Combining Fog and Cloud Computing to Support Spatial Analytics in Smart Cities

João Paulo Clarindo[1], João Pedro C. Castro[1,2], Cristina D. Aguiar[1]

[1] Institute of Mathematics and Computer Science – University of São Paulo – Brazil
jpcsantos@usp.br, cdac@icmc.usp.br
[2] Computing Center – Federal University of Minas Gerais – Brazil
jpcarvalhocastro@ufmg.br

Abstract. Spatial data generated by an Internet of Things (IoT) network is important to assist the spatial analytics process in issues related to smart cities. In these cities, IoT devices generate spatial data constantly. Thus, data can get increasingly voluminous very fast. In this article, we investigate the challenge of managing these data through the use of a spatial data warehouse designed over a parallel and distributed data processing framework extended with a spatial analytics system. We propose an architecture aimed to assist a smart city manager in the decision-making process. This architecture integrates a cloud computing layer, where these technologies are located, with a fog computing layer for extracting, transforming, and loading the data into the spatial data warehouse. Furthermore, we introduce a set of guidelines to aid smart city managers to implement the proposed architecture. These guidelines describe and discuss important issues that should be faced by the managers. We validate our architecture with a case study that uses real data collected by IoT devices in a smart city. This case study encompasses the execution of three different categories of spatial queries, demonstrating the architecture's efficacy and effectiveness to support spatial analytics in the context of smart cities.

Categories and Subject Descriptors: H.2.8 [**Database Applications**]: Spatial databases and GIS; H.3.4 [**Systems and Software**]: Information Networks; H.4.2 [**Information Systems Applications**]: Decision support

Keywords: cloud computing, fog computing, internet of things, parallel and distributed data processing, smart cities, spatial analytics, spatial data warehouse

1. INTRODUCTION

In the last few years, the world population has been growing rapidly. From projections made by the United Nations, the population will reach 8 billion people in 2025 [Fraga and Queirolo 2018]. Hence, providing the necessary infrastructure to accommodate a significant amount of people in cities can be a challenge for public authorities and companies. According to Ramaswami et al. (2016), the meta-principles for developing a sustainable and healthy city are "*improvements in transportation, basic sanitation and energy supply*", "*sustainability*", and "*technology integration*". Thus, the concept of smart cities emerged.

There are several definitions of smart cities in the literature, as discussed in [Ismagilova et al. 2019]. In this study, the authors state that there is no agreement regarding the most accepted definition. However, there are elements and terms that are frequently highlighted, such as Information and Communication Technology (ICT), Internet of Things (IoT), interaction, sustainability, citizens, and quality of life. Amongst these definitions, we choose to limit our scope to two that encompass the concepts of ICT and IoT. These definitions are described as follows. Peng et al. (2017) describe smart cities as being "essentially built by utilising a set of advanced ICT, including smart hardware devices (e.g., wireless sensors, smart meters, smart vehicles, and smartphones), mobile networks (e.g.,

Wi-Fi, 3G/4G/5G network), data storage technologies (e.g., data warehouse and cloud platform), and software applications (e.g., back-office control systems, mobile apps, big data analytical tools)". Additionally, Yeh (2017) defines that a smart city "involves the implementation and deployment of ICT infrastructures to support social and urban growth through improving the economy, citizens involvement, and government efficiency".

A network of IoT devices can be used to provide information in a smart city. According to Patel and Patel (2016), IoT can be classified as "*interconnected objects that have data regularly collected, analysed, and used to initiate action, providing a wealth of intelligence for planning, management and decision-making*". The different layers of an IoT architecture include: (i) the *smart device/sensor layer*, which is responsible for collecting data from the environment through the employment of connection standards such as Wi-Fi, GSM and Bluetooth; (ii) the *network layer*, which is composed of gateways and gateway networks that support different communication protocols for sending data to the *service layer*; and (iii) the *service layer*, in which data is processed and prepared to obtain the information required by a desired application [Patel and Patel 2016; Atzori et al. 2017].

The IoT technology is very important in a smart city environment. For instance, it is possible to apply this paradigm in the context of urban mobility, where sensors placed on streets and highways collect data on the number of vehicles and average vehicle speed in order to assist in decision-making for the improvement of urban traffic. Another IoT application scenario includes monitoring a public transport system, whose fleet contains sensors that collect data related to the number of passengers, vehicle type (e.g., buses, trams, etc.), route taken, and maximum speed, aiming to improve the existing lines. Other examples include pollution control, water consumption analysis, electric energy measurements, and tourist attraction measurements [Atzori et al. 2017].

Data generated on these scenarios usually include spatial data, which can be represented by geometries (such as points, lines, and polygons) or combinations of them. For example, sensors inserted in a highway can generate spatial data related to the region where it is located. Furthermore, smartphones can provide location data to determine the number of vehicles on a street that are not equipped with sensors based on Global Navigation Satellite Systems (GNSS). These positioning systems are generally composed of ground receivers and orbital platforms, including the North American Global Positioning System (GPS) and the Russian *Globalnaya Navigatsionnaya Sputnikovaya Sistema* (GLONASS) (English: Global Navigation Satellite System) [Bansal et al. 2021; Eldrandaly et al. 2019].

Performing analytical queries on data generated by an IoT network in a smart city can assist managers in the decision-making process. For instance, a smart city manager can be interested in determining "how many vehicles traveled through a given district, per day, per month". The query results can be displayed on a map according to the district, helping the manager to intuitively obtain the necessary knowledge. In order to enable the execution of this type of query, IoT data needs to be extracted, transformed, and loaded in a Spatial Data Warehouse (SDW). A SDW is a subject-oriented, integrated, time-variant, and non-volatile collection of conventional and spatial data. It provides support for the costly Spatial On-Line Analytical Processing (SOLAP) queries, which are analytical queries extended with spatial predicates [Han et al. 1998; Rivest et al. 2001].

In smart cities, IoT devices generate spatial data constantly [Bonomi et al. 2014]. Also, because sensors all over the city can collect and transmit masses of data, data scale becomes increasingly big [Chen et al. 2014]. To deal with big data, the management of the SDW can benefit from the use of a cloud computing environment as infrastructure and from the employment of a parallel and distributed data processing framework, such as Hadoop [Shvachko et al. 2010] and Spark [Zaharia et al. 2016], to reduce the complexity of the cloud. The processing of the spatial queries can also benefit from the use of Spatial Analytics Systems (SASs), which are developed on top of those frameworks to provide extended functionalities to deal with spatial data and spatial predicates [Castro et al. 2020].

On the other hand, for IoT applications that require low latency, using cloud computing environ-

ments may be inefficient, due to the delay caused by data transferring between the cloud and the devices [Javadzadeh and Rahmani 2020]. The paradigm called fog computing emerged to solve these problems. It employs computational resources close to the edge of the network, providing data processing, storage, and distribution services [Bonomi et al. 2014; Shi and Dustdar 2016; Javadzadeh and Rahmani 2020]. Some advantages of using fog computing include the wide geographic distribution of services, sensor networks distributed on a large scale, real-time interactions, predominance of wireless access, and heterogeneity, since sensors of different natures can exist on a single network.

The challenge is to propose an IoT architecture for smart cities that encompasses cloud computing, fog computing, and frameworks for parallel and distributed data processing, and also provides efficient support for storing SDWs and providing spatial analytics. Although there are some proposals in the literature [Silva et al. 2020; Zhang et al. 2020; Diaconita et al. 2018; Yuan and Zhao 2012] that investigate some related architectures, they do not consider all these technologies in the same setting. In this article, we overcome these shortcomings.

The contributions of our article are described as follows.

— Proposal of an architecture aimed to help smart city managers and residents in their decision-making process, by employing a SDW that uses a parallel and distributed data processing framework in the fog and in the cloud, aided by a SAS to process spatial queries.
— Definition of a set of guidelines to assist in the implementation of the proposed architecture.
— Validation of the efficacy and effectiveness of the proposed architecture through the execution of three different categories of spatial queries using an application that handles data generated from a real city.

A preliminary version of this article is described in [Santos et al. 2020]. Here, we provide a detailed description of the background and include new related work. We also further detail the previews guidelines and propose new ones. Furthermore, we improve the case study by managing data related to parking in addition to data related to measurement. Finally, we introduce new spatial queries and categorise them according to their type of spatial operation.

This article is organised as follows. Section 2 describes the technical background. Section 3 reviews related work and highlights the differentials of our proposal. Section 4 presents the proposed architecture. Section 5 introduces the guidelines for implementing the architecture. Section 6 describes the case study, details interesting spatial queries, and discusses their usefulness in the decision-making process. Section 7 concludes the article.

2. TECHNICAL BACKGROUND

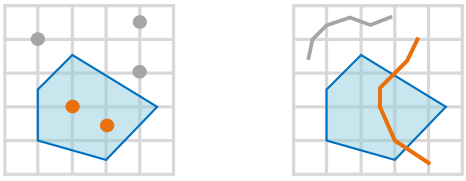
In this section, we detail notions related to spatial data manipulation (Section 2.1) and spatial data warehouses (Section 2.2).

2.1 Spatial Data Manipulation

Spatial data (or geographic data) are components that represent the geometry of spatial objects. Considering the vector data type, spatial data are of three types: points, lines, and polygons. They may have simple geometry, being defined considering one of these types, or complex geometry, being represented by sets of points, lines, and polygons. A point is stored as a set of coordinates, such as latitude and longitude. The other types of spatial data are represented by a data structure (for instance, a vector) that contains the points that form them.

There are several spatial operations that can explore the relationships between spatial data, as categorised in [Castro et al. 2020]. In this article, we are interested in topological predicates, metric

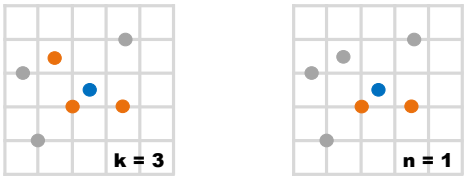
relationships, and type-dependent operations. Topological predicates can be evaluated by using the nine-intersection model, which relies on point sets and point set topology to evaluate the nine possible intersections of interior, exterior, and boundary of two spatial objects [Egenhofer 1989; Gaede and Günther 1998]. Two topological predicates used in our case study analysis are the containment and the intersection spatial join queries, which are defined as follows. Given an object o' with a spatial extent, the containment query, as illustrated in Figure 1a, finds all objects o enclosed by o'. Furthermore, given two collections R e S of spatial objects, the intersection spatial join query, as illustrated in Figure 1b, finds all pairs of objects $(o, o') \in R \times S$ having at least one point in common with o'.



(a) Containment Query (b) Intersection Query

Fig. 1: Examples of spatial queries with topological predicates. Object o' is represented in blue and objects o are represented in orange.

Metric relationships explore measures like distances and directions, where mathematical operations like the Euclidean and the Manhattan distances are used [Egenhofer 1989]. Two metric relationships used in our case study analysis are the k-nearest neighbours and the distance spatial join queries, which are defined as follows. Given an object o' with a spatial extent, the k-nearest neighbours query, as illustrated in Figure 2a, finds the k objects o next to o'. Furthermore, given two collections R e S of spatial objects, the distance spatial join query, as illustrated in Figure 2b, finds all pairs of objects $(o, o') \in R \times S$ where the distance between o and o' is less than or equal to a n value.



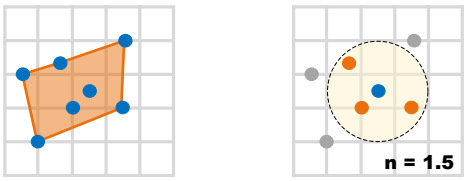
(a) k-NN Query (b) Distance Join Query

Fig. 2: Examples of spatial queries with metric relationships. Object o' is represented in blue and objects o are represented in orange.

Type-dependent operations are those that can be executed over only one specific type of spatial object. In our case study analysis, we are interested in spatial queries that involve generating a new geometry from a distance around a specific geometry. That is, we are interested in finding approximations of a spatial object using algorithms like convex hull [Brinkhoff et al. 1994] and buffer. Given a collection R of spatial objects, a convex hull query, as illustrated in Figure 3a, finds the smallest convex object o' that encloses all o objects in R. Also, given an object o' with a spatial extent, a buffer query, as illustrated in Figure 3b, finds all objects o enclosed or intersected in a zone that is drawn around o' within a specified distance n of o'.

2.2 Spatial data warehouses

In order to implement an SDW, it is necessary to be aware of spatial redundancy. According to the experiments conducted in [Mateus et al. 2016], higher spatial redundancy can lead to unsatisfactory performance results. Therefore, the authors propose that this type of data warehouse should be modelled using a special type of star schema, called Geographic Hybrid Star Schema. This schema



(a) Convex Hull Query (b) Buffer Query

Fig. 3: Examples of spatial queries with type-dependent operations. Object o' is represented in blue and objects o are represented in orange.

stores each geographic attribute of a spatial hierarchy in a different dimension table, avoiding spatial redundancy. We use as a basis these findings to propose the SDW that stores the spatial data of the case study described in Section 6. We also use the pictograms introduced in [Vaisman and Zimnyi 2014] to indicate the geographic attributes that are represented as points, lines, and polygons.

Figure 4 depicts the schema of the proposed SDW. It represents data related to sensors and parking lots scattered in a smart city that collects data on vehicle quantity and average speed. There are seven dimension tables: (i) Date and Time, storing the moment in which a measurement occurred; (ii) Report, storing the distance between the two street sensors that performed the measurement and their geographic locations; (iii) Garage, storing the geographic location of the sensors placed in parking garages and the total spaces available for vehicles to park there; and (iv) Road, District, and City, storing the geographic locations associated with the report and the garage. These three tables represent a spatial hierarchy and therefore their data are stored separately. The dimension tables are linked through two fact tables: (i) Measurement, which stores, for each measurement, the following numeric measures: vehicle count, measurement time, and vehicle speed; and (ii) Parking, which stores the vehicle count as its numeric measure.

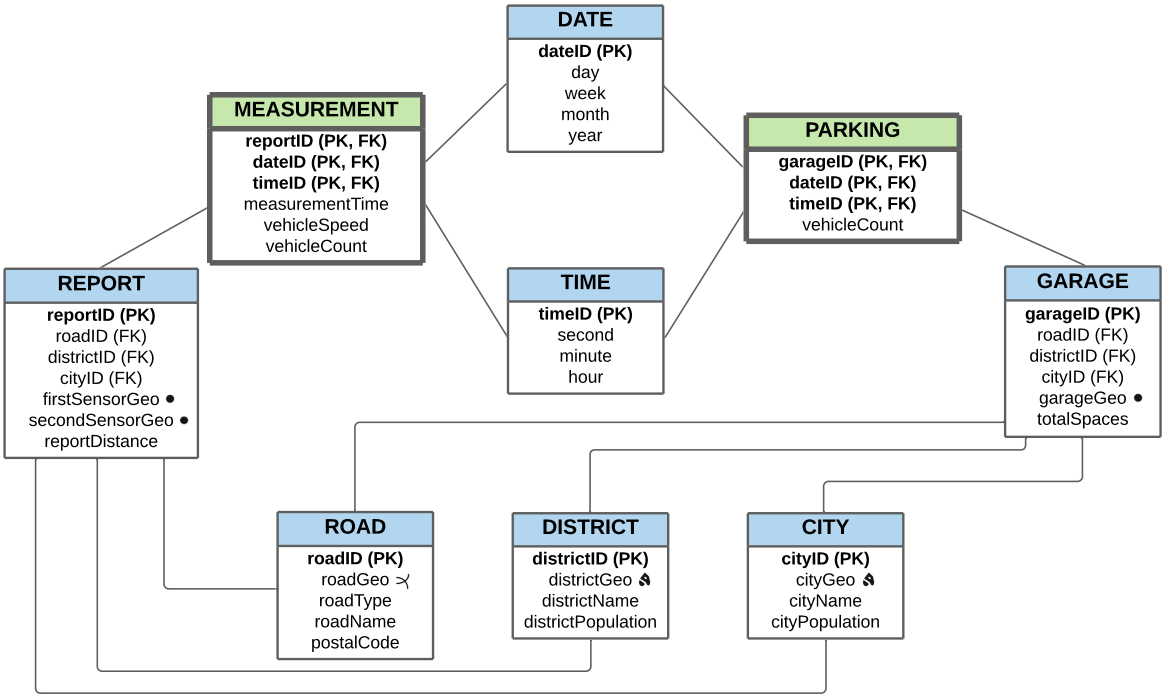


Fig. 4: Logical schema of the SDW proposed to support the case study, which models data related to sensors and parking lots scattered in a smart city that collects data on vehicle quantity and average speed.

3. RELATED WORK

Investigating IoT, big data, and smart cities in the same set is a quite recent area of research. In this section, we survey related work by classifying them into three classes: (i) literature reviews; (ii) use of data warehousing as an underlying technology; and (iii) use of IoT devices in the context of smart cities.

Considering the literature reviews, i.e., class (i), there are studies that present challenges related to IoT-generated data. Bansal et al. (2021) describe a systematic survey on the theme and challenges related to IoT Big Data (IoTBD), such as volume, variety and velocity. The authors also highlight trends in IoTBD research, such as leveraging edge and fog infrastructures and multi-cloud management. Javadzadeh et al. (2020) provide a systematic literature review of current research related to fog computing applications in smart cities. The review compares the approaches considering service objectives e application classification. It also highlights that only few studies found have dealt extensively with big data analysis. Therefore, there is a lack in the literature regarding this issue.

We now move our discussion to studies in the literature that propose the construction of data warehousing environments with data based on IoT, i.e., class (ii). We consider here studies that are not directly related to smart cities. The work of Rahman et al. (2019) investigate data sources from IoT-based soil nutrient measurement data to build a data warehouse that can be used to support analysis related to types of plants and fertilisers needed at a soil location. In [Al-Ali et al. 2017], the authors introduce an energy management system for smart homes that uses off-the-shelf business intelligence and big data analytics software packages to better manage energy consumption and to meet consumer demand. The study described in [Kumar et al. 2020] uses IoT to avoid spreading the COVID-19 virus. However, these related studies provide solutions to very specific problems. The architecture that we propose in this article is more generic and broad.

Let us now review studies classified in class (iii). Silva et al. (2020) present a big data analytics embedded smart city architecture. The authors use a Representational State Transfer (RESTful) for communication between devices and a parallel and distributed environment based on Hadoop. The RESTful service runs over a smart gateway between the devices and the cluster. The authors also introduce a case study involving real data on traffic, parking, air pollution, and water use in smart cities, comparing the performance between a cluster with a single Hadoop node, two Hadoop nodes and a JQuery-based system. Zhang et al. (2020) propose the SafeCity architecture, which contains three layers: (i) a data security layer to handle secure communication with devices; (ii) a data computation layer to process data in a Hadoop environment; and (iii) a decision-making layer to support AI-based operations for decision-making. The authors also present a system evaluation for data processing and computation, using real data from a smart city. Both [Silva et al. 2020] and [Zhang et al. 2020] do not provide spatial data manipulation and do not use concepts related to data warehousing environments.

Diaconita et al. (2018) propose an architecture based on Hadoop to efficiently manage smart cities. The architecture uses Hadoop-compatible solutions to resource management (e.g., Apache Hadoop YARN), bulk data transfer (e.g., Apache Sqoop[1]), sensors data ingestion (e.g., Apache Storm[2]), data processing (e.g., Apache Spark), and data warehousing (e.g., Apache Hive[3]). The work also describes a performance comparison of queries executed in HiveQL with several execution engines, like Spark, Apache Tez[4], and Hadoop MapReduce, using real data related to road networks, taxicabs and points of interest. Yuan and Zhao (2012) propose the architectural solution for SDWs in the context of IoT environments called SDWIT. It has the following layers: data processing layer, storage layer, and analysis application layer. SDWIT features include accessing and analysing IoT data in real time

[1]https://sqoop.apache.org/
[2]https://storm.apache.org/
[3]https://hive.apache.org/
[4]https://tez.apache.org/

over a traditional SDW. However, [Diaconita et al. 2018] do not focus on spatial data and [Yuan and Zhao 2012] do not consider the support of parallel and distributed data processing frameworks in their architecture. Furthermore, these two studies are not based on SASs and fog computing.

In contrast to the surveyed studies, we propose a novel architecture that employs both parallel and distributed data processing frameworks and SASs, allowing the execution of fast and reliable spatial decision-making analyses over IoT data from smart cities. Our architecture excels not only in the integration of these technologies with an SDW in the cloud, but also in the inclusion of a fog computing layer to handle spatial data analytics and Extract, Transform, and Load (ETL) processes. We also introduce a set of guidelines to aid smart city managers in the process of implementing our architecture. Further, we validate the architecture with a case study that describes an application that handles real data generated from a smart city. The aim of this case study is to investigate the efficacy and effectiveness of the architecture, but not its efficiency. Thus, carrying out performance evaluations is out of the scope of this article.

4. THE PROPOSED ARCHITECTURE

In this section, we describe a novel architecture for collecting and analysing, in a fast and reliable manner, data from IoT devices in smart cities. The proposed architecture is depicted in Figure 5. It achieves its goals through the employment of three different layers: (i) the terminal layer; (ii) the fog computing layer; and (iii) the cloud computing layer. We discuss each layer as follows.

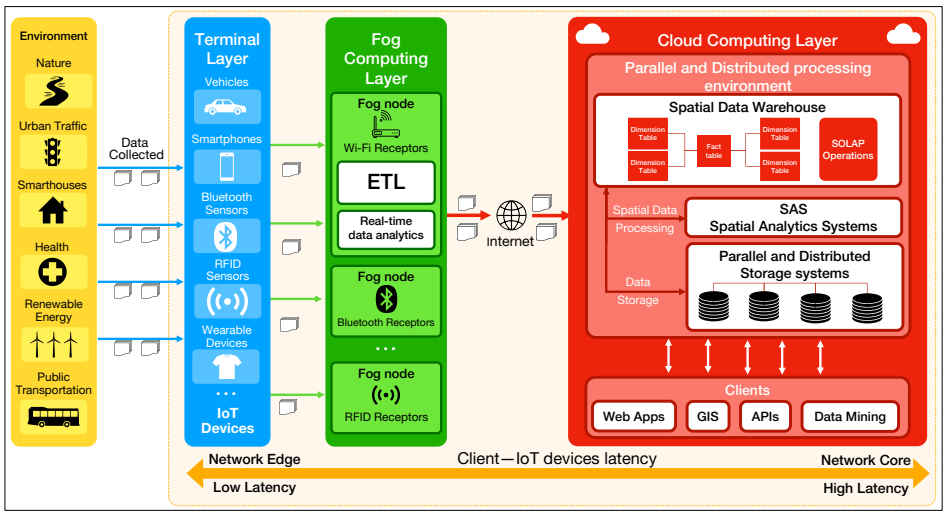


Fig. 5: Overview of the proposed architecture, which encompasses cloud computing, fog computing, and frameworks for parallel and distributed data processing, and also provides efficient support for storing SDWs and providing spatial analytics.

Terminal layer. The terminal layer consists of a network of IoT devices, which are interconnected by using technologies such as Radio Frequency Identification (RFID), Global Positioning System (GPS), and network communication standards, such as Ethernet and Bluetooth. These devices are available in many parts of a smart city, such as weather stations, traffic lights, and public transportation. The devices are aimed to collect spatial and conventional data.

Fog computing layer. Data collected by terminal layer devices are sent to receivers in the fog computing layer. These receivers, called fog nodes, can be limited with regard to data processing and storage. However, by being located close to the network edge, they are in an optimal position to allow the execution of real-time data analytics and ETL operations. This is due to the low latency in communication between the terminal layer devices and these nodes.

Cloud computing layer. After the data goes through the ETL process in the fog computing layer, it is sent to the cloud computing layer. In this layer, data are persisted in an SDW stored in a parallel and distributed storage system. This allows SOLAP queries to be processed with the help of a SAS, enhancing their performance considerably. Due to the scalable nature inherent to cloud computing environments, the number of nodes can increase or decrease according to the demand of queries from clients. Examples of clients include web applications, Geographic Information Systems (GIS), and different types of Application Programming Interfaces (APIs).

5. GUIDELINES FOR IMPLEMENTING THE PROPOSED ARCHITECTURE

In this section, we propose a set of guidelines to aid smart city managers in the process of implementing the proposed architecture to support spatial analytics in smart cities. Because the context behind each smart city may be different, it is not mandatory to follow every guideline in its completeness. Managers should choose the appropriated hints provided by the guidelines according to the specific characteristics of the smart city in which the architecture is being employed. Thus, a concise yet general description of each guideline is provided, allowing further specialisation based on the requirements imposed by each smart city application.

Guideline 1. Deploying IoT devices on the terminal layer. IoT devices must be deployed in the terminal layer considering the communication protocols supported by each sensor and the coverage of each device. A smart city manager must also consider the communication compatibility between these devices and the fog nodes. Some investigations found in the literature can be used to assist in the deployment of these devices. We indicate the work of Alablani and Alenazi (2020), which introduces an algorithm for sensor distribution and sink placement called EDTD-SC. This algorithm uses triangulation and clustering techniques to find optimal locations to improve sensor coverage over a smart city.

Guideline 2. Distributing fog nodes across the fog computing layer. After the disposition of the IoT devices in the terminal layer, a smart city manager must define which devices must be used as fog nodes. For instance, some approaches in the literature use Raspberry Pi computers[5], which are small single-boarded computers, as fog nodes, using containerisation over these resource-limited devices [Bellavista and Zanni 2017; Xu and Zhang 2019]. Because Raspberry Pi computers are low cost and support many communication protocols, they are a viable choice to the heterogeneous nature of an IoT network. Communication between the fog nodes and the cloud computing layer can be carried out using 4G/5G or Wi-Fi protocols. Each fog node uses the Docker container technology[6] for creating containers for each application available in a fog node (e.g., ETL and real-time data analytics). We indicate the reading of the work of Pérez de Prado et al. (2020) , which presents the challenges and opportunities on the schedule of smart containers in the Cloud-Fog-IoT interfaces.

Guideline 3. Securing the connection between IoT devices and fog nodes. A smart city manager must be concerned with the data flow security between the IoT devices and the fog nodes, as sensitive information may be transmitted. For instance, at the terminal layer, some security threats include signal jamming between IoT devices and Denial of Service (DoS) attacks [Puthal et al. 2019]. At the fog computing layer, security threats include phishing attacks, infected code injection, session hijacking, and distributed DoS [Puthal et al. 2019; Rauf et al. 2018]. To deal with these issues, smart city managers can take decisions using as a basis the work of Ni et al. (2018) . In this work, the authors discuss security-related challenges in fog-assisted IoT applications. The authors also review state-of-the-art solutions to address security and privacy issues in a fog environment deployed in an IoT network.

[5]https://www.raspberrypi.org/
[6]https://www.docker.com/

Guideline 4. Storing and querying data in the fog computing layer. Due to the intrinsic nature of his tasks, a smart city manager might require mechanisms for storing and querying data over a fog node. Multiple data management systems, like NoSQL databases (e.g., Couchbase Server[7] and Apache Cassandra[8]) can be employed [Yang 2017]. As suggested in Guideline 2, storage services can be operated by containers inserted in the fog node. Polyglot persistence [Medvedev et al. 2016; Sadalage and Fowler 2012] can be used in a fog node network since it takes into consideration the node processing and physical storage, as well as the queries that should execute in each node.

Guideline 5. Modelling the ETL process at the fog computing layer. To enable ETL processing in the fog computing layer, a smart city manager must define the conceptual and logical modelling of the ETL process. The conceptual model is a schema that represents the data flow between the fog computing layer and the cloud computing layer. The approaches described in [Ali and Wrembel 2017; El-Sappagh et al. 2011] propose techniques for the implementation and optimisation of ETL workflows at the conceptual level. The logical model provides a detailed description of an ETL workflow from the description of data and relationships. Furthermore, in [Ali and Wrembel 2017] is proposed the use of two logical implementations of an ETL workflow: (i) an architecture graph, which employs graphs to represent an ETL workflow, with the edges representing different types of relationships between ETL activities; and (ii) Parameterized Directed Acyclic Graphs (DAG-P), where vertices represent tasks and edges represent relationships between tasks.

Guideline 6. Enabling the ETL process at the fog computing layer. To enable ETL processing in the fog computing layer, a smart city manager must select tools that allow programming, scaling, and monitoring tasks in the ETL workflow. There are several tools on the market which support ETL and workflow monitoring. An example is Apache Airflow[9], which is an open-source platform that uses Directed Acyclic Graphs (DAGs) for authoring, scheduling and monitoring workflows. We indicate that the tasks be written using the Python programming language, since it is natively supported by Airflow. Airflow provides integration with Hadoop, Spark, and several cloud platforms. Another example is Pentaho Data Integration (or Kettle)[10], which provides a graphical environment for the creation of ETL workflow operations. This tool supports connections to several DBMSs and provides several plugins created by the community.

Guideline 7. Choosing the appropriate SAS to implement the SDW in the cloud computing layer. The SDW application should process SOLAP queries efficiently. Therefore, a smart city manager must select a SAS that is able to completely fulfil the requirements of the SDW application. Because there are several SASs available in the literature with different characteristics and capabilities, choosing the most appropriate SAS can become considerably challenging. Thus, smart city managers should use as a basis of choice the state-of-the-art user-centric comparison of existing SASs described in [Castro et al. 2020]. For a system-centric view of SASs, the work of Pandey et al. (2018) should be referred, as it compares several SASs in the literature based on their query processing performance.

Guideline 8. Multidimensional modelling of data contained in the cloud computing layer A smart city manager should perform multidimensional modelling at the logical and the physical level. At the logical level, the manager should on the use of schemas such as the star schema, the snowflake schema, and the Geographic Hybrid Star Schema. At the physical level, the manager should use structures such as star join bitmap indexes and materialised views. For a conventional star schema modelling, the work of [Kimball et al. 2011] can be used. In a spatial data context, Vaisman and Zimányi (2014) presents design discussions to model an SDW and also introduce the concept of trajectory data warehouses, which can be appropriate for IoT devices that collect data from sensors

[7] https://www.couchbase.com
[8] http://cassandra.apache.org
[9] http://airflow.apache.org/
[10] https://sourceforge.net/projects/pentaho/

installed in, for instance, in vehicles. The work of Mateus et al. (2016) details the design of novel schemas for an SDW deployed in a cloud environment. The authors also evaluate the performance of SOLAP query processing with the employment of a cloud-based spatial index.

Guideline 9. Configuring the SDW to process SOLAP queries on the cloud computing layer. After choosing the appropriate SAS, a smart city manager must configure the SDW environment in the cloud computing layer to process SOLAP queries. The parallel and distributed data processing framework and the distributed file system must be compatible with the chosen SAS. There are several Platform-as-a-Service (PaaS) on the market that support these frameworks natively, such as Microsoft Azure[11] and Amazon Web Services[12]. The manager must consider the periodicity in which data should be extracted from the fog computing layer, as well as carefully specify data distribution over the SDW. These SOLAP services must support APIs and GIS applications in order to visualise the results of SOLAP queries.

Guideline 10. Ensuring secure spatial queries processing in the cloud computing layer. Since cloud computing environments can be virtually accessed from anywhere, smart city managers should be concerned with security issues related to cloud applications. In the same way that fog nodes are subject to security threats like phishing and DDoS attacks (Guideline 3), a cloud environment also has security concerns [Almorsy et al. 2010; Balani and Varol 2020]. Solutions for issues related to security in a cloud computing environment are presented in [Singh et al. 2016]. Another way to ensure confidentiality is the encryption of the data stored in the SDW. Data encryption should be done carefully so that it does not compromise the performance of the SDW application. To this end, we indicate the reading of the work of Lopes et al. (2021) , which proposes an encryption methodology for a cloud DW stored according to the star schema, considering the capability of processing analytical queries over the encrypted DW.

6. CASE STUDY

In this section, we describe a case study that illustrates the use of the proposed architecture. We define the requirements of a spatial application, whose objective is to process data collected from multiple IoT devices in the context of smart cities. This case study is aimed to validate the efficacy and effectiveness of the proposed architecture. Focusing on details regarding the performance and reliability of the architecture is out of the scope of the article.

For this case study, we use a dataset provided by Ali et al. (2015), which contains vehicle traffic data observed between two street sensors and vehicle count data observed in sensors placed in parking garages. These sensors are distributed in the municipality of Aarhus, Denmark. The dataset, which is publicly available in the authors' website[13], contains both conventional (e.g., distance in meters between the sensors, type of road, etc.) and spatial data (e.g., the sensors locations, represented by points) referring to the period from February to June 2014. The sensors generated data every five minutes.

As the dataset only provides data regarding the sensor location (i.e., points), we extended it with new information to enrich the analyses performed in our spatial application. To this end, we use road (i.e., lines) and city (i.e., polygons) data obtained by Geofabrik[14] from OpenStreetMap, and statistical district data obtained from OpenDataDK[15]. We guarantee the spatial relationship between the data in the sense that a road intersects with sensors, a district contains multiple roads, and a city contains several districts.

[11] http://azure.microsoft.com
[12] http://aws.amazon.com
[13] http://iot.ee.surrey.ac.uk:8080/
[14] https://www.geofabrik.de/
[15] https://www.opendata.dk/city-of-aarhus/statistikdistrikter

The requirements imposed by the SDW application are described as follows. The application should be deployed in the cloud and should communicate with a SAS to process its queries. Furthermore, data handled by the application should be stored in an SDW designed according to the logical schema depicted in Figure 4, which should also be located in the cloud. Another requirement of the application is that it should support different types of spatial queries based on the definitions of Gaede and Günther (1998) , such as spatial join, containment, and k-nearest neighbour queries. The application should also provide good performance results. Finally, it is important to highlight that the developers who are going to implement the application have some previous knowledge of the SQL programming language.

According to the proposed architecture (Section 4) and guidelines (Section 5), the case study application should be implemented as follows: (i) use of the Apache Airflow to perform the ETL process; (ii) storage of the SDW data in the HDFS as the application requires data storage in the cloud; and (iii) selection of Sedona [Yu et al. 2019] as the SAS to process the spatial queries aimed to decision-making, as it complies with the application's requirements regarding performance and spatial queries, as well as supports the SQL programming language through the use of SedonaSQL [Pandey et al. 2018; Castro et al. 2020].

In Section 6.1, we describe aspects related to data loading, including the pre-processing on the data. Once the process of loading the data provided by the IoT sensors into the SDW is complete, smart city managers are able to execute different types of analyses using SedonaSQL. We define some query examples that address key points of the application's requirements, which are divided into three different categories: (i) spatial queries with a topological predicate (Section 6.2); (ii) spatial queries with metric relationships (Section 6.3); and (iii) spatial queries with type-dependent operations (Section 6.4). These queries may be employed by a smart city manager to support spatial analytics. We employ QGIS[16] to visualise the results of the queries.

6.1 Data Loading into the Cloud Layer

The dataset used in this case study consists of 449 reports and 8 garages. Data from these reports and garages are stored in Comma-Separated Values (CSV) files. To be loaded into the SDW, the data must go through an ETL process in the fog computing layer (Guideline 6). Thus, Apache Airflow should be employed to: (i) extract the data from the CSV files; (ii) perform transformations to arrange the data according to the logical schema depicted in Figure 4; and (iii) load the data into HDFS for later use by Sedona. To accurately simulate the fog computing layer, Airflow was executed in a Docker container.

Furthermore, in order to employ SedonaSQL for processing spatial queries over the SDW stored in HDFS, it is necessary to load its tables into structures called DataFrames. These structures, which resemble relational tables, do not transform the textual representations of the spatial data into spatial objects by default. Therefore, it is necessary to carry out the transformations. SedonaSQL provides a function to convert Well-Known Text (WKT) representations into spatial objects. An example of using this function during the process of loading the Report table is detailed in the following query:

```
SELECT reportID, roadID, districtID, cityID, reportDistance,
       ST_GeomFromWKT(firstSensorGeo) AS firstSensorGeo,
       ST_GeomFromWKT(secondSensorGeo) AS secondSensorGeo
FROM sensor
```

[16] https://qgis.org/

6.2 Spatial Queries with Topological Predicate

A containment query is to return the quantity of vehicles that traveled in Aarhus University/Community Hospital district, grouped by day and month. An interesting knowledge that can be obtained from this type of analysis is to identify the days in which the district had the largest number of vehicles and to investigate whether a holiday or an event happened, as displayed in Figure 6. The following command expresses this query:

```
SELECT day, month, SUM(vehicleCount)
FROM measurement, date, report, district, road
WHERE ST_Contains(districtGeo, roadGeo)
    AND ST_Intersects(roadGeo, ST_MakeLine(firstSensorGeo, secondSensorGeo))
    AND measurement.reportID = report.reportID
    AND measurement.dateID = date.dateID
    AND report.districtID = district.districtID
    AND report.roadID = road.roadID
    AND district.name = 'Universitetet/Kommunehospitalet'
GROUP BY day, month
ORDER BY day, month
```

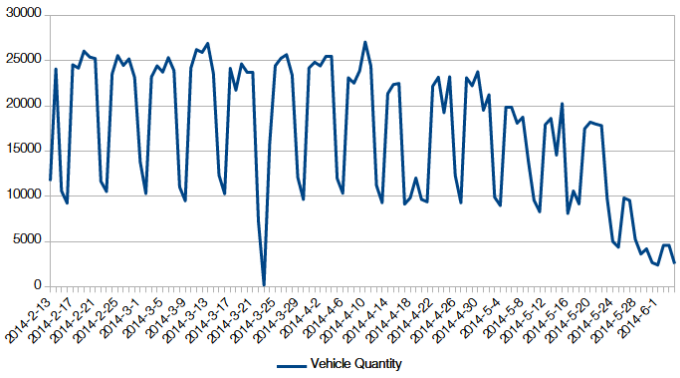


Fig. 6: Containment query returning the quantity of vehicles that traveled in Aarhus University/Community Hospital district grouped by day and month, considering the period between 2014-2-13 and 2014-6-1.

By interpreting the query results, a smart city manager can obtain different types of knowledge. For instance, the measurement of zero vehicles on March 25, 2014 (2014-3-25) could indicate that this was a day in which the sensors in the designated district were entirely disabled. Another interesting knowledge that can be obtained is that the traffic in this district seems more intense in weekdays when compared to weekends.

Another analysis, which resembles an intersection spatial join query, is to return the districts in which the average vehicle speed reported from the set of sensors that intercept it is greater than 60 km/h (37.28 mph). This analysis is necessary to check if there are districts where the maximum permitted speed is not being respected by the drivers. The query results are depicted in Figure 7, with each selected district being highlighted in red and the average vehicle speed (in km/h) displayed in its centre. The following command expresses this query:

```
SELECT districtGeo, AVG(vehicleSpeed) AS a
FROM measurement, report, district
WHERE ST_Intersects(ST_MakeLine(firstSensorGeo, secondSensorGeo), districtGeo)
    AND measurement.reportID = report.reportID AND report.districtID = district.districtID
GROUP BY districtGeo
HAVING a >= 60
```

The analysis of the query results indicate that the average vehicle speed is higher in the northern districts of the municipality of Aarhus. A smart city manager can extract different types of knowledge from this information. An example is the fact that drivers can be less inclined to drive over the speed limit in central areas of the municipality (highlighted in purple in Figure 7), probably due to the increased number of pedestrians in these areas. Another example resides in the assumption that the average speed in the northern districts is higher due to the fact that some of them connect with external highways (displayed as pink lines in Figure 7).

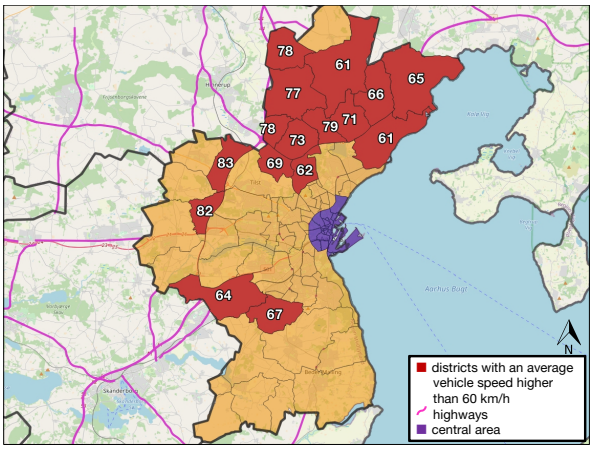


Fig. 7: Intersection spatial join query returning the districts of the municipality of Aarhus, in which the average vehicle speed reported from the set of sensors that intercept it is greater than 60 km/h (37.28 mph).

6.3 Spatial Queries with Metric Relationships

A k-nearest neighbours query is to return the average vehicle speed identified by the 10 nearest reports from the Aarhus Cathedral, which is represented by a point (10.210556, 56.156944). This type of analysis is necessary to verify if drivers are respecting the speed limit in the surrounding area of a highly frequented point of interest, as shown in Figure 8. The following command expresses this query:

```
SELECT AVG(vehicleSpeed), ST_MakeLine(firstSensorGeo, secondSensorGeo) AS reportGeo
FROM measurement, report
WHERE measurement.reportID = report.reportID
GROUP BY reportGeo
ORDER BY ST_Distance(reportGeo, ST_GeomFromWKT('POINT(10.210556 56.156944)'))
LIMIT 10
```

The results displayed in Figure 8 can enable smart city managers to perform a wide variety of analyses. In particular, one can identify that the highest average speeds around Aarhus Cathedral can often be observed in the main streets of its district, which are highlighted in red. smart city managers can also observe that these speeds do not go over 33 km/h. This can indicate that drivers do not tend to speed up in this region, a fact that might indicate the occurrence of heavy traffic.

An interesting query is to compare the amount of occupied parking spots and the amount of vehicles transiting in the streets in the last week of May. In order to perform such analysis, a distance spatial join is executed, which considers the sensors in the streets that are located at a distance of at most 100 m from sensors placed in parking garages. Only parking garages that had at least one vehicle parked in the period were considered. The following command expresses this query:

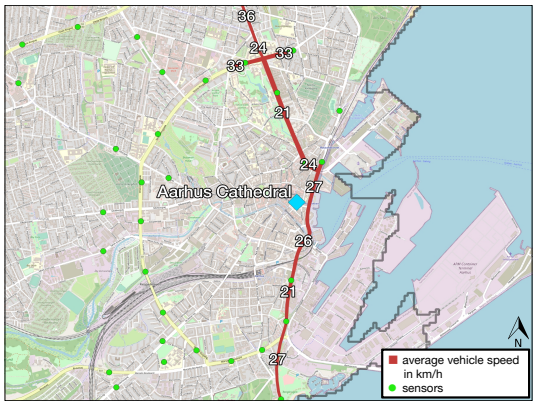


Fig. 8: K-nearest neighbours query returning the average vehicle speed identified by the 10 nearest reports from the Aarhus Cathedral.

```
SELECT garage.garageID AS "Garage Name",
    road.roadName AS "Street Name",
    ROUND(AVG(measurement.vehiclecount),0) AS "Average number of vehicles in traffic",
    ROUND(AVG(parking.vehiclecount),0) AS "Average number of parked vehicles"
FROM (SELECT reportID, SUM(vehiclecount) AS vehiclecount
      FROM measurement, date
      WHERE measurement.dateID = date.dateID AND month = 5 AND week = 4
      GROUP BY reportID) AS measurement,
     (SELECT parking.garageID, AVG(vehiclecount) AS vehiclecount
      FROM parking, date
      WHERE parking.parkingID = date.dateID AND month = 5 AND week = 4
      GROUP BY garageID) AS parking,
      report, garage, road
WHERE report.reportID = measurement.reportID
      AND parking.garageID = garage.garageID
      AND garage.roadID = road.roadID
      AND ST_Distance(
          ST_MakeLine(report.firstsensorgeo, report.secondsensorgeo),garage.geo) <= 0.01
GROUP BY garage.garageID
```

The query results are displayed in Table I. By analysing these results, a smart city manager can realise that there is no direct relationship between the average number of vehicles in traffic and the average number of parked vehicles in the period. This is clear due to the fact that there are streets with: (i) a high number of vehicles in traffic and few vehicles parked (such as Kalkværksvej); (ii) a high number of vehicles in traffic and a high amount of vehicles parked (such as Værkmestergade); and (iii) a few number of vehicles in traffic and a high amount of vehicles parked (such as Østergade). However, it is important for a smart city manager to run this analysis on different weeks and months in order to investigate if this lack of relationship is dependent of the time period.

Table I: Distance join query returning the average number of vehicles in traffic and the average number of parked vehicles, considering the sensors in the streets of the municipality of Aarhus that are located at a distance of at most 100 m from sensors placed in parking garages.

Garage Name	Street Name	Average number of vehicles in traffic	Average number of parked vehicles
BRUUNS	Værkmestergade	5,605	166
BUSGADEHUSET	Frederiksgade	4,223	84
KALKVAERKSVEJ	Kalkværksvej	4,458	49
MAGASIN	Åboulevarden	4,223	108
SALLING	Østergade	3,618	191

6.4 Spatial Queries with Type-Dependent Operations

A convex hull query is to return the minimal convex polygon (i.e., the *convex hull*) that contains all the sensors in the municipality of Aarhus. This analysis can be useful to verify the coverage area of the sensors in the municipality, as displayed in Figure 9. In order to compute the result of this query, it is necessary to first aggregate the sensor geometries, as displayed in the following command:

```
SELECT ST_ConvexHull(ST_Collect(sensors.sensorGeo))
FROM (SELECT report.firstSensorGeo AS sensorGeo FROM report
      UNION
      SELECT report.secondSensorGeo AS sensorGeo FROM report) AS sensors
```

The results displayed in Figure 9 reveal that several districts in the extremities of the municipality are beyond the sensor coverage area. Further, it is possible to visualise that there are sensors positioned outside the municipality limits. A smart city manager should analyse these results to better determine the sensor distribution across the municipality districts.

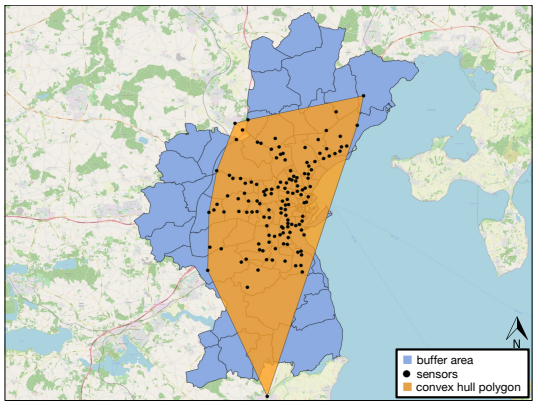


Fig. 9: Convex Hull query returning the minimal convex polygon that contains all the sensors in the municipality of Aarhus.

Another analysis, which represents a buffer query, is to return the average vehicle count of the sensors located within a 100 m buffer of each school in the municipality of Aarhus, considering the five highest measurements. Data relating to schools in the municipality of Aarhus was extracted from OpenStreetMap and contains as attributes the identifier, name of the school and its representation as a point. This type of analysis can be useful to determine the movement around school areas in different regions of the municipality. The following command expresses this query:

```
SELECT ROUND(AVG(measurement.vehicleCount),0) as AVGvehicleCount, schoolID
FROM schools, report, measurement,
    ST_Buffer(schools.geom, 0.01) AS schoolbuffer,
    ST_MakeLine(firstSensorGeo, secondSensorGeo) AS sensorset
WHERE ST_Intersects(sensorset,schoolbuffer)
    AND measurement.reportID = report.reportID
GROUP BY schoolID
ORDER BY AVGvehicleCount DESC
LIMIT 5
```

Figure 10 shows the visualisation of the query results. It is important to highlight that the buffers are not depicted as regular circular shapes due to the map projection adopted. Through the analysis of the query results, a smart city manager can verify if the movement is higher around a certain school in comparison with the others.

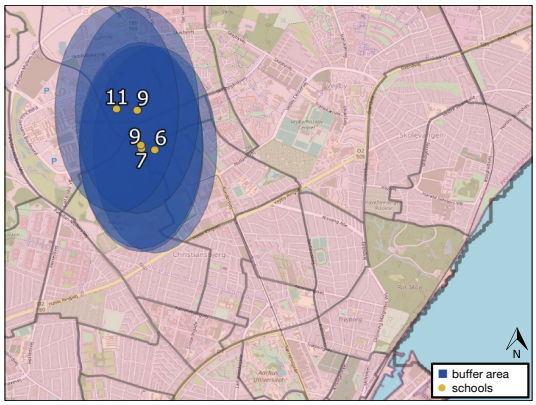


Fig. 10: Buffer query returning the average vehicle count of the sensors located within a 100 m buffer of each school in the municipality of Aarhus, considering the five highest measurements.

It is also possible for the manager to seek a more detailed analysis, investigating the average vehicle count per hour for a certain school, as depicted in Figure 11. The following command expresses the query necessary to run this analysis:

```
SELECT SUM(measurement.vehiclecount), hour, day
FROM schools, report, measurement, date, time
    ST_Buffer(schools.geom, 0.01) AS schoolbuffer,
    ST_MakeLine(firstSensorGeo, secondSensorGeo) AS sensorset
WHERE ST_Intersects(sensorset,schoolbuffer)
    AND measurement.reportID = report.reportID
    AND measurement.dateID = date.dateID
    AND measurement.timeID = time.timeID
    AND schools.schoolID = '4448940550'
    AND month = 2
GROUP BY hour, day
ORDER BY day, hour
```

After interpreting the results depicted in Figure 11, a smart city manager can obtain some knowledge regarding the movement of vehicles around the analysed school. For instance, it is possible to realise that the number of vehicles in the vicinity of the school is greater during business hours. A smart city manager can also observe that there is a small decline in the vehicle count at 7h, a fact that can be further investigated.

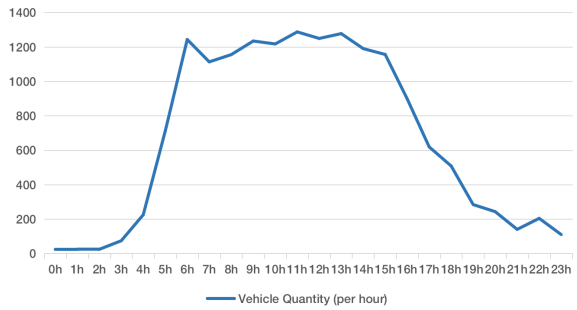


Fig. 11: Buffer query returning the average vehicle count per hour and day in February 2014, considering the period from 0h to 23h.

7. CONCLUSIONS AND FUTURE WORK

In this article, we propose an architecture aimed to help smart city managers to enable analyses over IoT data in the context of smart cities. The architecture is composed of three layers. The cloud computing layer is a cloud computing environment based on parallel and distributed data processing frameworks and spatial analytics systems. The fog computing layer is responsible for extracting, transforming and loading the data into a spatial data warehouse. The terminal layer consists of a network of interconnected IoT devices aimed to collect spatial and conventional data.

Based on our architecture, we introduce a set of guidelines to aid smart city managers in the process of implementing our architecture. We provide a concise yet general description of each guideline, allowing further specialisation based on the requirements imposed by each smart city application. The proposed guidelines focus on the following issues: deploying IoT devices, distributing fog nodes, securing the connection between IoT devices and fog nodes, data storing and querying, modelling and enabling the ETL process, choosing an appropriate spatial analytics system to implement the spatial data warehouse, modelling and configuring the spatial data warehouse, and ensuring secure spatial queries.

Furthermore, we validate the proposed architecture and guidelines by employing them to implement a spatial data warehousing application that analyses data collected from real IoT devices located in the municipality of Aarhus, Denmark. Three different categories of spatial queries were executed, i.e., spatial queries with topological predicate, spatial queries with metric relationships, and spatial queries with type-dependent operations. We also highlight important findings that can be obtained from these queries and how these findings can assist smart city managers in the decision-making process. Although our case study encompasses the municipality of Aarhus, our architecture can be applied to any smart city that employs spatial data generated by IoT devices to improve government intelligence.

We are currently investigating data mining aspects to encompass in our proposed architecture and to enhance the spatial analytics in the context of smart cities. Future work also includes the development of additional case studies, using real data from sensors that collect measurements in different contexts, such as public transport and air and water pollution. To comply with this extension, it is necessary to specify new spatial queries that are important to the decision-making in this context as well as to investigate new query categories, such as spatial queries with numerical operations, geometric set operations, and directional relationships. Another future work is to improve the performance of the spatial queries carried out using our proposed architecture. New case studies should also focus on investigating performance and reliability issues of the architecture. In addition to execute the expensive star join operation of data warehousing environments, these queries also require the computation of costly operations over the spatial data. Furthermore, they should consider the support of the underlying spatial analytics system to comply with our architecture.

ACKNOWLEDGMENT

This work was supported by Brazilian National Council for Scientific and Technological Development (CNPq) and by the São Paulo Research Foundation (FAPESP). C. D. Aguiar has been supported by the grant #2018/22277-8, FAPESP.

REFERENCES

Al-Ali, A. R., Zualkernan, I. A., Rashid, M., Gupta, R., and Alikarar, M. A smart home energy management system using IoT and big data analytics approach. *IEEE Transactions on Consumer Electronics* 63 (4): 426–434, 11, 2017.

Alablani, I. and Alenazi, M. EDTD-SC: An IoT Sensor Deployment Strategy for Smart Cities. *Sensors* 20 (24): 7191, 12, 2020.

Ali, M. I., Gao, F., and Mileo, A. CityBench: A configurable benchmark to evaluate RSP engines using smart city datasets. In *Arenas M. et al. (eds) The Semantic Web - ISWC 2015. ISWC 2015. Lecture Notes in Computer Science*. Vol. 9367. Springer, Bethlehem, PA, USA, pp. 374–389, 2015.

Ali, S. M. F. and Wrembel, R. From conceptual design to performance optimization of ETL workflows: current state of research and open problems. *VLDB Journal* 26 (6): 777–801, 12, 2017.

Almorsy, M., Grundy, J., and Müller, I. An Analysis of the Cloud Computing Security Problem. In *Proceedings of the APSEC 2010 Cloud Workshop*. APSEC, Sydney, 2010.

Atzori, L., Iera, A., and Morabito, G. Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks* vol. 56, pp. 122–140, 3, 2017.

Balani, Z. and Varol, H. Cloud Computing Security Challenges and Threats. In *8th International Symposium on Digital Forensics and Security, ISDFS 2020*. IEEE, Beirut, Lebanon, 2020.

Bansal, M., Chana, I., and Clarke, S. A Survey on IoT Big Data. *ACM Computing Surveys* 53 (6): 1–59, 2, 2021.

Bellavista, P. and Zanni, A. Feasibility of fog computing deployment based on docker containerization over RaspberryPi. In *ICDCN '17: 18th International Conference on Distributed Computing and Networking*. ACM, New York, NY, USA, pp. 1–10, 2017.

Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. Fog computing: A platform for internet of things and analytics. *Studies in Computational Intelligence* vol. 546, pp. 169–186, 2014.

Brinkhoff, T., Kriegel, H. P., Schneider, R., and Seeger, B. Multi-Step Processing of Spatial Joins. *ACM SIGMOD Record* 23 (2): 197–208, 5, 1994.

Castro, J. P., Carniel, A., and Ciferri, C. Analyzing spatial analytics systems based on Hadoop and Spark: A user perspective. *Software: Practice and Experience* 50 (12): 2121–2144, 12, 2020.

Chen, M., Mao, S., and Liu, Y. Big data: A survey. *Mobile Networks and Applications* 19 (2): 171–209, 1, 2014.

Diaconita, V., Bologa, A. R., and Bologa, R. Hadoop oriented smart cities architecture. *Sensors (Switzerland)* 18 (4): 1–20, 4, 2018.

Egenhofer, M. J. A formal definition of binary topological relationships. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 367 LNCS. Springer Verlag, Berlin, Heidelberg, Germany, pp. 457–472, 1989.

El-Sappagh, S. H. A., Hendawi, A. M. A., and El Bastawissy, A. H. A proposed model for data warehouse ETL processes. *Journal of King Saud University - Computer and Information Sciences* 23 (2): 91–104, 7, 2011.

Eldrandaly, K. A., Abdel-Basset, M., and Shawky, L. A. Internet of Spatial Things: A New Reference Model With Insight Analysis. *IEEE Access* vol. 7, pp. 19653–19669, 2019.

Fraga, E. and Queirolo, G. Crescimento populacional fará mundo mudar de cara até 2100. `https://folha.com/ne67804j`, 2018. [Online; access sep. 20].

Gaede, V. and Günther, O. Multidimensional access methods. *ACM Computing Surveys* 30 (2): 170–231, 6, 1998.

Han, J., Stefanovic, N., and Koperski, K. Selective materialization: An efficient method for spatial data cube construction. In *LNCS*. Vol. 1394. Springer, Berlin, Heidelberg, Germany, pp. 144–158, 1998.

Ismagilova, E., Hughes, L., Dwivedi, Y. K., and Raman, K. R. Smart cities: Advances in research—An information systems perspective. *International Journal of Information Management* vol. 47, pp. 88–100, 2019.

Javadzadeh, G. and Rahmani, A. M. Fog Computing Applications in Smart Cities: A Systematic Survey. *Wireless Networks* 26 (2): 1433–1457, 2, 2020.

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., and Becker, B. *The Data Warehouse Lifecycle Toolkit*. Vol. 3. John Wiley & Sons Inc, Hoboken, NJ, 2011.

Kumar, K., Kumar, N., and Shah, R. Role of IoT to avoid spreading of COVID-19. *International Journal of Intelligent Networks* vol. 1, pp. 32–35, 2020.

Lopes, C. C., Cesário-Times, V., Matwin, S., Ciferri, C. D. d. A., and Ciferri, R. R. An Encryption Methodology for Enabling the Use of Data Warehouses on the Cloud. In *Research Anthology on Artificial Intelligence Applications in Security*. IGI Global, Hershey, PA, USA, pp. 528–559, 2021.

Mateus, R. C., Siqueira, T. L. L., Times, V. C., Ciferri, R. R., and Ciferri, C. D. A. Spatial data warehouses and spatial OLAP come towards the cloud: design and performance. *Distributed and Parallel Databases* 34 (3): 425–461, 9, 2016.

Medvedev, A., Zaslavsky, A., Santiago, M. I., Haghighi, P. D., and Hassani, A. Storing and indexing IoT context for smart city applications. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9870 LNCS. Springer Verlag, St. Petersburg, Russia, pp. 115–128, 2016.

Ni, J., Zhang, K., Lin, X., and Shen, X. S. Securing Fog Computing for Internet of Things Applications: Challenges and Solutions. *IEEE Communications Surveys and Tutorials* 20 (1): 601–628, 1, 2018.

Pandey, V., Kipf, A., Neumann, T., and Kemper, A. How good are modern spatial analytics systems? *Proc. VLDB Endow.* 11 (11): 1661–1673, July, 2018.

Patel, K. K. and Patel, S. M. Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. *IJSR* 6 (5): 6122–6132, 2016.

Peng, G. C. A., Nunes, M. B., and Zheng, L. Impacts of low citizen awareness and usage in smart city services: the case of London's smart parking system. *Information Systems and e-Business Management* 15 (4): 845–876, 11, 2017.

Pérez de Prado, R., García-Galán, S., Muñoz-Expósito, J. E., Marchewka, A., and Ruiz-Reyes, N. Smart Containers Schedulers for Microservices Provision in Cloud-Fog-IoT Networks. Challenges and Opportunities. *Sensors* 20 (6): 1714, 3, 2020.

Puthal, D., Mohanty, S. P., Bhavake, S. A., Morgan, G., and Ranjan, R. Fog Computing Security Challenges and Future Directions [Energy and Security]. *IEEE Consumer Electronics Magazine* 8 (3): 92–96, 5, 2019.

Rahman, A., Ermatita, and Budianta, D. Data Warehouse Design for Soil Nutrients with IoT Based Data Sources. In *Proceedings - 1st International Conference on Informatics, Multimedia, Cyber and Information System, ICIMCIS 2019*. Institute of Electrical and Electronics Engineers Inc., Jakarta, Indonesia, Indonesia, pp. 181–186, 2019.

Ramaswami, A., Russell, A. G., Culligan, P. J., Sharma, K. R., and Kumar, E. Meta-principles for developing smart, sustainable, and healthy cities. *Science (New York, N.Y.)* 352 (6288): 940–3, 5, 2016.

Rauf, A., Shaikh, R. A., and Shah, A. Security and privacy for IoT and fog computing paradigm. In *2018 15th Learning and Technology Conference, L and T 2018*. IEEE, Jeddah, KSA, pp. 96–101, 2018.

Rivest, S., Bédard, Y., and Marchand, P. Toward better support for spatial decision making: defining the characteristics of Spatial On-Line Analytical Processing (SOLAP). *Geomatica* 55 (4): 539–555, 2001.

Sadalage, P. J. and Fowler, M. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional, Chicago, IL, USA, 2012.

Santos, J. P. C., Castro, J. P. d. C., and Ciferri, C. D. d. A. SOLAP Query Processing over IoT Networks in Smart Cities: A Novel Architecture. In *Anais do XXI GeoInfo - Simpósio Brasileiro de Geoinformática*. INPE, São José dos Campos, Brazil, pp. 118–129, 2020.

Shi, W. and Dustdar, S. The Promise of Edge Computing. *Computer* 49 (5): 78–81, 5, 2016.

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. The Hadoop Distributed File System. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, Incline Village, NV, USA, pp. 1–10, 2010.

Silva, B. N., Khan, M., and Han, K. Integration of Big Data analytics embedded smart city architecture with RESTful web of things for efficient service provision and energy management. *Future Generation Computer Systems* vol. 107, pp. 975–987, 6, 2020.

Singh, S., Jeong, Y. S., and Park, J. H. A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications* vol. 75, pp. 200–222, 11, 2016.

Vaisman, A. and Zimnyi, E. *Data Warehouse Systems: Design and Implementation*. Springer Publishing Company, Incorporated, Berlin, Heidelberg, Germany, 2014.

Xu, Q. and Zhang, J. PiFogBed: A Fog Computing Testbed Based on Raspberry Pi. In *2019 IEEE IPCCC*. IEEE, London, United Kingdom, 2019.

Yang, S. IoT Stream Processing and Analytics in the Fog. *IEEE Communications Magazine* 55 (8): 21–27, 2017.

Yeh, H. The effects of successful ICT-based smart city services: From citizens' perspectives. *Government Information Quarterly* 34 (3): 556–565, 9, 2017.

Yu, J., Zhang, Z., and Sarwat, M. Spatial data management in apache spark: the geospark perspective and beyond. *GeoInformatica* 23 (1): 37–78, 2019.

Yuan, L. and Zhao, J. Construction of the system framework of Spatial Data Warehouse in Internet of Things environments. In *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, Nanjing, China, pp. 54–58, 2012.

Zaharia, M., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., and Venkataraman, S. Apache Spark. *Communications of the ACM* 59 (11): 56–65, 10, 2016.

Zhang, H., Babar, M., Tariq, M. U., Jan, M. A., Menon, V. G., and Li, X. SafeCity: Toward Safe and Secured Data Management Design for IoT-Enabled Smart City Planning. *IEEE Access* vol. 8, pp. 145256–145267, 2020.