device will be $(P_{toto} + P_{pathalout})$.

6.5 Performance Evaluation

In this section, we illustrate the efficacy of our system by developing a test-bed as well as using a simulation toolkit.

6.5.1 Experimental Test-bed

We have developed an experimental test-bed for evaluating the efficacy of the proposed system. We have used the compute engine and app engine of Google Cloud Platform (GCP) to carry out the spatio-temporal data analysis. In the test-bed, we

have used Raspberry Pi 3 as the fog device. We have designed the android application using Android Studio 4.1 with *Firebase database* support. It is used to collect VGI data and personalized health data from the users. In Fig.6.5 (a), the user is asked to select the option between VGI and personalized health data. The personalized health data refers to the information collected from the BAN. When the user selects the personalized health data option, the BAN sensors are synced, and information is logged. On the other side, if the user selects the VGI option, then the next page of the app opens (as illustrated in Fig.6.5(b)). Here, the user is asked to select the type of event (such as accident or water contamination, etc.) he/she wants to log. In our case study, we have provided a list of event-types, where the user may select one or more than one. Also, the user can add any other events not listed in the selection menu. Next, the user logs the severity of the event on a scale (0,5) and provides the location of the event occurrence.

As soon as the user submits the information, the data is sent to the cloud server. The app can also collect the contextual data (location, acceleration, proximity, temperature, and light sensor data) from the smartphone's in-built sensors using the *Android sensor framework*. The application can also communicate with wearable devices such as smart-watch (Fitbit), body temperature measuring module, and SPO2 tracking module. In the Raspberry Pi 3, we have installed the *Eddystone Bluetooth Beacon*, for sending data periodically. For evaluating the spatio-temporal analysis, we have implemented the methods in GCP and QGIS framework. For this prediction, we have considered a region, *Bankura* district of West Bengal, India. The experimental and simulation results are discussed in subsequent sections of chapter.

6.5.2 Mobility analysis

In this section, we evaluate our proposed framework with five baseline methods to demonstrate the system's efficacy in terms of accuracy and delay of extracting path in the time of exigency. Here, we have used VM with $4 vCPU, 15 GB\, memory$ of Google Cloud Platform (GCP), and TensorFlow platform for implementing the proposed path finding algorithm, and obtaining the results.

To illustrate the efficacy of the proposed method, we have designed and executed a large set of experiments on mobility datasets and road-network. We consider a region of $10.8 km^2$ with 16×10^3 nodes in the underlying road network. We simulated several scenarios where varied segments in the study-region become the victim (or affected) region, and un-reachable. Our framework finds out the path when these scenarios occur. Fig. 6.6 represents the precision metric of the path finding module.

It is observed that our method achieves $0.98-0.90$ precision value with 10×10^2 and 50×10^2 edges in the road-graph, respectively. The significance of this result is that with a large number of edges, our framework is capable to efficiently extract the path to reach the destination avoiding the blockage or affected regions.

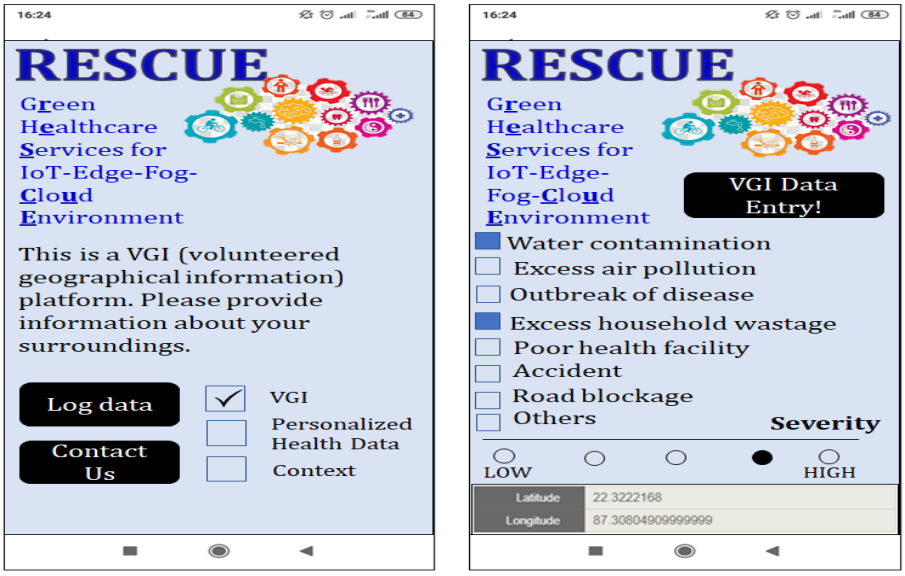


Figure 6.5: Frontend of RESCUE android application: (a) Home page of the android app; (b) Data collection page of the app.

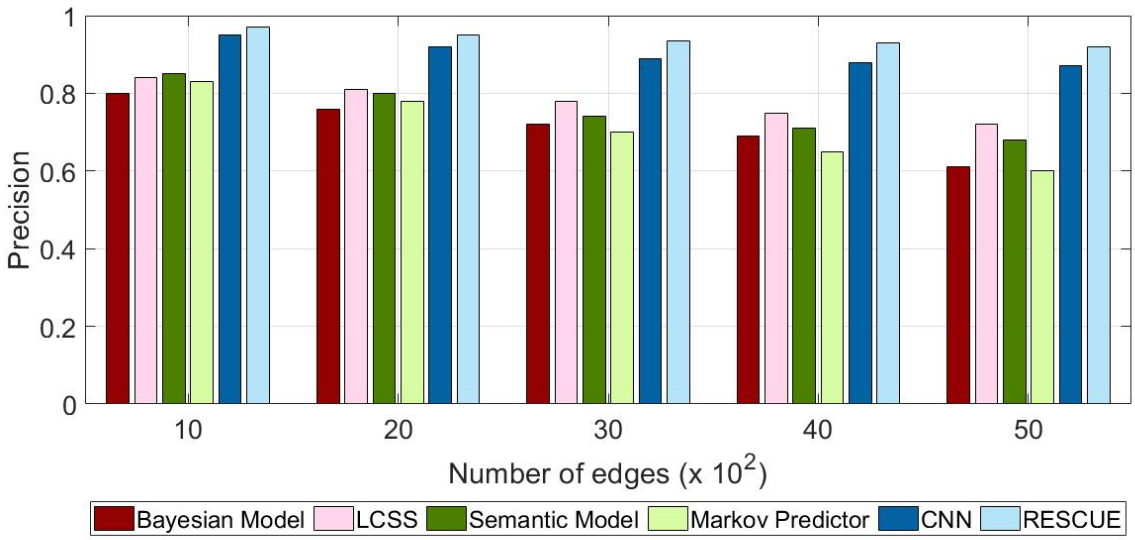


Figure 6.6: Precision value of path-finding algorithm

Fig. 6.7 illustrates the runtime of the path-finding algorithm compared to the baseline methods. It is observed that with more numbers of edges, our framework, significantly outperforms others. It shows high accuracy, precision as well as less

execution time compared to other existing approaches. Table 6.3 summarizes the experimental results compared with baseline methods. The accuracy value is measured by the extraction of the most optimal path in the region. Stability represents the robustness of the system. We have measured both the stability and accuracy metrics by simulating 10 simulated scenarios and report the average results for all baselines and RESCUE framework. The *learning cost* of the model estimates the time to learn the parameters of the models using the training dataset. Here, the area of the study region has been considered as the input cardinality of the model. The learning cost is categorized into three categories concerning model training time: Low (6-12mins), Medium (13-20mins), and High (above 20mins). It is observed that the neural network model has *high* learning cost. The modelling cost consists of the data pre-processing time, data segmentation, and generating the structure, if required. The modelling cost is categorized into three categories: Low (time 0-5mins), Medium (6-8mins), and High (above 9-15mins). Here, we have used the categorical values, as different variations of the baseline models may provide different training time or modelling time. Hence, we have considered a range of values. In several aspects, RESCUE has outperformed other approaches to a significant margin. The key reason behind this result is that our framework models the study-region in a graph-based structure in a hierarchical manner, and computes the path effectively.

Table 6.3: Performance metrics of the proposed mobility analytics module with baseline methods

Metric	Bayesian Model	LCSS	Semantic Model	Markov Model	Neural Network Model	RESCUE
Accuracy	78.1%	74.06%	79.8%	77.71%	89.48%	93.56%
Stability	35.8%	23.02%	49.86%	30.06%	85.09 %	91.87%
Learning Cost	Low	Medium	Low	Medium	High	Medium
Modelling Cost	Medium	Low	Low	Low	High	Low

6.5.3 Simulation results using iFogSim

The proposed healthcare framework has been simulated in iFogSim [144]. Here, Eclipse IDE has been used for implementation, and JProfiler has been installed and integrated with Eclipse IDE. The proposed healthcare framework has been created in iFogSim, and the respective codes are written, compiled, and executed. The created topology for the proposed healthcare framework is shown in Fig. 6.8. As observed in Fig. 6.8, there are data sensors (collecting contextual data, mobility-related data, blood pressure data, pulse rate data, body temperature data), and ECG monitoring module under each edge device. There are six edge devices, which are connected to

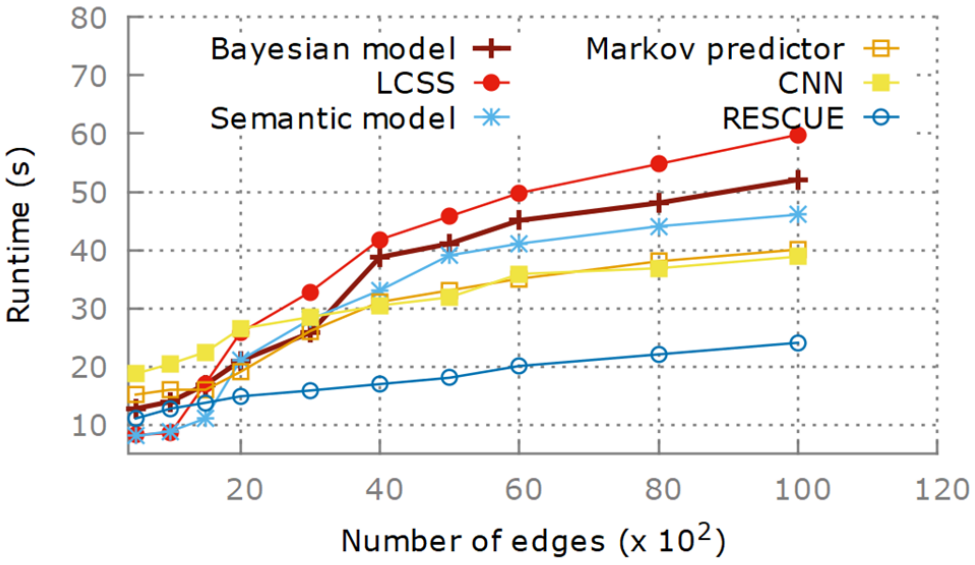


Figure 6.7: Comparison of runtime of path-finding algorithm

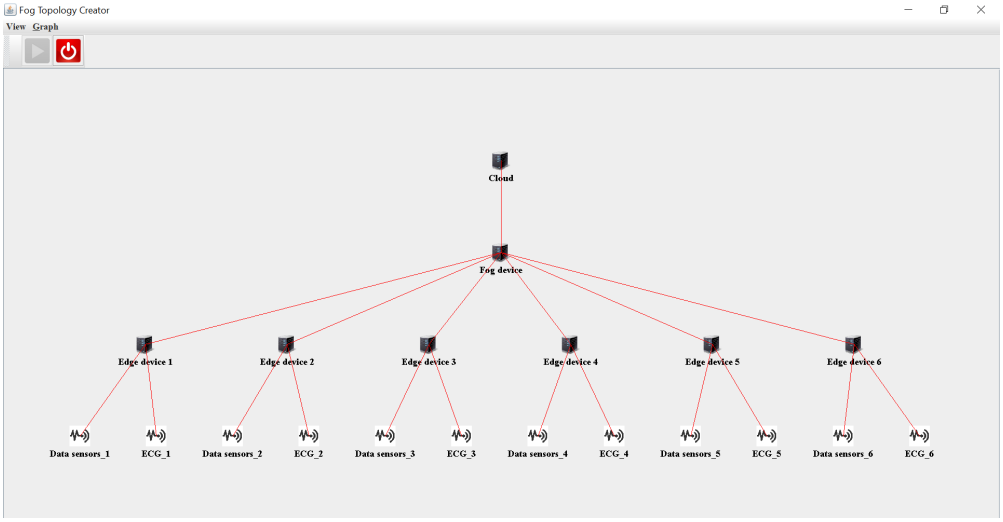


Figure 6.8: Created topology of proposed healthcare framework in iFogSim

a fog device. The fog device is connected with the cloud. We have also created the cloud only healthcare framework in iFogSim, and written, compiled, and executed the corresponding codes.

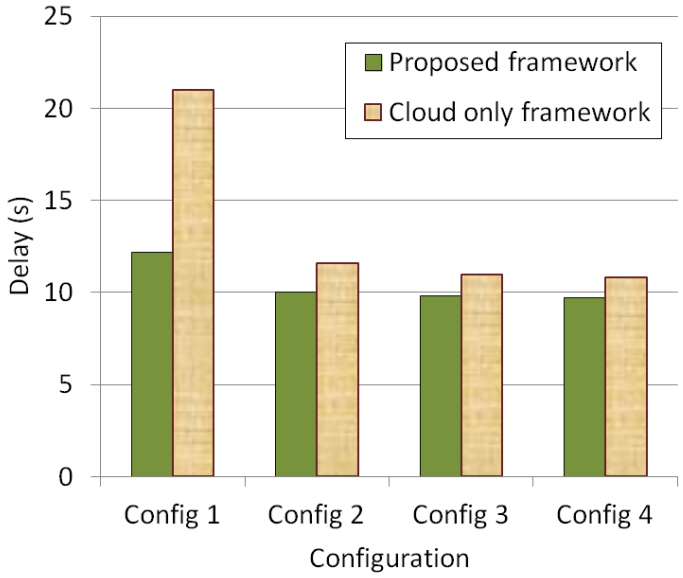


Figure 6.9: Delay in execution of Fog-based topology and Cloud-only topology for e-Healthcare

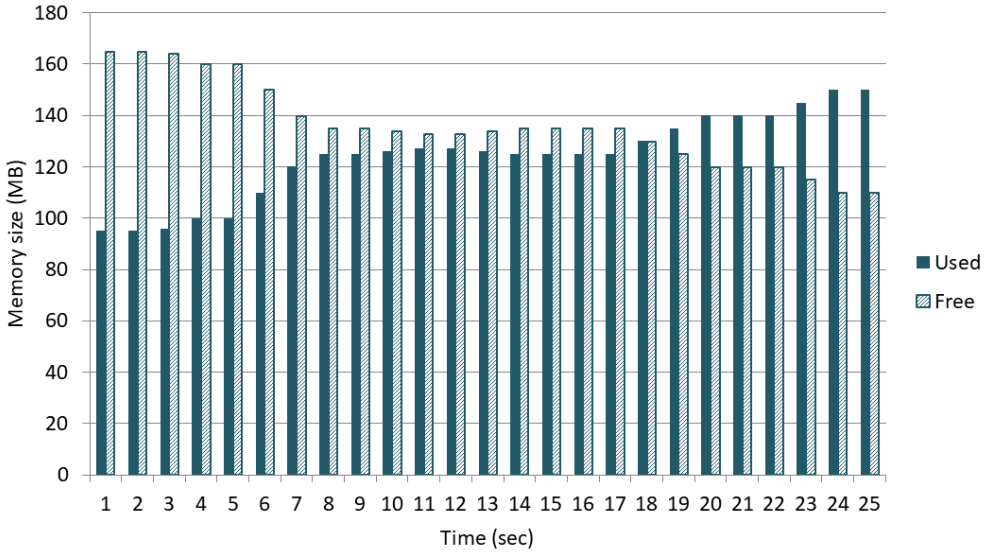


Figure 6.10: Memory usage during execution of the created topology

Four different configurations are considered, presented as follows.

- Config1:- Host storage: 1 GB, Cloud VM with CPU 3 GHz, RAM 4 GB, fog processor 3 GHz, RAM 4 GB

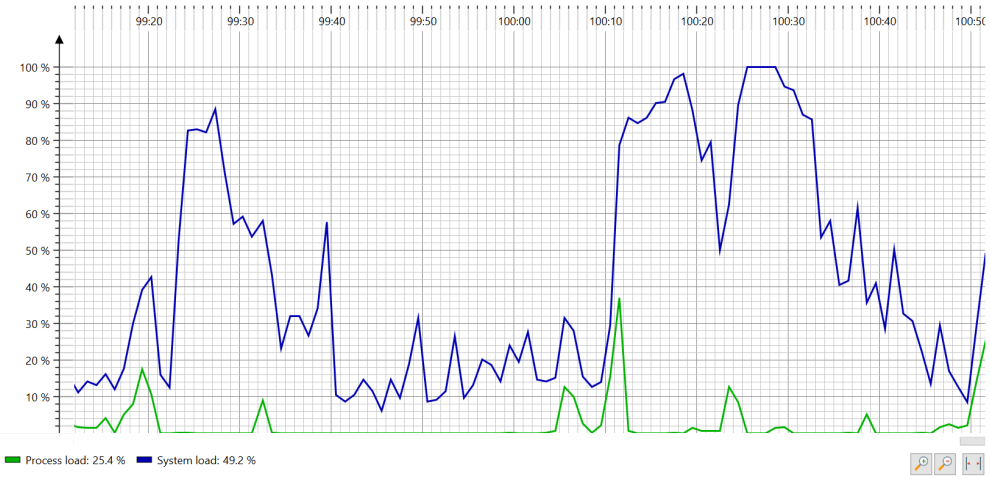


Figure 6.11: CPU load during execution of the created topology

- Config2:- Host storage: 1.5 GB, Cloud VM with CPU 3 GHz, RAM 4 GB, fog processor 3 GHz, RAM 4 GB
- Config3:- Host storage: 2 GB, Cloud VM with CPU 3 GHz, RAM 4 GB, fog processor 3 GHz, RAM 4 GB
- Config4:- Host storage: 2.5 GB, Cloud VM with CPU 3 GHz, RAM 4 GB, fog processor 3 GHz, RAM 4 GB

The execution delay (in second (s)) in the case of the proposed and cloud-only healthcare framework is monitored and compared. The results are presented in Fig.6.9. It is noted that the RESCUE has $\sim 10\% - 40\%$ less execution delay than the cloud only framework.

During the execution of code and created topology for the proposed framework, the JProfiler has been attached with the current Java Virtual Machine (JVM) to monitor the memory usage and CPU load. The CPU load and memory usage while executing the created topology and the respective codes, have been monitored using JProfiler. The memory usage is presented in 6.10, where the used and free memory (heap) size are shown with respect to the time of execution of the simulated framework. The CPU load has been presented in Fig. 6.11, where the process and system load are marked in green and blue lines, respectively.

6.5.4 Visualization of use cases using real-life data

Our experiment location is Bankura district, West Bengal, India. We take two layers (Health centers and road network) of Bankura district. The layers are presented in

Fig. 6.12.

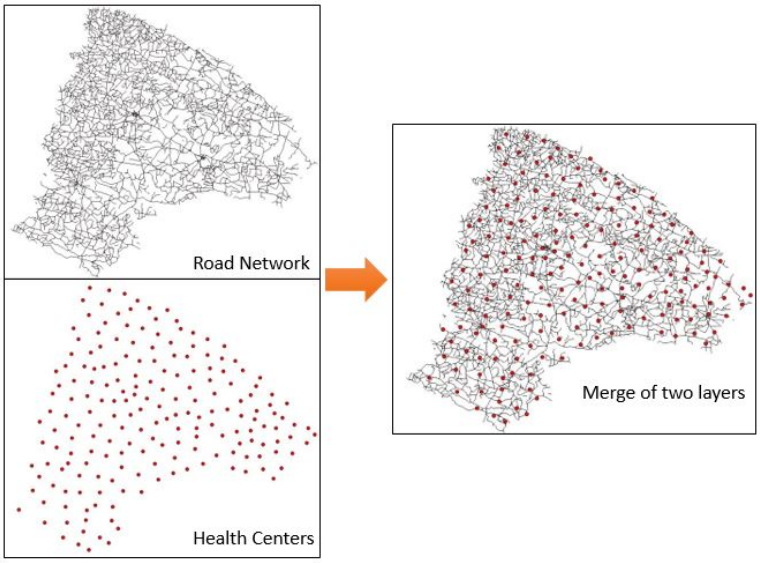


Figure 6.12: Road network and health centers of Bankura district, West Bengal, India

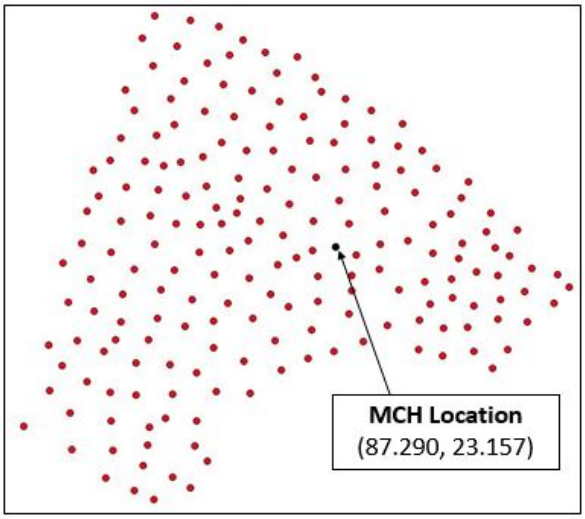


Figure 6.13: MRI facilitate hospital

Table 6.4: MRI facilitate hospital details for GQ1

H.ID	H.Latitude	H.Longitude	H.facility	H.Time	H.Price
MCH	87.290	23.157	MRI, ECG	11:00-16:00	INR 12000

Consider a user searching for MRI facilitate hospital details in Bankura district using geospatial query (GQ_1) of section 6.3.3, where District = 'Bankura'. So, the query syntax for the user will be as follow:

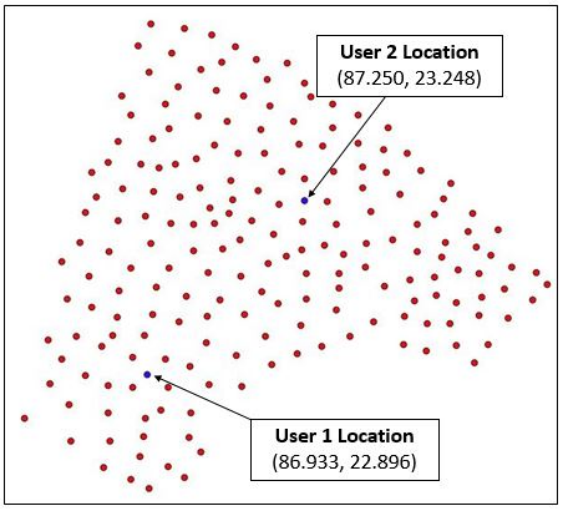


Figure 6.14: Locations of two users

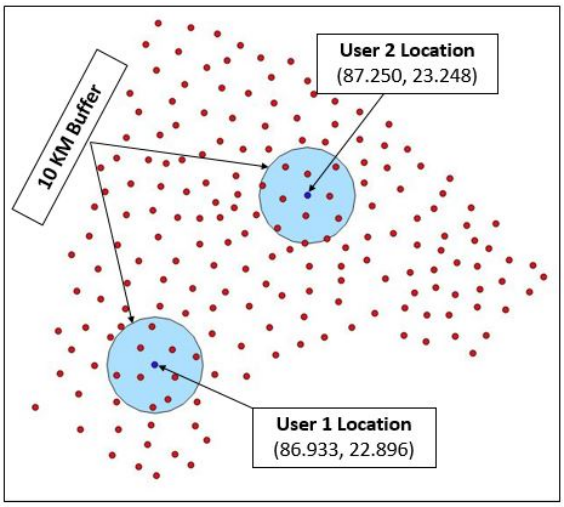


Figure 6.15: 10 kilometers buffer area of user locations

```
SELECT H.ID, H.Latitude, H.Longitude, H.facility, H.Time, H.Price
FROM Hospital H
WHERE H.facility='MRI' AND District='Bankura';
```

The details of the geospatial query(GQ_1) result is presented in Table 6.4 and pictorially represented in Fig. 6.13. The MRI facility is only available in the Medical College and Hospital(MCH)(87.290, 23.157).

Suppose, two users are searching hospitals using geospatial query (GQ_2) of section 6.3.3, where radius 'r'= 10 km. Two users' locations are (86.933, 22.896) and (87.250, 23.248). So, the query syntax for User-1 and User-2 will be as follows:

```
SELECT H.ID, H.Latitude, H.Longitude, H.distance
FROM Hospital H
WHERE Overlap (H.shape, Buffer((86.933, 22.896), 10))=1
ORDER BY H.distance ASC;

SELECT H.ID, H.Latitude, H.Longitude, H.distance
FROM Hospital H
WHERE Overlap (H.shape, Buffer((87.250, 23.248), 10))=1
ORDER BY H.distance ASC;
```

Users' locations are shown in Fig. 6.14 with navy blue points. 10 kilometers of buffer has been generated surroundings of both the users and shown in Fig. 6.15 with sky blue circles. The details of the hospitals of each user is presented in the Table 6.5 and Table 6.6.

Table 6.5: Hospital details within 10 kilometer radius of user-1 (86.933, 22.896)

H.ID	H.Latitude	H.Longitude	H.Distance (K.M.)
BPHC12	86.930	22.808	0.6237
HC06	86.928	22.977	0.7392
HC10	86.906	22.931	3.0229
PC16	86.904	22.872	3.2422
PC17	86.902	22.933	3.4695
SH2	86.968	22.929	3.9140
PC13	86.976	22.872	4.8048
RH09	86.979	22.826	5.1545
PHC4	86.854	22.874	8.8245

Table 6.6: Hospital details within 10 kilometer radius of user-2 (87.250, 23.248)

H.ID	H.Latitude	H.Longitude	H.Distance (K.M.)
RH21	87.252	23.295	0.3365
PC31	87.246	23.207	0.4979
PC24	87.244	23.148	0.8584
MCH	87.290	23.157	4.4938
PC35	87.206	23.308	4.9250
RH25	87.298	23.248	5.3611
HC	87.200	23.242	5.5846
PHC25	87.189	23.181	6.8228
PC33	87.311	23.306	6.8201
BPHC28	87.316	23.202	7.3756
PC40	87.163	23.270	9.7178

The terms used in the tables are as follows: HC- Health Center, PHC- Primary Healthcare Center, BPHC- Block level Primary Healthcare Center, MCH- Medical College and Hospital, RH- Rural Hospital, PC- Private Healthcare Center, SH- Subdivisional Hospital.

6.5.5 Theoretical Analysis

In section 6.4, the theoretical model of calculating latency and power consumption of the user device has been presented. In this section, we will calculate the same to compare with the cloud-only scenario. The integrated health, movement, and environmental data amount are considered 2.2-3 GB. The data transmission speed of the network is considered as 100-200 Mbps.

Figs. 6.16 and 6.17 present the latency for detecting health status in case of the indoor and outdoor scenarios, respectively, while using the proposed method and cloud-only scheme. The latency for detecting health status for indoor and outdoor users in the case of a cloud-only scheme [234] is determined to compare with the proposed method. Figs. 6.18 and 6.19 present the power consumption of the user device, i.e., smartphone in case of the indoor and outdoor scenarios during the health status detection period respectively while using the proposed method and cloud-only scheme. The power consumption of the user device during the health

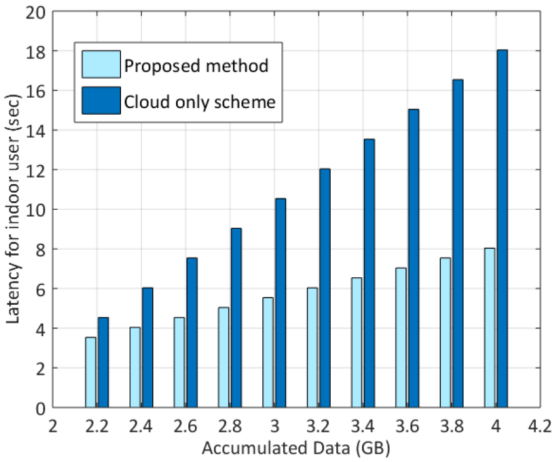


Figure 6.16: Health status detection latency (indoor scenario)

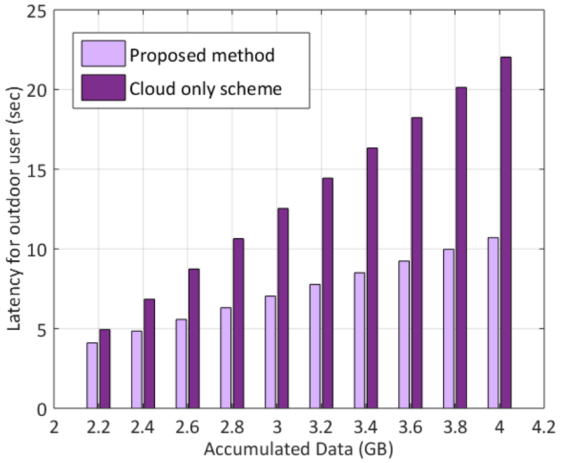


Figure 6.17: Health status detection latency (outdoor scenario)

status detection period in the case of a cloud-only scheme [234] is determined for indoor and outdoor users to compare with the proposed method.

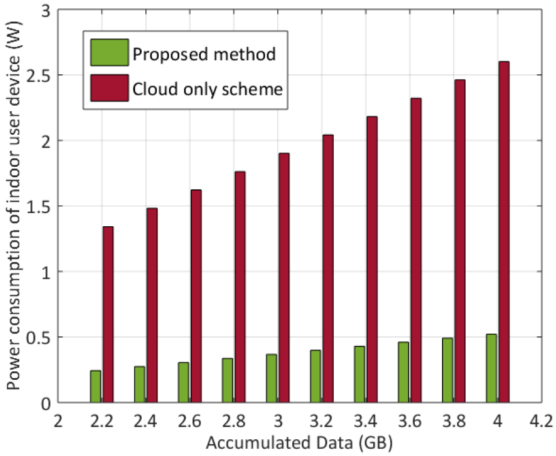


Figure 6.18: Power consumption of user device during health status detection period (indoor scenario)

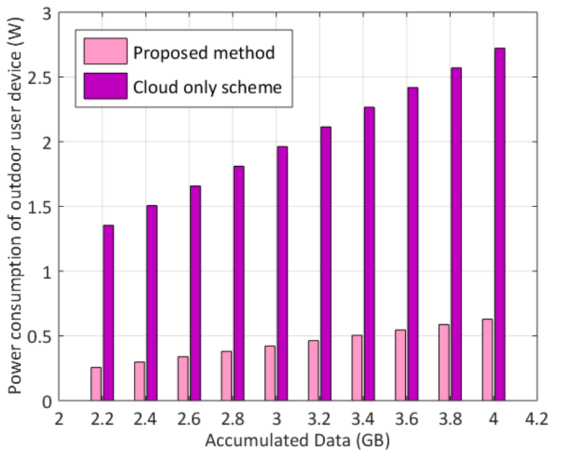


Figure 6.19: Power consumption of user device during health status detection period (outdoor scenario)

From Figs. 6.16 and 6.17, it is observed that the proposed method reduces latency up to 55% for the indoor user and up to 51% for the outdoor user than the cloud-only scheme. From Figs. 6.18 and 6.19, it is observed that the proposed method reduces power consumption of the user device up to 81% for the indoor user and up to 80% for the outdoor user than the cloud only scheme. In the cloud-only paradigm, data processing, and health status detection take place inside the cloud. But in the proposed framework, the SCceNB/fog device/RSU processes the data to

detect the health status before forwarding to the cloud. Thus the data transmission latency is reduced, which subsequently reduces the power consumption of the user device. As a result, the proposed framework provides faster and green health service provisioning to the user.

6.6 Summary

This chapter presents an end-to-end framework, namely RESCUE, which is capable of assisting users in the time of emergency (say, natural disaster) by predicting path in the post-disaster scenario when several roads are non-reachable. The framework's major working modules are: accumulating and refining crowd-sourcing data and extracting the correlations among the event and the spatial location. RESCUE collects the BAN data consisting of health-parameters' data of users and other environmental parameters' data from IoT devices. It analyses the data in the fog nodes, and in case any abnormality is found, the data is sent to the cloud server immediately. Further, RESCUE is conducive to predict routes to users in the time of exigency, avoiding the affected regions in minimal commute time. While the IoT paradigm enables global connectivity worldwide, communicating with billions of interconnected devices, the whole process's energy consumption is rising steeply. In this regard, *Green IoT computing* has a pivotal role in reducing environmental problems, and subsequently creating a sustainable environment by emphasizing energy-efficient technologies. It reduces the greenhouse gas emission and provides a *smarter* and *greener* view of varied applications. To this end, the major aim of this chapter is also to reduce energy consumption and contributing towards sustainable urban development. It may be noted that the mobility analysis module is also capable of predicting routes from source to destination, which requires less fuel consumption. The autoencoder process of the mobility analysis module helps in facilitating this feature as well. In all sense, RESCUE provides a *green healthcare and user assistance* framework in the time of exigency.

Chapter 7

Conclusion and Future Directions

This chapter concludes the thesis. A summary of the key contributions of the thesis discusses here. Future scopes of the thesis are discussed at the end of this chapter.

7.1 Summary of Contributions

Chapter 1 discussed the basics of various computing paradigms and different kinds of geospatial queries. The objective and motivation behind this thesis works are also elaborated in this chapter. The major contributions of this thesis are pointed here.

Chapter 2 investigated existing works in the geospatial cloud, fog, and edge computing paradigms and generated a taxonomy on it. Different kinds of geospatial applications on computing domains are presented in a table.

Chapter 3 presented a geospatial query resolution framework using an orchestration engine. The operations like filtration, buffer creation, intersection, and display of data are realized, helping in the efficient resolution of spatial queries. The orchestration engine abstracts the user query done by feature service, processing service, and map service, respectively. All the available services are published with metadata in the service catalog. The sequence of services is automated by the orchestration engine getting information from the catalog service. According to the need for geospatial queries, synchronizing such services and executing several virtual machines is challenging. The parallel execution of some services in the cloud may decrease the spatial query execution time.

Chapter 4 investigated the accuracy of geospatial query execution time estimation and computational resources. It helps in query scheduling based on the user-deadline and budget of the query processing. Furthermore, we can monitor the queries' progress, and for bulk query processing, particular queries taking an

unreasonably long time can be identified and eliminated apriori. Also, it helps in system sizing or obtaining the approximate estimation of the total budget or resource utilization. Our proposed framework, LYRIC, has three main components. First, it models the cost of an incoming spatio-temporal query based on the known PostgreSQL's cost model. However, instead of the PostgreSQL default parameters, LYRIC extracts the accurate CPU and disk-access cost and effectively predicts the query execution time. Next, it identifies several spatio-temporal services required to complete the query processing task and further decomposes it into a query tree. LYRIC is capable of considering the user-defined timeline and given budget for each query. The framework utilizes the concept of cooperative game theory to obtain the trade-off between more resources and budget or cost. LYRIC is deployed in GCP, and real-life experiments with mobility datasets and simulations yield encouraging results.

Chapter 5 proposed a fog computing-based architecture, namely Spatio-Fog, for geospatial query resolution. The fog devices of different regions contain geospatial data of the respective regions. When a query is received from a mobile device regarding the current region, the fog device resolves the query after analysing the data and responds to the mobile device. Otherwise, if the query is regarding other regions, the fog device responds using the corresponding region's cloud servers or fog device. Theoretical analysis shows that the proposed framework reduces the power consumption and delay by approximately 43-47% and 47-83% respectively than the existing system. The experimental results illustrate that the proposed framework reduces the power consumption and delay by 30-60% approximately than the existing query resolution system. Thus, we conclude that the proposed framework is a green and delay aware framework.

Chapter 6 presented an end-to-end framework, namely RESCUE, which is capable of assisting users in the time of emergency (say, natural disaster) by predicting path in the post-disaster scenario when several roads are non-reachable. The framework's major working modules are: accumulating and refining crowd-sourcing data and extracting the correlations among the event and the spatial location. RESCUE collects the BAN data consisting of health-parameters' data of users and other environmental parameters' data from IoT devices. It analyses the data in the fog nodes, and in case any abnormality is found, the data is sent to the cloud server immediately. Further, RESCUE is conducive to predict routes to users in the time of exigency, avoiding the affected regions in minimal commute time. While the IoT paradigm enables global connectivity worldwide, communicating with billions of interconnected

devices, the whole process's energy consumption is rising steeply. In this regard, *Green IoT computing* has a pivotal role in reducing environmental problems and subsequently creating a sustainable environment by emphasizing energy-efficient technologies. It reduces the greenhouse gas emission and provides a *smarter* and *greener* view of varied applications. To this end, this chapter's major aim is to reduce energy consumption and contribute towards sustainable urban development. It may be noted that the mobility analysis module is also capable of predicting routes from source to destination, which requires less fuel consumption. The autoencoder process of the mobility analysis module helps in facilitating this feature as well. In all sense, RESCUE provides a *green healthcare and user assistance* framework in the time of exigency.

7.2 Future Directions

In this thesis, we have addressed several challenges for geospatial query processing in cloud-fog-edge computing environments. However, there are several other issues in this domain are remain unresolved. These unresolved issues are pointed towards the future scope of this area. In this section, we are discussed such future scopes.

- **Multi-cloud environment:** Several cloud service providers take part in a multi-cloud environment. Each cloud provider facilitates different geospatial services. The user can select geospatial services and cloud resources based on their geospatial service requirements and budget. Game theory can be a promising approach for such selection procedures.

- **Use of dew computing:** Network connectivity becomes a problem if the mobile client is present at a remote location. Hence, the client needs service with minimal network facility, even in offline mode. If such a mobile user has a geospatial query, then to resolve it, dew computing can be a solution. Thus, creating, updating, and access to raw spatial data set for resolving the geospatial query for the devices connected in offline mode, use of dew computing will be a promising future research scope in this domain.

- **Hierarchical distribution:** The geospatial query processing framework further extends in the hierarchical distributed platform, such as MapReduce, where specific geospatial query processing tasks will be segmented into ”map” and ”reduce” tasks maintaining the spatial proximity rules.

- **Advanced energy preserving model:** Intermediate devices are not always used while participating in geospatial query processing. Dynamic voltage and frequency scaling (DVFS) can be a promising model to maximize the power and energy savings of the computing devices when they are not required. It helps to reduce the overall energy consumption of the query processing framework.
- **Advanced learning approaches:** Reinforcement based learning techniques help in geospatial query optimization. As huge computational power is required for spatio-temporal query processing, an effective reinforcement learning agent will optimize spatial-join and search strategies.
- **Security aspects:** Geospatial data security is one of the major concerns in this domain. Data can tamper between two intermediate fog and edge nodes while data transferring happens for query processing. Blockchain technology helps in this regard for its distributed nature. Geospatial data will generate from the secured devices, and generated data will be treated as transactions on the blockchain. Data will validate with a smart contract. Although any geospatial data tamper during the transmission, it can easily be identified through this modern technology.
- **Geospatial services as FaaS:** A more obvious technology to use for implementation of the distributed geospatial services would be Function-as-a-Service (FaaS). The healthcare service can be provisioned as a FaaS layer, where the spatio-temporal data analytics or machine learning models can be executed independently to deploy different functionalities to individual user. Also, FaaS layer can determine the geospatial service cost.

7.3 Final Remarks

Various researches are going on different computing domains. This thesis investigated how the geospatial queries are performed in Cloud, Fog, Edge computing environments. We have also measured the energy and delay consumption in all computing environments. The proposed layered architectures, algorithms, mathematical models optimizes the computing resources considering user budget and query deadline. The research outcome of this thesis will inspire future scholars to work in interdisciplinary domains like GIS and Computer Science.

Publications

1. **Jaydeep Das**, Shreya Ghosh, Soumya K. Ghosh, and Rajkumar Buyya. *LYRIC: Deadline and Budget Aware Spatio-Temporal Query Processing in Cloud*. IEEE Transactions on Services Computing, April 2021. DOI:10.1109/TSC.2021.3073006

2. **Jaydeep Das**, Anwesha Mukherjee, Soumya K. Ghosh, and Rajkumar Buyya. *Spatio-Fog: A Green and Timeliness-oriented Fog Computing Model for Geospatial Query Resolution*. Simulation Modelling Practice and Theory, Elsevier, Volume 100, Article 102043, ISSN: 1569-190X, April 2020.

3. **Jaydeep Das**, Shreya Ghosh, Anwesha Mukherjee, Soumya K. Ghosh, and Rajkumar Buyya. *RESCUE: Green Healthcare Services Using Integrated IoT-Edge-Fog-Cloud Computing Environments*. Journal of Software: Practice and Experience.(In Revision)

4. **Jaydeep Das**, Soumya K. Ghosh, and Rajkumar Buyya. *Geospatial Edge-Fog Computing: A Systematic Review, Taxonomy, and Future Directions*. Book Chapter in Mobile Edge Computing, Springer, ISBN: 978-3-030-69892-8, 2021.

5. **Jaydeep Das**. *Geolocation aware IoT and Cloud-Fog based Solutions for Healthcare.* Book chapter in Machine Learning, Big Data and IoT for Medical Informatics, ISBN: 978-0-128-21777-1, June 2021.

6. **Jaydeep Das**, Anwesha Mukherjee, Soumya K. Ghosh, and Rajkumar Buyya. *Geo-Cloudlet: Time and Power Efficient Geospatial Query Resolution using Cloudlet*. In Proceedings of the 11^{th} International Conference on Advanced Computing (ICoAC) 18-20 December 2019, Chennai, India, pp. 180-187.

7. **Jaydeep Das**, Sourav Kanti Addya, Soumya K. Ghosh, and Rajkumar Buyya. *Optimal Geospatial Query Placement in Cloud*. In Proceedings of International Conference on Intelligent and Cloud Computing (ICICC), 16-17 December 2019, Springer, Bhubaneswar, India, pp. 335-344.

8. **Jaydeep Das**, Arindam Dasgupta, Soumya K. Ghosh, and Rajkumar Buyya. *A Learning Technique for VM Allocation to Resolve Geospatial Queries*. In Proceedings

of the 5th International Conference on Advanced Computing, Networking, and Informatics (ICACNI), 1-3 June 2017, Springer, Goa, India, pp. 577-584.

9. **Jaydeep Das**, Arindam Dasgupta, Soumya K. Ghosh, and Rajkumar Buyya. *A Geospatial Orchestration Framework on Cloud for Processing User Queries*. In Proceedings of the 5th IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 19-21 October 2016, IEEE, Bangaluru, India, pp. 1-8.

10. Shreya Ghosh, **Jaydeep Das**, Soumya K. Ghosh, Rajkumar Buyya. *CLAWER: Context-aware Cloud-Fog based Workflow Management Framework for Health Emergency Services*. In Proceedings of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 11–14 May 2020, Melbourne, Australia, pp. 810-817.

11. Shreya Ghosh, **Jaydeep Das**, Soumya K. Ghosh. *LOCATOR : A CLoud-FOg-enabled Framework for Facilitating Efficient LoCATiOn based SeRvices*. In Proceedings of the 12th IEEE International Conference on COMmunication Systems & NETworkS (COMSNET), 7-11 January 2020, Bengaluru, India, pp. 87-92.

12. Omprakash Chakraborty, **Jaydeep Das**, Arindam Dasgupta, Pabitra Mitra, and Soumya K. Ghosh. *A Geospatial Service Oriented Framework for Disaster Risk Zone Identification*. In Proceedings of the 16th International Conference on Computational Science and Its Applications (ICCSA), 4-7 July 2016, Springer, Beijing, China, pp. 44-56.

Bibliography

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[2] C. Yang and Q. Huang, *Spatial cloud computing: a practical approach*. CRC Press, 2013.

[3] 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.

[4] S. Supavetch and S. Chunithipaisan, "Interface independent geospatial services orchestration," *Information Technology Journal*, vol. 10, no. 6, pp. 1126–1137, 2011.

[5] L. Di, P. Zhao, W. Yang, G. Yu, and P. Yue, "Intelligent geospatial web services," in *Proceedings of IEEE International Geoscience and Remote Sensing Symposium,(IGARSS)*, vol. 2. IEEE, 2005, pp. 1229–1232.

[6] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, no. 4, pp. 14–23, 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*, vol. 7, no. 1, pp. 141–154, 2016.

[8] A. V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[9] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big data and internet of things: A roadmap for smart environments*. Springer, 2014, pp. 169–186.

[10] A. Mukherjee, P. Deb, D. De, and R. Buyya, "C2OF2N: A low power cooperative code offloading method for femtolet-based fog network," *The Journal of Supercomputing*, vol. 74, no. 6, pp. 2412–2448, 2018.

[11] H. Das, R. K. Barik, H. Dubey, and D. S. Roy, *Cloud Computing for Geospatial Big Data Analytics: Intelligent Edge, Fog and Mist Computing*. Springer, 2018, vol. 49.

[12] S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya, G. S. Wander, and R. Buyya, "Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments," *Future Generation Computer Systems*, vol. 104, pp. 187–200, 2020.

[13] R. R. Vatsavai, B. Ramachandra, Z. Chen, and J. Jernigan, "geoEdge: a real-time analytics framework for geospatial applications," in *Proceedings of the 8th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, 2019, pp. 1–4.

[14] F. W. Nugroho, S. Suryono, and J. E. Suseno, "Fog computing for monitoring of various area mapping pollution carbon monoxide (co) with ordinary kriging method," in *2019 Fourth International Conference on Informatics and Computing (ICIC)*. IEEE, 2019, pp. 1–6.

[15] T. Tsubaki, R. Ishibashi, T. Kuwahara, and Y. Okazaki, "Effective disaster recovery for edge computing against large-scale natural disasters," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–2.

[16] S. Shekhar and S. Chawla, *A tour of spatial databases*. Prentice Hall Upper Saddle River, 2003.

[17] R. H. Güting, "An introduction to spatial database systems," *the VLDB Journal*, vol. 3, no. 4, pp. 357–399, 1994.

[18] H.-H. Park, C.-G. Lee, Y.-J. Lee, and C.-W. Chung, "Early separation of filter and refinement steps in spatial query optimization," in *dasfaa*. IEEE, 1999, p. 161.

[19] E. P. Chan, "Buffer queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 895–910, 2003.

[20] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 201–212.

[21] J. Zhang, N. Mamoulis, D. Papadias, and Y. Tao, "All-nearest-neighbors queries in spatial databases," in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*. IEEE, 2004, pp. 297–306.

[22] Y. Chen and J. M. Patel, "Efficient evaluation of all-nearest-neighbor queries," in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE, 2007, pp. 1056–1065.

[23] E. H. Jacox and H. Samet, "Spatial join techniques," *ACM Transactions on Database Systems*, vol. 32, no. 1, p. 7, 2007.

[24] T. Brinkhoff, H.-P. Kriegel, R. Schneider, and B. Seeger, *Multi-step processing of spatial joins*. ACM, 1994, vol. 23, no. 2.

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

[26] P. Schut and A. Whiteside, "Opengis web processing service," *OpenGIS Standard, Version 1.0. 0, OGC 05-007r7*, 2007.

[27] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, pp. 27–42, 2017.

[28] E. Ahmed and M. H. Rehmani, "Mobile edge computing: opportunities, solutions, and challenges," 2017.

[29] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of the Sixth International Conference on Advances in Future Internet*. Citeseer, 2014, pp. 48–55.

[30] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[31] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[32] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3512–3524, 2018.

[33] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[34] 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.

[35] R. K. Barik, H. Dubey, and K. Mankodiya, "SOA-FOG: secure service-oriented edge computing architecture for smart health big data analytics," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2017, pp. 477–481.

[36] T. N. Gia and M. Jiang, "Exploiting fog computing in health monitoring," *Fog and Edge Computing: Principles and Paradigms*, pp. 291–318, 2019.

[37] D. Chemodanov, P. Calyam, and K. Palaniappan, "Fog computing to enable geospatial video analytics for disaster-incident situational awareness," *Fog Computing: Theory and Practice*, pp. 473–503, 2020.

[38] M. A. Zamora-Izquierdo, J. Santa, J. A. Martínez, V. Martínez, and A. F. Skarmeta, "Smart farming iot platform based on edge and cloud computing," *Biosystems engineering*, vol. 177, pp. 4–17, 2019.

[39] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," 2015.

[40] C. Chang, S. N. Srirama, and R. Buyya, "Internet of things (iot) and new computing paradigms," *Fog and edge computing: principles and paradigms*, pp. 1–23, 2019.

[41] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[42] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for iot: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.

[43] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE access*, vol. 5, pp. 9882–9910, 2017.

[44] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource management approaches in fog computing: a comprehensive review," *Journal of Grid Computing*, pp. 1–42, 2019.

[45] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–37, 2019.

[46] P. Jiang, T. Fana, H. Gao, W. Shi, L. Liu, C. Cérin, and J. Wan, "Energy aware edge computing: A survey," *Computer Communications*, vol. 151, pp. 556–580, 2020.

[47] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog computing in healthcare–a review and discussion," *IEEE Access*, vol. 5, pp. 9206–9222, 2017.

[48] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, "Edge-oriented computing paradigms: A survey on architecture design and system management," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–34, 2018.

[49] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*. Springer, 2018, pp. 103–130.

[50] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application management in fog computing environments: A taxonomy, review and future directions," *ACM Computing Surveys*, 2020.

[51] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2017.

[52] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[53] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE access*, vol. 6, pp. 47 980–48 009, 2018.

[54] S. B. Nath, H. Gupta, S. Chakraborty, and S. K. Ghosh, "A survey of fog computing and communication: current researches and future directions," *arXiv preprint arXiv:1804.04365*, 2018.

[55] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K.-K. R. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live vm migration framework," *IEEE Access*, vol. 5, pp. 8284–8300, 2017.

[56] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the internet of things: A survey," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–41, 2019.

[57] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–43, 2017.

[58] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586–2595, 2017.

[59] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289—-330, 2019.

[60] P. Zhang, M. Zhou, and G. Fortino, "Security and trust issues in fog computing: A survey," *Future Generation Computer Systems*, vol. 88, pp. 16–27, 2018.

[61] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[62] Z. Liu, "Typical characteristics of cloud gis and several key issues of cloud spatial decision support system," in *2013 IEEE 4th International Conference on Software Engineering and Service Science*. IEEE, 2013, pp. 668–671.

[63] A. Rezgui, Z. Malik, and C. Yang, "High-resolution spatial interpolation on cloud platforms," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 377–382.

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

[65] Z. Li, C. Yang, Q. Huang, K. Liu, M. Sun, and J. Xia, "Building model as a service to support geosciences," *Computers, Environment and Urban Systems*, vol. 61, pp. 141–152, 2017.

[66] T. Xing, S. Zhang, and L. Tao, "Cloud-based spatial information service architecture within lbs," *Positioning*, vol. 2014, 2014.

[67] Y. Shi and F. Bian, "The design and application of the gloud gis," in *International Conference on Geo-Informatics in Resource Management and Sustainable Ecosystem*. Springer, 2014, pp. 56–67.

[68] Y. Wang, S. Wang, and D. Zhou, "Retrieving and indexing spatial data in the cloud computing environment," in *IEEE International Conference on Cloud Computing*. Springer, 2009, pp. 322–331.

[69] L.-Y. Wei, Y.-T. Hsu, W.-C. Peng, and W.-C. Lee, "Indexing spatial data in cloud data managements," *Pervasive and Mobile Computing*, vol. 15, pp. 48–61, 2014.

[70] V. Siládi, L. Huraj, N. Polčák, and E. Vesel, "A parallel processing of spatial data interpolation on computing cloud," in *Proceedings of the Fifth Balkan Conference in Informatics*, 2012, pp. 193–198.

[71] R. C. Mateus, T. L. L. Siqueira, V. C. Times, R. R. Ciferri, and C. D. de Aguiar Ciferri, "Spatial data warehouses and spatial olap come towards the cloud: design and performance," *Distributed and parallel databases*, vol. 34, no. 3, pp. 425–461, 2016.

[72] S. J. Park and J. S. Yoo, "Leveraging cloud computing for spatial association mining," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2014, pp. 4152–4153.

[73] Y. Zhong, J. Han, T. Zhang, and J. Fang, "A distributed geospatial data storage and processing framework for large-scale webgis," in *2012 20th International Conference on Geoinformatics*. IEEE, 2012, pp. 1–7.

[74] R. Sugumaran, J. Burnett, and A. Blinkmann, "Big 3D spatial data processing using cloud computing environment," in *Proceedings of the 1st ACM SIGSPATIAL international workshop on analytics for big geospatial data*, 2012, pp. 20–22.

[75] G. Zhang, Q. Huang, A.-X. Zhu, and J. H. Keel, "Enabling point pattern analysis on spatial big data using cloud computing: optimizing and accelerating ripley's k function," *International Journal of Geographical Information Science*, vol. 30, no. 11, pp. 2230–2252, 2016.

[76] S. You, J. Zhang, and L. Gruenwald, "Large-scale spatial join query processing in cloud," in *2015 31st IEEE International Conference on Data Engineering Workshops*. IEEE, 2015, pp. 34–41.

[77] S. You, J. Zhang, and L. Gruenwald, "Spatial join query processing in cloud: Analyzing design choices and performance comparisons," in *2015 44th International Conference on Parallel Processing Workshops*. IEEE, 2015, pp. 90–97.

[78] V. Prokhorenko and M. A. Babar, "Architectural resilience in cloud, fog and edge systems: A survey," *IEEE Access*, vol. 8, pp. 28 078–28 095, 2020.

[79] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: architecture and service modes," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 18–24, 2015.

[80] A. Mukherjee, D. G. Roy, and D. De, "Mobility-aware task delegation model in mobile cloud computing," *The Journal of Supercomputing*, vol. 75, no. 1, pp. 314–339, 2019.

[81] J. Michel and C. Julien, "A cloudlet-based proximal discovery service for machine-to-machine applications," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2013, pp. 215–232.

[82] M. Uehara, "Mist computing: Linking cloudlet to fogs," in *International Conference on Computational Science/Intelligence & Applied Informatics*. Springer, 2017, pp. 201–213.

[83] J. S. Preden, K. Tammemäe, A. Jantsch, M. Leier, A. Riid, and E. Calis, "The benefits of self-awareness and attention in fog and mist computing," *Computer*, vol. 48, no. 7, pp. 37–45, 2015.

[84] 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.

[85] R. K. Barik, A. C. Dubey, A. Tripathi, T. Pratik, S. Sasane, R. K. Lenka, H. Dubey, K. Mankodiya, and V. Kumar, "Mist data: leveraging mist computing for secure and scalable architecture for smart and connected health," *Procedia Computer Science*, vol. 125, pp. 647–653, 2018.

[86] S. Ghosh, A. Mukherjee, S. K. Ghosh, and R. Buyya, "Mobi-IoST: mobility-aware cloud-fog-edge-iot collaborative framework for time-critical applications," *IEEE Transactions on Network Science and Engineering*, 2019.

[87] M. Mishra, S. K. Roy, A. Mukherjee, D. De, S. K. Ghosh, and R. Buyya, "An energy-aware multi-sensor geo-fog paradigm for mission critical applications," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2019.

[88] A. Olasz and B. Nguyen Thai, "Geospatial big data processing in an open source distributed computing environment," *PeerJ Preprints*, vol. 4, p. e2226v1, 2016.

[89] E. M. Xavier, F. J. Ariza-López, and M. A. Ureña-Cámara, "A survey of measures and methods for matching geospatial vector datasets," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, pp. 1–34, 2016.

[90] M. R. Palattella, R. Soua, A. Khelil, and T. Engel, "Fog computing as the key for seamless connectivity handover in future vehicular networks," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 1996–2000.

[91] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[92] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular adhoc network with fog computing," in *2015*

IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, 2015, pp. 1202–1207.

[93] M. Arif, G. Wang, V. E. Balas, O. Geman, A. Castiglione, and J. Chen, "Sdn based communications privacy-preserving architecture for vanets using fog computing," *Vehicular Communications*, p. 100265, 2020.

[94] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*. ACM, 2015, pp. 37–42.

[95] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[96] B. Wu, X. Wu, and J. Huang, "Geospatial data services within cloud computing environment," in *2010 International Conference on Audio, Language and Image Processing*. IEEE, 2010, pp. 1577–1584.

[97] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 1, pp. 108–119, 2015.

[98] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *International conference on wireless algorithms, systems, and applications*. Springer, 2015, pp. 685–695.

[99] P. Bhattacharya, S. Tanwar, R. Shah, and A. Ladha, "Mobile edge computing-enabled blockchain framework—a survey," in *Proceedings of ICRIC 2019*. Springer, 2020, pp. 797–809.

[100] Q. Li, S. Meng, S. Zhang, J. Hou, and L. Qi, "Complex attack linkage decision-making in edge computing networks," *IEEE Access*, vol. 7, pp. 12 058–12 072, 2019.

[101] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, "A secure iot service architecture with an efficient balance dynamics based on cloud and edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4831–4843, 2018.

[102] K. Hammoudi, F. Dornaika, B. Soheilian, and N. Paparoditis, "Extracting wire-frame models of street facades from 3d point clouds and the corresponding cadastral map," *IAPRS*, vol. 38, no. Part 3A, pp. 91–96, 2010.

[103] P. K. Agarwal, L. Arge, and A. Danner, "From point cloud to grid dem: A scalable approach," in *Progress in Spatial Data Handling*. Springer, 2006, pp. 771–788.

[104] Y. Hu, "Geo-text data and data-driven geospatial semantics," *Geography Compass*, vol. 12, no. 11, p. e12404, 2018.

[105] M. J. De Smith, M. F. Goodchild, and P. Longley, *Geospatial analysis: a comprehensive guide to principles, techniques and software tools*. Troubador publishing ltd, 2007.

[106] A. Kamilaris and F. O. Ostermann, "Geospatial analysis and the internet of things," *ISPRS international journal of geo-information*, vol. 7, no. 7, p. 269, 2018.

[107] K. Puri, G. Areendran, K. Raj, S. Mazumdar, and P. Joshi, "Forest fire risk assessment in parts of northeast india using geospatial tools," *Journal of forestry research*, vol. 22, no. 4, p. 641, 2011.

[108] M. Sharifikia, "Vulnerability assessment and earthquake risk mapping in part of north iran using geospatial techniques," *Journal of the Indian Society of Remote Sensing*, pp. 708–716, 2010.

[109] N. Wood, J. Jones, J. Schelling, and M. Schmidtlein, "Tsunami vertical-evacuation planning in the us pacific northwest as a geospatial, multi-criteria decision problem," *International journal of disaster risk reduction*, vol. 9, pp. 68–83, 2014.

[110] E. M. Delmelle, H. Zhu, W. Tang, and I. Casas, "A web-based geospatial toolkit for the monitoring of dengue fever," *Applied Geography*, vol. 52, pp. 144–152, 2014.

[111] A. I. J. Tostes, F. de LP Duarte-Figueiredo, R. Assunção, J. Salles, and A. A. Loureiro, "From data to knowledge: city-wide traffic flows analysis and prediction using bing maps," in *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing*, 2013, pp. 1–8.

[112] A. Kotsev, S. Schade, M. Craglia, M. Gerboles, L. Spinelle, and M. Signorini, "Next generation air quality platform: Openness and interoperability for the internet of things," *Sensors*, vol. 16, no. 3, p. 403, 2016.

[113] A. Kamilaris, A. Assumpcio, A. B. Blasi, M. Torrellas, and F. X. Prenafeta-Boldú, "Estimating the environmental impact of agriculture by means of geospatial and big data analysis: The case of catalonia," in *From Science to Society*. Springer, 2018, pp. 39–48.

[114] I. A. Jalil, A. R. A. Rasam, N. A. Adnan, N. M. Saraf, and A. N. Idris, "Geospatial network analysis for healthcare facilities accessibility in semi-urban areas," in *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*. IEEE, 2018, pp. 255–260.

[115] A. Kamilaris and A. Pitsillides, "A web-based tourist guide mobile application," in *Proceedings of the International Conference on Sustainability, Technology and Education (STE), Kuala Lumpur, Malaysia*, vol. 29, 2013.

[116] S. Ghosh, A. Chowdhury, and S. K. Ghosh, "A machine learning approach to find the optimal routes through analysis of gps traces of mobile city traffic," in *Recent Findings in Intelligent Computing Techniques*. Springer, 2018, pp. 59–67.

[117] S. Ghosh and S. K. Ghosh, "Thump: Semantic analysis on trajectory traces to explore human movement pattern," in *Proceedings of the 25th International Conference Companion on World Wide Web*, 2016, pp. 35–36.

[118] M. Van Setten, S. Pokraev, and J. Koolwaaij, "Context-aware recommendations in the mobile tourist application compass," in *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer, 2004, pp. 235–244.

[119] J. S. Brownstein, C. C. Freifeld, B. Y. Reis, and K. D. Mandl, "Surveillance sans frontieres: Internet-based emerging infectious disease intelligence and the healthmap project," *PLoS medicine*, vol. 5, no. 7, 2008.

[120] O. Chakraborty, A. Das, A. Dasgupta, P. Mitra, S. K. Ghosh, and T. Mazumder, "A multi-objective framework for analysis of road network vulnerability for relief facility location during flood hazards: A case study of relief location analysis in bankura district, india," *Transactions in GIS*, vol. 22, no. 5, pp. 1064–1082, 2018.

[121] A. Dasgupta, S. K. Ghosh, and P. Mitra, "A technique for assessing the quality of volunteered geographic information for disaster decision making," in *International Conference on Computational Science and Its Applications*. Springer, 2018, pp. 589–597.

[122] S. Pal and S. K. Ghosh, "Rule based end-to-end learning framework for urban growth prediction," *arXiv preprint arXiv:1711.10801*, 2017.

[123] V. Miz and V. Hahanov, "Smart traffic light in terms of the cognitive road traffic management system (ctms) based on the internet of things," in *Proceedings of IEEE East-West Design & Test Symposium (EWDTS 2014)*. IEEE, 2014, pp. 1–5.

[124] E. D. Ayele, K. Das, N. Meratnia, and P. J. Havinga, "Leveraging ble and lora in iot network for wildlife monitoring system (wms)," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. IEEE, 2018, pp. 342–348.

[125] N. Cressie, *Statistics for spatial data*. John Wiley & Sons, 2015.

[126] S. Bhattacharjee, P. Mitra, and S. K. Ghosh, "Spatial interpolation to predict missing attributes in gis using semantic kriging," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 8, pp. 4771–4780, 2013.

[127] A. C. Clements, H. L. Reid, G. C. Kelly, and S. I. Hay, "Further shrinking the malaria map: how can geospatial science help to achieve malaria elimination?" *The Lancet infectious diseases*, vol. 13, no. 8, pp. 709–718, 2013.

[128] K. Forsythe, K. Paudel, and C. Marvin, "Geospatial analysis of zinc contamination in lake ontario sediments," *Journal of Environmental Informatics*, vol. 16, no. 1, pp. 1–10, 2010.

[129] E.-S. E. Omran, "A proposed model to assess and map irrigation water well suitability using geospatial analysis," *Water*, vol. 4, no. 3, pp. 545–567, 2012.

[130] F. Liu, Y. Guo, Z. Cai, N. Xiao, and Z. Zhao, "Edge-enabled disaster rescue: a case study of searching for missing people," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 6, pp. 1–21, 2019.

[131] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, 2018.

[132] A. Mukherjee, S. Ghosh, A. Behere, S. K. Ghosh, and R. Buyya, "Internet of health things (ioht) for personalized health care using integrated edge-fog-cloud network," *Journal of Ambient Intelligence and Humanized Computing*, 2020.

[133] X. Zhou, C. Xu, and B. Kimmons, "Detecting tourism destinations using scalable geospatial analysis based on cloud computing platform," *Computers, Environment and Urban Systems*, vol. 54, pp. 144–153, 2015.

[134] R. K. Barik, R. K. Lenka, N. Simha, H. Dubey, and K. Mankodiya, "Fog computing based sdi framework for mineral resources information infrastructure management in india," *arXiv preprint arXiv:1712.09282*, 2017.

[135] M. P. Armstrong, S. Wang, and Z. Zhang, "The internet of things and fast data streams: prospects for geospatial data science in emerging information ecosystems," *Cartography and Geographic Information Science*, vol. 46, no. 1, pp. 39–56, 2019.

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

[137] X. Cao and S. Madria, "Efficient geospatial data collection in iot networks for mobile edge computing," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2019, pp. 1–10.

[138] R. Dautov, S. Distefano, D. Bruneo, F. Longo, G. Merlino, A. Puliafito, and R. Buyya, "Metropolitan intelligent surveillance systems for urban areas by harnessing iot and edge computing paradigms," *Software: Practice and Experience*, vol. 48, no. 8, pp. 1475–1492, 2018.

[139] B. Denby and B. Lucia, "Orbital edge computing: Nanosatellite constellations as a new class of computer system," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 939–954.

[140] T. Higashino, "Edge computing for cooperative real-time controls using geospatial big data," in *Smart Sensors and Systems*. Springer, 2017, pp. 441–466.

[141] L. Klein, "Geospatial internet of things: Framework for fugitive methane gas leaks monitoring," in *International Conference on GIScience Short Paper Proceedings*, vol. 1, no. 1, 2016.

[142] S. Liu, X. Chen, B. Qi, and L. Zherr, "Performace oriented edge computing of geospatial information with 3d scenery," in *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE, 2018, pp. 853–858.

[143] W. Richardson, H. Krishnaswami, R. Vega, and M. Cervantes, "A low cost, edge computing, all-sky imager for cloud tracking and intra-hour irradiance forecasting," *Sustainability*, vol. 9, no. 4, p. 482, 2017.

[144] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[145] R. Mahmud and R. Buyya, "Modelling and simulation of fog and edge computing environments using ifogsim toolkit," *Fog and edge computing: Principles and paradigms*, pp. 1–35, 2019.

[146] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "Fogbus: A blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, vol. 154, pp. 22–36, 2019.

[147] P. Yue, H. Zhou, J. Gong, and L. Hu, "Geoprocessing in cloud computing platforms–a comparative analysis," *International Journal of Digital Earth*, vol. 6, no. 4, pp. 404–425, 2013.

[148] A. Dasgupta and S. Ghosh, "A framework for ubiquitous geospatial information integration on mobile device using orchestration of geoservices," *International Journal Of UbiComp (IJU)*, vol. 1, no. 3, pp. 69–88, 2010.

[149] S. S. Walia, A. Dasgupta, and S. K. Ghosh, "Geospatial orchestration framework for resolving complex user query," in *International Conference on Computational Science and Its Applications*. Springer, 2011, pp. 643–651.

[150] L. Bernard and A. Wytzisk, "A web-based service architecture for distributed spatiotemporal modeling," in *Proceedings of the 5th AGILE Conference on Geographic Information Science*, vol. 2002, 2002, pp. 25–27.

[151] A. Barker and R. Buyya, "Decentralised orchestration of service-oriented scientific workflows." in *CLOSER*, 2011, pp. 222–231.

[152] C. Peltz, "Web services orchestration and choreography," *Computer*, vol. 36, no. 10, pp. 46–52, 2003.

[153] P. Yue, P. Baumann, K. Bugbee, and L. Jiang, "Towards intelligent giservices," *Earth Science Informatics*, vol. 8, no. 3, pp. 463–481, 2015.

[154] V. Cardellini, V. Di Valerio, and F. L. Presti, "Game-theoretic resource pricing and provisioning strategies in cloud systems," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 86–98, 2020.

[155] G. Yao, Q. Ren, X. Li, S. Zhao, and R. Ruiz, "A hybrid fault-tolerant scheduling for deadline-constrained tasks in cloud systems," *IEEE Transactions on Services Computing*, 2020.

[156] R. Marcus, S. Semenova, and O. Papaemmanouil, "A learning-based service for cost and performance management of cloud databases," in *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 2017, pp. 1361–1362.

[157] W. Wu, Y. Chi, S. Zhu, J. Tatemura, H. Hacigümüs, and J. F. Naughton, "Predicting query execution time: Are optimizer cost models really unusable?" in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 1081–1092.

[158] W. Wu, Y. Chi, H. Hacígümüş, and J. F. Naughton, "Towards predicting query execution time for concurrent and dynamic database workloads," *Proceedings of the VLDB Endowment*, vol. 6, no. 10, pp. 925–936, 2013.

[159] A. Ganapathi, H. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. Jordan, and D. Patterson, "Predicting multiple metrics for queries: Better decisions enabled by machine learning," in *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE, 2009, pp. 592–603.

[160] M. Akdere, U. Çetintemel, M. Riondato, E. Upfal, and S. B. Zdonik, "Learning-based query performance modeling and prediction," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 2012, pp. 390–401.

[161] S. Ramesh, A. Baranawal, and Y. Simmhan, "A distributed path query engine for temporal property graphs," in *Proceedings of* 20^{th} *IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*. IEEE, 2020, pp. 499–508.

[162] X. Zhou, J. Sun, G. Li, and J. Feng, "Query performance prediction for concurrent queries using graph embedding," *Proceedings of the VLDB Endowment*, vol. 13, no. 9, 2020.

[163] Z. Chu, J. Yu, and A. Hamdulla, "A novel deep learning method for query task execution time prediction in graph database," *Future Generation Computer Systems*, 2020.

[164] J. Duggan, O. Papaemmanouil, U. Cetintemel, and E. Upfal, "Contender: A resource modeling approach for concurrent query performance prediction." in *EDBT*, 2014, pp. 109–120.

[165] M. Ahmad, A. Aboulnaga, S. Babu, and K. Munagala, "Modeling and exploiting query interactions in database systems," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 183–192.

[166] M. Ahmad, A. Aboulnaga, S. Babu, and K. Munagala, "Qshuffler: Getting the query mix right," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE, 2008, pp. 1415–1417.

[167] M. Ahmad, A. Aboulnaga, and S. Babu, "Query interactions in database workloads," in *Proceedings of the Second International Workshop on Testing Database Systems*. ACM, 2009, p. 11.

[168] A. D. Popescu, V. Ercegovac, A. Balmin, M. Branco, and A. Ailamaki, "Same queries, different data: Can we predict runtime performance?" in *2012 IEEE 28th International Conference on Data Engineering Workshops*. IEEE, 2012, pp. 275–280.

[169] P. Yue, L. Di, W. Yang, G. Yu, and P. Zhao, "Semantics-based automatic composition of geospatial web service chains," *Computers & Geosciences*, vol. 33, no. 5, pp. 649–665, 2007.

[170] P. Zhao, L. Di, G. Yu, P. Yue, Y. Wei, and W. Yang, "Semantic web-based geospatial knowledge transformation," *Computers & Geosciences*, vol. 35, no. 4, pp. 798–808, 2009.

[171] L. Di, P. Zhao, W. Yang, and P. Yue, "Ontology-driven automatic geospatial processing modeling based on web-service chaining," in *Proceedings of the sixth annual NASA earth science technology conference*. Citeseer, 2006, pp. 27–29.

[172] P. Yue, J. Gong, and L. Di, "Augmenting geospatial data provenance through metadata tracking in geospatial service chaining," *Computers & Geosciences*, vol. 36, no. 3, pp. 270–281, 2010.

[173] V. W. Chu, R. K. Wong, S. Fong, and C.-H. Chi, "Emerging service orchestration discovery and monitoring," *IEEE Transactions on Services Computing*, vol. 10, no. 6, pp. 889–901, 2015.

[174] X. Tan, L. Di, M. Deng, A. Chen, F. Huang, C. Peng, M. Gao, Y. Yao, and Z. Sha, "Cloud and agent-based geospatial service chain: A case study of submerged crops analysis during flooding of the yangtze river basin," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 3, pp. 1359–1370, 2014.

[175] C. Jatoth, G. Gangadharan, and R. Buyya, "Computational intelligence based qos-aware web service composition: a systematic literature review," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 475–492, 2015.

[176] P. J. Haas, J. F. Naughton, S. Seshadri, and A. N. Swami, "Selectivity and cost estimation for joins based on random sampling," *Journal of Computer and System Sciences*, vol. 52, no. 3, pp. 550–569, 1996.

[177] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," in *ACM Sigmod Record*, vol. 19, no. 2. ACM, 1990, pp. 322–331.

[178] S. Ghosh, S. K. Ghosh, and R. Buyya, "Mario: A spatio-temporal data mining framework on google cloud to explore mobility dynamics from taxi trajectories," *Journal of Network and Computer Applications*, vol. 164, p. 102692, 2020.

[179] K. Lee, L. Liu, R. Ganti, M. Srivatsa, Q. Zhang, Y. Zhou, and Q. Wang, "Lightweight indexing and querying services for big spatial data," *IEEE Transactions on Services Computing*, vol. 12, no. 3, pp. 630–643, 2018.

[180] R. Li, H. He, R. Wang, Y. Huang, J. Liu, S. Ruan, T. He, J. Bao, and Y. Zheng, "Just: Jd urban spatio-temporal data engine," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1558–1569.

[181] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing en-

vironments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[182] 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.

[183] 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.

[184] S. Li, S. Dragicevic, F. A. Castro, M. Sester, S. Winter, A. Coltekin, C. Pettit, B. Jiang, J. Haworth, A. Stein *et al.*, "Geospatial big data handling theory and methods: A review and research challenges," *ISPRS journal of Photogrammetry and Remote Sensing*, vol. 115, pp. 119–133, 2016.

[185] J.-G. Lee and M. Kang, "Geospatial big data: challenges and opportunities," *Big Data Research*, vol. 2, no. 2, pp. 74–81, 2015.

[186] 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.

[187] J.-H. Hong, Z. L.-T. Su, and E. H.-C. Lu, "A recommendation framework for remote sensing images by spatial relation analysis," *Journal of Systems and Software*, vol. 90, pp. 151–166, 2014.

[188] P. Gerolymatos, S. Sioutas, N. Nodarakis, A. Panaretos, and K. Tsakalidis, "SMaRT: A novel framework for addressing range queries over nonlinear trajectories," *Journal of Systems and Software*, vol. 105, pp. 79–90, 2015.

[189] G. Roumelis, M. Vassilakopoulos, A. Corral, and Y. Manolopoulos, "Efficient query processing on large spatial databases: A performance study," *Journal of Systems and Software*, vol. 132, pp. 165–185, 2017.

[190] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 187–260, 1984.

[191] I. Kamel and C. Faloutsos, "Hilbert R-tree: An improved R-tree using fractals," Tech. Rep., 1993.

[192] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+-Tree: A dynamic index for multi-dimensional objects." Tech. Rep., 1987.

[193] G. R. Hjaltason and H. Samet, "Speeding up construction of PMR quadtree-based spatial indexes," *The VLDB Journal—The International Journal on Very Large Data Bases*, vol. 11, no. 2, pp. 109–137, 2002.

[194] T. Dokeroglu, M. A. Bayir, and A. Cosar, "Robust heuristic algorithms for exploiting the common tasks of relational cloud database queries," *Applied Soft Computing*, vol. 30, pp. 72–82, 2015.

[195] M. Malensek, S. Pallickara, and S. Pallickara, "Analytic queries over geospatial time-series data using distributed hash tables," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1408–1422, 2016.

[196] K. Karantzalos, D. Bliziotis, and A. Karmas, "A scalable geospatial web service for near real-time, high-resolution land cover mapping," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 10, pp. 4665–4674, 2015.

[197] 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.

[198] P. Schut and A. Whiteside, "OpenGIS web processing service, Version 1.0. 0, OGC 05-007r7, Open Geospatial Consortium," *Inc*, vol. 87, 2007.

[199] G. Buehrer, B. W. Weide, and P. A. Sivilotti, "Using parse tree validation to prevent SQL injection attacks," in *Proceedings of the 5th international workshop on Software engineering and middleware*. ACM, 2005, pp. 106–113.

[200] R. P. Haining and R. Haining, *Spatial data analysis: theory and practice*. Cambridge university press, 2003.

[201] T. Brinkhoff, H.-P. Kriegel, and B. Seeger, *Efficient processing of spatial joins using R-trees*. ACM, 1993, vol. 22, no. 2.

[202] A. Mukherjee, D. De, and R. Buyya, "E2R-F2N: Energy-efficient retailing using a femtolet-based fog network," *Software: Practice and Experience*, vol. 49, no. 3, pp. 498–523, 2019.

[203] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[204] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, "A survey on the role of IoT in agriculture for the implementation of smart farming," *IEEE Access*, vol. 7, pp. 156 237–156 271, 2019.

[205] J. J. Rodrigues, D. B. D. R. Segundo, H. A. Junqueira, M. H. Sabino, R. M. Prince, J. Al-Muhtadi, and V. H. C. De Albuquerque, "Enabling technologies for the internet of health things," *IEEE Access*, vol. 6, pp. 13 129–13 141, 2018.

[206] K. A. Eldrandaly, M. Abdel-Basset, and L. A. Shawky, "Internet of spatial things: A new reference model with insight analysis," *IEEE Access*, vol. 7, pp. 19 653–19 669, 2019.

[207] A. Gilchrist, *Industry 4.0: the industrial internet of things*. Springer, 2016.

[208] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[209] L. Yushi, J. Fei, and Y. Hui, "Study on application modes of military internet of things (MIOT)," in *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, vol. 3. IEEE, 2012, pp. 630–634.

[210] D. E. Zheng and W. A. Carter, *Leveraging the internet of things for a more efficient and effective military*. Rowman & Littlefield, 2015.

[211] A. Taherkordi, F. Eliassen, M. Mcdonald, and G. Horn, "Context-driven and real-time provisioning of data-centric iot services in the cloud," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 1, pp. 1–24, 2018.

[212] P. Vinten-Johansen, H. Brody, N. Paneth, S. Rachman, D. Zuck, M. Rip, H. C. A. D. Zuck *et al.*, *Cholera, chloroform, and the science of medicine: a life of John Snow*. Medicine, 2003.

[213] M. A. Albreem, A. A. El-Saleh, M. Isa, W. Salah, M. Jusoh, M. Azizan, and A. Ali, "Green internet of things (IoT): An overview," in *2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*. IEEE, 2017, pp. 1–6.

[214] V. Namboodiri and L. Gao, "Energy-aware tag anticollision protocols for RFID systems," *IEEE Transactions on Mobile Computing*, vol. 9, no. 1, pp. 44–59, 2009.

[215] T. Li, S. S. Wu, S. Chen, and M. C. Yang, "Generalized energy-efficient algorithms for the RFID estimation problem," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1978–1990, 2012.

[216] C.-S. Lee, D.-H. Kim, and J.-D. Kim, "An energy efficient active rfid protocol to avoid overhearing problem," *IEEE Sensors Journal*, vol. 14, no. 1, pp. 15–24, 2013.

[217] S. K. Singh, M. Singh, and D. Singh, "A survey of energy-efficient hierarchical cluster-based routing in wireless sensor networks," *International Journal of Advanced Networking and Application (IJANA)*, vol. 2, no. 02, pp. 570–580, 2010.

[218] R.-S. Chang and C.-J. Kuo, "An energy efficient routing mechanism for wireless sensor networks," in *20th International Conference on Advanced Information Networking and Applications-Volume 1 (AINA'06)*, vol. 2. IEEE, 2006, pp. 5–pp.

[219] C. Schurgers and M. B. Srivastava, "Energy efficient routing in wireless sensor networks," in *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No. 01CH37277)*, vol. 1. IEEE, 2001, pp. 357–361.

[220] S. K. Singh, M. Singh, and D. K. Singh, "Energy-efficient homogeneous clustering algorithm for wireless sensor network," *International Journal of Wireless & Mobile Networks (IJWMN)*, vol. 2, no. 3, pp. 49–61, 2010.

[221] J.-S. Leu, T.-H. Chiang, M.-C. Yu, and K.-W. Su, "Energy efficient clustering scheme for prolonging the lifetime of wireless sensor network with isolated nodes," *IEEE communications letters*, vol. 19, no. 2, pp. 259–262, 2014.

[222] J. Huang, Y. Meng, X. Gong, Y. Liu, and Q. Duan, "A novel deployment scheme for green internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 196–205, 2014.

[223] P. Bharti, D. De, S. Chellappan, and S. K. Das, "HuMAn: complex activity recognition with multi-modal multi-positional body sensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 857–870, 2018.

[224] S. Khalifa, G. Lan, M. Hassan, A. Seneviratne, and S. K. Das, "Harke: Human activity recognition from kinetic energy harvesting data in wearable devices," *IEEE Transactions on Mobile Computing*, vol. 17, no. 6, pp. 1353–1368, 2017.

[225] D. Riboni and C. Bettini, "COSAR: hybrid reasoning for context-aware activity recognition," *Personal and Ubiquitous Computing*, vol. 15, no. 3, pp. 271–289, 2011.

[226] M. F. Goodchild, "Citizens as sensors: the world of volunteered geography," *GeoJournal*, vol. 69, no. 4, pp. 211–221, 2007.

[227] C. Goranson, S. Thihalolipavan, and N. di Tada, "VGI and public health: possibilities and pitfalls," in *Crowdsourcing geographic knowledge*. Springer, 2013, pp. 329–340.

[228] H. Senaratne, A. Mobasheri, A. L. Ali, C. Capineri, and M. Haklay, "A review of volunteered geographic information quality assessment methods," *International Journal of Geographical Information Science*, vol. 31, no. 1, pp. 139–167, 2017.

[229] R. K. Barik, H. Dubey, K. Mankodiya, S. A. Sasane, and C. Misra, "GeoFog4Health: a fog-based SDI framework for geospatial health big data analysis," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 2, pp. 551–567, 2019.

[230] P. Mooney, P. Corcoran, and B. Ciepluch, "The potential for using volunteered geographic information in pervasive health computing applications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 4, no. 6, pp. 731–745, 2013.

[231] C. Tsigkanos, M. Garriga, L. Baresi, and C. Ghezzi, "Cloud deployment tradeoffs for the analysis of spatially distributed internet of things systems," *ACM Transactions on Internet Technology (TOIT)*, vol. 20, no. 2, pp. 1–23, 2020.

[232] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-aware application module management for fog computing environments," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 1, pp. 1–21, 2018.

[233] S. Ghosh, S. K. Ghosh, and R. Buyya, "MARIO: A spatio-temporal data mining framework on google cloud to explore mobility dynamics from taxi trajectories," *Journal of Network and Computer Applications*, vol. 164, p. 102692, 2020.

[234] P. D. Kaur and I. Chana, "Cloud based intelligent system for delivering health care as a service," *Computer methods and programs in biomedicine*, vol. 113, no. 1, pp. 346–359, 2014.

Author's Biography

Jaydeep Das received the B.Tech [2011], and M.E. [2014] degrees from West Bengal University of Technology, and Jadavpur University, Kolkata, India, respectively. He worked towards his Ph.D. dissertation [2015-2021] at the Advanced Technology Development Centre, Indian Institute of Technology Kharagpur, India. He also worked as a Senior Scientific Officer [2014] with the School of Information Technology, IIT Kharagpur. He served as Quality Analyst for client Oracle India Pvt. Ltd. of Randstad India Limited, Bangalore [2012]. His broad research area includes geospatial cloud computing, geospatial information system, and geospatial query resolution using spatial services in the cloud, fog, edge computing environments.

He received Best Paper Award in Computing in 5th International Conference on Advanced Computing, Networking, and Informatics (ICACNI 2017) conference, held at NIT Goa. Jaydeep qualified for various competitive examinations like GATE 2012, UGC-NET June 2019, and WBSET December 2018.

Apart from the research work, Jaydeep assisted faculties in various undergraduate and postgraduate courses of Computer Science and Engineering department during his Ph.D. tenure. He was actively involved in assisting online courses on the National Programme on Technology Enhanced Learning (NPTEL) platform by Govt. of India. He also helped to organize DST workshops at IIT Kharagpur and support the community. Jaydeep loves to solve sudoku, crosswords, and puzzles in his leisure time.

Google Scholar Link:
https://scholar.google.com/citations?user=jQdOJaQAAAAJ&hl=en&oi=ao

Contact Information:

Email ID: jaydeep@iitkgp.ac.in