SOLAP Query Processing over IoT Networks in Smart Cities: A Novel Architecture

João Paulo Clarindo dos Santos[1], João Pedro de Carvalho Castro[1,2], Cristina Dutra de Aguiar Ciferri[1]

[1]Institute of Mathematics and Computer Science – University of São Paulo – Brazil

[2]Computing Center – Federal University of Minas Gerais – Brazil

jpcsantos@usp.br, jpcarvalhocastro@ufmg.br, cdac@icmc.usp.br

Abstract. *Spatial data generated by an Internet of Things (IoT) network is important to assist the decision-making in issues related to smart cities. In these cities, IoT devices generate spatial data constantly. Thus, data get increasingly voluminous very fast. In this paper, we investigate the challenge of managing these data through the use of a spatial data warehouse and spatial on-line analytical processing designed over a parallel and distributed processing framework extended with a spatial analytics system. We propose a novel architecture aimed to assist a smart city manager in decision-making, which integrates a cloud layer where these technologies are located with a fog computing layer for extracting, transforming and loading. Furthermore, we introduce a set of guidelines to aid smart cities managers to implement the proposed architecture. We validate our architecture with a case study that uses real data collected by IoT devices in a smart city.*

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. It "*involves the implementation and deployment of information and communication technology (ICT) infrastructures to support social and urban growth through improving the economy, citizens involvement, and government efficiency*" [Yeh 2017].

A network of Internet of Things (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 a public transportation system, whose fleet contains sensors that collect data related to the number of passengers, vehicle type (buses, trams, etc.), route taken, and maximum speed, aiming to improve the existing lines. Another IoT application scenario includes air monitoring, with sensors scattered around the city collecting data about pollution in order to identify if air quality improvements are necessary in certain regions [Atzori et al. 2017].

An IoT network contained in a smart city tends to generate spatial data, usually represented by geometries (such as points, lines, and polygons) or combinations of these. For example, smartphones can contain sensors that use location data to connect people with the same hobbies or relationship status living in the same area. The spatial properties of IoT devices can be determined directly by the sensors, using satellite positioning techniques, like GPS and GLONASS [van der Zee and Scholten 2014, 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 manager of a public transportation system can be interested in determining *how many passengers were transported last month, considering the type of vehicle, route, and region*. The query results can be displayed on a map according to the region, helping the manager to obtain the necessary knowledge in an intuitive manner. 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). An SDW is a subject-oriented, integrated, time-variant and non-volatile collection of conventional and spatial. 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 SDWs can benefit from the use of a cloud computing environment as infrastructure and from the employment of parallel and distributed processing frameworks, such as Hadoop [Shvachko et al. 2010] and Hadoop Spark [Zaharia et al. 2016], to reduce the complexity of the cloud. The processing of the SOLAP queries can also benefit from the use of spatial analytics systems (SASs), which are developed on the top of parallel and distributed processing frameworks to provide extended functionalities to deal with spatial data [Castro et al. 2020].

The challenge is to propose an IoT architecture for smart cities that encompasses all these technologies and also provides efficient support for storing SDWs and processing SOLAP queries. Although there are some proposals of architecture proposed in the literature [Yuan and Zhao 2012, Bonomi et al. 2014, Eldrandaly et al. 2019], they do not focus on SDWs and SOLAP in a parallel and distributed processing environment. In this paper, we overcome these shortcomings.

The contributions of our paper are described as follows.

- The proposal of an architecture aimed to help smart cities managers and residents in their decision-making process through the employment of an SDW that uses a parallel and distributed data processing framework in the cloud and also uses SASs to process SOLAP queries.
- The definitions of guidelines to assist in the proposed architecture implementation.
- The validation of the efficacy and effectiveness of the architecture with a case study that describes an application that handles real data generated from a smart city.

This paper is organized as follows. Section 2 reviews related work, Section 3 presents the proposed architecture, Section 4 introduces the guidelines for implementing the architecture, Section 5 describes the case study, and Section 6 concludes the paper.

2. Related Work

There are studies in the literature that present challenges related to IoT-generated data, considering general [Patel and Patel 2016, Atzori et al. 2017] and smart city scenarios [Arasteh et al. 2016, van der Zee and Scholten 2014, Theodoridis et al. 2013]. In the context of big data, some work has been done to manipulate IoT spatial data. These proposals are described as follows.

Yuan and Zhao (2012) propose an architectural solution for SDWs in the context of IoT environments (SDWIT). This architecture has the following layers: data processing layer, storage layer, and analysis application layer. SDWIT features include accessing and analysing IoT data in real time over a traditional SDW. However, the authors did not consider parallel and distributed data processing frameworks in their architecture, making SOLAP operations difficult for very large SDWs.

Bonomi et al. (2012) introduce a highly virtualized platform called *fog computing*, which "*provides computing, storage and networking services between end devices and traditional cloud computing data centres, typically, but not exclusively located at the edge of network*". Fog computing aims at low latency between the edge and the core of the network, very large number of nodes, and wide-spread geographical distribution. Therefore, the platform is appropriate for operations that have IoT services. The fog computing platform is expanded in [Bonomi et al. 2014] to deal with a massively distributed number of sources at the edge. Regarding applications that require analytics over longer periods or wider scenarios, like an SDW, these proposals only suggest that the corresponding operations should be performed in the cloud. No further investigation is conducted.

Eldrandaly et al. (2019) define the concept of Internet of Spatial Things (IoST), which "*is an integrative paradigm of embedded smart devices concerned with collecting spatial data of objects to serve a significant purpose*". The authors also introduce a framework for an IoST network that uses a fog computing platform for real-time spatial computing. Data are collected and then sent to a fog node for temporary storage and processing. Finally, data are extracted, transformed and loaded into a cloud database. Although the framework provides analytics operations that are defined according to the requirements of the enterprise, these operations do not focus on the processing of queries extended with spatial predicates over SDWs.

In contrast to the described approaches, we propose a novel architecture that employs both parallel and distributed data processing frameworks and SASs, allowing the execution of fast and reliable 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 layer to handle SOLAP data analytics and SDW Extract, Transform, and Load (ETL) processes.

We also introduce a set of guidelines to aid smart cities 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 paper.

3. 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 architecture (Figure 1) achieves these goals through the employment of three different layers: (i) the terminal layer; (ii) the fog layer; and (iii) the cloud layer. We discuss each layer as follows.

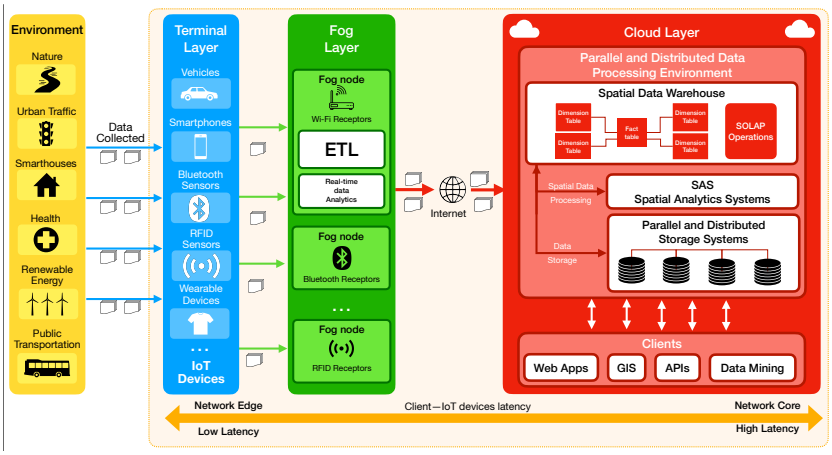

Figure 1. Architecture overview.

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), Wireless Sensor Network (WSN), 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 layer. Data collected by terminal layer devices are sent to receivers in the fog 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 layer. After the data goes through the ELT/ELT process in the fog layer, it is sent to the cloud 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).

4. Guidelines for Implementing the Proposed Architecture

In this section, we propose a set of guidelines to aid smart cities managers in the process of implementing the proposed architecture. Because the context behind each smart city may be different, is not mandatory to follow every guideline in its completeness. Managers should choose the appropriated hint 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 specialization 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. For instance, vehicle sensors can use GPS, while temperature sensors can use 4G/5G protocols. A smart city manager must also consider the communication compatibility between these devices and the fog nodes. We recommend the framework proposed by [Theodoridis et al. 2013] to assist these managers in the process of integrating these devices in a smart city scenario.

Guideline 2. Distributing fog nodes across the fog 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[1], which are small single-boarded computers, as fog nodes, using containerization over these resource-limited devices [Bellavista and Zanni 2017, Xu and Zhang 2019]. Each fog node uses the Docker container technology[2] for creating containers for each application available in fog node (i.e. ETL and real-time data analytics). 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 layer can be carried out using 4G/5G or Wi-Fi protocols.

Guideline 3. Securing the connection between IoT devices and fog nodes. A smart city manager must be concerned with the dataflow between the IoT devices and the fog nodes, as sensitive information may be transmitted. Malicious attacks in a fog computing environment must also be considered. To deal with these issues, smart cities managers can take decisions using as a basis the work of [Mukherjee et al. 2017]. In this work, the authors determine the impact of security problems on a fog network and also provide solutions to increase the security of these environments.

[1] https://www.raspberrypi.org/
[2] https://www.docker.com/

Guideline 4. Configuring the ETL process in the fog layer. To enable ETL processing in the fog layer, a smart city manager must select tools that allow programming, scaling and monitoring the tasks of the ETL workflow. There are several tools on the market which support ETL and workflow monitoring. An example is Apache Airflow[3], which is an open-source platform that uses directed acyclic graphs (DAGs) for authoring, scheduling and monitoring workflows. The tasks of the process should be written in the Python programming language, since it is natively supported by Airflow. Airflow provides integration with Hadoop, Spark and several cloud platforms.

Guideline 5. Enabling real-time data analytics in the fog layer. In a fog node, data are loaded constantly, enabling real-time data analytics. To this end, a smart city manager can use multiple data management systems, like NoSQL databases (i.e. Couchbase Server[4] and Apache Cassandra[5]) and event streaming platforms such as Apache Kafka[6], operated in containers inserted in the fog node. These platforms support communication by APIs, enabling real-time spatial data analytics over SDWs.

Guideline 6. Choosing the appropriate SAS to implement the SDW in the cloud layer. The SDW application should process SOLAP queries efficiently. Therefore, a smart city manager must select a SAS that is able to completely fulfill the requirements of the SDW application. Because there several SASs available in the literature with different characteristics and capabilities, the choice of the most appropriate SAS burdens the selection process considerably. 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].

Guideline 7. Configuring the SDW to process SOLAP queries on the cloud layer. After choosing the appropriate SAS, a smart city manager must configure the SDW environment in the cloud layer to process SOLAP queries. The parallel and distributed 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[7] and Amazon Web Services[8]. The smart city manager must consider the periodicity that data should be extracted from the fog layer, as well as carefully specify data distribution over the SDW. The SOLAP services must support APIs and GIS applications in order to visualize the result of SOLAP queries.

Guideline 8. Ensuring secure SOLAP query processing in the cloud layer. Since cloud computing environments can be virtually accessed from anywhere, smart cities managers should be concerned with security issues related to SDW applications. That is, smart cities managers should define restrict protocols with regard to roles, permissions, and data confidentiality. A solution to ensure confidentiality is the encryption of the data stored in the SDW. Data encryption should be done carefully to not compromise the performance of the SDW application. To this end, the encryption methodology proposed in [Lopes et al. 2014] can be employed, as it allows the efficient processing of analytical queries over encrypted data warehouses.

[3] http://airflow.apache.org/
[4] https://www.couchbase.com
[5] http://cassandra.apache.org
[6] http://kafka.apache.org
[7] http://azure.microsoft.com
[8] http://aws.amazon.com

5. 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. For this case study, we use a dataset provided by [Ali et al. 2015], which contains vehicle traffic data observed between two points. These data were collected from sensors distributed in the municipality of Aarhus, Denmark. The dataset, which is publicly available in the authors' website[9], contains both conventional (i.e., distance in meters between the sensors, type of road, etc.) and spatial data (i.e., the sensors locations, represented by points) referring to the period from February to June 2014. As this 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[10] from OpenStreetMaps, and statistical district data obtained from OpenDataDK[11]. 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.

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 2, which should also be located in the cloud. There are six dimension tables in the SDW: (i) Date and Time, storing the moment in which a measurement occurred; (ii) Report, storing the distance between the two sensors that performed the measurement and their geographic locations; and (iii) Road, District and City, storing the geographic locations associated with the report. The dimension tables are linked through the fact table Measurement, which stores both the measurement time and the vehicle speed. The fact table also stores the vehicle count for each measurement of the IoT sensors.

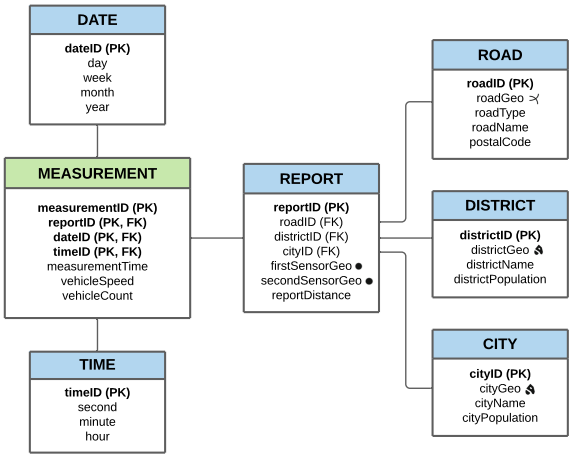

Figure 2. Logical schema of the SDW stored in the cloud.

[9] http://iot.ee.surrey.ac.uk:8080/
[10] https://www.geofabrik.de/
[11] https://www.opendata.dk/city-of-aarhus/statistikdistrikter

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 3) and guidelines (Section 4), 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 GeoSpark [Yu et al. 2019] as the SAS to process the SOLAP queries, 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 GeoSparkSQL [Pandey et al. 2018, Castro et al. 2020].

Data loading into the cloud layer. The dataset used in this case study consists of 449 reports. Data from these reports 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 layer (Guideline 4). 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 2; and (iii) load the data into HDFS for later use by GeoSpark. To accurately simulate the fog layer, Airflow should be executed from a Docker container.

Converting textual representations of spatial data into spatial objects. In order to employ GeoSparkSQL for processing SOLAP 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. GeoSparkSQL 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
```

Once the process of loading the data provided by the IoT sensors into the SDW is complete, smart cities managers are able to execute different types of SOLAP queries using GeoSparkSQL. Some query examples that address key points of the application's requirements are defined as follows. We employ QGIS[12] to visualize the query results.

Spatial Join Query. This query returns 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 3, 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:

[12]https://qgis.org/

```
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 cities' 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 3), 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 3).

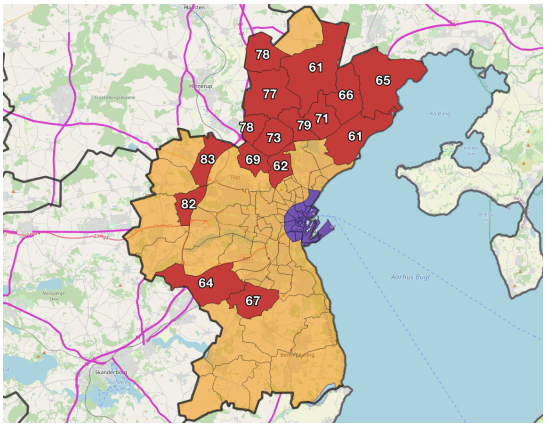

Figure 3. Spatial join query results.

Containment query. This query returns the quantity of vehicles that travelled in Aarhus University/Community Hospital district grouped by day. 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 4. The following command expresses this query:

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

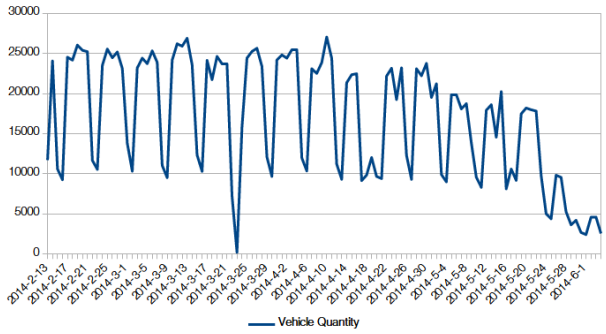

Figure 4. Containment query results.

By interpreting the query results, a smart cities manager can obtain different types of knowledge. For instance, the measurement of zero vehicles in 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.

K-nearest neighbours query. This query returns 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 accessed point of interest, as shown in Figure 5. 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
```

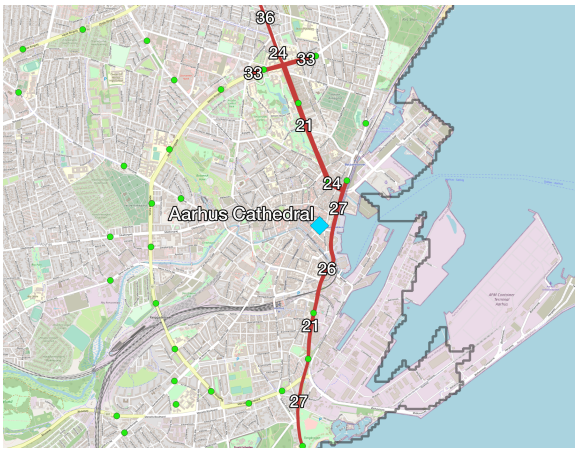

Figure 5. K-nearest neighbours query results.

The results displayed in Figure 5 can enable smart cities 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 cities 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.

6. Conclusions and Future Work

In this paper, we propose a novel architecture for enabling the execution of fast and reliable analyses over IoT data from smart cities. The architecture employs parallel and distributed data processing frameworks and spatial analytics systems in a cloud computing environment. Besides this cloud layer, our architecture also includes a fog layer, responsible for handling both real time data analytics and ETL processes; and a terminal layer, where the IoT devices are located. Further, we introduce a set of guidelines in order to aid smart cities managers in the process of implementing our architecture. Finally, we validate the proposed architecture by employing it to implement an SDW application that analyses data collected from real IoT devices in a smart city.

Future work includes describing additional case studies with sensors that collect measurements from different contexts, such as temperature and pollution levels. Another future work consists in the proposal of algorithms to optimize SOLAP query processing using as a basis the components of the proposed architecture.

Acknowlegments

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

References

Ali, M. I., Gao, F., and Mileo, A. (2015). CityBench: A configurable benchmark to evaluate RSP engines using smart city datasets. In *LNCS*, volume 9367, pages 374–389.

Arasteh, H., Hosseinnezhad, V., Loia, V., Tommasetti, A., Troisi, O., Shafie-khah, M., and Siano, P. (2016). Iot-based smart cities: A survey. In *2016 IEEE 16th EEEIC*, pages 1–6. IEEE.

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

Bellavista, P. and Zanni, A. (2017). Feasibility of fog computing deployment based on docker containerization over RaspberryPi. In *ACM ICPS*, pages 1–10. ACM.

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

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the MCC '12*, page 13.

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

Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile Netw Appl*.

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

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

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

Han, J., Stefanovic, N., and Koperski, K. (1998). Selective materialization: An efficient method for spatial data cube construction. In *LNCS*, volume 1394, pages 144–158.

Lopes, C. C., Times, V. C., Matwin, S., Ciferri, R. R., and Ciferri, C. D. A. (2014). Processing olap queries over an encrypted data warehouse stored in the cloud. In *16th DaWaK*, pages 195–207. Springer.

Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M. A., Choudhury, N., and Kumar, V. (2017). Security and Privacy in Fog Computing: Challenges. *IEEE Access*, 5:19293–19304.

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

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

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

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

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The Hadoop Distributed File System. In *2010 IEEE 26th MSST*, pages 1–10.

Theodoridis, E., Mylonas, G., and Chatzigiannakis, I. (2013). Developing an IoT Smart City framework. In *4th IISA*, pages 180–185.

van der Zee, E. and Scholten, H. (2014). Spatial dimensions of big data: Application of geographical concepts and spatial technology to the internet of things. *SCI*, 546:137–168.

Xu, Q. and Zhang, J. (2019). PiFogBed: A Fog Computing Testbed Based on Raspberry Pi. In *2019 IEEE IPCCC*. Institute of Electrical and Electronics Engineers Inc.

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

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

Yuan, L. and Zhao, J. (2012). Construction of the system framework of Spatial Data Warehouse in Internet of Things environments. In *5th IEEE ICACI*, pages 54–58.

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. (2016). Apache Spark. *Communications of the ACM*, 59(11):56–65.