Master in Technology & Innovation Management (MTIM)

School of Production Engineering and Management

Technical University of Crete

Big Data Analytics for Smarter Cities:

Method – Architecture – Algorithmic Apparatus

Eirini Kontaki

M.Sc. Dissertation

February 24th, 2020

ABSTRACT

A smart city is a concept that integrates multiple Information and Communication Technology (ICT) solutions to efficiently manage city's assets. City's assets include, but are not limited to, transportation systems, waste management, water management, safety systems, local departments' information systems and other community services. In addition, the concept of the smart city is to make traditional networks and services more efficient with the use of technologies, including Internet of Things (IoT) and Big Data, to improve the quality of people's life and to boost the operation of city's businesses. Moreover, a smart city is an inherently distributed setting composed of a number of sites. Sites may range from conventional or modern data gathering devices, such as sensors, RFIDs, smartphones, and so on, which collect streams of relevant data, to intermediate routers that convey these data to local departments' systems and finally to city authorities' data centers. Valuable information hidden in the streaming data gathered by the distributed setting should be provided in real-time and in a continuous fashion to timely support decision making procedures. However, extracting value, in the form of analytics, out of the massive flows of data that stream-in, such a distributed setting is an intriguing task.

This thesis focuses on highlighting the importance of big data analysis in a smart city context by reviewing existing technologies that enable city authorities to produce valuable insights and facilitate the decision-making process both in the short and in the long run. More specifically, we identify the needs of the city's core systems, to facilitate its operational activities and coordinate its services, approaching a smart city as an instrumented city. We consider instrumentation as the process in which an interconnected system provides solutions and shares data to enable collaboration across entities and domains for the benefit of the society and its people, as well. This work analyzes arising issues in three vertical city domains: (i) smart transportation, (ii) smart resources and (iii) smart environment and outlines potential benefits associated with IoT-powered cities that handle large amounts of data. In this direction, we incorporate all the above in a fully-fledged architectural design proposal tailored for smart cities. After a thorough study on existing big data technologies, we provide indicative solutions using big data platforms addressing specific technical requirements and review existing algorithms that can be implemented with such platforms.

As an in-depth investigation, we selected the setup of Heraklion Smart City as a case study to exemplify the advantages in terms of business's logic efficiency using big data technologies along with the aforementioned solutions methodology and proposed architecture. In this context, we present the processing pipeline of big data on various Heraklion city's core systems including parking, water and air quality monitoring, reporting future benefits for the stakeholders.

Acknowledgments

I wish to thank the members of my dissertation committee: Prof. Minos Garofalakis, Dr. Nikolaos Giatrakos and Prof. Nikolaos Matsatsinis for generously offering their time, support, guidance and good will throughout the preparation and review of this document. I especially, owe my gratitude to Dr. Nikolaos Giatrakos for his intellectual enrichment, thoughtful and generous criticism and strong support. He provided me with valuable guidance and motivated me to improve my research skills with his close attention to my work and critical thinking. Moreover, I would like to express my sincere gratitude to Prof. Constantine Stephanidis for his support and encouragement throughout my studies.

I am also grateful to my beloved husband, Manos Zidianakis, who provided me with incredible support, encouragement and advice. This thesis was enriched significantly through our fruitful discussions, while his guidance was invaluable in order to shape this work to its current state.

I owe my thanks to my colleagues and friends, Ioanna and Argiro for their valuable comments during the preparation process of my thesis presentation.

Finally, I would like to thank my parents, Katerina and Nikos, who supported me from the beginning of my studies and never stop giving of themselves in countless ways, as well as my two brothers, Petros and Manos, who are always there for me. The last word goes to Giorgos, my baby boy, who has been the light of my life for the last nine months and who has given me the extra strength and motivation to prove that there is nothing that cannot be done.

Table of Contents

ABSTRACT II

1 INTRODUCTION 1

1.1 The rising need of smart cities 1

1.2 IoT-based smart cities 1

1.3 Big Data in smart city's ICT network 3

1.4 Contribution 4

1.5 Thesis outline 4

2 A SMART CITY'S ICT NETWORK 6

2.1 Main sources of data generation in a Smart City Infrastructure 6

2.1.1 Smart Devices 6

2.1.2 Web Services 7

2.1.3 Sensor devices 7

3 Smart City Focus Areas 9

3.1.1 Smart Transportation Management 9

3.1.1.1 Road traffic monitoring 9

3.1.1.2 Parking management 11

3.1.2 Smart Resource Management 13

3.1.2.1 Water management 13

3.1.2.2 Waste management 15

3.1.3 Smart Environmental Management 17

3.1.3.1 Air Quality Monitoring 18

3.1.3.2 Weather Monitoring 19

4 BUSINESS OBJECTIVES AND KEY PERFORMANCE INDICATORS 21

4.1 Business Objectives for Smart Transportation 23

4.2 Business Objectives for Smart Resources 24

4.3 Business Objectives for Smart Environment 25

5 TOWARDS ACHIEVING THE BUSINESS OBJECTIVES 26

5.1 Big data processing pipeline 26

5.2 Business logic scenarios for big data processing pipelines 28

5.3 Big Data volume analysis 30

5.4 Technological Challenges 31

5.4.1 Challenge 1: Horizontal scalability 31

5.4.2 Challenge 2: Vertical scalability 32

5.4.3 Challenge 3: Federated scalability - Communication reduction 33

5.4.4 Challenge 4: Fast setup of new big data processing pipelines 33
5.4.5 Challenge 5: Cross platform execution of big data analytics tasks 34
5.4.6 Challenge 6: Optimized big data pipelines' execution over heterogeneous clusters or clouds 34
6 PROPOSED ARCHITECTURAL FRAMEWORK 35
6.1 A Generic Architectural Framework 35
6.2 Architectural Framework Materialization 39
6.2.1 Platforms for Workflow Design Facilities and Cross Platform Optimization 40
6.2.2 Visualization and Analytics 41
6.2.3 Optimizer Components 44
6.2.4 Big Data Management Platforms 46
6.2.4.1 Data Ingestion Layer 46
6.2.4.2 Batch Layer 48
6.2.4.3 Speed Layer 50
6.2.4.4 Serving Layer 52
7 ALGORITHMIC APPARATUS & IMPLEMENTATIONS 53
7.1 In-situ Processing Algorithms & Implementations 53
7.2 In-network Processing Algorithms & Implementations 55
7.3 Data Summarization Techniques & Implementations 55
7.4 Machine Learning and Data Mining Algorithms and Libraries 58
8 CASE STUDY: HERAKLION SMART CITY 64
8.1 Identifying the needs of Heraklion city 64
8.2 The IoT infrastructure 66
8.3 The WSN architecture 66
8.4 Data processing pipelines for Heraklion Smart City 68
8.4.1 Parking management using Apache Storm 68
8.4.2 Water management using Apache Spark 72
8.4.2.1 Water level and water quality monitoring in city's water reservoirs 72
8.4.2.2 Water leakage detection 73
8.4.3 Environmental monitoring using Apache Flink 76
8.5 Using Apache Beam for cross platform data processing 80
9 CONCLUSION AND FUTURE WORK 84
REFERENCES 85

List of Figures

Figure 1: The core domains in a smart city framework (Source: Deloitte insights) 2
Figure 2: A smart on-road traffic management infrastructure (Source: ScienceSoft) 10
Figure 3: Smart city pilot project for parking optimization 12
Figure 4: A smart parking infrastructure using magnetic sensors (source: Libelium) 13
Figure 5: An example of Smart Water Network infrastructure 14
Figure 6: Common situation of garbage cans' filling status in todays' cities 16
Figure 7: Scenario for municipal waste management17
Figure 8: Mobile application examples illustrating air pollution values and protection recommendations 18
Figure 9: Indicative application example for sun protection 20
Figure 10: A cloud-computing infrastructure based on IBM Aspera software 27
Figure 11: Data analytic pipeline 27
Figure 12: Framework architecture. Addressing the requirements arising from the technological challenges 35
Figure 13: Framework architecture. A more focused analysis 37
Figure 14: Lambda Architecture from a high-level perspective 38
Figure 15: Framework architecture. Big data platforms 39
Figure 16: The Beam model – Data processing pipelines (Source: Data Science Central) 40
Figure 17: Visual analytics tools: an example of Kibana 41
Figure 18: Visual analytics tools: an example of Grafana 42
Figure 19: Visual analytics tools: an example of Apache Zeppelin 42
Figure 20: Visual analytics tools: an example of Zoomdata 43
Figure 21: Visual analytics tools: an example of Tableau 43
Figure 22: Visual analytics tools: an example of Qlik 44
Figure 23: High-level flow of MESOS resource allocation 46
Figure 24: Framework architecture. A more focused analysis based on specific tools 48
Figure 25: An RDD distributed across the worker nodes (Source: DevOps) 49
Figure 26: Data stream in Apache Flink (task manager) 51
Figure 27: Core components of the Flink CEP library 51
Figure 28: Geometric Monitoring method rationale 54
Figure 29: Example of different push-pull application for an AND operator receiving 3 events (e1, e2, e3) at input 55
Figure 30: Clustering output depicting traffic propagation relationship 60
Figure 31: Clustering output depicting traffic split relationship 61
Figure 32: Clustering output depicting traffic merge relationship 61
Figure 33: Random Forest Algorithm graph 62
Figure 34: Support Vector Machine Algorithm 63
Figure 35: k-Nearest Neighbors (kNN) Algorithm 63
Figure 36: The Heraklion Smart City ecosystem 66
Figure 37: Pilot sensor infrastructure in Heraklion city (Source: www.smartcity.heraklion.gr) 67
Figure 38: A database schema approach 67
Figure 39: An Apache Storm topology for parking monitoring (logical view) 68
Figure 40: Tuples emitted throughout the Apache Storm topology 69
Figure 41: A data pipeline for water monitoring in Heraklion city 74
Figure 42: Tuples emitted throughout the Spark topology 75
Figure 43: A physical dataflow plan for air quality monitoring 77
Figure 44: Tuples emitted throughout the Apache Flink topology 78
Figure 45: A Data processing pipeline with the Beam model 81
Figure 46: PCollections emitted through the Beam pipeline 82

List of Tables

Table 1: Different sensing technologies 8
Table 2: Sub-dimensions of KPIs 9
Table 3: Measurable Objectives – Key Performance Indicators (KPIs) 22
Table 4: Data volumes extracted from a smart city IoT infrastructure 30
Table 5: Exemplary Data Summarization Techniques and their Utility in the Smart City Context 56
Table 6: Measured parameters per area of interest 66

1 INTRODUCTION

1.1 The rising need of smart cities

During the past decades, the planet has witnessed rapid urbanization and overpopulation growth with the world's population reaching 7.7 billion in mid-2019. According to [180], the global population is expected to reach 8.5 billion in 2030, 9.7 billion in 2050 and 10.9 billion in 2100, while by 2050 the population that lives in urban areas will be increased reaching 70% of the total population globally [155].

Urbanization is considered as one of the main driving forces for the change of ecosystem services, including water and energy consumption, that lead directly to climate change. The World Health Organization (WHO) [178] reports that rapid, unplanned and unsustainable patterns of urban development are making developing cities focal points for many emerging environment and health hazards. As urban populations grow, the quality of global ecosystems and the urban environment will play an increasingly important role in public health with respect to issues ranging from waste disposal to provision of safe water and sanitation.

Cities, despite their population size, need to become more resource efficient and technology driven in order to mitigate their impact and exploit innovative solutions towards enhancing urban sustainability throughout the world. According to [99], a sustainable city is based on the following four main aspects:

- Economic: The ability to generate income and employment for the livelihood of the inhabitants.
- Social: The ability to ensure well-being (safety, health, education, etc.) of the citizens can be equally delivered despite differences in class, race or gender.
- Environmental: The ability to protect future quality and reproducibility of natural resources.
- Governance: The ability to maintain social conditions of stability, democracy, public participation in online city services, and justice.

1.2 IoT-based smart cities

According to [31], sustainable cities lack of innovative solutions based on Internet of Things (IoT) applications in several urban domains where such solutions can have a substantial contribution to the goals of environmentally sustainable development. In this context, **smart cities** come to focus on searching greener and more energy efficient urban dynamics by integrating Information and Communication Technologies (ICTs). Smart cities were firstly introduced as a concept in scientific publications in 1992 and emerged as a fast-growing topic of scientific enquiry [124]. Since then, many different perceptions of smart cities prevail with a divergence and lack of cohesion. However, smart cities appear to be the hottest emerging research and business topic of the 21st century [38].

IoT-based cities become instrumented and interconnected in a way to properly confront with challenges and threats to their core systems. They deploy a mechanism that assists stakeholders to comprehend existing problems, to extract the necessary knowledge and take the appropriate actions.

One of the leading tech companies, IBM, underlines that cities need to become smarter by seizing opportunities through managing the resources efficiently and giving the accurate information in order to prevent an unexpected event [3]. It also reports that cities can become smarter when using new technologies to transform their core systems and optimize the use of limited resources. Pervasive new technologies provide a much greater scope for instrumentation, interconnection and intelligence of a city's core systems.

One of the core mechanisms of smart cities is a self-monitoring and self-response system, enabling them to identify and fix problems promptly, recover rapidly from disasters, gather data to make better decisions and deploy resources effectively [125].

The overall smart cities' objective is to manage assets and resources efficiently in areas such as transportation, health, education, air quality, energy and water services [42]. In that, smart cities aim to enhance the quality of life by facilitating everyday activities in order to support future generations with respect to economic, social and environmental aspects.

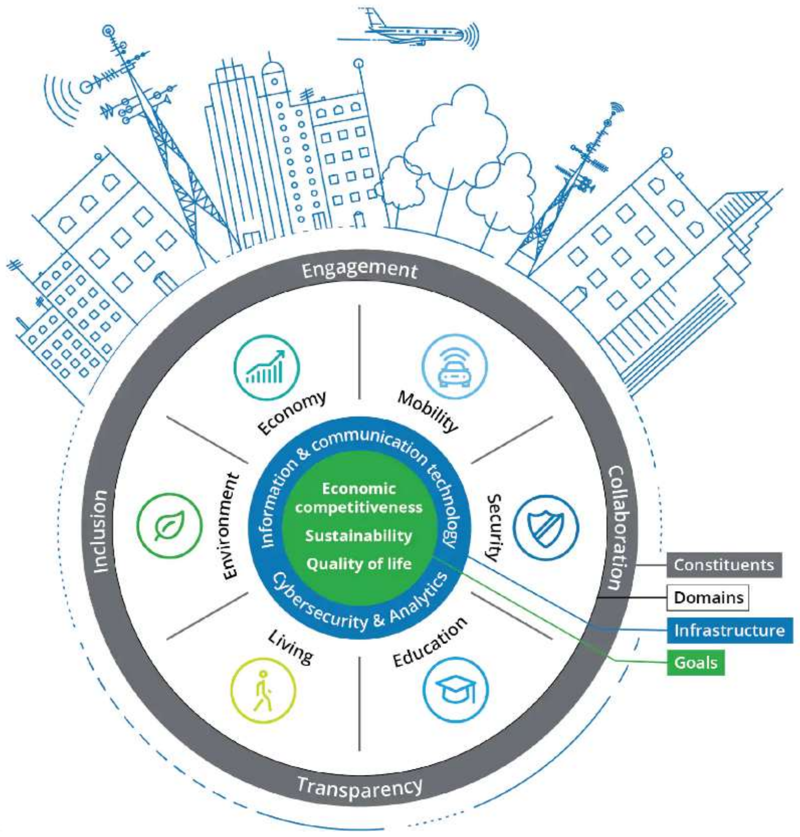


Figure 1: The core domains in a smart city framework (Source: Deloitte insights [52])

ICTs are key drivers of smart city initiatives [96] and enhance the management and functioning of a city since they offer a number of potential opportunities [133]. Towards that direction, national governments and local authorities strongly support technological growth by integrating ICT solutions to municipality and city landscapes, thus absorbing the concept of a smart city [44]. It is worth pointing out that little effort has been made globally towards facing environmental aspects with the assistance of advanced technological solutions [32, 95, 109].

1.3 Big Data in smart city's ICT network

Since digitization has become an integral part of everyday life and many cities have already adapted technologies in order to meet the aforementioned needs, data collection has resulted in the accumulation of huge amounts of data that can be used in various beneficial application domains. Data is being generated from multiple sources, such as sensor and smart devices, cameras, smart phones, social media, commercial transactions, advertising application and so on. All this is made possible due to the continuous and rapid diffusion of electronic devices capable of retrieving and transmitting data, which have supported the growth of the Internet of Things (IoT) [124, 173]. Smart cities' IoT-like infrastructure allows data to be collected: (a) in huge volumes, (b) in the form of continuous, volatile, high speed data streams and (c) in both structured and unstructured format in multiple modalities (video, image, GPS, time series). Data is so complex that it is difficult to ingest, store, manage and process using traditional database and software tools [112]. Gartner, a leading research and advisory company in Information Technology, summarizes **big data** as high volume, high velocity and/or high variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision-making, and process automation [68].

Adapting an ICT network in a smart city setup implies the use of different computing technologies depending on the requirements and the demand on different sectors. New technological advances allow wireless communications offering great standardization of low-power communication protocols, which make it possible to obtain sensor data almost everywhere and at any time. A smart city network comprises a number of technological components, including the physical sensor infrastructure, the network infrastructure, an IoT middleware for digesting heterogeneous data streaming in from the network, data processing and analytics platforms and visualization tools illustrating the outcomes extracted from analytics procedures [103].

Although there are many different domains where big data find application in, this thesis focuses on their impact in smart cities towards a sustainable economic growth and an enhanced life quality.

1.4 Contribution

Smart cities constitute the cornerstone towards creating sustainable urban dynamics to mitigate excessive resource consumption for the benefit of the environment and human well-being. This thesis focuses on highlighting the importance of big data analysis in a smart city context by reviewing existing technologies that enable city authorities to produce valuable insights and facilitate the decision-making process both in the short and in the long run.

Towards that direction, this work:

(i) emphasizes on specific smart city focus areas (i.e., transportation, resource and environmental management) presenting the benefits of the provision of ICTs,

(ii) presents usual problems related to the aforementioned focus areas,

(iii) reviews existing technological solutions with a reference to existing sensing technologies,

(iv) analyzes and translates usual problems into business objectives and key performance indicators,

(v) underlines the technological challenges of utilizing large amounts of data and interprets them in technical requirements,

(vi) provides an in-depth analysis of existing technologies for big data processing,

(vii) incorporates the technical requirements into a full-fledged architecture of a smart city in refer to existing technologies for big data analysis and

(viii) summarizes algorithmic suites that can be implemented with the suggested technologies for big data analysis.

(ix) The aforementioned are pursued through a real smart city scenario of Heraklion city, according to which we design big data processing pipelines for a number of core services such as parking, water and air quality monitoring, reporting future benefits for the stakeholders.

1.5 Thesis outline

In **Chapter 2**, we present the ICT network of a distributed setting with a reference to different sensing technologies that are commonly used in various smart environments.

In **Chapter 3**, we introduce the aspects of the main smart city domains on which we are going to focus (i.e., transportation, resources and environmental aspects) and identify potential issues and difficulties towards realizing the concept of a sustainable city infrastructure. In addition, we report indicative examples of existing technologies adopted by smart cities setups based on the literature underlining the physical structure of the ICT network with a reference to existing sensing technologies.

In **Chapter 4**, we analyze and translate the reported issues into business objectives and key performance indicators that need to be served by big data analytics tasks.

Chapter 5 introduces the business logic scenarios for big data processing pipelines by presenting indicative configuration examples in terms of data recording and reporting intervals followed by the data flow, application logic and the mechanism that will communicate the data to the stakeholders. Moreover, this chapter underlines the technological challenges of utilizing large amounts of data and translate them into specific technical requirements.

In **Chapter 6**, we incorporate all the above in a full-fledged architectural design proposal tailored for smart cities and analyze indicative solutions on big data platforms according to the specific requirements.

Chapter 7 reviews existing algorithms and algorithmic suites that are generic enough to accommodate broad categories of analytics tasks useful within a smart city context. Through this chapter, we provide specific algorithmic solutions that can be implemented within the aforementioned big data platforms in various parts of the proposed architecture.

Finally, **Chapter 8** focuses on exemplifying the presented methodology, architecture and algorithmic apparatus with a real case study, the Heraklion Smart City, by elaborating indicative big data pipelines of big data platforms and reporting future benefits.

2 A Smart City's ICT Network

The evolutionary ICT landscape that is progressively embedded in everyday life environments encompasses large amount of smart computing technologies. Harrison et al. [90] refers to smart computing as the integrated hardware, software and network technologies that provide Information Technology systems with real-time information and advanced analytics aiming to assist humanity to make more efficient and intelligent decisions in a proactive fashion.

ICTs are able to record and process raw data, extracted among the various city services (e.g., transportation, health, education, police, water, etc.), and eventually transmit them via wireless or wired network infrastructure to city authorities' data centers, public or hybrid clouds. There, powerful computing resources or High Performance Computing (HPC) infrastructures are able to handle such large amounts of information. The main challenge is to ensure that big smart city data processing will be horizontally (i.e., adding more nodes to a system) and vertically (i.e., adding more resources to a single node) scalable. This should be achieved with respect to both the accumulated data volumes and the number of employed devices collecting relevant streams of data, resilient to changes. Data processing should not only serve city authorities' mission but also remain open for third party innovation in the city, by offering functionality and open data middleware technologies (i.e., Application Programming Interfaces) [24, 103].

The analysis in the following chapter is based on the architectural approach adopted by the authors of [103]:

(a) The **physical sensor infrastructure**,
(b) The **network infrastructure** for communicating and fusing sensor data,
(c) An **IoT middleware** for digesting potentially heterogeneous information streaming in from the network,
(d) **Data processing and analytics platforms** within data centers and clouds, and
(e) **Visualization tools** that present the outcomes of analytics tasks so that the value extracted out of the analytics procedures is presented in a comprehensive fashion.

2.1 Main sources of data generation in a Smart City Infrastructure

A smart city infrastructure generates an enormous amount of data that need to be handled, stored and made available. The main sources of data generation are smart devices, web services and sensing devices.

2.1.1 Smart Devices

Smart mobile devices (smartphones, tablets) constitute nowadays, a very common source of data gathering anywhere and anytime at a very low cost. Smartphones are used for many purposes to cover different needs. Modern smartphones are equipped with sensing technologies, such as image sensors, sound and touch sensors, accelerometers, GPS, gyroscopes and magnetometers. They are able to generate various information regarding an individual's location, orientation, health, daily movement, etc. Over 5 billion people around the world use smartphones every day to make calls, communicate through social media and search for information, while it is estimated that the accumulated digital world of data will increase from 33 zettabytes in 2018 to around 175 zettabytes by 2025 [160].

2.1.2 Web Services

Web services, such as social media, generate huge amounts of data that may be of interest in a smart city environment. However, the data veracity and quality should be carefully taken into account before being used. Web services generate unstructured data, which contain a significant amount of uncertain and imprecise data. Techniques such as semantic web and linked data [57] have been exploited in order a smart city's stakeholders to be able to obtain information from websites, as well [150].

2.1.3 Sensor devices

Nowadays, sensors of any size together with actuators and microelectronic processors are installed to everyday objects towards adapting ICTs in smart environments. In general, smart sensor devices or sensor nodes are capable of measuring data input from the physical environment and intercommunicate through wireless connection with each other comprising a Wireless Sensor Network (WSN) infrastructure [30]. Sensing devices generate data either through periodic observations or through event-based observations. IoT devices programmed with the former contain information on a configurable frequency basis, whereas event-based observations [71, 62] are reported only upon the detection of a change in the parameter they measure [115].

The main components of a sensor node are (i) sensory devices, (ii) processors, (iii) transceivers, and (iv) the energy source. Sensor nodes can serve as the sources of smart city data per se, or as relay nodes forwarding required information to intermediate backbone routers. Deploying low-cost sensors and various types of smart objects to collect data from public infrastructure, allows smart cities to increase operational efficiency by obtaining a picture of the actual physical realm in a timely fashion.

In Table 1, we present different types of sensing devices that can be used in various application areas in a city in refer to the type of energy they detect as signals [32]. Additionally, each sensor type is classified to one or more smart city focus areas that are going to be explicitly discussed in the following chapter.

Table 1: Different sensing technologies

Sensor types	Examples	Application areas
Location sensors	GPS, active badges	Road traffic monitoring, Waste management
Optical/vision sensors	Photo-diode, color sensor, IR and UV sensor	Road traffic monitoring, Air pollution intensity
Image sensor	Stereo-type camera, infrared	Road traffic monitoring, Parking management
Sound sensors	Microphones/ultrasonic sensors	Road traffic monitoring, Parking management, Water management, Waste management
Temperature sensors	Thermometers	Weather management
Humidity sensors	Electrolyte sensor, polymer-based sensors, hygrometers	Weather management
Chemical sensors	Gas sensors, dissolved oxygen (DO) sensor, conductivity sensor, turbidity sensor and pH/ oxidation reduction potential (ORP) sensor	Waste management, Air pollution intensity, Water monitoring
Meteorological sensors	Barometer, pressure gauges	Weather monitoring
Motion sensors	Speedometer, mercury switches, tachometer	Road traffic monitoring
Physical movement sensors	Accelerometers	Road traffic monitoring
Identification and traceability sensors	RFID, NFC	Road traffic monitoring Waste management
Magnetic field sensors	Magnetic sensor	Road traffic monitoring Parking management

The selection of the appropriate sensors in a given context conceals important challenges that need to be taken into account including, but not limited to, the accuracy, environmental conditions, range, calibration, resolution, cost, robustness and repeatability.

3 Smart City Focus Areas

In this chapter, we introduce three distinct areas of a smart city, related to transportation, resources and environmental aspects, subdivided into six vertical sectors, referred to as sub-dimensions in Table 2. Specifically, we underline the importance of the deployment of ICTs in terms of the environmental impact and human well-being, while we project potential benefits towards a more efficient resource management. The presented focus areas will constitute the focal point of this work in terms of leveraging big data with an emphasis on extracting knowledge that may prove valuable to governments and city authorities.

Table 2: Sub-dimensions of KPIs

Dimension label	Dimension	Sub-dimension label	Sub-dimension
D1	Smart Transportation	D1.1	Road traffic monitoring
		D1.2	Parking management
D2	Smart Resources	D2.1	Water management
		D2.2	Waste management
D3	Smart Environment	D3.1	Air pollution
		D3.2	Weather monitoring

3.1.1 Smart Transportation Management

Mobility is one of the most important facilities to support the functioning of any urban area [154]. However, it is a fact that transport produces negative impacts affecting the life quality in cities, causing air pollution, street congestion, time and fuel consumption, while it entails high transfer costs. In this context, smart mobility is considered an important aspect to focus on enabling city authorities to locate traffic bottlenecks, through efficient road traffic monitoring and parking management, and adapt proactive measures for congestion mitigation.

The key stakeholders regarding the initiatives of smart mobility can be classified into public transport authorities, private companies, citizens, environmental agencies, as well as public bodies and local governments.

3.1.1.1 Road traffic monitoring

Traffic congestion can be caused by physical bottlenecks, traffic incidents, work zones, traffic control devices, special events or weather conditions. Congestion has a negative impact on both the economy and the environment. Therefore, municipalities tend to adapt ICTs to monitor and mitigate congestion. Smart technological infrastructures provide necessary and real-time information that upon being analyzed can decrease the exposure of vehicles to highly congested areas and recommend efficient driving patterns.

In order to achieve the above, wireless network sensors distributed along a city center are considered an effective solution for large areas coverage in comparison to wired sensors networks that require high deployment and maintenance costs [149]. A wireless sensor network allows multiple sensory devices to be networked together and share geographically distributed information through wireless communication protocols and are able to detect the presence, speed, density and occupancy ratios of vehicles. Moreover, fetching GPS data from drivers' smart devices or even historical data, smart solutions for traffic management can predict through machine learning techniques, which areas are more prone to traffic congestion and thus take measures to prevent it.

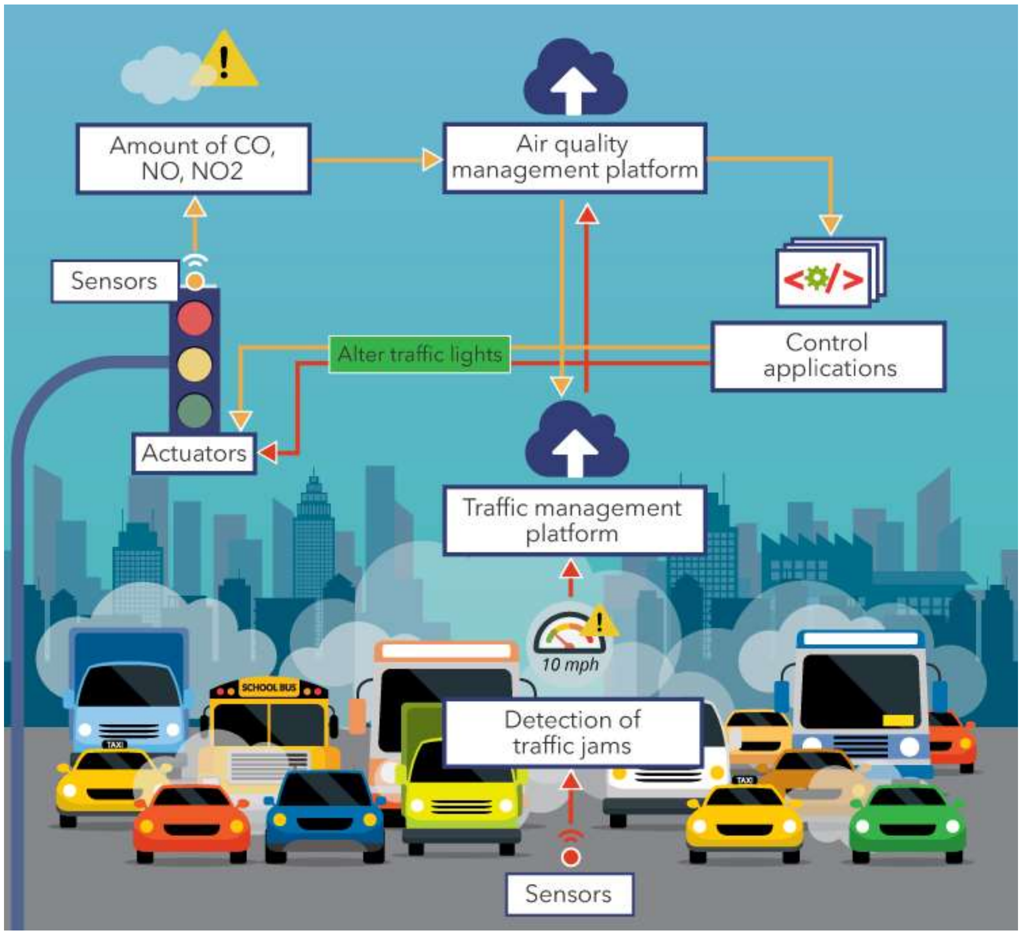


Figure 2: A smart on-road traffic management infrastructure (Source: ScienceSoft [145])

Roads equipped with tags and sensors forward important information to traffic control sites and transportation vehicles to better route the traffic and provide citizens and authorities with real-time transportation information [24]. Magnetic sensors are the most widely used sensors in the transportation field especially in cases of moving vehicles detection. They are considered sensitive, robust on demanding weather conditions (e.g., fog, rainfall, snow, storm, etc.) and easy to maintain [184, 43, 129]. These sensors are able to detect magnetic anomalies in Earth's magnetic field originating from the presence of cars and are based on magnetoresistors, which can obtain information such as the number of vehicles and speed statistics [103, 149]. Alternative sensors for vehicle detection are optical sensors [105], RFID and NFC [24], acoustic sensors and video cameras, although they have been proven less accurate and robust than magnetic sensors [184].

The authors of [43] present the physical architecture of a prototype wireless sensor network in a traffic surveillance system consisted of a number of sensor nodes and a single access point. The latter has superior computation resources, enhanced radio communication and unlimited power supply. A processor processes the raw data in the sensor node to extract useful information. In the specific setup, the output of the sensor node is transmitted to the access point through either a direct communication or a multi-hop communication across other sensor nodes. Eventually, the access point processes the data collected from all the nodes in the network to extract more information and visualize them into a meaningful format in order this information to be shared to the stakeholders (i.e., end-users, GPS transceivers, radiophones, authorities, etc.).

In addition, smart cities are able to confront high volumes of traffic congestions through interconnected **smart traffic lights** consisted of sensors able to collect information about traffic flow patterns, such as vehicle speed, traffic density, waiting time at the lights and traffic jams [6]. For instance, whenever a traffic management platform receives a congestion alert from the sensor nodes, it sends a command to the traffic lights' actuators to alter the signals and re-route part of the traffic.

Dublin city, in terms of the initiative Smart Dublin [152], has embedded in the city's traffic lights a system that dynamically responds to congestion using sensors at traffic intersections to detect the number of vehicles in each lane and the number of pedestrians waiting at the signals. Additionally, Los Angeles, as being one of the most traffic-affected cities in the world, is deploying a network of smart controllers to make automatic traffic lights adjustments, reacting to changing traffic conditions in real time. The system uses magnetic sensors in the road to measure the flow of traffic, cameras and a centralized computer system that makes constant adjustments to keep cars moving as smoothly as possible [145, 163].

Moreover, a number of sensors allocated in the city center of Santander and embedded under the asphalt, collect data on rainfall and road traffic, which is sent to a control center to help the city provide services more efficiently [161]. Specifically, in the SmartSantander testbed [42] a sensor node wakes up within predefined time intervals, takes readings from the sensors, transmits the data, and returns to sleep mode in order to save more power. Every minute, traffic intensity information is provided with the node ID and its localization (latitude and longitude), while the current occupancy and the number of vehicles on the lane are recorded [170].

In such cases, big data analytics are essential in order to build an intelligent decision system installed in a data center. They process the extracted values in order to make decisions and send the appropriate instructions to the lights and signals. Additional smart traffic light systems, adapting different smart city technologies (e.g. magnetic sensors, car's on-board software and GPS, cameras, etc.), have already been tested and installed in Pittsburgh, USA with over 20% of reduced emissions due to reduced traffic jams and waiting times [6], in Santander, Spain [42], in Oulu, Finland [108] and so on.

3.1.1.2 Parking management

Finding a parking place given the rapidly increasing number of vehicles in large urban centers makes smart parking systems an inevitable need towards reducing the traffic congestion, time and fuel consumption. This raises also serious problems of pollution and noise that need to be mitigated. According to the survey of [116], smart parking systems include parking availability monitoring, parking reservations and dynamic pricing schemes to benefit the drivers, the parking operators and the municipal police. Moreover, the authors of [85] present parking predictive models including a comparison of static and dynamic models that were used to address parking lot management providing decision support to drivers that need to reach their destination in the shortest time possible. An indicative pilot IoT-based setup introduced by Deutsche Telekom is illustrated in Figure 3 [58].

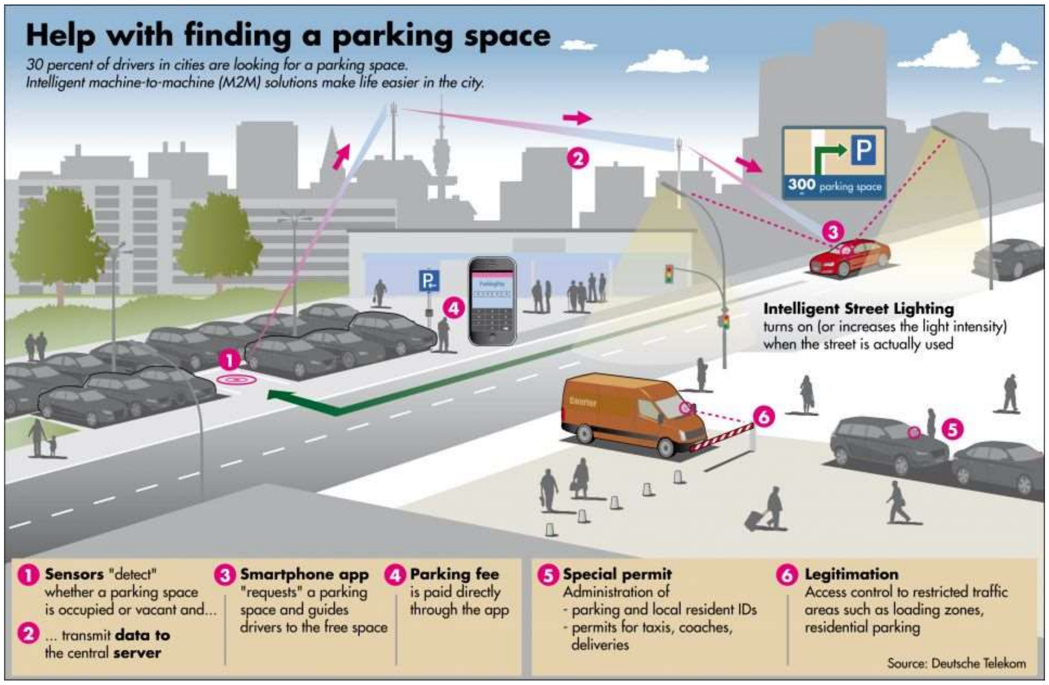

Help with finding a parking space
30 percent of drivers in cities are looking for a parking space.
Intelligent machine-to-machine (M2M) solutions make life easier in the city.
300 parking space
Intelligent Street Lighting
turns on (or increases the light intensity) when the street is actually used
1 Sensors "detect" whether a parking space is occupied or vacant and...
2 ... transmit data to the central server
3 Smartphone app "requests" a parking space and guides drivers to the free space
4 Parking fee is paid directly through the app
5 Special permit
Administration of
- parking and local resident IDs
- permits for taxis, coaches, deliveries
6 Legitimation
Access control to restricted traffic areas such as loading zones, residential parking
Source: Deutsche Telekom

Figure 3: Smart city pilot project for parking optimization

According to the setup in Figure 3, the following main components are considered necessary in order to collect and process data in an integrated parking management infrastructure:

- **Parking Sensor Nodes**: Indicative types of sensors proposed in the past for parking monitoring include infrared sensors, ultrasonic sensors and image sensors [184, 107, 40], however, each one of them introduces restrictions that can lead to erroneous judgments. On the other hand, magnetic sensors have been used and installed in situ in cities such as Heraklion [103] and Barcelona [26] and have been reported as more precise and robust for the collection of parking availability information [149, 138]. Magnetic sensors are installed along the public roads (on-street parking) and in designated parking areas, while they provide information in predefined intervals regarding the occupation status of the specific parking spots.

- **Processing Unit**: It comprises a processor (e.g. Raspberry pi, Orange pi, Arduino, etc.) that collects the data extracted from the nodes, processes and transmits the information to the data servers.

- **Cloud platform**: A central data center based on cloud computing platforms that allow online storage of all the records related to parking areas.

- **Mobile application**: A mobile application with an embedded navigation system can be proven a cost and time saver for drivers that need to find a parking place with the least effort. All the necessary information can be visualized in an intuitive way for the end-users, illustrating the parking spots that are occupied or available on a real-time parking map. The information provided can be personalized according to the audience that is addressed (e.g. authorities, citizens, municipal police, etc.). Moreover, a dedicated web application deployed by the authors of [103] aims to facilitate the municipal police to track cars that park in prohibited areas, such as parking lots for individuals with disabilities, pedestrian crossings and other.

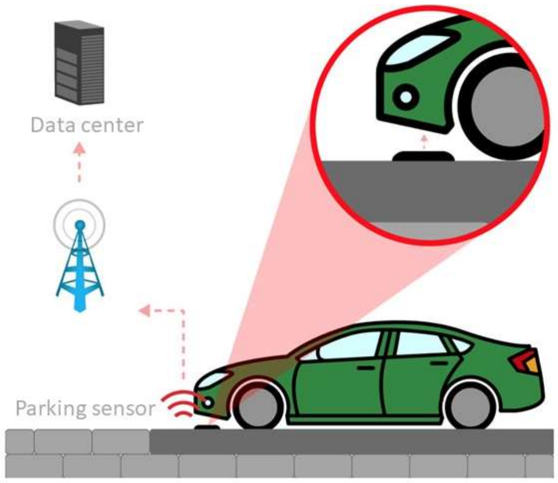


Figure 4: A smart parking infrastructure using magnetic sensors (source: Libelium)

3.1.2 Smart Resource Management

Pervasive sensing technologies enable techniques to optimize resource allocation in a city context through data that provide necessary information to predict and avoid disastrous events (e.g. floods, water contamination, etc.) or even confront with every day challenges including waste collection and irrigation.

3.1.2.1 Water management

The issues arising regarding water management are twofold. The integration of technologies, such as smart pipes and sensor networks, smart metering and Global Positioning System (GPS) coordinates, are essential for smart cities in order to be able to mitigate: (a) problems in the urban water sector and prevent water contamination and (b) excessive water consumption and water loss from leaks. Daily, monthly and yearly reports are necessary to draw important conclusions towards that direction.

Since water covers a 71% of the Earth’s surface, it is a vital resource for the humanity and the environment itself. However, climate change along with rapid urbanization and inappropriate urban planning have led to water-related issues that call for immediate action adopting strategies for a smart water resource management. Such issues may include flooding disasters, water pollution and water shortages [130]. Cutting-edge ICTs are able to monitor water in polluted and contaminated areas (i.e., the quality and quantity of water in rivers and lakes), caused by agricultural fertilizers, industrial wastes, etc., in order important decisions to be made promptly to protect citizens from consuming contaminated drinking water in a timely manner [127, 41].

Specifically, studies suggest that approximately 783 million people lack access to clean water, 2,5 billion lack access to adequate sanitation, and 6 to 8 million are dying per year due to water-related diseases and disasters [98]. More specific, smart water management systems aim to provide access to safe water, to manage demand and supply, and to develop a pricing mechanism.

Moreover, in case limited quantities in water reservoirs are foreseen, municipalities could monitor relentless consumption, while at the same time the quality of drinking water needs to be continuously controlled to protect people from contamination. To tackle these issues, governments and municipalities have activated water management systems that aim at planning, developing, distributing and managing the optimum use of water resources [144]. Yet nowadays, many cities lack of water ICT standards in a way that water consumption, distribution, system identification and equipment maintenance cannot be properly monitored and controlled. However, the provision of IoT technologies in water management can lead to energy costs reduce and minimization of human intervention. With the use of sensors (e.g. water leak sensors) and actuators, real-time data can be monitored and proactive improvements can take place in water management infrastructures. An indicative example of a smart water infrastructure, introduced by [158], is presented in Figure 5.

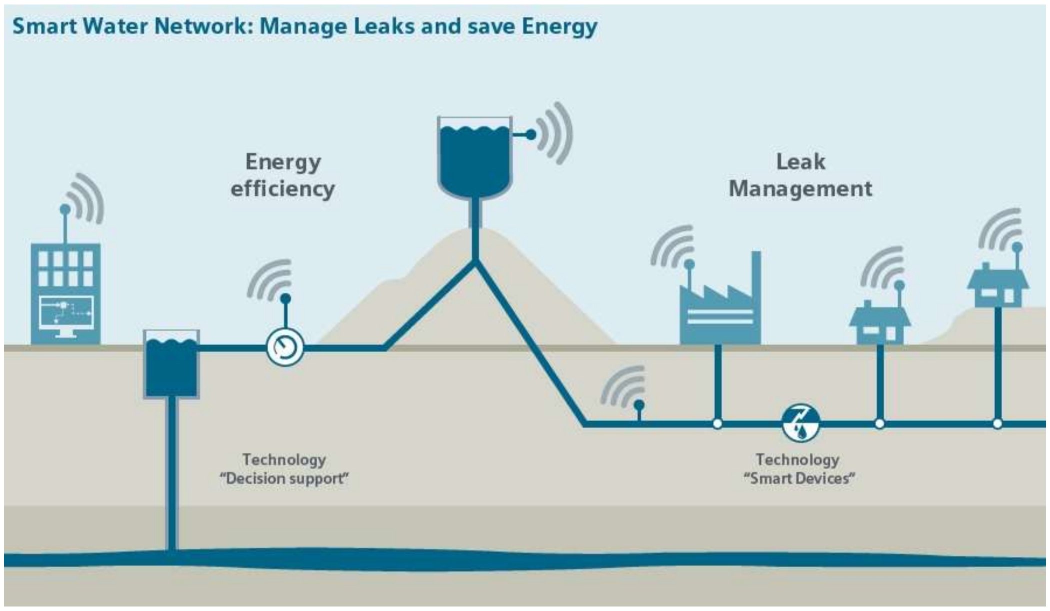

Smart Water Network: Manage Leaks and save Energy
Energy efficiency
Leak Management
Technology "Decision support"
Technology "Smart Devices"

Figure 5: An example of Smart Water Network infrastructure

The report of [148] has presented an IoT-based solution comprising three main components that are considered necessary to collect and process data in a typical water management infrastructure:

i. **Sensors:** Installations of specialized sensors across the water grid accompanied with microcontrollers (e.g., Arduino, Raspberry Pi) that are able to collect information (i.e., water level, water temperature, conductivity, etc.) and through a communication device to forward it to a central server.

ii. **Centralized server**: The sensors transmit data to a local server and a central server fetches processes and extracts useful information. Data are then translated into measurable results through data analytics platforms. For instance, if a leakage occurs and the chemical composition of water changes, the cloud platform triggers an output defined by the users.

iii. **Data visualization tools:** With the use of a cloud platform, the extracted results can be easily illustrated in a comprehensive way, through a web interface, allowing authorities to manage the water supply infrastructure more efficiently and solve upcoming issues proactively.

Dedicated sensors installed in water reservoirs can be used to record the quality and quantity of water (i.e., water level, water temperature, conductivity, PH and ions {NH_4^+, NO_3^-, Cl^-, and Ca^{2+}}), thus providing meaningful results to stakeholders, such as municipal authorities and citizens [103]. For instance, a number of sensors deployed at different levels (e.g. 20%, 40%, 60% and 80%) in water tanks can provide through a communication protocol the appropriate information to data servers, through which notifications will be sent to stakeholders whenever the water level goes below the threshold.

An indicative example introduced by [41], is a water monitoring system installed in Bristol city that controls the water quality. The system comprises a multi-sensor (i.e., a dissolved oxygen (DO) sensor, a conductivity sensor, a turbidity sensor and a pH/ oxidation reduction potential (ORP) sensor) and a camera based on a wireless sensor network solution.

Another example is deployed by SmartSantander [172]. Through its smart water initiative the city mitigates water consumption costs by empowering residents to manage their water use through a smartphone app. They have access to real-time data on water quality and consumption, track trends over time, and receive service alerts.

Moreover, smart water meters installed in buildings' water supply infrastructure could indicate excessive consumption, leakage identification and so on. Addressing this issue, actuators are able to monitor the water consumption per household and control the water flow in cases of excessive usage during periods that the water reserves of the city are below limits [148].

3.1.2.2 Waste management

Nowadays, waste collection is an arising matter that needs to be triggered in respect to dynamic route optimization for routes followed by waste collecting trucks, cost efficiency and environmental pollution. Oftentimes, waste containers are emptied from cleaners at certain intervals, according to their daily routines regardless whether they are filled up or semi-empty (Figure 6) [65]. This is not a very efficient approach since it leads to the unproductive use of waste containers and unnecessary fuel consumption by waste collecting trucks. This aspect, might also affect the urban living having both an environmental and a socioeconomic impact.

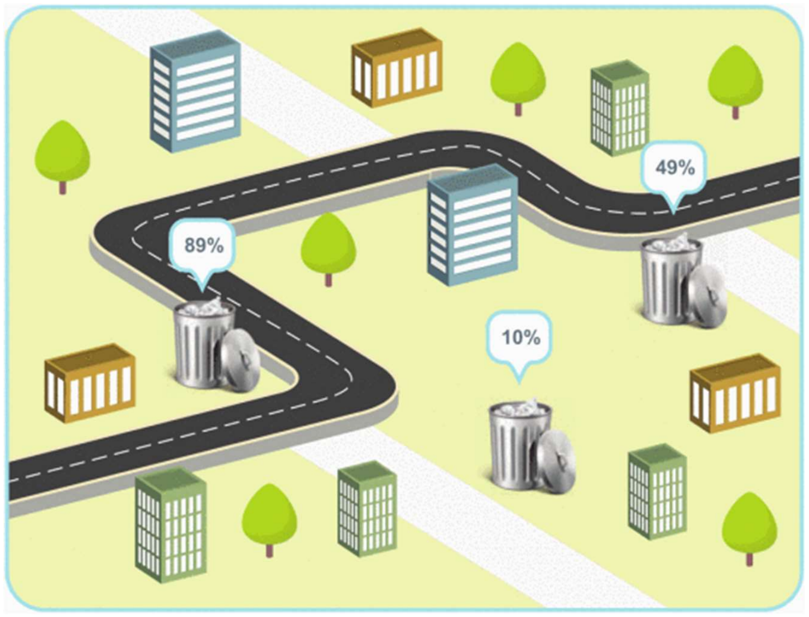


Figure 6: Common situation of garbage cans' filling status in todays' cities

Advanced technologies enable municipalities to use intelligent ways focusing on managing the waste cost efficiently, being more environmentally responsible, including waste recycling and re-use processing, and thus improve the human well-being [7]. Sensoneo, an enterprise specialized in smart waste management solutions [146] underlines that this can be achieved through embedded monitoring systems providing cities and businesses with data-driven decision-making, and optimization of waste collection schedules and routes, frequencies, vehicle load and trash container load. Pick-up services depending on real-time requirements could optimize the routing operations of the cleaning service. IoT components incorporated into trucks with surveillance systems that monitor the recycled waste and into waste containers to gather data about the level of the waste, can offer a more efficient process of waste collectors. Such components include, but are not limited to, RFIDs, sensors, cameras, and actuators [120].

According to [7], a waste management network may consist of:

(a) the **physical infrastructure** that involves among others the waste bins, waste location, the fleet of trucks, recycling and processing of recyclable waste,

(b) the **IoT infrastructure** that includes technologies such as RFID tags, Near Field Communications (NFC), sensors (e.g. capacity, weight, temperature, humidity and chemical sensors), Wireless Sensor Networking (WSN) for broader communication, actuators, cameras, GPS and a cloud server able to collect the data and communicate them to end-users, and

(c) **software analytics**, which involve: (i) dynamic scheduling and routing processing, (ii) GPS coordinates that focus on the processing of spatiotemporal information and (iii) decision support systems.

Smart waste monitoring systems presented by [64, 119], integrate sensor nodes in every trash container in order to measure the fullness and capacity of each bin. The sensors report the collected data to a central gateway node installed in every sensor cluster, which forwards the information to a back-end server through wireless internet connection. Sensor clusters are consisted of sensor nodes installed in a specific range and density within predefined geographic areas. The data collected and stored to data centers can be analyzed and visualized with the assistance of an analytics module and afterwards can be send through a web platform or a mobile application to the city's cleaning service.

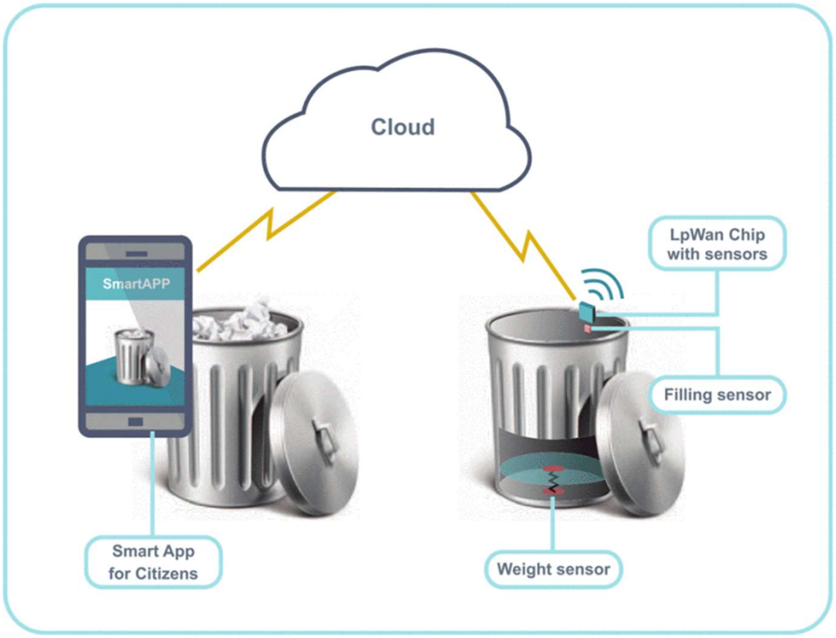


Figure 7: Scenario for municipal waste management proposed by [65]

A real use case adopted by Dublin Smart City [152] concerns smart bins that are able to provide real time information to the waste division collectors and send e-mails whenever the waste reaches 85% capacity. A smart compaction module, installed in each bin, accumulates the trash towards improving the efficiency of the public waste collection service.

In addition, Nordsens solution [131] deploys the appropriate infrastructure to city's waste containers aiming to minimize collection costs and manage the collection time. More specific, it monitors the fill levels of waste containers, equipped with optical sensors able to generate 3D depth maps of a bin's content. The extracted data are gathered and analyzed in an effort to optimize routing and navigation for collections, while it predicts waste generation patterns.

Moreover, the city of Stockholm [156] has installed more than 150 smart bins based on solar-powered software, mobile devices and sensors that report in real time when they are about to become full and need to be emptied. These leads to more efficient waste collection services with fewer garbage collection runs, lower costs and reduced emissions.

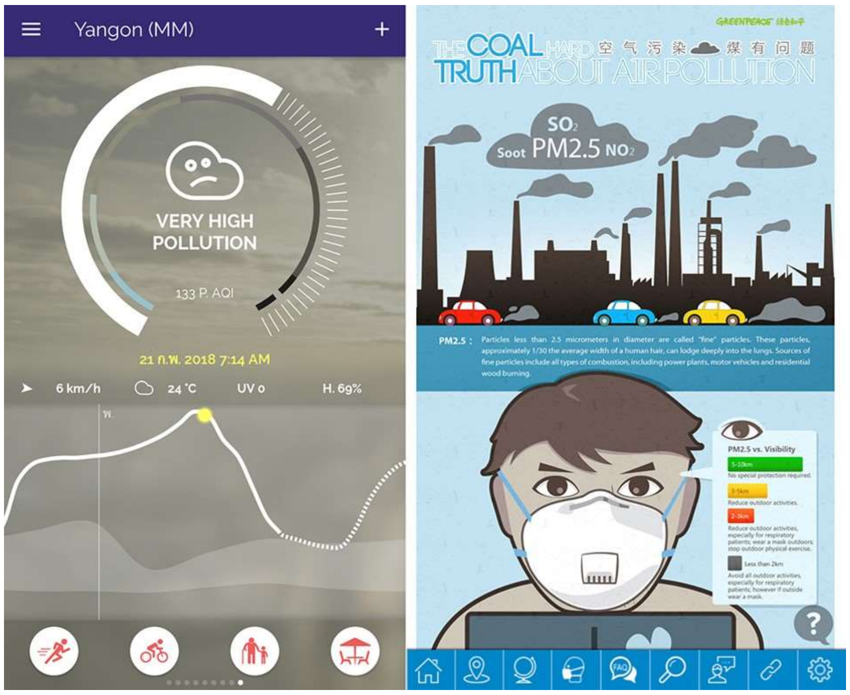
3.1.3 Smart Environmental Management

In this section, we refer to specific environmental sustainability issues that arise due to various human factors. Such issues can be proactively forecasted through proper observations of weather and air quality data in order to prevent health or environmental problems from expanding.

3.1.3.1 Air Quality Monitoring

Air pollution is being a major environmental challenge that causes many hazardous effects on both humans and the environment. Health issues as well as global warming effects increase due to the intensive air pollution stemming mostly from transportation, road traffic, home heating and industrial emissions in urban centers. The World Health Organization reports that ambient air pollution accounts for million deaths per year due to stroke, heart disease, lung cancer and chronic respiratory diseases [179].

ICTs have been adapted in order to monitor, forecast and report the status of ambient air pollution in smart cities using cutting-edge technologies able to measure a large amount of data and extract valuable results. The aim is to provide real-time air quality information to inform people and guide their daily decision-making. Such data should be accessible to stakeholders, including vulnerable groups of people (e.g. elderly, individuals with chronic diseases, genetic susceptible, etc.), environmental scientists, prefectural civil protection agencies and policy makers, while advanced warnings of potential health-damaging events in the form of national air quality indices could be also made available [106, 136]. An example of real-time mobile applications, is presented in Figure 8 [117]. Real-time information can help advice the public to take proper actions according to their individual health needs and also raise public awareness adapting more environment-friendly solutions (e.g. using vehicles with less CO_2 emissions, home heating, etc.) [183]. Based on low-cost and accurate sensors, smart cities are able to deploy a ubiquitous monitoring system using WSN nodes distributed at large geographic areas. This system could constitute an informative tool for the public able to notify stakeholders, through proactive alert services, forecasting and warnings, aiming to protect especially people of high risk.



PM2.5 : Particles less than 2.5 micrometers in diameter are called "fine" particles. These particles, approximately 1/30 the average width of a human hair, can lodge deeply into the lungs. Sources of fine particles include all types of combustion, including power plants, motor vehicles and residential wood burning.

PM2.5 vs. Visibility

5-10km
No special protection required.

3-5km
Reduce outdoor activities.

2-3km
Reduce outdoor activities, especially for respiratory patients; wear a mask outdoors; stop outdoor physical exercise.

Less than 2km
Avoid all outdoor activities, especially for respiratory patients; however if outside wear a mask.

Figure 8: Mobile application examples illustrating air pollution values and protection recommendations

In the past, various sensing technologies have been proposed aiming to monitor air pollutants, such as gases (e.g. SO_2, NO, NO_2, CO_2, O_3), smoke concentration, and dust particles accumulated in urban centers. MESSAGE project [122] was deployed to monitor air pollutants in the city of Cambridge using a combination of dedicated sensors attached on street- and traffic lights. Additionally, OPENSENSE [134] has developed a network of fixed and mobile sensors mounted on trams and public buses in Zurich and Lausanne by using, among others, electrochemical gas sensors (NO_2, CO, SO_2, H_2S) and optical particulate matter detectors. In this context, data can be aggregated through monitoring stations that cover different regions and indirect data sources, such as historical time-series data, social network tweets, real-time traffic data, and city layout [8].

The IoT infrastructure for air quality monitoring deployed by [103] comprises an integrated sensor node network (including the appropriate sensors, microcontrollers and transceivers), a data center (could be cloud-based) that stores and processes the extracted data and an API for mobile and web interfaces that illustrates the results. An alerting and prediction system is hosted in the main server to process streaming data from the database and provide timely notifications based on machine learning libraries.

In addition, , air quality sensors were installed near schools, in urban centers, near major roads, on public lighting infrastructure and so on, enabling authorities to monitor the generated data and apply measures to reduce industrial pollution, as well as vehicle and commercial building carbon emissions. In this context, cities could emphasize on expanding public transport, imposing restricted car access in city centers, forcing stricter vehicle emission standards, promoting renewable energy resources, etc.

3.1.3.2 Weather Monitoring

Weather monitoring and forecasting also plays an important role on the everyday life due to its numerous applications in agriculture, farming, fishery, shipping, military operations, etc. [54]. The integration of ICTs in weather monitoring systems, able to process both real-time and historic data, can lead meteorologists to important conclusions regarding environmental threats that may affect the planet in the long run. At the same time, giving access to information such as ambient temperature, ambient humidity, atmospheric pressure, wind speed, wind direction and luminosity, smart cities may protect, among others, susceptible groups of people to expose themselves to extreme weather conditions. An indicative example of an application developed by [123] that aims to inform citizens about the weather status is illustrated in Figure 9. In addition, through temperature and humidity observations a city, applying the appropriate technological mechanisms, is able to predict the energy load from its own buildings, inform and encourage citizens to use renewable energy sources [151].

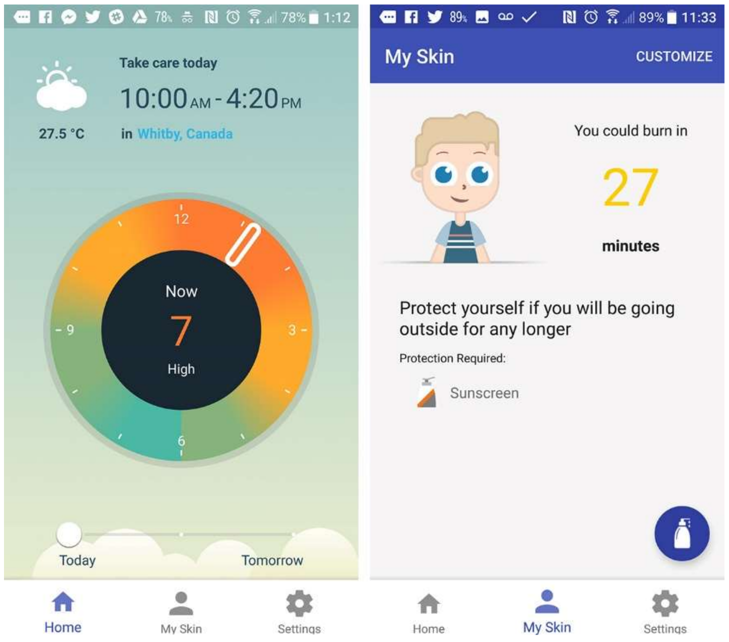


Figure 9: Indicative application example for sun protection

The report of [147] presents a Sensor Node consisted of (i) meteorological sensors (temperature, humidity, wind speed and direction, rain, and atmospheric pressure sensors), (ii) a low power microcontroller that processes the sensor readings and (iii) a wireless transceiver that transmits the data to a database. An additional API platform allows the data to be transferred from the database to a web interface or a mobile application. In that way, stakeholders, including farmers, gardeners, meteorologists and people suffering from health diseases, could be properly informed about extreme weather conditions and potential disastrous events (e.g. floods, storms, etc.).

4 Business Objectives and Key Performance Indicators

In the previous chapter, we introduced the focus areas that we are going to exploit in this work presenting the benefits of the provision of ICTs. As a next step, we need to analyze and translate usual problems related to the aforementioned domains into concrete objectives.

Specifically, this chapter aims at presenting the business logic that will facilitate IoT-based cities to mitigate resource consumption by setting specific goals. To that end, we define the appropriate business objectives together with a set of Key Performance Indicators (KPIs) that will enable stakeholders to assess how the use of ICTs has an impact on the environmental sustainability of smart cities focusing on (a) Smart Transportation Management, (b) Smart Resources Management and (c) Smart Environmental Management.

Identifying Key Performance Indicators (KPIs) of technologies established in smart city infrastructures aims at making cities smarter and more sustainable, while they provide cities all the means for self-assessment. The KPIs presented in this chapter enable stakeholders develop strategies and understand the progress related to the use of ICTs in smart sustainable cities, while managing the city assets efficiently. As [99] indicates, the selection of KPIs for ICTs in smart cities is based on the following principles.

- ➢ **Comprehensiveness**: The set of indicators should cover all the aspects of smart sustainable cities. The indicators for evaluation should be aligned to the measured subject, i.e., the impact of technologies on the sustainability of cities.
- ➢ **Comparability**: The KPIs should be defined in a way that the data can be compared scientifically between different cities according to different phases of urban development. It should also be possible to extend and amend the set of KPIs according to the actual stage of development.
- ➢ **Availability**: The KPIs should be quantitative, while historic and current data should be either available or easy to collect.
- ➢ **Independency**: The KPIs in the same dimension should be independent or almost orthogonal in order overlaps of KPIs to be avoided.
- ➢ **Simplicity**: The indicators should be clearly formulated and easy to understand. The calculation of the associated data should be intuitive and simple.
- ➢ **Timeliness**: is the ability to produce KPIs with respect to emerging issues in smart cities.

According to [99], Key Performance Indicators can be utilized by:

- o Cities and municipal administrations, including policy-making organizations, and government sectors, enabling them to develop strategies and understand the progress related to the use of ICTs.
- o City inhabitants and non-profit organizations, enabling them to understand the development and progress of smart cities with respect to the impact of ICT.
- o Development and operation organizations of smart cities, including planning unit and service providers, operation and maintenance organizations, helping them fulfill the

tasks of sharing information related to the use of ICT and its impact on the sustainability of cities.

- Evaluation agencies and academia, supporting them in selection of relevant KPIs for assessing the contribution from ICT in the development of smart cities.

In an effort to identify the KPIs of technologies established in the presented focus areas in an IoT-based city infrastructure, Table 3 summarizes some indicative examples of hypothetical current and target performance that the provision of ICT technologies and the utilization of big data processing platforms aim to accomplish. A further analysis of the objectives in Table 3 will follow in the next chapter.

Table 3: Measurable Objectives – Key Performance Indicators (KPIs)

Objective	**Current Performance**	**Target Performance**	**Influence on Business KPIs**
Reduced on-road congestion (as less as possible)	minutes	seconds	- Reduced number of vehicles on the road - Reduced waiting time at traffic lights - Minimize time intervals while searching for parking
Parking monitoring (as precise as possible)	thousands of illegal parking per year	tens of illegal parking per year	- Reduced illegal parking
Water monitoring (as precise as possible)	tens of tons per year	tons per year	- Reduced water leakage - Water supply network monitoring
	millions of infections per year	hundreds of infections per year	- Reduced disease transmissions through water contamination
Reduced environmental pollution (as less as possible)	Manual monitoring	Automated monitoring	- Efficient waste collection services - Automated monitoring system for recycling items - Decreased exposure of vulnerable groups to air pollution
Weather forecasts (as precise as possible)	Automated monitoring	Automated monitoring	- Reduced disasters caused by extreme weather conditions - Reduced health incidents due to high temperature and humidity

4.1 Business Objectives for Smart Transportation

After a thorough literature review presented in Chapter 3 and taking into consideration the serious problems arising from overpopulated cities, the authors underline the most significant business objectives aiming to mitigate excessive resource consumption and create more sustainable urban dynamics with the provision of ICTs.

BO1. Reduced congestion rates through early warnings

Road surveillance cameras, personal mobile devices and integrated on-road sensor technologies allow authorities to monitor the driving behavior and suggest efficient driving patterns. In particular, drivers can avoid intense congestion through message signs, real-time notices through the media (e.g., radio) or updated suggestions for alternative routes provided through Global Positioning Systems (GPS) embedded in vehicles and mobile devices. The road status can be provided by road surveillance cameras, intelligent traffic lights, personal mobile devices and integrated on-road sensor technologies able to monitor the driving behavior.

BO2. Reduced waiting time at traffic lights

Smart traffic lights established in large urban centers conquer environmental pollution stemming from traffic congestion due to increased redundant waiting times [66]. Parameters such as the vehicle speed, traffic density and waiting time, can be monitored and processed in order dynamic commands to be transmitted to traffic lights and operate more efficiently and effectively. At the same time, dynamic response to congestion can be provided to drivers through early notifications while alternative routes can be suggested according to vehicles' geospatial position.

BO3. Minimized time intervals while searching for parking

Vehicles need long time intervals to spot a parking space. An automatic notification system for the nearest available parking lots to their final destination, including people with special needs, could allow drivers to minimize not only these time intervals but also fuel consumption. The geographic location of sensor nodes installed in different spots could extract the appropriate data to inform drivers where free parking lots can be found. Presenting all the available parking lots of an area facilitates both drivers and parking companies.

BO4. Reduced illegal parking

Illegal parking is a common habit and can lead to reduced traffic speeds, congestion and accidents especially in large cities. Smart monitoring systems, able to control the status of prohibited parking (on-road) spots, automatically notify the municipal police to take immediate actions and thus enforce citizens' compliance with parking regulations. Parking regulation enforcement raises the conscience of citizens towards respecting road regulations and in that way, it facilitates street management and eliminates traffic congestion and road accidents.

4.2 Business Objectives for Smart Resources

BO5. Reduced water leakage

Reduced water leakage can be achieved through monitoring the water supply network in order to act immediately when a breakdown in the network appears or to take proactive actions beforehand. An automated monitoring system triggers an alert notification (e-mail, SMS) whenever the water level goes below a predefined threshold or in case it has uncommon behavior.

BO6. Water level monitoring

Monitoring of the water level of seas, rivers and lakes during the rainy seasons can mitigate potential inconvenience, such as floods or drought. In case of water scarcity during the dry seasons, citizens could be informed to consume less amounts of water. Additionally, smart water supply networks can be automatically adjusted in a way to supply limited quantity of water in buildings whenever needed.

BO7. Water quality monitoring

The continuous measurement of the water resources' (i.e., rivers, lakes, etc.) quality will prevent citizens from hazardous water usage. In case of uncommon indications, the government or municipalities can get properly informed to take protective measures and prevent citizens from consuming contaminated drinking water thus avoiding disease transmissions.

BO8. Efficient waste collection services

Optimizing the waste collection routes can facilitate the cleaning service insuring cost efficiency. Waste collection services can be notified in real-time with push notifications based on GPS coordinates of the mobile devices of truck drivers, whenever bins are overflowed with trash and need immediate intervention.

Furthermore, a dedicated mobile application will enable citizens to share the status of waste containers, in case they are full and require immediate actions, along with the geographic position of the specific item. Both a web-based platform monitored by stakeholders and a mobile application used by the cleaning service could be automatically updated.

BO9. Automated monitoring system for recycling items

Frequently, people accidentally or not knowing put non-recyclables in recycling bins, which unfortunately breaks down the recycling process. Sensors integrated in the trucks could scan the recycling trash and identify the existence of organic elements. In case non-recyclable items are identified, an automated monitoring system can inform the waste collectors to return the trash back to the household.

4.3 Business Objectives for Smart Environment

BO10. Protect vulnerable groups from exposure to high concentrations of air pollutants

High accumulations of particles and toxic substances (e.g., carbon release steaming from increased use of fireplaces, industrial emissions, etc.) on air worsen the health condition of people with respiratory and allergic diseases. People with respiratory and allergic diseases, including asthma, bronchitis and pneumonia, are prone to high concentrations of air pollutants both in developed and less-developed cities. A dedicated alerting system based on geospatial-referenced data could proactively inform susceptible end-users to avoid exposure to areas with intense pollutant concentration. Push notifications can be send whenever the values exceed a predefined threshold, providing: (a) the status of air quality, (b) the level of hazardous substances (e.g. CO_2) in the atmosphere, (c) suggestions for alternative routes to be followed and (d) forecasts emphasizing on extreme pollution events.

BO11. Reduced disasters caused by extreme weather conditions

Monitoring and forecasting weather conditions can be proven very valuable for smart cities through automated alerts addressed to stakeholders, including citizens, farmers, municipalities and meteorologists. Rain and wind forecasts allow citizens to avoid accidents and protect their property (houses, farming, etc.) from extreme weather events such as floods, storms, hurricanes and drought.

BO12. Reduce health incidents due to high temperature and humidity

High levels of temperature and humidity may have a negative impact to people with health issues, particularly to those afflicted with cardiovascular diseases, senior citizens and people with disabilities. For that reason, forecasts of these parameters (e.g. excessive humidity, very high/low temperatures, etc.) could be forwarded to the vulnerable with the appropriate warnings about upcoming extreme weather conditions in order to act proactively.

5 TOWARDS ACHIEVING THE BUSINESS OBJECTIVES

5.1 Big data processing pipeline

A big data processing pipeline typically consists of four distinct phases, not necessarily sequential: (a) data generation, (b) data acquisition, (c) data storage and (d) data analysis [97, 60]. Each of these phases encapsulate challenges, which if not efficiently surmounted, big data will not be easily explored in order to deliver valuable results and facilitate the decision-making process. More specifically:

1. **Data generation** refers to the way data is generated from various data sources including sensor networks. Nowadays, most of the data used to understand cities originate from digital sensors and are available in various forms, with temporal tags as well as geotags in many instances.
2. **Data acquisition**: One challenge that arises during the acquisition of data is to define the filters that should be deployed in order valuable data to be further processed and less important data to be discarded. In this phase, the appropriate metadata that will describe the form of the data recorded and the way they are recorded and measured will be automatically generated [113].
3. **Data storage** concerns persistently storing and managing large-scale datasets. Cloud Computing offers many benefits in terms of data storage, although it entails many challenges. Transferring data to the cloud can be proven a slow and unreliable process when using traditional transfer software technologies. An exemplary cloud computing architecture is illustrated in Figure 10, inspired by IBM.
4. **Data analysis**. There is a general challenge that arises in the foreground regarding leveraging big data gathered through ICT in smart cities. Cities have ended up with a vast amount of data and yet without any capability to use these data and without being able to extract information towards taking better decisions for the public good. To interpret such data, data-mining techniques together with visualization tools needs to be exploited and extended through which the visualization of correlations and patterns will be essential. Big data visualization brings opportunities of presenting data in a meaningful and intuitive way.

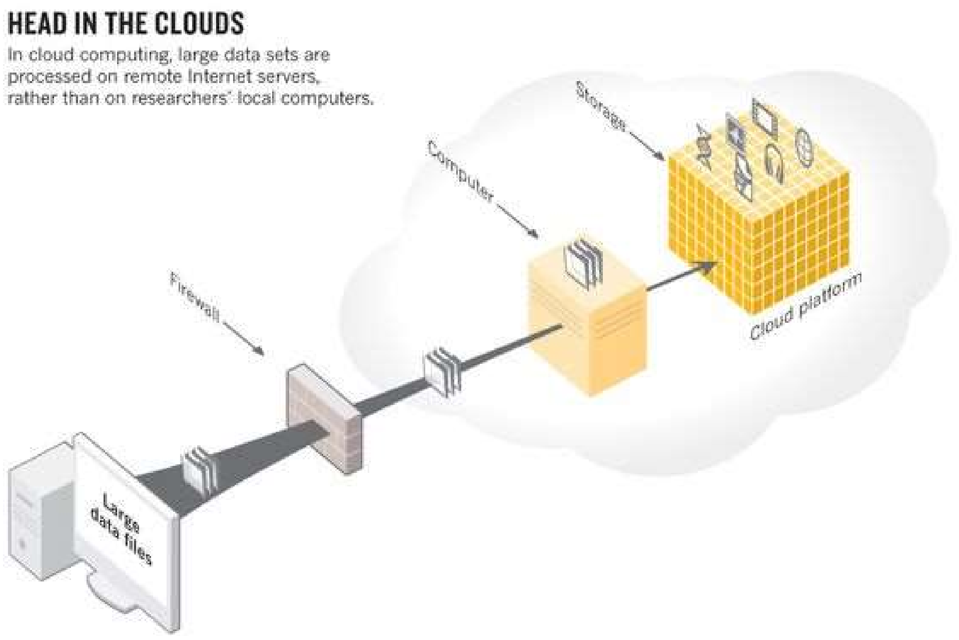

HEAD IN THE CLOUDS
In cloud computing, large data sets are processed on remote Internet servers, rather than on researchers' local computers.
Storage
Computer
Firewall
Cloud platform
Large data files

Figure 10: A cloud-computing infrastructure based on IBM Aspera software [126]

In an effort to enrich the aforementioned architecture, the authors of [28] have introduced an abstract data analytic pipeline, depicted in Figure 11. However, according to [142] a typical big data pipeline often consists of many separate solutions that cover one or more steps of the pipeline.

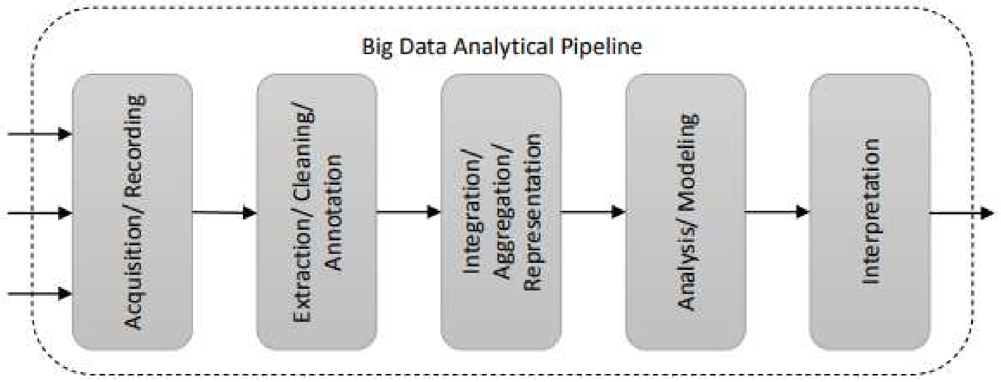

Big Data Analytical Pipeline
Acquisition/ Recording
Extraction/ Cleaning/ Annotation
Integration/ Aggregation/ Representation
Analysis/ Modeling
Interpretation

Figure 11: Data analytic pipeline

5.2 Business logic scenarios for big data processing pipelines

In this chapter, we outline indicative business logic scenarios for big data processing pipelines in order to materialize the aforementioned business objectives. More specifically, for each focus area, we underline an indicative configuration example of the available sensors in terms of data recording and reporting intervals followed by the data flow, application logic and the mechanism that will communicate the data to the stakeholders.

Road traffic monitoring:

- Record and count the number of vehicles passing certain points (through on-road sensor installations) **Every 1 min**
- Record the number of vehicles waiting in front of traffic lights **Every 1 min**
- Transmit the number of vehicles passing a certain point **Every 3 min**
- Transmit the number of vehicles standing in front of traffic lights **Every 2 min**
- If sensors return value equal to zero, then they send a notification to the server. That constitutes an indication that either cars do not change position due to traffic or no cars pass from the specific point at all. This information in correlation with other measurements of parameters such as CO_2 emissions can lead to more precise results.
- If the congestion takes place near to an intersection, the nearest traffic lights will be appropriately adjusted to mitigate traffic congestion.
- The server will calculate and estimate alternative and shorter routes for the drivers to follow. This information can be (a) presented through variable-message signs (VMS) on the road, (b) broadcasted through radio frequencies (TA) or (c) delivered to in-car navigation systems.
- The server will provide a dedicated API making traffic information open to the public.

Parking monitoring:

- Record sensor *status* (available or occupied) and *type* of parking place (PWD - people with disabilities or PWND - people with no disabilities) (through sensors embedded in the asphalt in parking lots positions) **Every 1 min**
- Record *start_time* and *end_time when sensor's status becomes occupied*
- Transmit the current sensor status and type of parking place to the server **Every 5 min**
- Check if a sensor status remains occupied for longer than the permitted time range and maximum stay
- Every time sensor' status changes, the corresponding signal will be send to the server
- Every time the sensor status remains occupied longer than the time range and the maximum stay and the sensor status is occupied, a warning will be sent to the server
- Whenever the sensor type is *PWD* and the parking place is *available* this information will be forwarded to the server
- The server will provide a dedicated API so as end-users to be informed through a mobile application
- Every time a sensor installed in prohibited parking spaces is occupied, an emergency notification will be forwarded to the server
- The server will propagate the event to the municipal police through available communication channels (Rest API)

Water leakage detection:

- Record the water flow and pressure **Every 5 sec**
  (through sensors installed in a city's water supply network)
- Transmit the mean average of the aggregated data to the server **Every 5 min**
- If the water pressure in certain spots of the pipeline network deviates from the normal values, then an alert will be sent to the municipal water authority

Water level monitoring in water tanks:

- Record the water quantity consumed per household **Every 30 min**
  (through sensors installed in the pipes of the water supply network of each building)
- Record the water level of the city's water tanks **Every 10 min**
- Transmit the mean average of the aggregated data to the server **Every 30 min**
- If water level in water tanks is below threshold, then sensors will send data to the server
- If the water quantity consumed per household exceeds normal values, then sensors will send data to the server
- The server will send a signal and activate valve actuators, embedded in the pipes in different city territories, to control the water flow
- The server will propagate the event to municipal water authorities through available communication channels (Rest API)

Water quality monitoring in water tanks:

- *Record the water quality parameters in water tanks* **Every 10 min**
  (through sensors installed in water tanks of a city and in households)
- Transmit the data to the server **Once a day**
- In case the values of specific measurements (i.e. temperature, conductivity, pH, water height, or chloride, ammonium, nitrate and calcium ions, etc.) exceed for instance two or three standard deviations (95% - 99.7%) from the mean of the total number of values, that indicates that there are non-normally distributed variables. Therefore, an emergency notice will be forwarded to municipal authorities through a dedicated mobile/web application.

Waste management:

- Record the status of trash cans and waste containers **Every 5 min**
  (through embedded sensors in containers able to track different capacity levels: 10%, …, 80%, 90%, 100%)
- Transmit the data to the server **Every 60 min**
- If the level of the trash exceeds 90% of the bins' total capacity, then an alert will be forwarded to the waste collection service through a dedicated mobile app
- Citizens will be able to use the mobile app to inform waste collectors regarding overloaded trash cans or containers based on GPS coordinates
- The server will calculate and estimate optimized routes for the city's cleaning service through a web platform or a mobile application based on GPS routing.

Air quality monitoring:

- Record values indicating air quality **Every 5 min**
- Transmit the data to the server **Every 30 min**
- If sensors detect high levels of air pollutants (e.g. SO_2, NO, NO_2, CO_2, O_3, smoke concentrations, etc.), then emergency notifications will be transmitted to the server
- The server will forward the information to a mobile application available for citizens with health issues and through a web interface giving access to professionals, e.g. environmental scientists, prefectural civil protection agencies, etc.

Weather monitoring:

- Record values of weather parameters **Every 1 min**
- Transmit the mean average of the aggregated data to the server **Every 5 min**
- If sensors detect indications of weather conditions that exceed predefined thresholds, then notifications will be instantly transmitted to the server
- The server will forward the information to registered users with health issues through a dedicated mobile application and make it available to stakeholders, such as meteorologists, farmers, etc. through a web interface

5.3 Big Data volume analysis

The following table comprises an estimation of the data volume generated from a sensor network in the aforementioned focus areas. In addition, it presents an estimated range of sensors that can be installed in a smart city setup, in accordance with the existing infrastructure of the biggest urban centers. More specific, some indicative published records include ca. 12.500 intersections with traffic signals in New York City (NYC) [128], 18,6 million parking spaces in Los Angeles [121], 17.000 water tanks scattered throughout NYC [2] and 3.600 installed air quality sensors across Newcastle [5]. As reported by Data-Smart City Solutions [51], Environmental Defense Fund (EDF) in collaboration with Google embedded thousands of mobile sensors in vehicles and bikes, which collected more than 150 million data points over 750 hours of driving.

Table 4: Data volumes extracted from a smart city IoT infrastructure

Parameters	**Number of sensors**	**Data volume (per sensor, per day)**	**Total data volume (per day)**	**Frequency of updating data**	**Frequency of sending data**
Number of vehicles passing certain points	< 10.000	< 11.500 KB	< 115 GB	Every time a vehicle is detected	Every 3 min
Number of vehicles waiting in front of traffic lights	< 12.500	< 19.200 KB	< 240 GB	Every 1 min	Every 2 min
Parking sensor status: available / occupied	< 18.600.000	< 11.500 KB	< 214 TB	Every 1 min	Every 5 min
Water pressure (Pa)	< 15.000	< 11.500 KB	< 173 GB	Every 1 min	Every 5 min
Water level of seas, rivers and lakes (m)	< 1.000	< 40 KB	< 40 MB	Every 60 min	Once a day

Water quantity consumed per household (m^3)	< 10.000.000	< 1.920 KB	< 19,2 TB	Every 30 min	Every 30 min
Water level in water tanks (m)	< 17.000	< 1.920 KB	< 33 GB	Every 10 min	Every 30 min
Water temperature, conductivity, PH and ions (i.e., NH4+, NO3−, Cl−, and Ca2+)	< 1.000	< 11.500 KB	< 11,5 GB	Every 10 min	Twice a day
Waste container status (capacity levels: 10%,…, 80%, 90%, 100%) (cm)	< 12.000	< 960 KB	< 11,5 GB	Every 5 min	Every 60 min
Air quality (SO_2, NO, NO_2, CO_2, O_3)	< 15.000	< 1.920 KB	< 28 GB	Every 5 min	Every 30 min
Temperature (^oC), humidity (g/m^3), wind speed (mph) and direction, rain (mm), and atmospheric pressure (atm)	< 1.000	< 11.500 KB	< 11,5 GB	Every 1 min	Every 5 min

It should be noticed that the "Frequency of updating information" involves queries sent from sensor nodes to report an event in the aforementioned intervals, whereas the "Frequency of sending information" refers to the information transmitted from the sensor nodes to the server. In any of the above cases, whenever sensors record data that reach or exceed a predefined threshold, they transmit the specific event-based observations to the server within a recent time window.

5.4 Technological Challenges

Below we address the technological challenges and identify the requirements that arise from huge data volumes in terms of data storage and data analytics applied in smart cities.

5.4.1 Challenge 1: Horizontal scalability

Smart cities need to leverage enormous amounts of data quickly and efficiently (see Table 4). Handling sheer data volumes that are constantly increasing at a high pace, requires data processing to take place over a clustered computer architecture composed of multiple computing machines each equipped with multiple CPUs and CPU cores, which we classify under the general term "processing units". Each such processing unit is supposed to undertake a piece of data to process so that various data pieces can be processed in parallel among the processing units. For instance, in case analytics need to be extracted out of the generated data (e.g. average/min/max number of vehicles during rush hours) part of the processing can be assigned to various cluster nodes each with its own CPU cores. Finally, the combined partial results can be computed to an overall result. The above refers to horizontal scalability, which is the ability of the processing to scale with increasing data volumes that arrive at high speeds.

Requirement 1a: The processing of data needs to be distributed to different nodes in order to deliver useful results to multiple users from different servers. Efficient data processing through concurrent processes increases the throughput of the system, which is the number of tuples being processed per time unit (i.e., seconds, milliseconds). Parallel processing or parallelism allows every single processing unit to process data in parallel. For instance, in both the car parking services and the congestion monitoring services, datasets from various servers that are distributed across a city could be processed in parallel to achieve the most efficient suggestions for citizens (e.g., closest available parking lots, quickest route, etc.). At the same time, high-throughput distributed messaging platforms would prove necessary allowing stakeholders to receive prompt alerts.

Requirement 1b: In a complex environment, such as a smart city, resources should be allocated to different processing units in response to changes in workload and resource availability. Elastic resource allocation allows new processing units to be progressively occupied whenever the data volume and velocity increases and thus complex processing needs to be executed. According to [174], cloud elasticity is the ability of the cloud infrastructure to rapidly change (expand or shrink) the number of resources allocated to a service during runtime. It focuses on meeting the actual varying demands on the service, while minimizing the resource provisioning costs.

Requirement 1c: Smart city infrastructures generate high volumes of data within seconds, minutes or even milliseconds, which need of immediate actions. Data can be processed either through batch data processing or through stream processing. The former is an efficient way of processing large volumes of data over a period at once although it does not provide results in real-time. On the other hand, stream processing is considered more appropriate in cases where continuous data processing, online data analysis and real-time answers are needed.

5.4.2 Challenge 2: Vertical scalability

The demanding technological infrastructure of the use cases presented in this work, i.e. smart transportation, smart resource and smart environmental management, involve a complex setup consisted of numerous sensors that generate big streaming data that arrive at high speeds continuously. For instance, constant reports of average temperature in the atmosphere and intensity of air pollutants are streaming operations that involve real-time analytics. Taking into consideration the number of sensors utilized in the proposed setup along with the number of streams generated by each sensor, the complexity of data streaming processing requires efficient ways to be handled. More advanced big data analytics involve quantifying pairwise correlations of streams originating from sensors. The computational complexity of judging such pairwise correlations is quadratic to the number of input streams, thus the targeted smart city use cases may operate over tens of thousands of streams. Therefore, vertical scalability, i.e., the ability of the computation to scale with the increasing number of incoming streams, is an essential issue that needs to be accounted for to meet the set business goals.

Requirement 2: Instead of naively adding more resources to the cluster to cover the potentially quadratic (or higher) complexity of certain analytics tasks, one needs to come up with scalable real-time algorithms, focusing on the particular problems. Choosing representative sensor streams and relying on stream approximation techniques with predefined accuracy guarantees is a standard way to proceed with in order to achieve vertical scalability. In other words, an approximation technique may sample only a number of streams to analyze [73] or apply some dimensionality-reduction technique [72, 75] to reduce the size of the streams and speed up the computation. More specific, state-of-the-art site sampling techniques [73], perform analytics using a sample of sites proportional to $\sqrt{N}$ instead of N, where N is the total number of sites in the network. Similarly, dimensionality reduction techniques applied for sensor data, pinpoint interesting phenomena such as outliers mentioned in [72, 75] to reduce the amount of communicated data and the number of pairwise similarity comparisons. These two approaches combined could be considered a plausible way to proceed.

5.4.3 Challenge 3: Federated scalability - Communication reduction

Communication reduction is of the essence in smart city settings since one of the most basic ingredient of their physical architecture setup involves battery-powered sensors and wireless or wired network infrastructure. In massively distributed settings, such as the smart city scenario discussed in this work, the amount of transmitted messages need to be reduced for two reasons. Firstly, data transmission through battery-powered sensors is the main culprit in energy drain [76]. If sensor nodes deplete their remaining energy quickly, they will soon die. Consequently, when a fraction of nodes in the network dies, the whole network's graph connectivity will be lost. Secondly, when leaving energy consumption issues aside and allowing all streams to be pushed towards a corporate data center (cluster) or the cloud, the network links transferring these streams will eventually become overloaded. Thus, lags in communication will appear and the required needs for online stream analysis and real-time delivery of big data analytics to smart city applications will not be satisfied.

Requirement 3: Besides using stream approximation and summarization techniques as those mentioned in Requirement 2, there are a couple of standard tools to employ in our setup in order to achieve communication reduction. On the one hand, in-situ processing constructs qualifying conditions and places local filters to sensors aiming to minimize communication by performing as much of the computation locally at the sensor as possible. While in-situ filters may be easy to set for simple analytic tasks, they are much more challenging to adjust for generic, non-linear event query operators [74, 71, 62]. On the other hand, in-network processing is an additional tool since it pushes the evaluation of analytic operations, defined on the union of local data streams from different sensors, to sensors or network elements that are near to the relevant sensor sources. In that way, the qualified portion of local data streams that passes through in-situ filters is synthesized early, extracting compact aggregate information to forward it further in the network. In this context, an optimizer module instructing in-situ filter construction and executing in-network processing filters would be required [63].

5.4.4 Challenge 4: Fast setup of new big data processing pipelines

In rapidly changing environments, like the one we examine in this thesis, new big data processing pipelines may need to be defined at any given time according to the needs of new analytics tasks that domain experts and data analysts desire to follow. However, data analysts and domain experts often do not possess the necessary programming skills to materialize the necessary data processing pipeline in a timely fashion. It may take days, weeks or months to prepare and correctly execute newly constructed big data processing pipelines for data analysts that are not data scientists.

Requirement 4: A visual tool for setting up the desired data processing pipelines that would automatically translate the setting to optimized code is highly desirable. In contrast to mainstream solutions, what is needed here is to design and develop a flexible, pluggable, distributed software architecture that will be largely programmable and set up by graphical data processing workflows. This will allow experts with no programming skills to take the most out of the underlying computational resources, by interactively querying data-at-rest and data-at-motion and experimenting with a multitude of otherwise highly-opaque models with complex parameter configurations.

5.4.5 Challenge 5: Cross platform execution of big data analytics tasks

Nowadays, numerous applications have been developed to cover different needs. These produce diverse and large datasets that have to be analyzed and ultimately translated into valuable conclusions for the decision-making process. To cover that need big data analytics platforms are being continuously evolving in order to allow end-users to have access to a range of different analytics tasks. Thus, emerging applications use or exploit the use of a diversity of platforms for effective or efficient data processing. However, applications are typically tied to one single platform, whereas different data analytical pipelines need to intercommunicate with one another through cross-platforms for data processing.

Requirement 5: A systematic solution that enables efficient and easy-of-use cross-platform optimization for data analytics is required. A dedicated middleware, able to answer queries among big data platforms, would prove inevitable in order to allow applications to run over one or multiple platforms efficiently.

5.4.6 Challenge 6: Optimized big data pipelines' execution over heterogeneous clusters or clouds

There is a recent trend in major cloud providers, such as Amazon AWS [93], to provide High Performance Computing (HPC) cloud infrastructure by allowing clients use CPU and GPU servers on-demand, optimized for specific applications. Emerging cloud applications (i.e., machine learning, artificial intelligence and big data) require high performance computing systems able to sustain the exponentially increased amount of data without consuming excessive power. In the smart city context, certain authorities may have securely exposed their data and big data processing pipelines and in that, they should be able to examine the potential of speeding up the execution of their analytics tasks leveraging such infrastructure.

Requirement 6: To address this challenge, an optimization module at the city authority's side is needed in order to help deciding which parts of the set up big data processing pipelines are worthwhile of being executed over GPU accelerators [103].

6 PROPOSED ARCHITECTURAL FRAMEWORK

This chapter focuses on presenting the architectural framework of a smart city that exploits technologies to support the focus areas elaborated in Chapter 3. The proposed framework introduces specific technological solutions to address the challenges of Chapter 5.4 and ultimately to fulfil the business objectives defined in Chapter 4.

6.1 A Generic Architectural Framework

In the following architecture, we present a city governance model that comprises the city main authority, referred to as Smart City Governance, and three sub-authorities regarding (i) Transportation Management, (ii) Resource Management and (iii) Environmental Management.

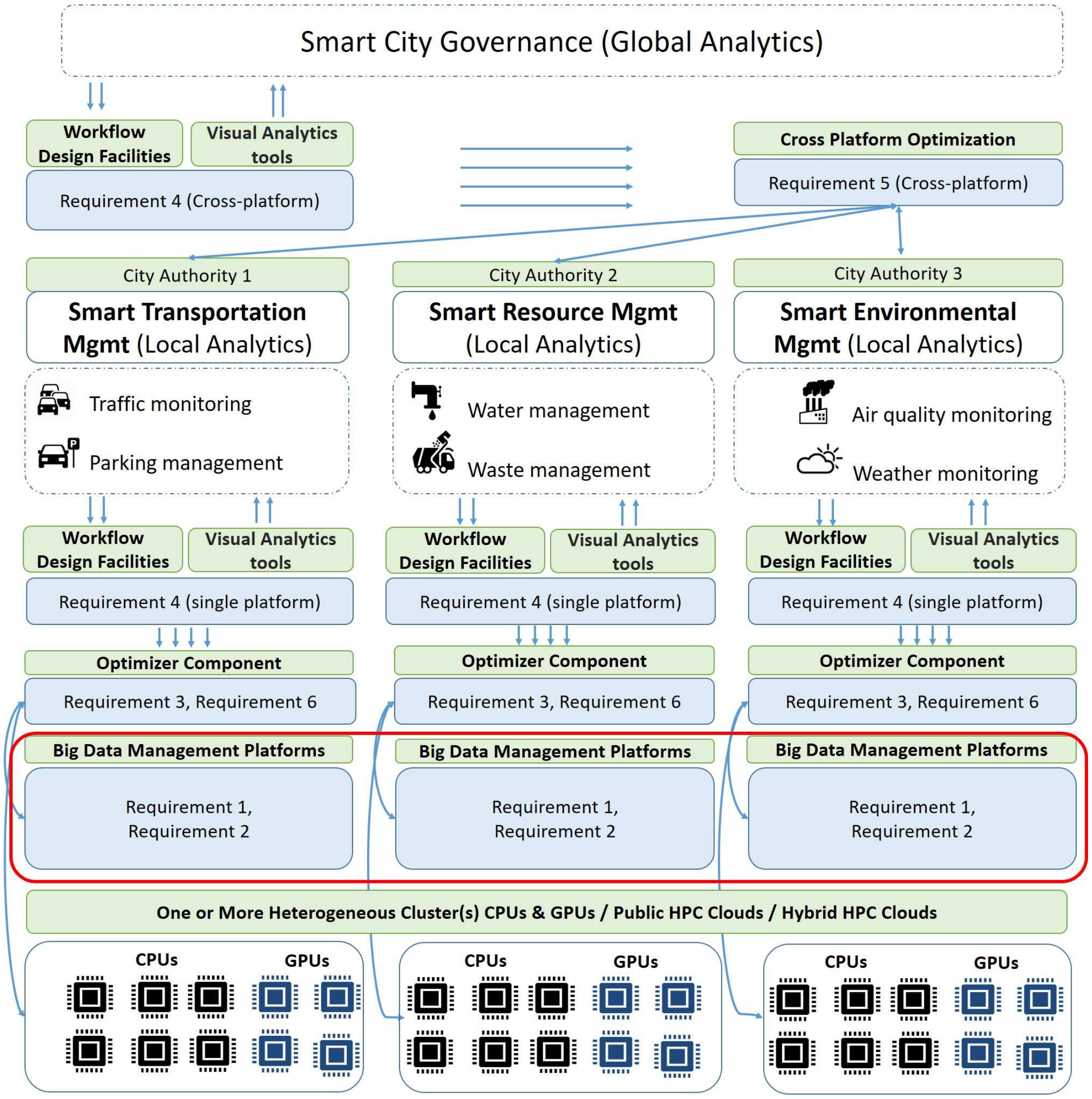

Smart City Governance (Global Analytics)
Workflow Design Facilities
Visual Analytics tools
Requirement 4 (Cross-platform)
Cross Platform Optimization
Requirement 5 (Cross-platform)
City Authority 1
Smart Transportation Mgmt (Local Analytics)
Traffic monitoring
Parking management
City Authority 2
Smart Resource Mgmt (Local Analytics)
Water management
Waste management
City Authority 3
Smart Environmental Mgmt (Local Analytics)
Air quality monitoring
Weather monitoring
Workflow Design Facilities
Visual Analytics tools
Requirement 4 (single platform)
Optimizer Component
Requirement 3, Requirement 6
Big Data Management Platforms
Requirement 1, Requirement 2
One or More Heterogeneous Cluster(s) CPUs & GPUs / Public HPC Clouds / Hybrid HPC Clouds
CPUs
GPUs

Figure 12: Framework architecture. Addressing the requirements arising from the technological challenges

Streaming event data arriving at multiple, potentially geographically dispersed, platforms should be efficiently processed in situ and then combined to provide holistic answers to global application queries. In our context, **Smart City Governance** needs to monitor and combine the information extracted locally from each sub-authority. To achieve this, **Visual Analytics tools**, enable Smart City Governance to query, visualize, alert on and understand metrics extracted locally from each sub-authority. Application queries can be processed through a technological component, named **Data Workflow Design Facilities**, that enables stakeholders to create intuitively new data processing pipelines combining different big data platforms, based on *Requirement 4*. Afterwards, specific requirements originating from the main authority are forwarded to a **Cross Platform Optimization** middleware (*Requirement 5*). The optimizer ensures the appropriate management of the queries and determines efficient execution mechanisms, combining different applications that run over multiple big data platforms.

The middleware communicates with the sub-authorities where analytics tasks are deployed locally. Each city authority submits queries using different platforms to extract numerical and graphical analytics results and forwards them to the **Optimizer Component**. The optimizer module needs to be able to instruct in-situ filter construction and execute in-network processing filters based on *Requirement 3*. The flood of high speed of streaming data abstracted from sensor feeds can easily overwhelm the, often, limited CPU memory capacities of a stream processor. To overcome this issue, the optimizer component decides which tasks need to be performed in one or more **Heterogeneous Clusters** by distributing them to CPU and GPU servers upon demand depending on the required computing performance, based on *Requirement 6.* This can be also achieved through public or hybrid High Performance Computing (HPC) clouds.

Then, the optimizer will instruct **Big Data Management Platforms** how to parallelize the required work and how they need to be handled, including assigning tasks and aggregating partial results, based on *Requirements 1 and 2*. At the same time,

The following figure (Figure 13) illustrates a more detailed (zoomed-in) view of the "Big Data Management Platforms" component. Big data Platforms receive as input the aggregated data arriving from WSN nodes, GPS devices and other sources. Since the data comes from various sources, it needs to be cleansed and transformed in a comprehensive format in order to be further analyzed. In this context, comes **data ingestion** that is the operation of fetching raw data from external sources and converting it into a format suitable for processing. In the presented architecture, a "**publish and subscribe"** messaging system is added before batch and speed layer, to provide instant event notifications for the distributed applications. The publish/subscribe (pub/sub) model enables event-driven architectures and asynchronous parallel processing, while improving performance, reliability and scalability [25]. In a pub/sub model, any message published to a topic is immediately received by all the subscribers to the topic. In particular, neither senders ("publishers") nor receivers ("subscribers") need to be aware of who uses and who provides the information of messages. Instead, message brokers receive and transmit the data associated with one or more publishers or subscribers and ensure that the messages are delivered to the correct subscribers. A number of publish/subscribe platforms both for batch and speed layer capable to handle a vast amount of streams of records, will be introduced in the following section.

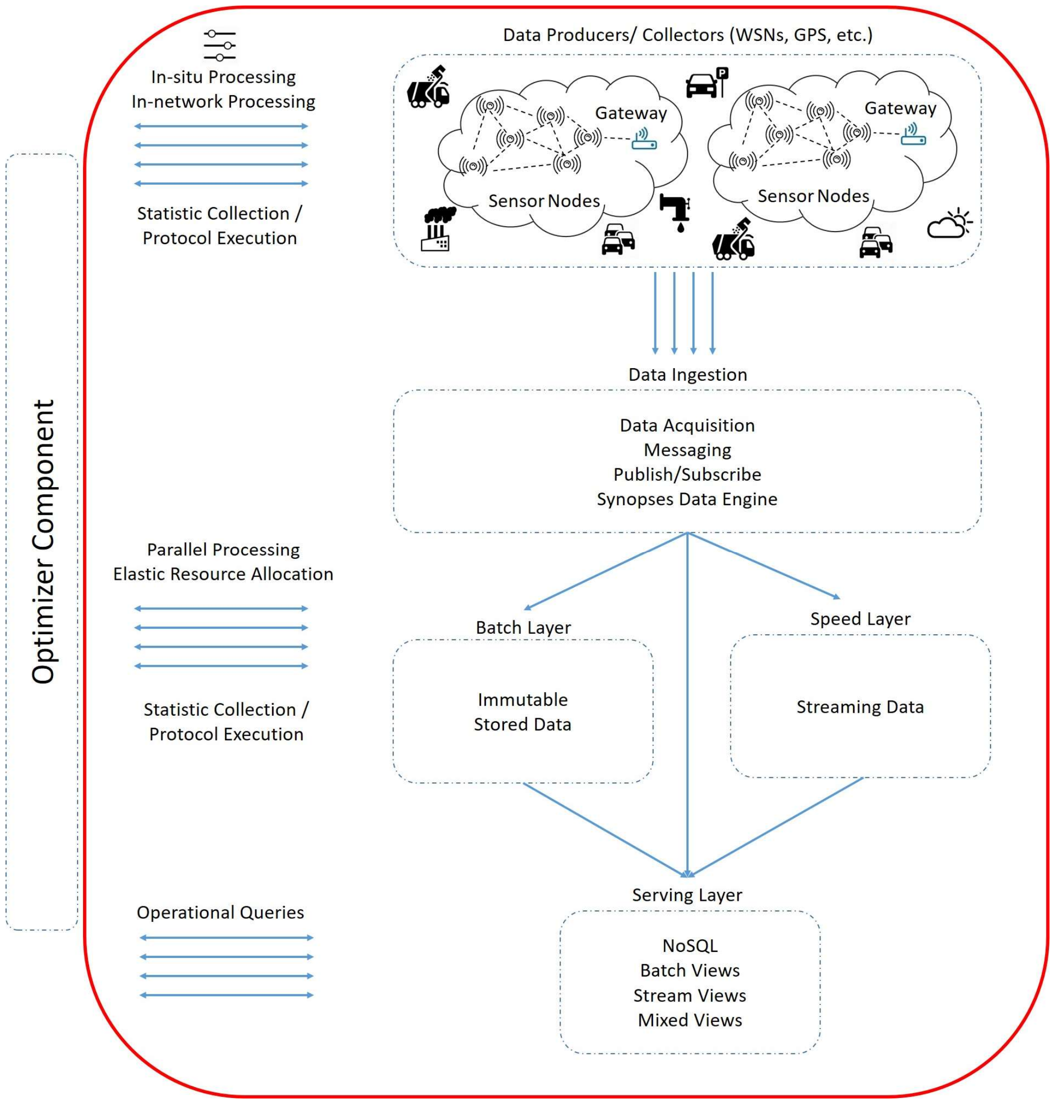


Figure 13: Framework architecture. A more focused analysis

The complexity of a smart city's infrastructure entails the challenge of effectively analyzing the exponentially growing data. For that purpose, advanced statistical analysis is required in any smart city setup to extract valuable insights and allow stakeholders to take proactive decisions by exploring easily and receiving fast comprehensive answers to their queries. In this regard, a **Synopses Data Engine** (SDE) [166] will answer analytic queries using statistical sampling techniques and probabilistic data structures, without processing the entire dataset. SDE allows data analysts to run high-speed interactive analytics over billions of records.

Afterwards, our architecture comprises a combination of both batch (stored data) and stream processing (real-time data), known as "Lambda Architecture". **Lambda Architecture** is a data-processing architecture designed to handle massive quantities of data by taking advantage of both batch and stream-processing methods with a hybrid approach. The architecture consists of three layers: (i) batch, (ii) speed and (iii) serving layer. It aims to serve high throughput and satisfy the needs for a fault-tolerant system, able to serve a wide range of workloads and use cases, in which low-latency reads and updates are required. Lambda Architecture depends on a data model with an append-only, immutable data source that serves as a system of record and it is considered a robust solution that handles a wide array of data processing and querying challenges at scale [35]. It attempts to balance the features of batch and real-time modes, where real time ingestion means importing the data as it is produced by the source and ingesting data in batches means importing discrete chunks of data at regularly scheduled intervals. In particular, it comprises the following steps [114]:

1. All **data** entering the system is dispatched to both the batch layer and the speed layer for processing.
2. The **Batch Layer**: (i) manages historical data from the master dataset (an immutable, append-only set of raw data), and (ii) makes predictions by pre-computing the upcoming batch views based on machine learning models. More specifically, the batch layer receives arriving data, combines it with historical data and re-computes results by iterating over the entire master data set. It may process incoming data slower than the speed layer, it, however, produces most accurate results [182].
3. The **Speed Layer** processes data streams in real time and deals with recent data only, while it processes data in a smaller scale. It is responsible for immediate processing of incoming data, although it may entail data inaccuracy in comparison to the batch layer.
4. The **Serving Layer** aggregates and merges results from batch computations and the speed layer into a complete answer. Any incoming query can be answered by merging results from both batch and real-time views.

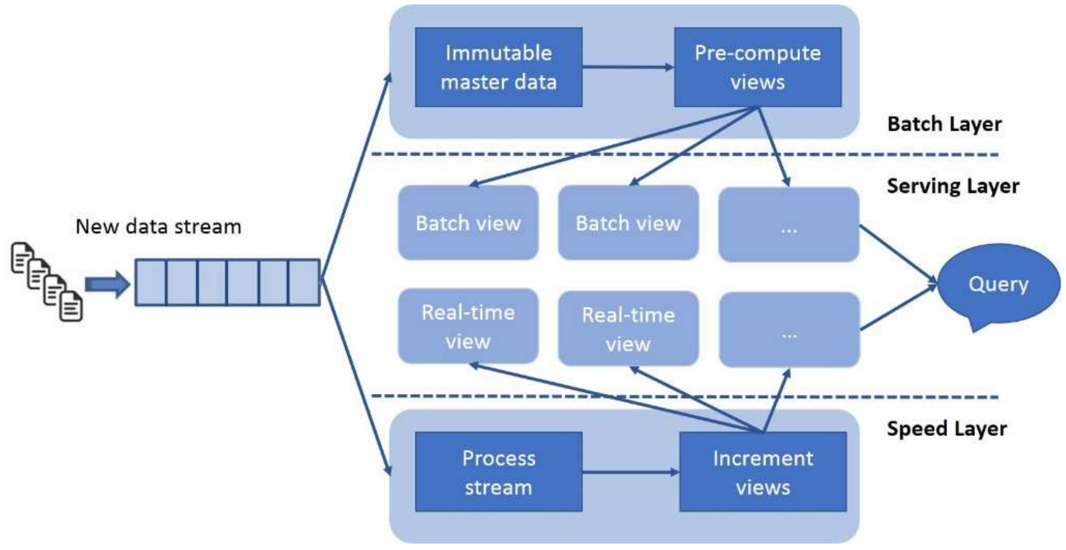


Figure 14: Lambda Architecture from a high-level perspective

Mixed views, of batch and streaming data, may be applied in certain use case scenarios, where data need to be stored and processed in batches in the master data set off-line in order to extract rules. At the same time, the speed layer forecasts these rules to proactively apply them and push notification alerts to the stakeholders.

In cases where simple transactions of queries are needed, NoSQL databases can be applied to quickly serve operational needs and provide effectively interactive responses on an ad-hoc basis without necessarily falling in the stream processing category.

6.2 Architectural Framework Materialization

Trying to address the aforementioned requirements implies the adoption of more than a single data processing platform depending on the technological infrastructure that each authority deploys. In this chapter, we propose specific big data platforms that comply on a great extent with the aforementioned requirements based on the previous framework architecture, presented in Figure 12, and aligned with the main city components.

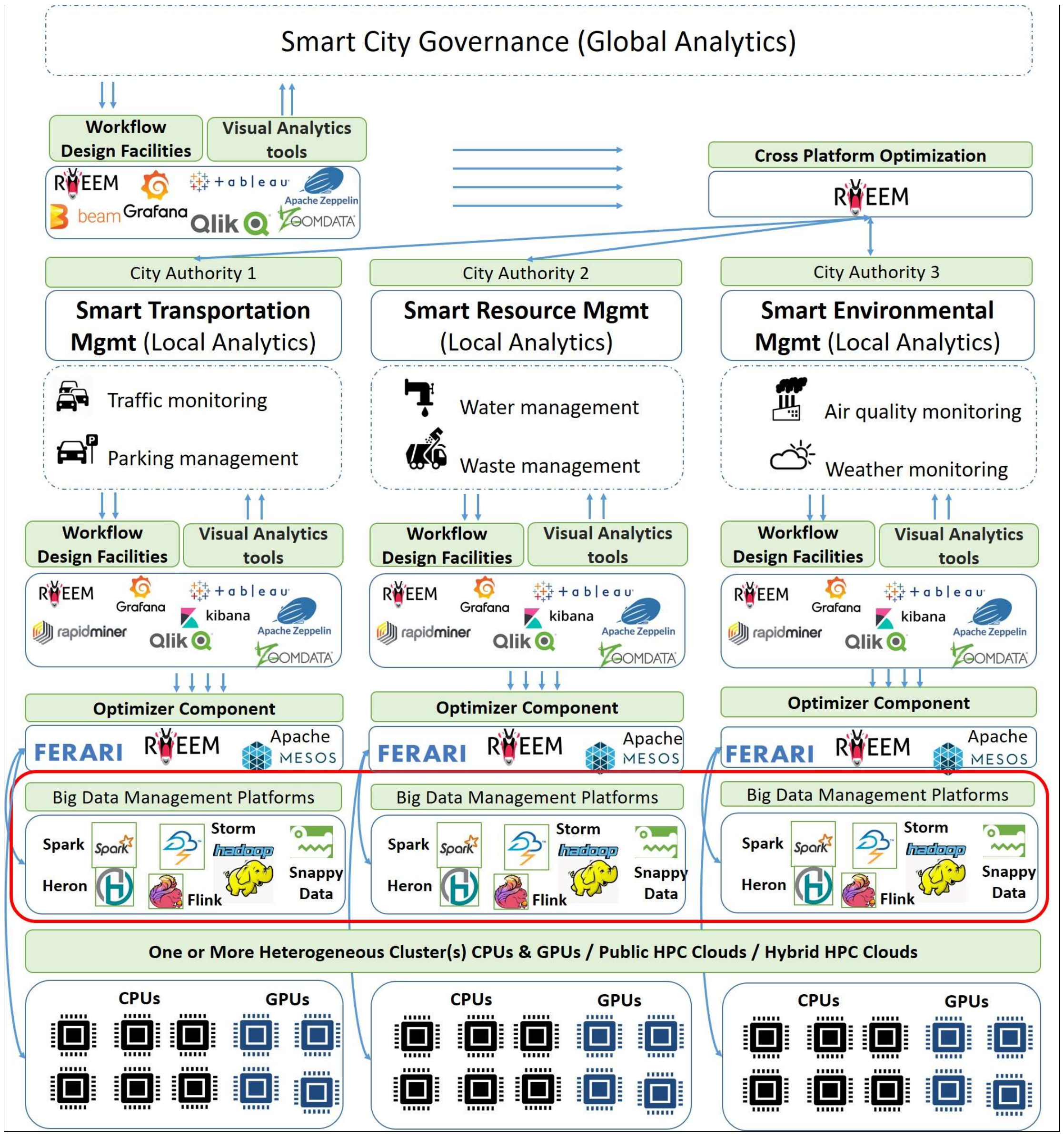

Smart City Governance (Global Analytics)
Workflow Design Facilities
Visual Analytics tools
RHEEM
Grafana
tableau
Apache Zeppelin
beam
Qlik
ZOOMDATA
Cross Platform Optimization
City Authority 1
Smart Transportation Mgmt (Local Analytics)
Traffic monitoring
Parking management
City Authority 2
Smart Resource Mgmt (Local Analytics)
Water management
Waste management
City Authority 3
Smart Environmental Mgmt (Local Analytics)
Air quality monitoring
Weather monitoring
kibana
rapidminer
Optimizer Component
FERARI
Apache MESOS
Big Data Management Platforms
Spark
Storm
hadoop
Snappy Data
Heron
Flink
One or More Heterogeneous Cluster(s) CPUs & GPUs / Public HPC Clouds / Hybrid HPC Clouds
CPUs
GPUs

Figure 15: Framework architecture. Big data platforms

6.2.1 Platforms for Workflow Design Facilities and Cross Platform Optimization

Smart cities need to integrate various types of real time data (structured and unstructured) selected from different sources. Using more than one data processing platform to perform a data analytics task is very common nowadays, thus there is an urgent need for cross-platform data processing. For that purpose, and in order to address the fourth and fifth requirement (see Chapter 5.4.4 and 5.4.5), we propose the following solutions:

- **Rheem** has been recently introduced by [4] as the first general-purpose cross-platform system that tackles the challenge of enabling users and applications to run data analytic tasks efficiently on more than one data processing platforms. Rheem operates as an orchestrator of tasks performed in various platforms and is able to: (a) suggest the most suitable platform for a given task, (b) perform inter-platform parallelism to prevent slow platforms from dominating execution time and (c) move data across platforms while minimizing the costs in both runtime and monetary costs. Rheem supports a variety of big data platforms such as Spark, Flink and GraphX, however, its support for streaming processing is limited to JavaStreams.

- **Apache Beam** [176, 169] is an evolution of the Cloud Dataflow [8689] model created by Google to process massive amounts of data. It is a unified programming model (Software Development Kit - SDK) that defines and executes data processing pipelines that implement both batch and streaming data processing jobs (**B**atch & str**EAM**) running on multiple execution engines. It binds frameworks, such as Hadoop, Spark, Flink, JStorm and Google Cloud Dataflow, and data sources while it provides an abstraction to the application logic from a big data ecosystem [29]. Although Apache Beam does not include an optimizer, it allows users to indicate any of the available runners (Apache Spark, Apache Flink, Google Cloud Dataflow, Apache Apex, Apache Samza, etc.) in which each query will be executed (Figure 16). Moreover, it allows users to write their pipelines in a language more familiar to them and integrated with other tooling, while it has experienced significant growth due to its both community and feature set. In that context, Apache Beam can be used from a main authority to unify different technologies, frameworks, APIs, languages, and software development kits that may be used from the sub-authorities to aggregate and analyze data.

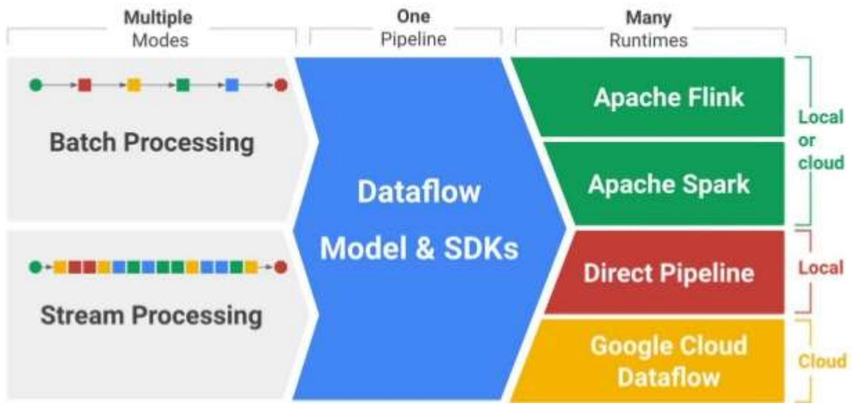


Figure 16: The Beam model – Data processing pipelines (Source: Data Science Central [48])

- **Rapidminer** eliminates the complexity of data distributed analytics solutions (e.g. Hadoop) that they usually lack a user interface and require specialized IT and analytics skills. It constitutes a data-science software platform that provides an integrated environment for data preparation, machine learning, deep learning, text mining, and predictive analytics. It supports machine-learning processes such as data preparation, visualization, model validation and optimization [94]. An extension for RapidMiner, Radoop [139], allows users to, easily, execute in Hadoop in order to apply minimum effort without needing to have specialized Hadoop skills. Radoop is a data-mining tool that provides easy-to-use operators for running distributed processes on Hadoop. Rapidminer is a visual workflow designer suggested to be used for local data analytics that will be executed from each city authority separately for batch processing.

6.2.2 Visualization and Analytics

Visual analytics tools are necessary to present the extracted results to end-users in an intuitive and user-friendly way. Visual analytics solutions tend to solve the scalability and complexity of issues that arise from big data applications, including the visualization of streaming and historical data. Among a large number of visual analytics platforms that have been developed for big data, below are some indicative examples:

- **Kibana** [88] is the world’s most popular open source log-analysis platform that provides users with a tool for exploring, visualizing, and sharing dynamic dashboards on top of the log data stored in Elasticsearch clusters. Kibana’s core feature is data querying and analysis, including full-text data queuing. Using various methods, users can search the data indexed in Elasticsearch for specific events or strings within their data for root cause analysis and diagnostics. Based on these queries, users can use Kibana’s visualization features, which allow users to visualize data in a variety of different ways, using charts, tables, geographical maps and so on.

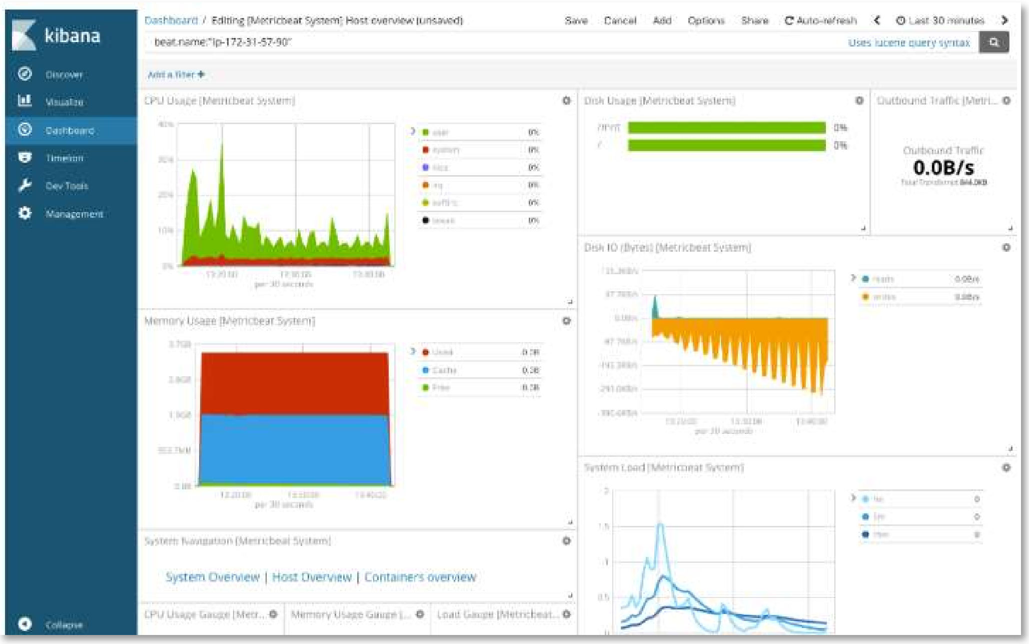

kibana
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
Collapse
Dashboard / Editing [Metricbeat System] Host overview (unsaved)
Save
Cancel
Add
Options
Share
Auto-refresh
Last 30 minutes
beat.name:"ip-172-31-57-90"
Uses lucene query syntax
Add a filter +
CPU Usage [Metricbeat System]
Disk Usage [Metricbeat System]
Outbound Traffic [Metri...
Outbound Traffic
0.0B/s
Disk IO (Bytes) [Metricbeat System]
Memory Usage [Metricbeat System]
System Load [Metricbeat System]
System Navigation [Metricbeat System]
System Overview | Host Overview | Containers overview
CPU Usage Gauge [Metr...
Memory Usage Gauge [...
Load Gauge [Metricbeat...

Figure 17: Visual analytics tools: an example of Kibana

- **Grafana** [87], is an open source metric analytics & visualization suite. It is most commonly used for visualizing time series data for infrastructure and application analytics. It can be used on top of a variety of different data stores, but the most commonly used are Graphite, Elasticsearch and InfluxDB. It is used in different domains including industrial sensors, home automation, weather, and process control. It applies data from multiple data stores in order to assist end-users to comprehend metrics through visualization while it allows queries submission and alert notification. Among a large number of data sources, Grafana supports Druid, PostgreSQL and JSON.

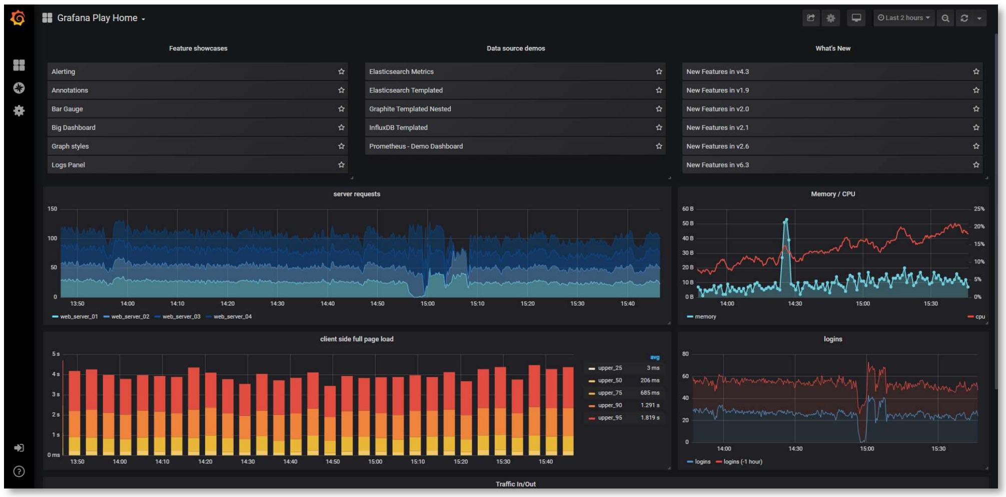


Figure 18: Visual analytics tools: an example of Grafana

- **Apache Zeppelin** [23], a visualization platform that enables data-driven, interactive data analytics and collaborative documents with SQL, Scala, R and python. It is integrated with distributed, general-purpose data processing systems such as Apache Spark and Apache Flink. Apache Zeppelin has an interactive surface that allows users to view instantly the results of their analytics providing immediate access to these results [34].

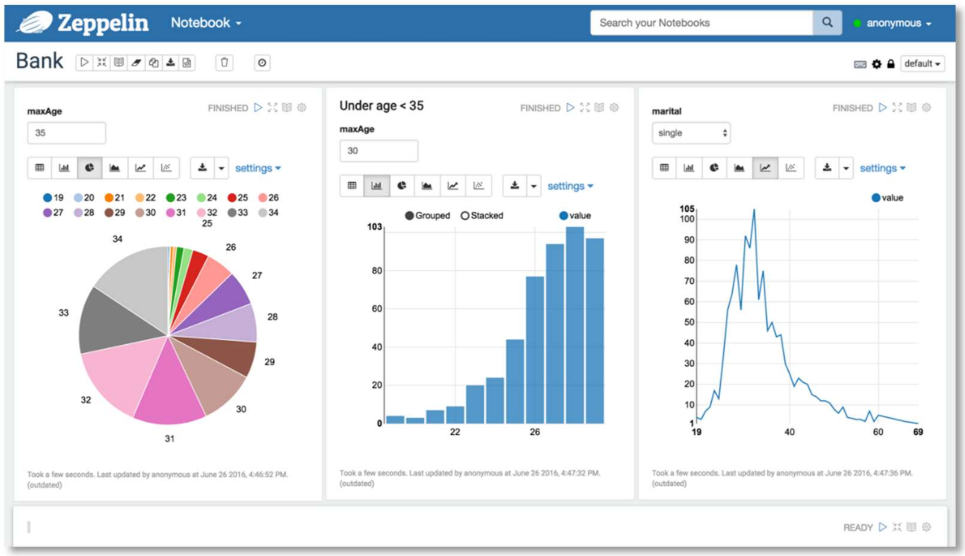


Figure 19: Visual analytics tools: an example of Apache Zeppelin

- **Zoomdata** [185], an open platform that provides big data visualization and analysis solutions. Zoomdata allows users to easily compare real-time streaming data with historical trends and makes it possible to merge data on the fly from multiple sources. It supports big data sources including Hadoop, Spark and NoSQL, as well as various traditional relational databases.

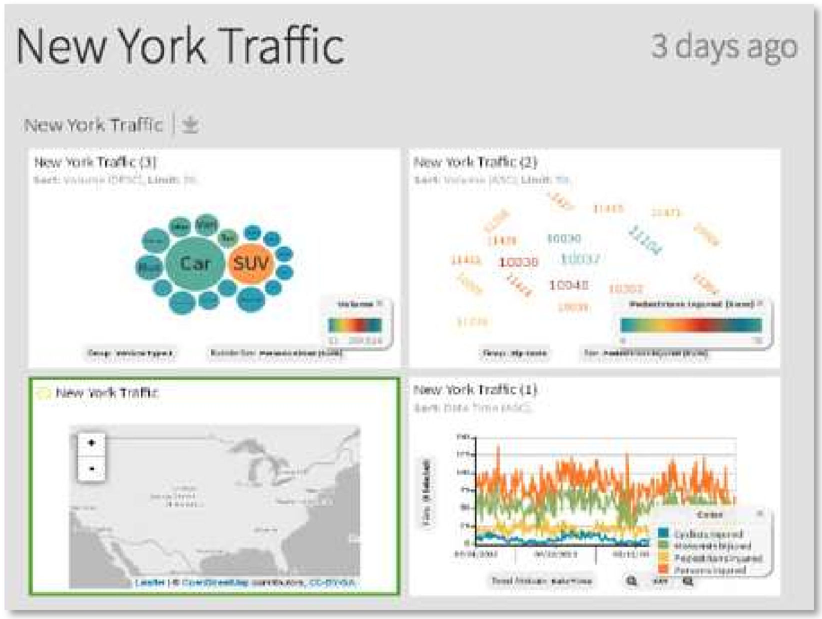


Figure 20: Visual analytics tools: an example of Zoomdata

- **Tableau** [157] is a business intelligence end-to-end analytics platform that provides powerful analytics for deeper knowledge, able to produce interactive visualizations. Tableau can be integrated with a large number of advanced database solutions including Hadoop, MySQL, SAP and other.

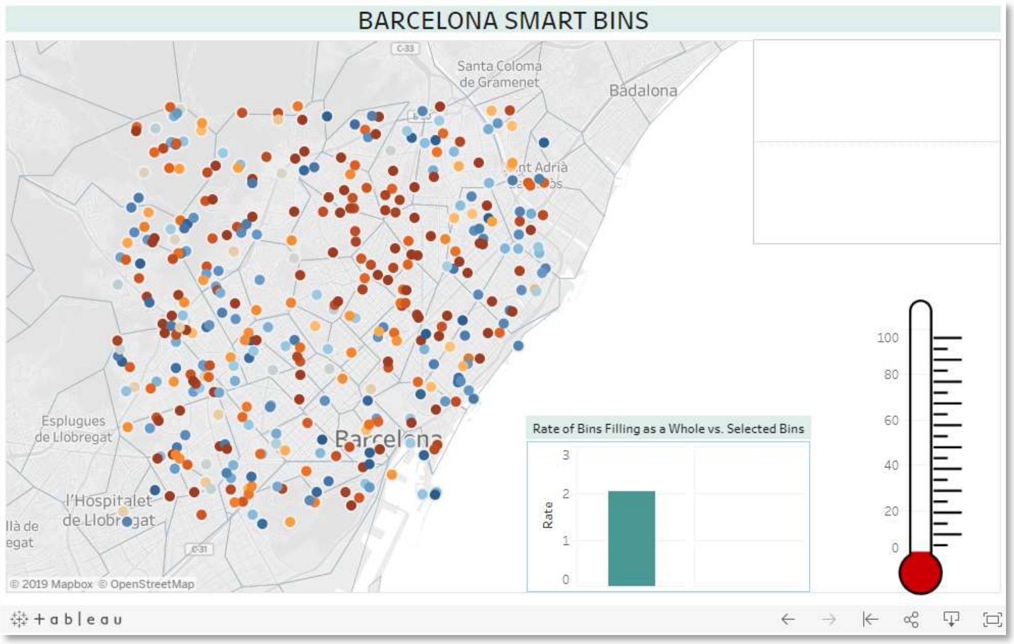


Figure 21: Visual analytics tools: an example of Tableau [59]

- **QlikView** [141] is an open data analytics platform that supports a complete portfolio of solutions providing advanced analytics across the spectrum of Business Intelligence needs. It offers the Qlik Analytics Platform (QAP), which is a developer platform for building custom analytic applications based on frontend and backend APIs. The main products of the company Qlik are the QlikView and QlikSense, where the former is suitable for guided analytics and the latter for self-service visualizations. With QlikView users pursue day-to-day tasks, analyzing data with a slightly configurable dashboard, while QlikSense allows associating different data sources and fully configurable visualizations [140].

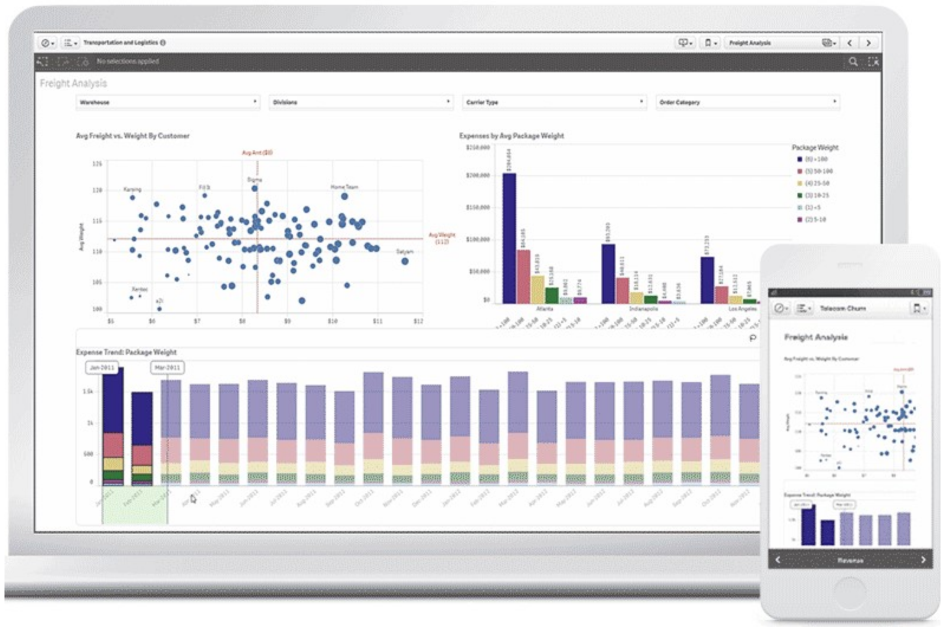

Transportation and Logistics
Freight Analysis
Warehouse
Divisions
Carrier Type
Order Category
Avg Freight vs. Weight By Customer
Expenses by Avg Package Weight
Package Weight
Expense Trend: Package Weight
Telecom Churn
Revenue

Figure 22: Visual analytics tools: an example of Qlik [141]

6.2.3 Optimizer Components

The following are considered the most appropriate optimizer tools that combined fulfill the technological requirements of our use case scenarios. More specific:

- **Rheem**. Additionally to the aforementioned, Rheem includes an optimizer that indicates in which big data platform the operators will be more efficiently executed. Rheem also provides a visual integrated development environment (IDE), called Rheem Studio that provides comprehensive facilities to computer programmers for software development. Its Graphical User Interface (GUI) is composed of four parts [4]: (i) a panel containing all Rheem operators, (ii) the drawing surface, (iii) a console for writing RheemLatin (a data-flow language) queries, and (iv) the output terminal. Rheem Studio enables users to compose their data analytic tasks in a drag-and-drop fashion and run common data analytic tasks without writing code [118].
- **FERARI** is a prototype introduced by [63] that enables real time Complex Event Processing (CEP) for large volume event data streams over distributed topologies. It aims at processing such event data streams efficiently and immediately recognizing interesting situations in real-time. FERARI orchestrates interactions among different CEP engines and chooses the best query execution plan with respect to low network

latency and reduced communication burden. It addresses the need of in-situ filter construction and in-network processing filter execution; however, it does not yet support elastic resource allocation.

FERARI constitutes a complete, multi-cloud based end-to-end CEP solution and its architecture comprises four main components.

a. **CE Query Authoring Tool** is a user-friendly, web-based query-authoring tool, which facilitates and speeds up the query formulation in order to underlie multi-cloud setup.
b. **FERARI Query Optimizer & Inter-cloud CEP** is responsible for synthesizing event information from different sites in the same subset.
c. **FERARI Intra-cloud CEP** allows different degrees of parallelization in different modules. It chooses the best query execution plan with respect to low latency and/or reduced inter-cloud communication burden.
d. **FERARI Dashboard** is a query analytics dashboard encompassing graph and map visualization tools to provide a holistic view with respect to the detected complex events to final stakeholders.

- **Apache MESOS** is a top-level open-source project able to manage computer clusters focusing on improved resource utilization by dynamically sharing resources among multiple frameworks. MESOS shares the available capacity of every connected computer in the cluster, among jobs of different natures, providing a unified view of resources on all computers. It supports resource allocation among heterogeneous clusters with different processing units, i.e., CPU and GPUs cores [19]. It supports a variety of workloads, ranging from batch processing (Hadoop), interactive analysis (Spark), real-time processing (Storm, Flink), data storage (HDFS, Cassandra), graph processing (Hama) and others.

  The main components of MESOS are:

  a. The **slaves,** which manage resources on individual nodes and are configured with a resource policy to reflect the business priorities
  b. A **master,** responsible for mediating the slave resources and frameworks i.e., who gets what resources
  c. The **framework schedulers** are application that run on MESOS and solve a specific use case, defining how to use the resources (task scheduling)

  More precisely, the MESOS master gets information on the available resources from the MESOS slaves, and based on resource policies, the master offers these resources to different frameworks. Different frameworks can choose to accept or reject the offer. If the framework accepts a resource offer, the framework allocates the corresponding resources to the framework, and then it is free to use them to launch tasks. The following figure illustrates the high-level flow of MESOS resource allocation, according to [102].

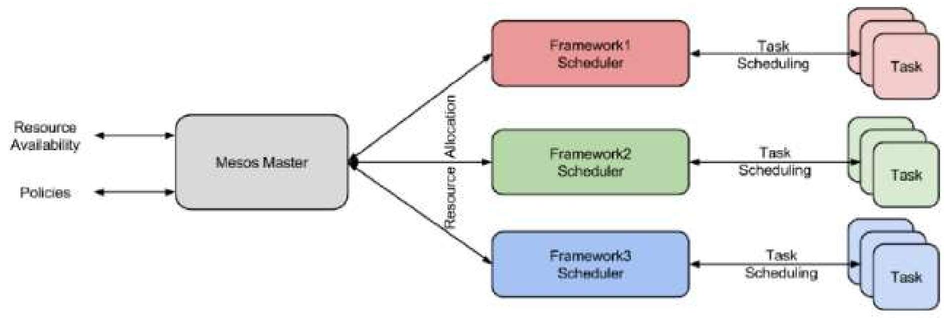


Figure 23: High-level flow of MESOS resource allocation

The main advantage of MESOS is the consolidation of various frameworks on a common infrastructure saving infrastructure costs and providing operational benefits. Last but not least, Apache MESOS provides applications with APIs for resource management and scheduling across entire datacenter and cloud environments.

6.2.4 Big Data Management Platforms

Among a very large amount of existing big data platforms, the following are recommended for realizing the requirements emerging from the potential challenges of horizontal and vertical scalability.

6.2.4.1 Data Ingestion Layer

The Data Ingestion Layer (Figure 24) comprises the following frameworks that are considered the most suitable for the proposed setup [159].

- **Apache Kafka** [17] is a distributed streaming platform that can be used to build real-time data pipelines and streaming applications and to vertical scale through Kafka brokers [164]. It is able to: (a) publish and subscribe to streams of records, (b) store partitions of records by replicating them across a number of servers in a fault-tolerant durable way and (c) process streams of records as they occur. More specific:
    - As a Messaging System, Kafka has two models: (i) queuing and (ii) publish-subscribe. In the former, producers send messages into the queue, where each message record is read by only one consumer, although multiple consumers can read different messages from the publish-subscribe model, the publisher sends messages into one or more topics and subscribers are able to consume all the messages of one or more topics [45].
    - Kafka runs as a cluster on one or more servers that can span multiple datacenters, which stores streams of records in categories called "topics". A topic is a category or feed name to which records are published. Topics in Kafka can have many consumers (applications) that subscribe to the data written to it. Each record consists of a key, a value, and a timestamp. Each partition has one server, which acts as the "leader" and zero or more servers, which act as "followers". The leader handles all read and write requests for the partition while the followers passively replicate the leader. If the leader fails, one of the followers will automatically become the new leader.

    - In refer to stream processing, Kafka takes continual streams of data from input topics, performs some processing on this input, and produces continual streams of data to output topics.

- **Redis** [143] is a key value data store that is extremely fast because the entire dataset is stored in the memory [100]. It is used as a database, cache and message broker. The publish/subscribe feature of Redis implements the messaging system where the senders (publishers) send the messages while the receivers (subscribers) receive them, through the channel [171]. It is one of the fastest NoSQL databases trading durability over speed. Redis can be considered more as a toolkit of useful data structure algorithms than as an ordinary member of a database group, because it contains a list of processes and functionalities like a blocking queue or stack, a publisher-subscriber system and a list of configurable features as expiry policies, durability levels and replication options [137, 27]. Redis supports two dimensions for vertical scaling [46]; if the Redis database is not in Cache mode the auto-scaling system will increase the available memory when under memory pressure. When in Cache mode, the auto-scaling will not operate and the administrators must select the amount of memory they need for the Redis database.

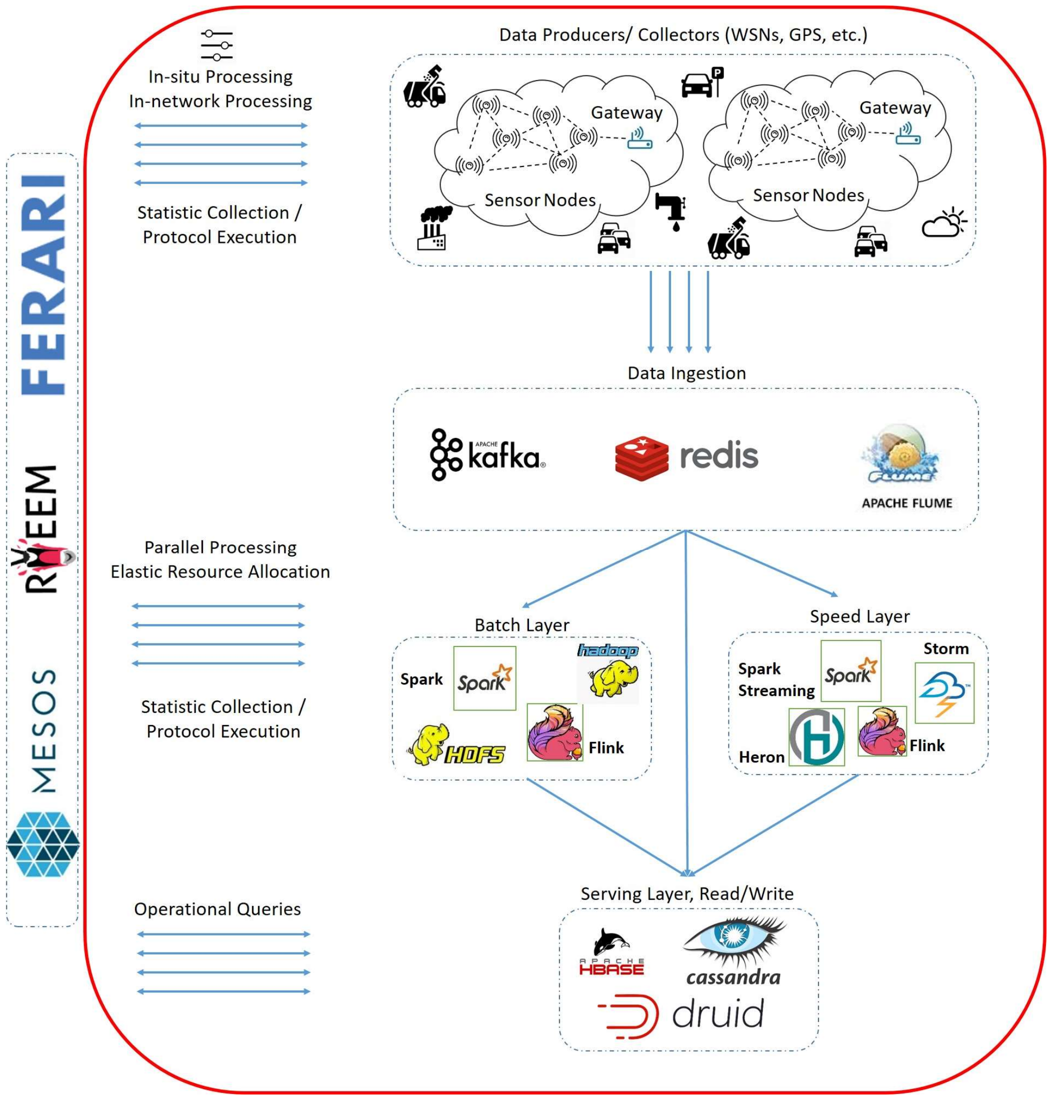


Figure 24: Framework architecture. A more focused analysis based on specific tools

- **Apache Flume** [14] is a distributed, reliable, and available service for collecting, aggregating, and moving large amounts of log data. Flume can be used to transport event data including but not limited to network traffic data, data generated by social media websites and email messages. It has a simple and flexible architecture based on streaming data flows and it is fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. Apache Flume uses a simple extensible data model that allows for online analytic application.

6.2.4.2 Batch Layer

Regarding the Batch Layer, we propose the following framework solutions as the most appropriate.

- **Apache Spark** is a unified analytics cluster-computing framework that supports modules for batch and streaming processing, SQL, machine learning and graph computation processing [50]. Spark uses a specialized fundamental data structure, the Resilient Distributed Datasets (RDDs), which is a collection of elements, partitioned across the nodes of the cluster and can automatically recover from node failures, as there is no single node having all the data. RDDs support two types of operations: (i) transformations that are performed on an RDD and creating a new dataset and (ii) actions, which return a value after running a computation in the dataset (Figure 25).

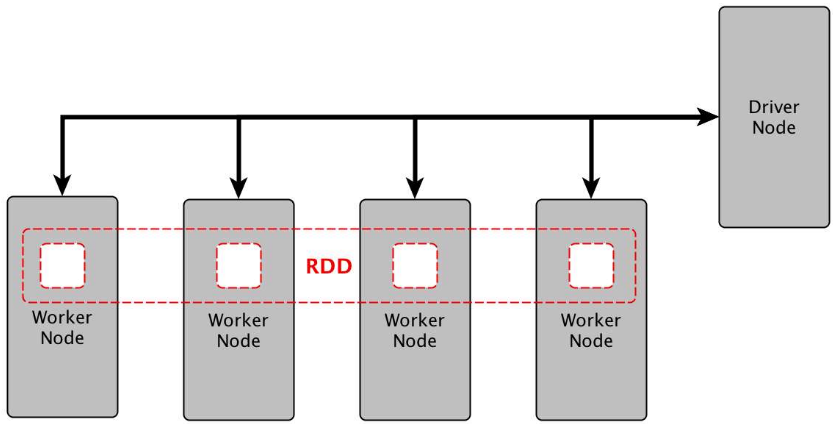


Figure 25: An RDD distributed across the worker nodes (Source: DevOps [55])

Apache Spark SQL was developed to confront with the limitations of MapReduce processing framework of Hadoop. Spark SQL is a module for structured data processing that allows developers to run SQL queries over imported data and existing RDDs [50]. It maintains MapReduce's linear scalability and fault tolerance and supports in-memory analytics, iterative algorithms and interactive queries. [153] Spark SQL provides a DataFrame API that can perform relational operations on both external data sources and Spark's built-in distributed collections at scale.

- **Hadoop** [15] is a popular open-source big data framework that allows distributed processing of large data sets across clusters of computers and it is designed to scale up from single servers to thousands of machines. It can be used for visualization processes and is composed of several components, such as MapReduce (parallel processing), YARN (job scheduling) and HDFS (distributed file system) that work together to process batch data.
  - **MapReduce** is a programming model that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. It refers to two distinct tasks that Hadoop programs perform: (a) the map job, which filters and sorts data, where individual elements of data sets are being broken down into tuples (i.e. key/value pairs) and (b) the reduce job that uses the output of the mapping process and combines the data tuples into smaller set of tuples. [18]
  - **HDFS** (Hadoop Distributed File System) is the primary data storage system used by Hadoop applications. It provides high-performance access to data across highly scalable Hadoop clusters and it enables efficient parallel processing since it breaks information into separate blocks and distributes them to different nodes in a cluster. [89] The HDFS is considered highly fault-tolerant since blocks of data are being replicated or copied and then

distributed to different nodes. Thus, in case one node crashes the replicas of data can be found to other nodes.

6.2.4.3 Speed Layer

The Speed Layer comprises the following frameworks:

- **Apache Spark Streaming** enables powerful interactive and analytical applications across both streaming and historical data. Spark Streaming is able to process 100,000-500,000 records/node/sec [135] and has some unique benefits over other traditional streaming systems, such as [49]: (a) fast recovery from failures, (b) better load balancing and resource usage, (c) combination of streaming data with static datasets and interactive queries and (d) native integration with advanced processing libraries (i.e., SQL, machine learning, graph processing). It integrates with a wide variety of popular data sources, including HDFS, Flume, Kafka, and Twitter. Moreover, Spark Streaming is able to seamlessly integrate with any other Spark components, such as Sparks SQL and MLlib. The concepts and use cases of Apache Spark Streaming looks similar to Apache Flink, however there are differences in the implementation. Spark Streaming is designed to deal with mini batches, which can deliver near real-time capabilities. Apache Flink delivers real-time processing due to its event level processing architecture [162].
- **Apache Storm** is a distributed real-time computation framework that processes fast large streams of data based on master-slave architecture [159]. As reported in [168], a Storm topology is directed graph of spouts and bolts. Spouts are sources of input data and bolts are an abstraction to represent computation on the stream. The spouts are the main entry point of data into the topology and connect to a source of data, such as Kafka [110], and emit the tuples as a stream. Whereas, bolts are considered as the operators of arbitrary computations, such as filtering and joining. They receive as input any number of streams, they process the data and they optionally emit streams to one or more bolts. Bolts are also able to read from or write data directly to a data store. These components are represented by vertices in the topology graph, where direct edges indicate how the streams are routed. Grouping in Storm can be done with the following five ways, which define how to send tuples between tasks [181].
    - **Shuffle**: Tuples are randomly distributed across receiving bolt's tasks and each task is guaranteed to receive an equal number of tuples.
    - **Fields grouping**: A field of a tuple is used as the key to partition the stream. Tuples with the same key will be mapped to the same task.
    - **All grouping**: Each tuple is broadcasted to all tasks of the corresponding bolt.
    - **Global grouping**: The entire stream is routed to one of the bolt's tasks, usually the task with the lowest ID.
    - **Direct grouping**: The producer of the stream decides which task of the consuming bolt will receive each tuple
    - **Partial Key grouping**: it works similarly to the fields grouping it however adapts the "power of two choices" technique of load balancing. Here, tuples

are load balanced between two downstream bolts, which provides better utilization of resources when the incoming data is skewed.

- **Apache Flink** is a very popular open-source stream processing framework with powerful stream and batch processing capabilities. Flink is suitable for running interactive, graph processing and machine learning applications for real time predictions [77], while it is able to process a large amount of data streams from different IoT devices at a scale of millions of events per second. It typically consists of streams and transformations and provides real-time data processing pipelines in a high-performance and fault-tolerant way. The central part of its fault tolerance mechanism is drawing consistent snapshots of the distributed data stream and operator state. These snapshots act as consistent checkpoints to which the system can fall back in case of a failure. The core element in Flink's distributed snapshotting are the stream barriers that separate the records in data streams into a set of records that goes into the current snapshot and the records that go into the next snapshot [13].

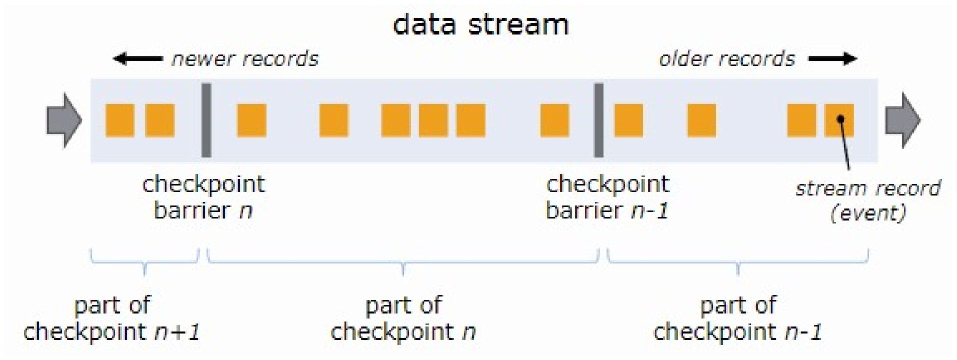


Figure 26: Data stream in Apache Flink (task manager)

Apache Flink also uses time windows to understand how the data streams flows and consequently which events are preceded by which. Moreover, it provides the Flink CEP (Complex Event Processing) library that allows programmers to define the pattern that will trigger an appropriate action (e.g. alert) whenever it is detected. CEP matches continuously incoming events against a specific pattern and discards the irrelevant for the query data at runtime, while it offers real time analytics capabilities [13].

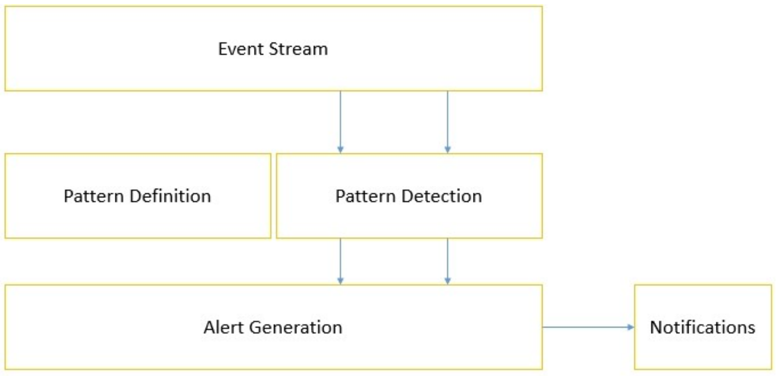


Figure 27: Core components of the Flink CEP library [53]

Flink supports connectivity with various data resources such as Apache Cassandra, Elasticsearch, Kafka, RabbitMQ and Hive [175].

- **Heron** is a real-time, distributed, fault-tolerant stream processing engine of Twitter, developed to overcome the limitations of Storm, such as scalability, debug-ability, manageability, and efficient sharing of cluster resources with other data services. [111] It runs topologies, i.e., directed acyclic graphs of spouts and bolts, where the programmer specifies the number of tasks for each spout and each bolt (i.e. the degree of parallelism), as well as how the data is partitioned as it moves across the spout and the bolt tasks (grouping).

6.2.4.4 Serving Layer

The Serving Layer is consisted of the following databases:

- **Apache HBase** [16] is an open-source, distributed, scalable, NoSQL big data store built to run on top of the Hadoop Distributed File System (HDFS). A column-oriented key/value data store that can host large tables – with billions of rows and millions of columns – and can provide real-time, random read/write access to Hadoop data. As opposed to other NoSQL databases, like Cassandra, HBase is architected to have strongly consistent reads and writes. Once a write has been performed, all read requests for that data return the same value. It is failure tolerant since it supports replicating data across different datacenters.
- **Apache Cassandra** [9] is an open-source, distributed, wide column store, NoSQL database management system that provides high availability and low latency. It has been proven fault-tolerant on commodity hardware or cloud infrastructure, due to the automatic replication of data to multiple nodes. Cassandra supports replication across multiple datacenters, while failed nodes can be replaced with no downtime.

- **Apache Druid** [56] is a column-oriented, open-source, distributed database and provides fast analytical queries, at high concurrency, on event-driven data. Druid is designed to quickly ingest massive quantities of event data, and provide low-latency queries on top of the data [91]. Druid can instantaneously ingest streaming data and provide sub-second queries to power interactive User Interfaces (UI). It is commonly used in business intelligence applications to analyze high volumes of real-time and historical data.

7 ALGORITHMIC APPARATUS & IMPLEMENTATIONS

In the previous chapter, we presented a generic architectural framework capable of serving a smart city's needs both within various vertical city sectors (transportation, resource and environmental management) and across such sectors for global smart city governance purposes. We also detailed a materialized view of the proposed architecture elaborating on technologies and platforms suitable for satisfying Requirements 1 to 6 as described upon referring to the technological challenges of a smart city environment in Chapter 5.4.

In this chapter, we briefly describe and provide references to algorithms and algorithmic suites that are generic enough to accommodate broad categories of analytics tasks useful within a smart city context. In other words, the basic concept is that having presented a generic architectural framework for big smart city data analysis and its materialization by state-of-the-art technological platforms, this section points to useful, commonly used algorithms that can be implemented within such platforms, in various parts of the proposed architecture, so that they can be utilized while performing analytics tasks. Although throughout our previous discussion we referred to some of such algorithms, this section provides a concise summary of relevant works.

In this point, it should be clarified that our goal is not to provide an exhaustive description of the respective algorithms, but instead to tip on the basic rationale behind their functionality and point potential adopters of the proposed architecture to respective algorithmic suite implementations.

7.1 In-situ Processing Algorithms & Implementations

The idea of in-situ processing is to install local filters at the various sites (i.e., sensors, smart devices or vertical smart city sectors' clusters upon performing global analytics) so that data communication is avoided unless it is absolutely necessary. We consider for instance, a water management scenario and assume we want to monitor the variance among the water levels in various neighboring water tanks and respective interconnecting pipes. The global task we may want to accomplish involves whether a variance function defined on the union of streams gathered by relevant sensors exceeds a threshold, i.e., $Var(\cup SensorStreams)$>T. The usefulness of such a monitoring task is to detect anomalies in the water flow and pinpoint potential points of leak. A key technique for monitoring such generic, complex, non-linear functions is the **geometric monitoring method** [74]. The key idea behind the geometric monitoring technique is that instead of monitoring the function value itself, the monitoring protocol tracks the potential position of the function input vector (formed by ∪SensorStreams in the example) in the input domain of the function (variance in the above example). Every site in the network accumulates local streams forming a local vector of observations. Then each site uses the aforementioned vector as a diameter around which a multi-dimensional sphere is drawn. If any site in the network finds its local sphere entering the area of the input domain of the function where the function crosses the threshold, all sites are notified to communicate their data. Otherwise, no communication is necessary. The whole rationale is depicted in Figure 28, where the gray shaded area depicts the area of the input domain where the function crosses the threshold. The individual vectors e+Dvi depict local stream evolution in each site (5 sites exist in this example), while the 3-d balls involve the corresponding spheres that are inscribed using e+Dvi in each diameter. The yellow colored area of the input domain is the convex hull induced by the corresponding e+Dvi vectors. Since in this particular example no multidimensional sphere crosses the threshold, no communication is necessary.

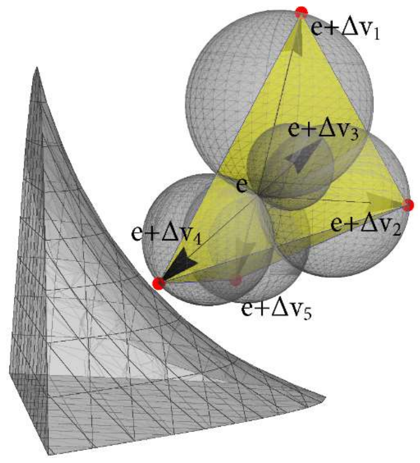


Figure 28: Geometric Monitoring method rationale introduced by [73]

Moreover, the technique of [73] combines in-situ processing with data summarization by executing the aforementioned geometric monitoring scheme only to a carefully-crafted sample of sites in the network. The latter simultaneously addresses vertical and federated scalability challenges as outlined in Chapter 5.4.

A second generic algorithm for in-situ processing especially in event-based analytic tasks regards the **push-pull** rationale [61]. To elaborate this technique, we consider the air quality monitoring scenario, in which we monitor the concentration of gases, such as SO_2, NO, NO_2, CO_2, O_3. In this context, we assume there is an analytics task, which says "trigger an alert when ***E1***: the CO_2 concentration exceeds a given threshold, this event is followed by (SEQUENCE) ***E2***: O_3 exceeds a different threshold AND ***E3***: SO_2, NO_2 concentrations follow similar trends, within a predefined time window *W*". Such a query can be expressed using logical operators and monitored functions as follows: AND (E3, SEQUENCE(E1,E2)). However, the query will never return TRUE, thus no alert will be provided unless all three events occur simultaneously within the window W. In order to reduce communication in such a case, where query operators (such as AND, SEQUENCE) require all of their inputs to output something, we can use the push-pull rationale. According to the push-pull mechanism, the transmission of frequent events is rendered conditional upon the occurrence of rare ones. So for instance, if E1 is a frequent event, but E2 and E3 are not, E1 will not be transmitted until E2 and E3 occur. In other words, E1 will be set in pull mode and it will be cached at the sensors, where it is detected, until a request for transmitting such an event to the respective sensors-sites is received. At the same time, E2 and E3 will be set in push mode, meaning that they will be transmitted immediately towards the query source as soon as they occur. The work of [61] provides a description of the push-pull mechanism and comments on implementation aspects over big data platforms (see Figure 29), while the geometric method and the push-pull rationale is implemented in the FERARI project [37].

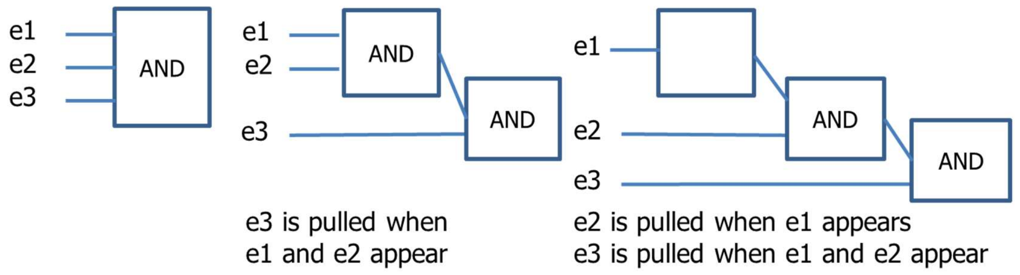

e1
e2
e3
AND
e1
e2
AND
e3
AND
e3 is pulled when e1 and e2 appear
e1
e2
AND
e3
AND
e2 is pulled when e1 appears
e3 is pulled when e1 and e2 appear

Figure 29: Example of different push-pull application for an AND operator receiving 3 events (e1, e2, e3) at input [61]

7.2 In-network Processing Algorithms & Implementations

A recent survey of in-network processing algorithms included in [61, 70] explains the idea behind all these algorithms, which is to map a query graph expressing the workflow of analytics tasks that need to be carried out to the physical network of sites (i.e., sensors, clusters, smart devices). Eventually, the execution of each operator will be assigned to a site that is near to the data sources. In that, data is aggregated early during the execution of parts of the analytics procedures and the total amount of communicated data is reduced because only compact, aggregated pictures of the data are further communicated in the network, towards the query source. The work of [61] proposes respective algorithms that aim at reducing the communication cost in the network, while they harness network latency. To accomplish the task of the aforementioned work, which essentially describes the algorithms employed in the FERARI optimizer discussed in the previous chapter, it proposes exhaustive, dynamic programming, greedy and heuristic algorithms for assigning operator execution to sites. These algorithms differ in their computational complexity as well as on their ability to come up with appropriate operator to site mappings in case of strict network latency constraints. Moreover, the implementation of the respective optimizer [37] couples in-situ with in-network processing. Although the optimization techniques are tailored for event processing applications, in principle, the proposed techniques can accommodate any analytics operator upon being stripped of from the push-pull, in-situ processing rationale.

7.3 Data Summarization Techniques & Implementations

Data synopsis techniques can be useful in the Smart City context especially for efficiently dealing with Requirements 1 to 3 described in Chapter 5.4. Below we outline existing implementations of data summarization algorithms and underline their utility in the scenarios presented in previous chapters. The literature regarding data summarization techniques is abundant. Therefore, instead of citing each related work separately, we refer to the works of [23, 67], which provide concise and comparative views of relevant algorithms.

Stream-lib [83] is a Java-based data synopses library. It includes algorithms for various useful analytics operations such as cardinality; set membership; top-k elements and frequency estimation, along with references to respective research works. Stream-lib supports stream processing in an online fashion, however, the provided implementation does not address horizontal scalability since it does not include provisions for parallel maintenance of single synopses or maintenance of multiple synopses.

Yahoo DataSketch [11] is another data summarization library, which includes implementations of sampling, count, frequency, cardinality and quantile estimation algorithms [10]. The provided implementations include Hive and Pig, but as is the case with Stream-lib, DataSketch is detached from parallelization aspects.

Proteus-backend [81] is a backend module that implements incremental version (~O(1) computational cost using approximations) of most common analytics operations. This module is implemented on Apache Flink and is perhaps the only one that combines the potential of data summarization with parallel processing over big data platforms.

SnappyData [167] is a high performance in-memory data platform for mixed workload applications, built on Apache Spark Streaming. The latter means that it employs the concept of micro-batches instead of being a purely streaming synopses data engine. SnappyData provides a unified programming model for streaming, transactions, machine learning and SQL Analytics in a single cluster. SnappyData's Synopses Data Engine offers a novel and scalable system able to analyze large data sets. It relies on two methods for approximations: (a) stratified sampling and (b) sketching. The former allows users to specify the common dimensions used for querying and ensures that each dimension or stratum with low representation will be captured in the sampled data set. The latter is used to capture data frequencies in a stream, i.e., how often an element is seen in a stream.

Individual repositories with a reach set of data stream summarization techniques, but deprived from parallelization aspects, are also provided open source [84, 78].

The following Table 5 lists some popular data summarization techniques implemented in the above referenced libraries along with their utility for Smart City sectors.

Table 5: Exemplary Data Summarization Techniques and their Utility in the Smart City Context

Synopsis	Estimates	Useful for Smart City Sector
Count-Min Sketch	Count, Frequency	Smart Transportation
HyperLogLog Sketch	Distinct Count	Smart Transportation
LSH	Similarity	Environmental Monitoring, Resource Management
Reservoir Sampling	Cardinality	All
Bloom Filter	Set Membership	Smart Transportation
Qdigest	Approximate quantiles	Environmental Monitoring, Resource Management

Sampling algorithms such as **Reservoir sampling** cited in Table 5 are useful across all smart city sectors examined in this thesis. The idea of maintaining a fair sample of measured quantities in traffic monitoring, water flow or environmental sensing scenarios is to reduce the amount of data that will be processed by the rest of the components of the proposed architecture. At the same time, it will be assured that the constructed samples preserve important statistical properties of the summarized data such as specific moments and cumulants. Cumulants refer to as a set of quantities that provide an alternative to the moments of the distribution [177].

The **Count-Min sketch summary** is designed so that it provides approximate count, frequency estimations over voluminous streaming data, with reduced memory requirements. In the traffic monitoring scenario, this summary can be used in order to maintain approximate counts of the number of vehicles that passed through each edge of the underlying road network. Road networks of large city centers consist of thousands or millions of edges (road segments) and thus, maintaining the exact counts per edge in memory as time passes may be quite a burden. Online and continuous queries over such streams require real-time answers, storing these data in HDFS or other permanent storage means. Counting the number of vehicles that pass through a road segment is quite useful for achieving the relevant business objectives mentioned in Chapter 4. For instance, such counts and their evolution over time can be used initially by data mining algorithms to cluster similar road edges together in order to detect afterwards relations among them and observe how the traffic flows.

Additionally, **Bloom filters** may also be of the essence for the smart traffic monitoring scenario. Bloom filters are a compact data summary capable of providing approximate answers with respect to intersections and union of sets. We consider that apart from the number of vehicles that pass through a road segment, information regarding vehicle IDs, such as the license plate, is also available. Thus, for each edge of the network we can maintain a set of vehicle IDs that used the respective edge while commuting. In order to monitor how the traffic flows among road edges similarity measures, such as the Jaccard Similarity of pairs of edges can be used. This technique provides the ratio of cardinality of the intersection over the cardinality of the union of vehicle id sets maintained per edge. Computing the size of the intersection of vehicle sets in edges is as simple as performing a logical conjunction (AND operation) among the respective Bloom filters maintained per edge. However, a logical disjunction (OR operation) is used for the cardinality of union of such sets.

The **HyperLogLog sketch** is a structure capable of providing approximate query answers regarding distinct count queries with reduced memory consumption. The idea of using such a structure in traffic monitoring scenarios is to extract information regarding the number of **distinct** vehicles that move in the city road network or parts of it. Such a piece of information can be useful for various reasons. For instance, similarly to the aforementioned examples, judging the similarity among the traffic of road segments according to Sorensen-Dice similarity index requires computing the intersection of the corresponding sets of vehicle IDs (e.g. using a Bloom filter) over the cardinalities of sets of (distinct) vehicles passing by each edge (one HyperLogLog sketch per road segment).

Taking into consideration the number of sensors utilized in resource management and environmental monitoring scenarios along with the number of streams generated by each sensor, executing analytics tasks such as anomaly or outlier detection becomes a challenge. Recent techniques for judging outliers [72, 76, 75] have pointed out the importance of judging outliers in such networks based on the similarity of the values produced by each sensor compared to other sensors. In other words, to extract outliers in sensor network settings over environmental or resource related data often regards quantifying pairwise correlations of streams originating from different sensors. The computational complexity of computing such pairwise correlations is quadratic to the number of input streams, thus the targeted smart city use cases may operate over tens of thousands of streams. The usefulness of **Locality Sensitive**

Hashing (LSH) techniques in these scenarios is two-fold. First, they compress sensor measurements so that communication towards the query source is reduced. Second, the Hamming weight (number of set bits) in the resulted LSH bitmaps can be used as a hash key based on which the processing of entire series of sensor readings are distributed among different machines within a cluster, simultaneously pruning the number of performed pairwise comparisons by hashing highly dissimilar bitmaps to different machines [76].

Finally, **Qdigest** is a data summarization technique that can provide approximate answers to quantile queries. The use of quantiles/percentiles is useful in both environmental monitoring and resource management scenarios. Indicatively, England has water quality consent limits that are based on the 90th and 95th percentiles of monitoring data not exceeding specified levels. The U.S. EPA has specifications for air quality monitoring that are, in effect, percentile limitations. These may, for example, specify that the ambient concentration of a compound cannot exceed a predefined percentile during the day. The U.S. EPA has provided guidance for setting aquatic standards on toxic chemicals that require estimating 99th percentiles and using this statistic to make important decisions about monitoring and compliance. They have also used the 99th percentile to establish maximum daily limits for industrial effluents [39].

7.4 Machine Learning and Data Mining Algorithms and Libraries

FlinkML [12] is a set of scalable ML algorithms and API adopted by Flink distributed framework. It contains algorithms for:

- supervised (SVM CoCoA, Multiple Linear Regression, Batch Gradient Descent) and unsupervised (k-Nearest Neighbors) learning,
- data pre-processing (Polynomial Features, Standard Scaler, MinMax Scaler), recommendation (Alternative Least Squares, Stochastic Outlier Selection) and
- other ML utilities involving (a) the computation of distance metrics such as Euclidean, squared Euclidean, Chebyshev, Cosine Similarity, Manhattan and Minkowsky distance as well as Tanimoto coefficient calculation, (b) cross validation methods such as Train-Test-(Holdout) splits, K-Fold splits, Multi-random Splits etc.

Most of the incorporated algorithms are destined for batch processing purposes and as such, they constitute offline algorithms built on top of the batch API of Flink. For the orchestration of machine learning tasks that run over distributed, parallel architectures ensuring horizontal scalability via parallel and elastic processing, an implementation of the parameter server paradigm [1] is also available [80]. Finally, FlinkML includes a connector to the SAMOA library discussed below.

Apache SAMOA [20] is a distributed streaming ML framework built for Apache Storm, Flink and Samza [21]. SAMOA supports Prequential Evaluation Task (which is an online ML model evaluation pipeline), Vertical Hoeffding Tree Classifiers, Adaptive Model Rules Regressors, Bagging and Boosting, Distributed Stream Clustering and Distributed Frequent Itemset Mining analytics.

Proteus-SOLMA [82] is a scalable set of online machine learning algorithms including classification, clustering, regression, ensemble, graph oriented, linear algebra operations, anomaly and novelty detection algorithms. It is implemented on top of Apache Flink using the batch and stream processing capabilities. SOLMA supports the computation of online moments and it, also includes a number of summarization techniques involving online sampling and incremental Principal Component Analysis (PCA), among others. Regarding classification algorithms, it includes Online Support Vector Machines (OSVM) for batch learning (Pegasos, Norma algorithms), Online bi-level stochastic gradient for SVMs (OBSG-SVM) and Online Passive Aggressive Classifier (PA). With respect to regression methods, it mainly incorporates Online Ridge Regression with L2 regularization (ORR), Online shrinkage via limit of Gibbs sampling with L1 regularization (SLOG), Aggregated Algorithm for Regression (ARR), Competitive Online Iterated Ridge Regression (COIRR).

Spark's MLlib [22] is perhaps the richest ML API implementation over a popular big data platform. It is mainly destined to support batch processing, included only limited support for online ML tasks. More precisely, MLlib supports the computation of basic statistics and measures such as pairwise correlations via Pearson's and Spearman's correlation coefficients, hypothesis testing via Pearson's Chi-Squared test. It further provides numerous extracting, transforming and selecting features and ML tuning facilities such as hyperparameter tuning, cross-validation and train-validation split. Indicatively incorporate algorithms include:

- Classification algorithms
  - Logistic Regression
  - Decision Trees and Random Forest
  - Gradient-boosted tree
  - Multilayer Perceptron
  - Linear SVM
  - One-vs-Rest
  - Naive Bayes
- Regression algorithms
  - Linear regression
  - Generalized Linear Regression
  - Decision Tree and Random Forest regression
  - Gradient-boosted tree regression
  - Survival regression
  - Isotonic regression
- Clustering algorithms
  - K-means
  - Latent Dirichlet Allocation (LDA)
  - Bisecting Mixture Model (BMM)
  - Gaussian Mixture Model (GMM)

Clustering techniques can be proven useful across various smart city vertical sectors. At this point, we exploit smart traffic monitoring scenarios since they constitute an interesting example for utilizing clustering techniques. As already discussed in Chapter 7.3, in order to observe how the traffic flows among road segments, we may employ information regarding the counts of vehicles per road segment as well as its evolution over time. According to the literature [132], hierarchical clustering on road network edges is performed using different similarity measures in order to detect traffic flow relationships, including traffic propagation (Figure 30), traffic split (Figure 31) and traffic merge (Figure 32). More precisely, the notion of distance between two road segments (edges) is defined (a) based on the corresponding traffic series count values, (b) based on the corresponding traffic series "shape" (normalized count vectors) and (c) according to the network graph topology. Then, a clustering method is employed that utilizes the different distance types to first cluster together edges of similar time series shape, then these clusters are further split based on the similarity of time series count values and finally based on their spatial proximity. The result of the clustering can be visualized, i.e., the technique leaves to the human operator the task of detecting traffic propagations, splits or merges, to illustrate the extracted, undergoing traffic flow as shown in Figure 30, Figure 31 and Figure 32.

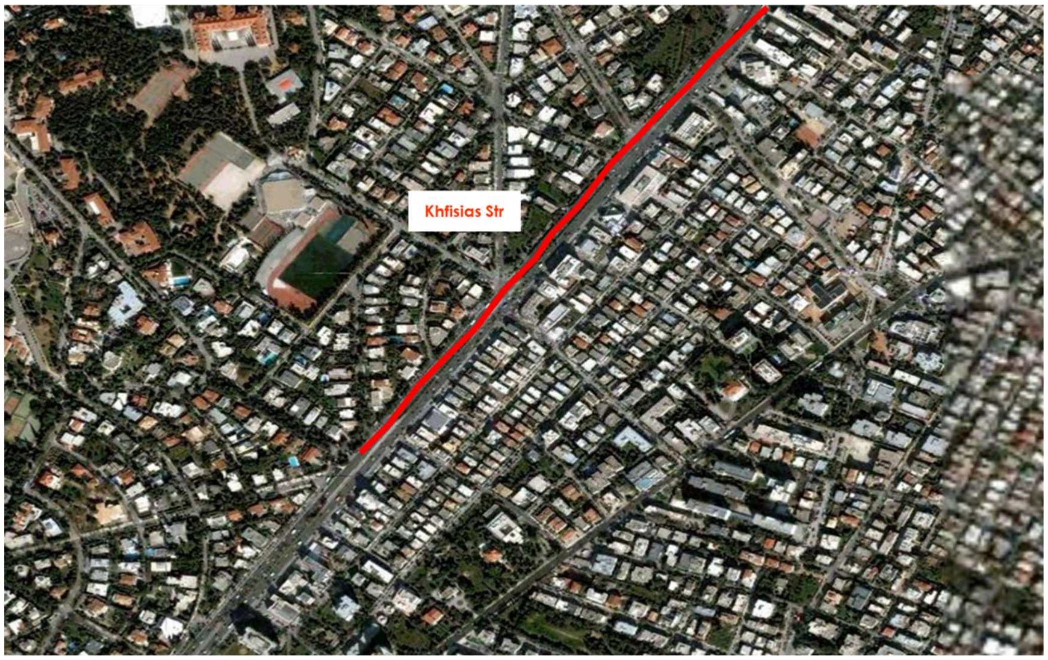


Figure 30: Clustering output depicting traffic propagation relationship [132]

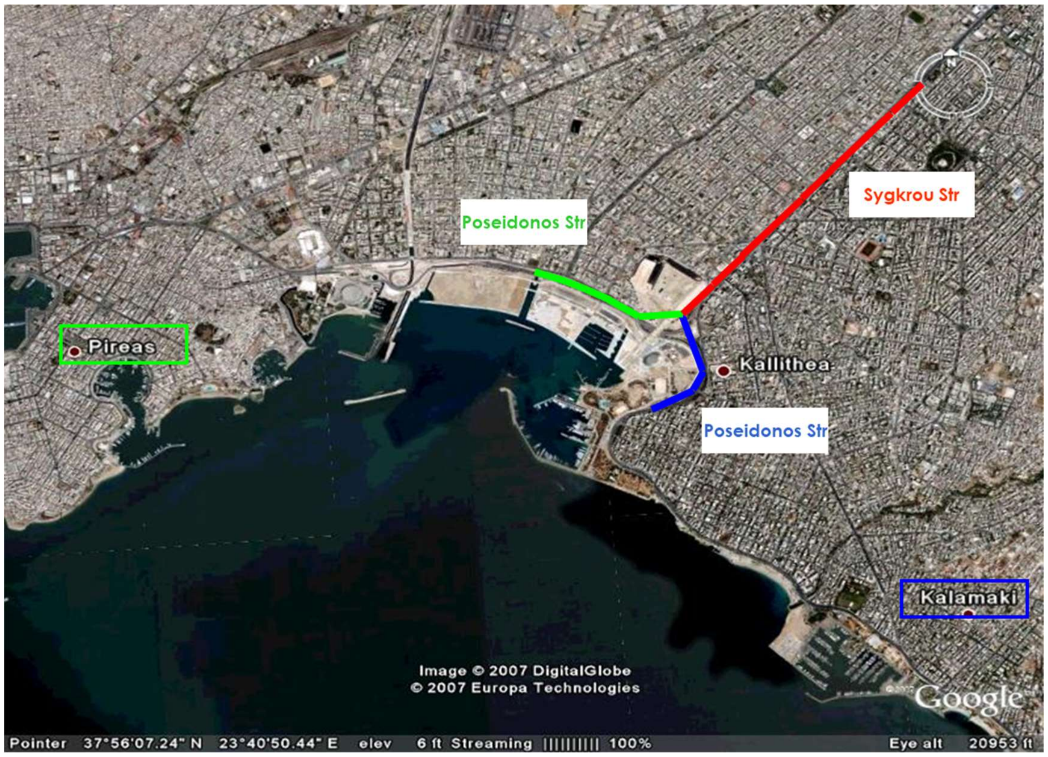

Sygkrou Str
Poseidonos Str
Pireas
Kallithea
Poseidonos Str
Kalamaki
Image © 2007 DigitalGlobe
© 2007 Europa Technologies
Google
Pointer 37°56'07.24" N 23°40'50.44" E elev 6 ft Streaming |||||||||| 100% Eye alt 20953 ft

Figure 31: Clustering output depicting traffic split relationship [132]

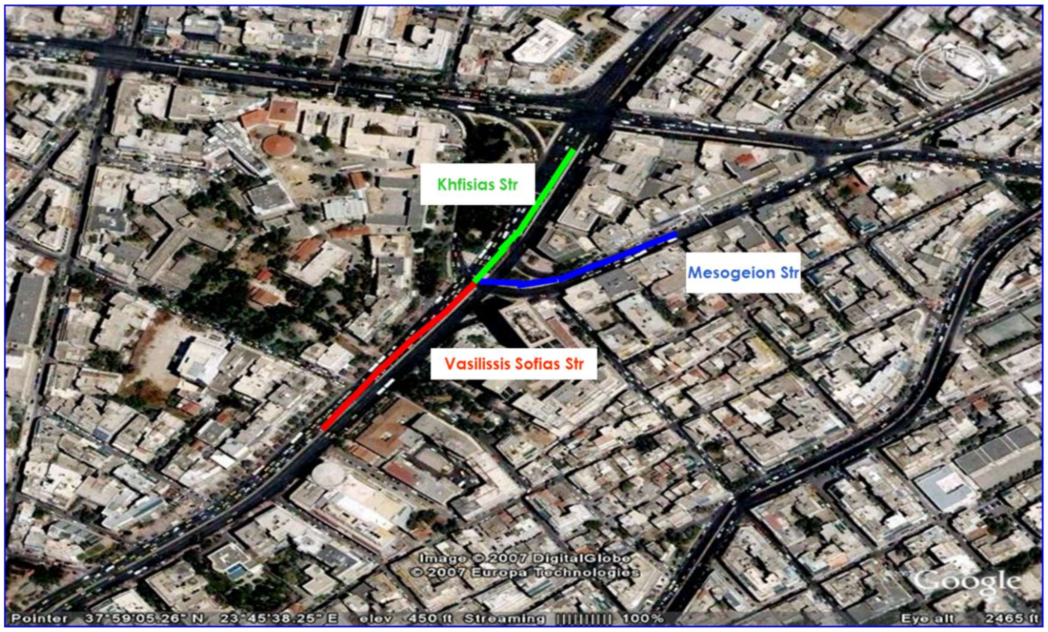

Khfisias Str
Mesogeion Str
Vasilissis Sofias Str
Image © 2007 DigitalGlobe
© 2007 Europa Technologies
Google
Pointer 37°59'05.26" N 23°45'38.25" E elev 450 ft Streaming |||||||||| 100% Eye alt 2465 ft

Figure 32: Clustering output depicting traffic merge relationship [132]

Classification techniques such as Random Forests can be used for air quality prediction purposes. Relevant data analysis may include air monitoring station data, meteorology data or even road traffic information. All these data are fetched at certain intervals (see Table 4). The city can be divided into a grid and each grid cell is regarded as one region. Those grid cells with air quality monitoring stations generate the data used for training a learning model (a number of decision trees in the case of random forests). Predictions should be made on the streaming data for characterizing the level of hazard as regards the concentration of certain pollutants in the near future. The Random Forest is a general term for methods using decision tree-type classifiers [69]. It uses recursive partitioning to generate many trees and then aggregates the results (Figure 33) [165]. Each tree is independently constructed using a bootstrap sample of the training data, which subdivides the parameter set first into several parts depending on one of the parameters, and subsequently repeats the process for each part. After all the trees are constructed, the streaming data are input into all decision trees. For each tree, a probability is estimated for the label (e.g. hazardous, non-hazardous, likely hazardous, etc.) of the predicted air quality. The final probability of each label (class) is computed by combining e.g., the majority voting among the highest probability labels across the trees of the forest.

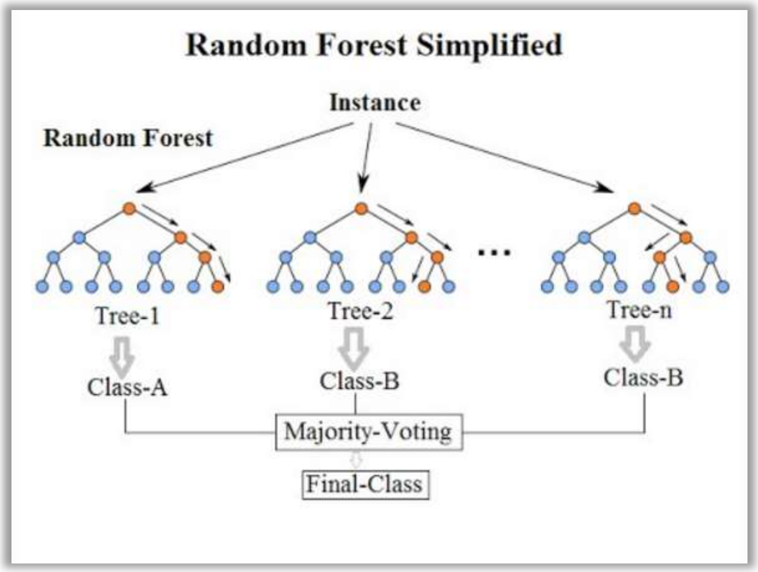


Figure 33: Random Forest Algorithm graph

SVM (Support Vector Machine) algorithms aim at separating data points in multidimensional spaces, so that they are as far apart as possible. These algorithms find a hyperplane (acting as a so-called soft margin) by selecting the most appropriate points in a training subset (the supporting vectors), that split data into categories (see Figure 34 [101]). When a new data point, e.g. readings from air quality monitoring stations, streams into the architecture, it is assigned to a category that gives the corresponding label to it.

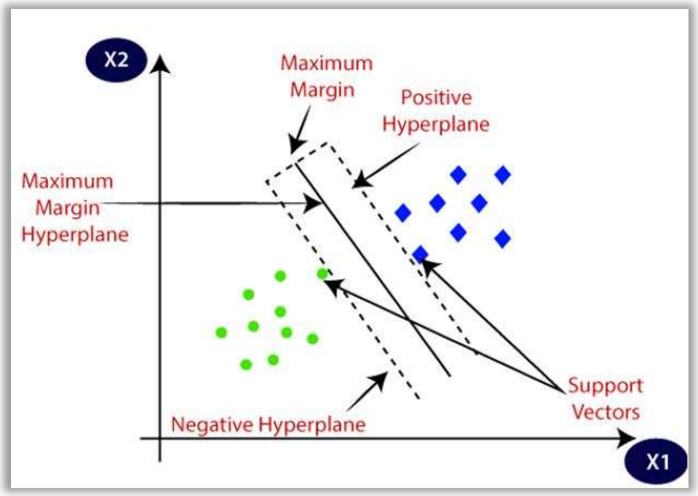


Figure 34: Support Vector Machine Algorithm

As a forecasting method, **k-Nearest Neighbor (kNN)** algorithm is a supervised machine learning technique that seeks to identify the past sequence, in time series, that is most similar to the one to be forecasted. It can be used to make the waste collection process more efficient by monitoring the occupancy level of garbage bins and calculating the travel distance in order to clear all the filled bins from trash. The sensor in each bin measures the level of garbage, while GPS data stemming from a track mark its current position. The idea is to instruct, in advance, a proper route for a track based on the current and expected garbage levels of bins. The kNN algorithm measures the distance between a query point (current status/capacity of a bin) and a set of scenarios in past data (see for example Figure 35 [79]). A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its *k* nearest neighbors measured by a distance function. Therefore, one can predict the bins that will be filled in the near future and proactively instruct the proper trajectories for garbage collecting trucks, in real-time as they commute.

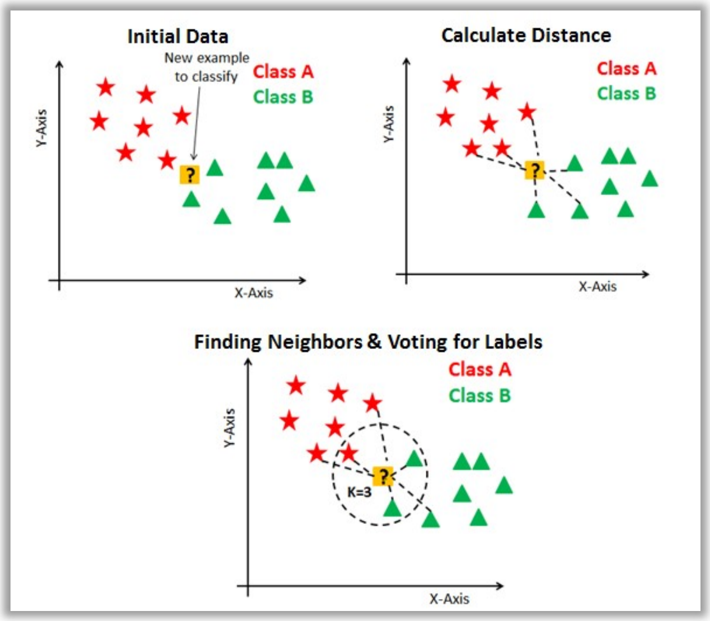


Figure 35: k-Nearest Neighbors (kNN) Algorithm

8 CASE STUDY: HERAKLION SMART CITY

In the previous chapters, we presented a holistic view of leveraging big data in smart cities setups. Starting initially from identifying the needs of divergent city authorities, translating these to business objectives and key performance indicators and realizing them to distinct requirements, we finally elaborated prevalent big data platforms to address these requirements. This chapter aims to exemplify the aforementioned analysis into a real smart city context and deploy big data platform topologies in an effort to facilitate the stakeholders to materialize more easily big data analysis in every day setups for different interdisciplinary scenarios.

To that end, we exploit the business logic of Heraklion Smart City as an indicative example to leverage generated big data aiming to extract valuable information. In the following analysis we introduce the needs of Heraklion city and we translate these needs into specific requirements using big data analytics tools in order to alleviate existing knowledge to a more powerful decision making process.

8.1 Identifying the needs of Heraklion city

Heraklion has embedded Information and Communication Technologies in many different infrastructures aiming to enhance the quality life of citizens and facilitate already existing services towards solving administrative issues. Moreover, the city focuses on ensuring a networked society that enjoys the benefits of intelligent management of city issues. Heraklion Smart City approaches, among others, the following domains: (a) environmental and weather monitoring, (b) air quality monitoring, (c) water quality and management monitoring, and (d) smart parking.

As stated in [92], the Municipality of Heraklion envisions a smart city able to address the following requirements:

A. Environmental and weather monitoring

- Optimization of the cooling, heating and lighting systems of buildings to achieve energy savings rates.
- Decrease of carbon dioxide emissions by actuating citizens to integrate cycling and walking in their everyday transportation within the city
- Integration of a dynamic system that records the variations of quantitative and qualitative waste production.
- Enhancement of the waste collection services through proper control and management of waste levels in smart bins.
- Social protection notifications regarding earthquakes, fires, extreme weather conditions and other. Towards that direction, social media can be also proven a useful communication tool to inform citizens in time about upcoming threatening events.

B. Air quality monitoring

Crete is heavily affected by air pollution due to high urbanization rates, seasonal tourism and excessive use of fireplaces in combination with high. Moreover, the dust transport from the deserts of North Africa and the Middle East, urban and industrial pollutants in Europe and Turkey and forest fires deteriorate air quality and affect the health of the public. In that context, the Municipality aims to:

- Control the greenhouse gas emissions and estimation of the carbon footprint in order to identify the areas that need of special attention and require proper interventions.
- Improve the microclimate of the densely built up city and reduce the energy consumption of the buildings through bioclimatic repairs of settlements and open spaces increasing green areas.

C. Water management

The region of Crete suffers often from lack of enough water resources due to the dry climate and towards utilizing the available resources efficiently aims at:

- Reducing water consumption of both citizens and businesses through appropriate actions and initiatives for the rational management of the existing water resources

D. Smart mobility and smart parking

Heraklion is a city that suffers from the outnumbered vehicles that search for a parking place, especially in the city center. It has been noticed that early in the mornings, public-parking garages and terrains have reached their parking occupancy. At the same time, searching for a parking space often provokes bottlenecks in the narrow streets of Heraklion, while traveling around the city. Due to the limited number of public garages, car drivers most often use on-road parking lots, taking the risk of occupying non-parking zones, as well. The latter constitutes the reason of the increased number of tickets issued from the municipal police for illegal parking the past years. Thus, the Municipality envisions to approach the following:

- Real-time traffic management improvements for intra-community and inter-municipal transportation with the assistance of modern intelligent technologies (web and mobile applications, etc.)
- Combining ICTs to monitor environmental and traffic issues
    - Pilot scheme of installing urban ICT equipment (signs, benches, lighting, shelters, bus stops, information and/or selling kiosks, environmental objects) while integrating monitoring systems (e.g. atmospheric quality, light pollution, noise, traffic, Wi-Fi, etc.)
    - Traffic monitoring on streets through sensor installations on the city buses
    - Traffic monitoring system using various sensors or through citizens' mobile phones
- Deploying cooperative transport systems on freight services.
- Development of an intelligent parking system providing real-time information for pricing and parking availability.
- Integrating smart parking monitoring system, informing authorities whenever a prohibited area is occupied by a vehicle.

8.2 The IoT infrastructure

The ecosystem adapted by Heraklion Smart city is based on three main components (Figure 36): (a) the IoT Devices, (b) the Open Data Middleware Layer and (c) the Visualization Layer [103].

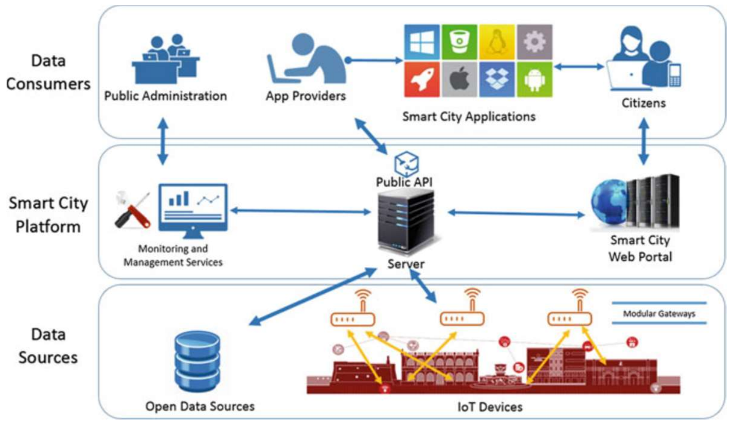


Figure 36: The Heraklion Smart City ecosystem [103]

8.3 The WSN architecture

The aim of the Heraklion Smart City initiative is to confirm whether an IoT infrastructure could be proven valuable to citizens and city authorities. A pilot Wireless Sensor Network has been deployed to aggregate specific information covering the previously presented requirements. According to [103], the necessary sensing infrastructure has been installed in dispersed spots around the city, including:

i. Water quality sensors installed in the city's water reservoirs.
ii. Parking sensors installed on the road surface. They provide at a programmable interval, a status for the occupation of the parking space, based on the field strength compared to a predefined threshold based on the measurement of the magnetic field strength in three orthogonal axes.

Table 6: Measured parameters per area of interest

Area of interest	Parameters
Environmental and weather monitoring	Ambient temperature, ambient humidity, atmospheric pressure, wind (speed and direction), dust, VOC (volatile organic compounds), noise level, luminosity
Air quality monitoring	Gases concentrations (SO_2, NO, NO_2, CO_2, O_3)
Water quality	Water quality and quantity (water level, water temperature, conductivity, PH and ions {NH_4+, NO_3-, $Cl-$ and Ca_2+})
Smart parking	Number of occupied illegal parking spots

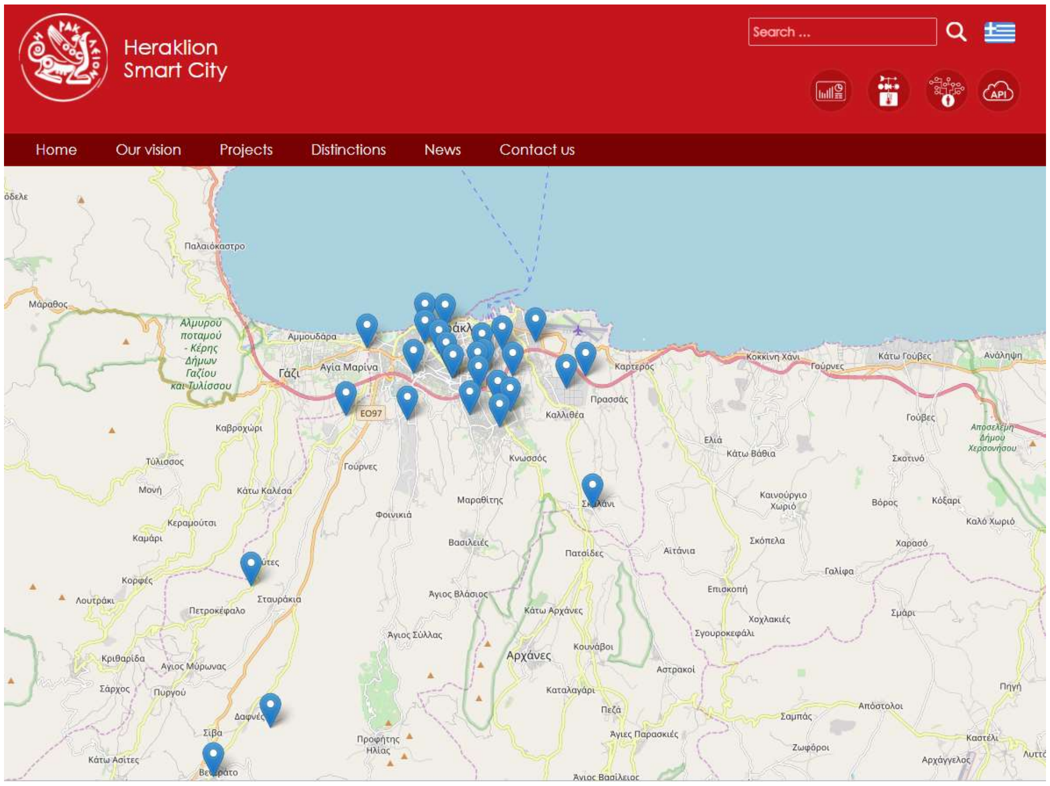
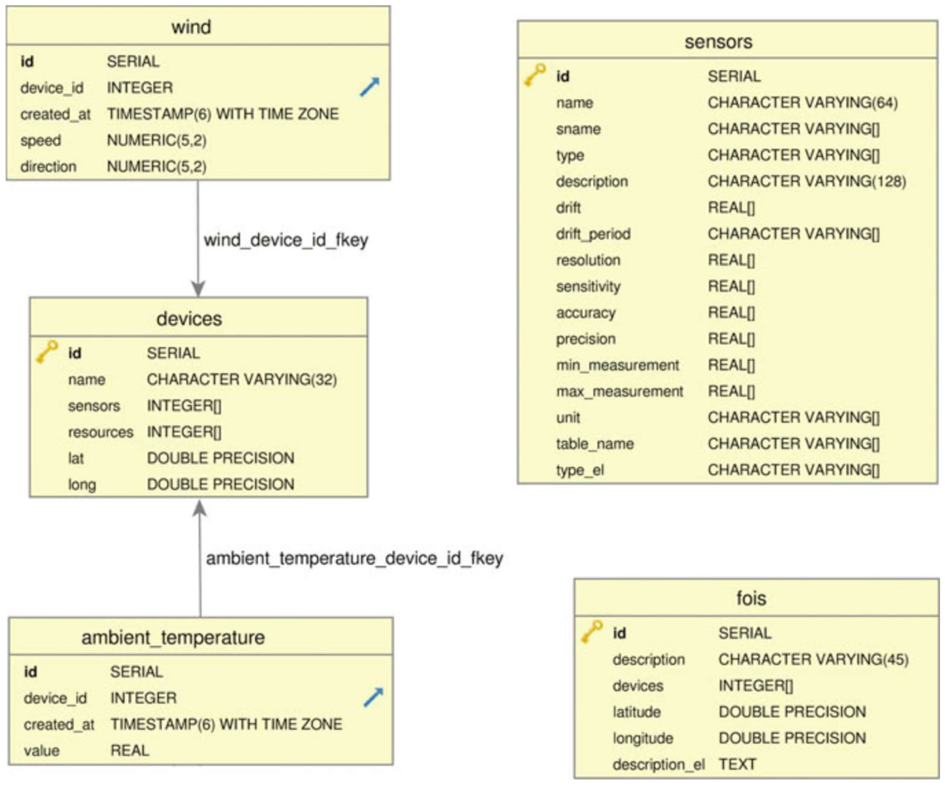


Figure 37: Pilot sensor infrastructure in Heraklion city (Source: www.smartcity.heraklion.gr) [103]



wind	
id	SERIAL
device_id	INTEGER
created_at	TIMESTAMP(6) WITH TIME ZONE
speed	NUMERIC(5,2)
direction	NUMERIC(5,2)

devices	
id	SERIAL
name	CHARACTER VARYING(32)
sensors	INTEGER[]
resources	INTEGER[]
lat	DOUBLE PRECISION
long	DOUBLE PRECISION

ambient_temperature	
id	SERIAL
device_id	INTEGER
created_at	TIMESTAMP(6) WITH TIME ZONE
value	REAL

sensors	
id	SERIAL
name	CHARACTER VARYING(64)
sname	CHARACTER VARYING[]
type	CHARACTER VARYING[]
description	CHARACTER VARYING(128)
drift	REAL[]
drift_period	CHARACTER VARYING[]
resolution	REAL[]
sensitivity	REAL[]
accuracy	REAL[]
precision	REAL[]
min_measurement	REAL[]
max_measurement	REAL[]
unit	CHARACTER VARYING[]
table_name	CHARACTER VARYING[]
type_el	CHARACTER VARYING[]

fois	
id	SERIAL
description	CHARACTER VARYING(45)
devices	INTEGER[]
latitude	DOUBLE PRECISION
longitude	DOUBLE PRECISION
description_el	TEXT

Figure 38: A database schema approach

8.4 Data processing pipelines for Heraklion Smart City

As illustrated in Figure 37, an indicative number of sensors have been integrated in Heraklion city's infrastructure. A fully integrated IoT-based network comprising thousands of sensors coupled with the escalation in big data acquisition would require additional means of data processing and analysis. To that end, this chapter aims to propose an architecture based on the big data technologies presented in Chapter 6, regarding the application scenario of Heraklion Smart City for batch and stream processing. The following analysis is based on the database schema illustrated in Figure 38 presented by [103].

8.4.1 Parking management using Apache Storm

A number of parking sensors dispersed in Heraklion city transmit real time data. Sensors are installed in the asphalt in zones where the parking is prohibited (e.g. crosswalk, pedestrian streets, spots where people with disabilities are able to pass through, etc.). Every time the sensors' status is occupied, the appropriate notification is forwarded to the Municipal Police. Data about parking behavior need to be processed at runtime in order to inform stakeholders with the current on-road status. Based on the architecture we presented in Chapter 6.2, we use Apache Storm, a stream processing engine.

In Figure 39, the presented Apache Storm topology constitutes the structure of a distributed computation regarding the parking-monitoring infrastructure of Heraklion Smart City. The presented graph comprises a spout (stream producer) and various bolts (operations) that receive and emit an unbounded sequence of tuples, namely streams. Streams of tuples flow from spouts to bolts through various stream groupings that determine how the tuples are routed in the topology (defined in Chapter 6.2.4.3). Spouts and bolts are connected with edges that indicate which streams are processed by which bolts.

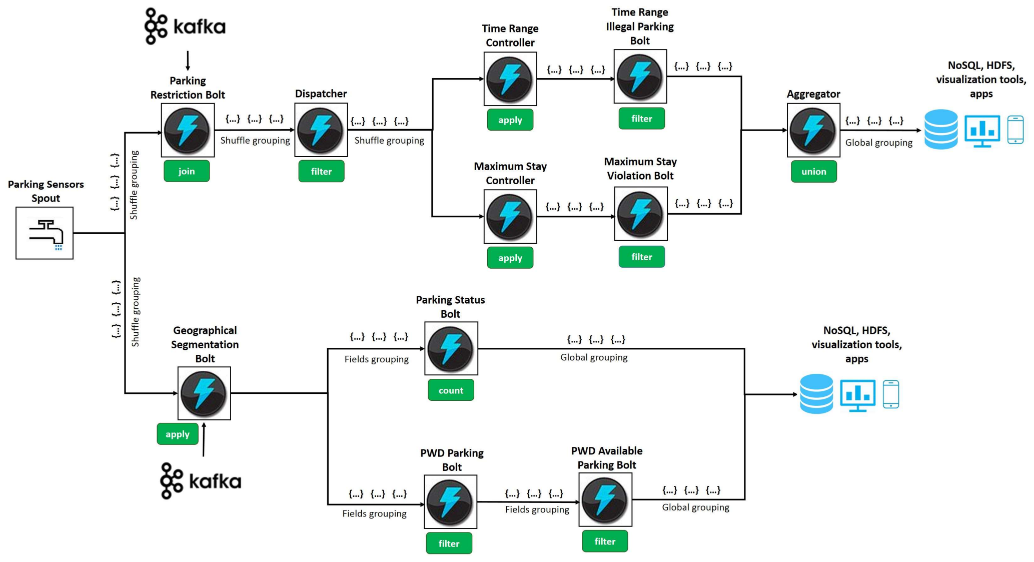
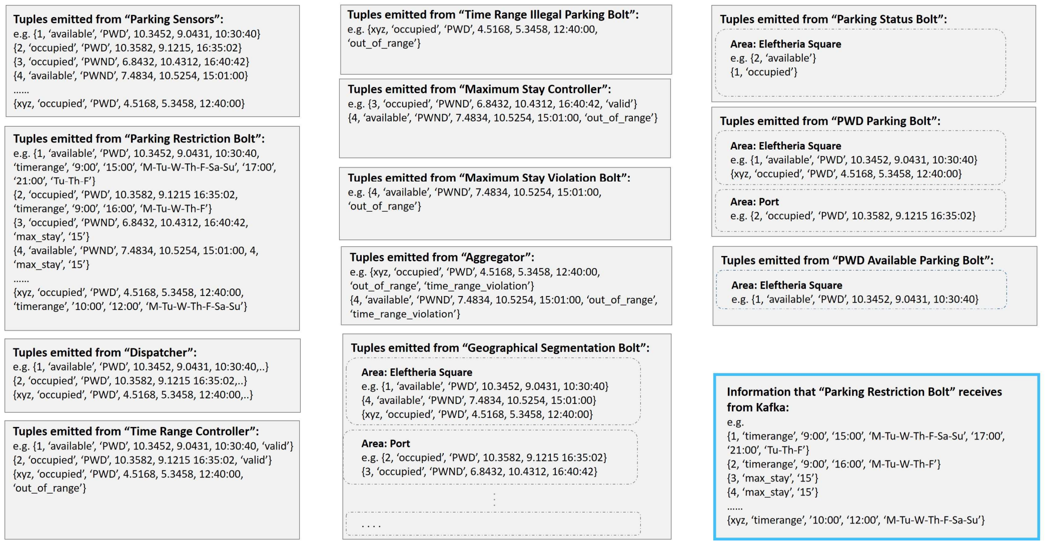


Figure 39: An Apache Storm topology for parking monitoring (logical view)



Tuples emitted from "Parking Sensors":
e.g. {1, 'available', 'PWD', 10.3452, 9.0431, 10:30:40}
{2, 'occupied', 'PWD', 10.3582, 9.1215, 16:35:02}
{3, 'occupied', 'PWND', 6.8432, 10.4312, 16:40:42}
{4, 'available', 'PWND', 7.4834, 10.5254, 15:01:00}
……
{xyz, 'occupied', 'PWD', 4.5168, 5.3458, 12:40:00}

Tuples emitted from "Parking Restriction Bolt":
e.g. {1, 'available', 'PWD', 10.3452, 9.0431, 10:30:40, 'timerange', '9:00', '15:00', 'M-Tu-W-Th-F-Sa-Su', '17:00', '21:00', 'Tu-Th-F'}
{2, 'occupied', 'PWD', 10.3582, 9.1215 16:35:02, 'timerange', '9:00', '16:00', 'M-Tu-W-Th-F'}
{3, 'occupied', 'PWND', 6.8432, 10.4312, 16:40:42, 'max_stay', '15'}
{4, 'available', 'PWND', 7.4834, 10.5254, 15:01:00, 4, 'max_stay', '15'}
……
{xyz, 'occupied', 'PWD', 4.5168, 5.3458, 12:40:00, 'timerange', '10:00', '12:00', 'M-Tu-W-Th-F-Sa-Su'}

Tuples emitted from "Dispatcher":
e.g. {1, 'available', 'PWD', 10.3452, 9.0431, 10:30:40,..}
{2, 'occupied', 'PWD', 10.3582, 9.1215 16:35:02,..}
{xyz, 'occupied', 'PWD', 4.5168, 5.3458, 12:40:00,..}

Tuples emitted from "Time Range Controller":
e.g. {1, 'available', 'PWD', 10.3452, 9.0431, 10:30:40, 'valid'}
{2, 'occupied', 'PWD', 10.3582, 9.1215 16:35:02, 'valid'}
{xyz, 'occupied', 'PWD', 4.5168, 5.3458, 12:40:00, 'out_of_range'}

Tuples emitted from "Time Range Illegal Parking Bolt":
e.g. {xyz, 'occupied', 'PWD', 4.5168, 5.3458, 12:40:00, 'out_of_range'}

Tuples emitted from "Maximum Stay Controller":
e.g. {3, 'occupied', 'PWND', 6.8432, 10.4312, 16:40:42, 'valid'}
{4, 'available', 'PWND', 7.4834, 10.5254, 15:01:00, 'out_of_range'}

Tuples emitted from "Maximum Stay Violation Bolt":
e.g. {4, 'available', 'PWND', 7.4834, 10.5254, 15:01:00, 'out_of_range'}

Tuples emitted from "Aggregator":
e.g. {xyz, 'occupied', 'PWD', 4.5168, 5.3458, 12:40:00, 'out_of_range', 'time_range_violation'}
{4, 'available', 'PWND', 7.4834, 10.5254, 15:01:00, 'out_of_range', 'time_range_violation'}

Tuples emitted from "Geographical Segmentation Bolt":
Area: Eleftheria Square
e.g. {1, 'available', 'PWD', 10.3452, 9.0431, 10:30:40}
{4, 'available', 'PWND', 7.4834, 10.5254, 15:01:00}
{xyz, 'occupied', 'PWD', 4.5168, 5.3458, 12:40:00}
Area: Port
e.g. {2, 'occupied', 'PWD', 10.3582, 9.1215 16:35:02}
{3, 'occupied', 'PWND', 6.8432, 10.4312, 16:40:42}
⋮
. . . .

Tuples emitted from "Parking Status Bolt":
Area: Eleftheria Square
e.g. {2, 'available'}
{1, 'occupied'}

Tuples emitted from "PWD Parking Bolt":
Area: Eleftheria Square
e.g. {1, 'available', 'PWD', 10.3452, 9.0431, 10:30:40}
{xyz, 'occupied', 'PWD', 4.5168, 5.3458, 12:40:00}
Area: Port
e.g. {2, 'occupied', 'PWD', 10.3582, 9.1215 16:35:02}

Tuples emitted from "PWD Available Parking Bolt":
Area: Eleftheria Square
e.g. {1, 'available', 'PWD', 10.3452, 9.0431, 10:30:40}

Information that "Parking Restriction Bolt" receives from Kafka:
e.g.
{1, 'timerange', '9:00', '15:00', 'M-Tu-W-Th-F-Sa-Su', '17:00', '21:00', 'Tu-Th-F'}
{2, 'timerange', '9:00', '16:00', 'M-Tu-W-Th-F'}
{3, 'max_stay', '15'}
{4, 'max_stay', '15'}
……
{xyz, 'timerange', '10:00', '12:00', 'M-Tu-W-Th-F-Sa-Su'}

Figure 40: Tuples emitted throughout the Apache Storm topology

More specific, the parking sensors installed in Heraklion's Smart City infrastructure send data to the "*Parking Sensors Spout"* every 1 minute (see Chapter 5.2). The spout wraps streaming data and emits tuples of key-value pairs in the form of

{id, 'status', 'type', lon, lat, 'timestamp'}

, where *id* is the key and *status, type, lon* (longitude)*, lat* (latitude) and *timestamp* are considered as the value. The *status* parameter can take a value of either *'available'* or *'occupied'* and *the type* parameter can take *'PWD'* or *'PWND'* values, defining whether a parking zone is for People with Disabilities (PWD). The *"Parking Sensors Spout"* replicates the stream of tuples to two different bolts; the first is called *"Parking Restriction* Bolt" and the second *"Geographical Segmentation* Bolt" by applying **shuffle grouping**. In that way, tuples are randomly distributed across both bolts' tasks and each task is guaranteed to receive an equal number of tuples. Both bolts enable the parallelization process for the aggregations of data, allowing multiple workers to process data in parallel.

The *"Parking Restriction Bolt"* reads data from Kafka, regarding the parking zones where parking is allowed for predefined time frames. The tuples have the form of

{id, 'time_descr', 'start_time', 'end_time', 'days_of_week'}

, where the *'time_descr'* parameter takes 'timerange' or *'max_stay'* values. The former refers to the zones where parking is allowed within a specified time range. For instance, there are parking lots in the city center defined as prohibited during shops' opening hours (e.g. 9:00-15:00 and 17:00-21:00 for Tuesdays, Thursdays and Fridays, etc.). The latter refers to the parking zones where vehicles are allowed to stop only for a short period of time (e.g. for 15 minutes). *"Parking Restriction Bolt"* applies a **join** operation where the tuples with the same 'id' are paired together (e.g. {2, 'occupied', 'PWD', 10.3582, 9.1215 16:35:02, 'timerange', '9:00', '16:00', 'M-Tu-W-Th-F'}, {3, 'occupied', 'PWND', 6.8432, 10.4312, 16:40:42, 'max_stay', '15'}, etc.).

These tuples are randomly distributed across the *"Dispatcher"* bolt's tasks and each task is guaranteed to receive an equal number of tuples. The *"Dispatcher"* **filters** the tuples and forwards the ones that apply the following condition to the *"Time Range Controller"* bolt

if {time_descr = 'timerange'}

, whereas it forwards the tuples that satisfy the condition below to the *"Maximum Stay Controller" bolt.*

if {time_descr = 'max_stay'}

The *"Time Range Controller"* is responsible to **apply** a user-defined function in order to monitor if an occupied parking lot applies the specific time range, defined through a Kafka topic. It then transmits tuples to the *"Time Range Illegal Parking Bolt"* through **shuffle grouping** in order to **filter** and forward only the tuples that satisfy the following condition to the *"Aggregator"* bolt:

if {restr_status = 'out_of_range'}

Similarly, the *"Maximum Stay Controller"* is responsible to **apply** a user-defined function in order to monitor if a parking lot that is occupied applies the maximum permitted stay limit, as defined through a Kafka topic. Namely, if the status of the sensor is *'occupied'* a **time window** monitors whether it remains occupied, for the past X consequent minutes. In case the status changes to *'available'* then the *'restr_status'* returns the value *'valid'. However,* if it remains occupied longer than X minutes the *'restr_status'* parameter returns *'out_of_range'.* Afterwards, the bolt transmits tuples to the *"Maximum Stay Violation Bolt"* through **shuffle grouping** in order to **filter** and forward only the tuples that satisfy the following condition to the *"Aggregator"* bolt:

if {restr_status = 'out_of_range'}

Eventually, the "Aggregator" bolt receives inputs from both the *"Time Range Illegal Parking Bolt"* and the *"Maximum Stay Permission Anomaly Bolt"* through a **union** operation. The "Aggregator" emits tuples in the following form and writes them to a NoSQL or HDFS database.

{4, 'available', 'PWND', 7.4834, 10.5254, 15:01:00, 'out_of_range', 'time_range_violation'}

At the same time, a replicate of the input data is transmitted from the *"Parking Sensors Spout"* to the *"Geographical Segmentation Bolt"* that reads from Kafka a grid of defined geographic areas that encompass a number of dispersed sensors around Heraklion city. The *"Geographical Segmentation Bolt"* is responsible to **apply** a user-defined function that correlates which sensors belong to which geographic areas, according to their longitude and latitude values. It maps tuples with the same key, namely the area, through **fields grouping** to the same task and transmits them to the *"Parking Status Bolt"* that **counts** the total number of available and occupied sensors (e.g. {count, 'status'}). Then, it groups all the streams together with a **global grouping** and writes them to a NoSQL or HDFS database. In that way, stakeholders will be able to retrieve information about the total number of available and occupied parking places through the appropriate visualization tool or mobile application.

Finally, a second stream of data through **fields grouping** maps the tuples with the same geographic area across *"PWD Parking Bolt"* 's tasks, which **filters** the tuples according to the condition below and groups them together.

```
if {type = 'PWD'}
```

Finally, it transmits data to the *"PWD Available Parking Bolt"*, which provides information about the available parking places for people with disabilities. More specifically, the bolt **filters** the tuples that apply the following condition and groups them together.

```
if {type = 'PWD' AND status = 'available'}
```

Afterwards, the bolt performs a **global grouping** to group all streams together and write them to a database for future analysis.

The aforementioned mechanism will allow monitoring the parking behavior of drivers in order to improve traffic efficiency and reduce the time spend in vehicles for searching a parking spot. This will have significant financial effects for Heraklion city in general, making it interesting to all the stakeholders, from citizens to municipal police. More specific, monitoring the traffic behavior will provide authorities with insights regarding rush hours in time windows of any day in a week; parking data in every single hour of a selected day; or average daily, weekly or monthly parking occupancy. Additionally, through data analysis, the city authority will be able to observe patterns and thus predict the parking demand and traffic situation. In this context, traffic can be proactively and more efficiently managed aiming to reduce or even avoid traffic congestion (e.g. during Christmas holidays public shuttles can be activated to load transport the public from the outskirts to the city center).

8.4.2 Water management using Apache Spark

8.4.2.1 Water level and water quality monitoring in city's water reservoirs

In an effort to protect citizens from water contamination, the municipal authority of Heraklion records the water quality of city's reservoirs to ensure that it complies with the quality standards for drinking water. In addition, data regarding the water level of city's reservoirs are collected in order to extract useful information about the water consumption aiming to prevent unfortunate events of water shortages.

Considering a complex IoT infrastructure for water status monitoring, Heraklion city could benefit from using Apache Spark to activate a smart monitoring system, including warnings and alerts about real time events, while utilizing historic data to generate general knowledge. Apache Spark is a distributed processing environment able to process reliably large volumes of data using distributed clusters for both batch and stream processing. To this end, sensor installations in the city's main water tanks emit data every 10 minutes and generate raw data that can be further exploited by authorities not only to stay informed about their current status, but also to calculate deviations of normal values based on previous data. Sensors record data regarding water temperature, conductivity, pH, water level, chloride, ammonium, nitrate, calcium ions, etc.

In Figure 41, we present a data pipeline aiming to monitor water quantity and quality in Heraklion water tanks with Apache Spark. Data are structured in Resilient Distributed Datasets (RDD), while each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. Employed sensors emit data in the following form:

{id, 'measurement', value, 'timestamp'}

, where *id* refers to the identifier of the sensor followed by the measurement type, the current value and the timestamp of sampling.

In the presented graph streaming data generated by sensors are collected to extract knowledge and provide stakeholders with an overview about the aforementioned parameters (e.g. status) including non-normal behaviors regarding the average values of different parameters, as further explained below.

According to Figure 41, data is processed through a **Window (P1)** function that keeps data frames generated in specific time windows (e.g. the past 10 minutes). Afterwards, a **Map (P2)** operation transforms raw data to key-value pairs consisted solely of the id and the measurement value of a sensor, minimizing that way the distributed data footprint. A **ReduceByKey (P3)** operation calculates the sum of all the values generated from the same sensor, it counts how many entries are recorded within the specified window (represented as *n* in equation (2)), it returns the last entry as the current value and it calculates the following equation (1). The processed data have the form of *(id, (sum, count, numerator, cur_value)).*

$$y = \Sigma(\mathrm{xi} - avg)\hat{}2 \qquad (1)$$

To calculate the average value of the entries, the RDD partition sums up the value of each entry and divides the total number by the count of the entries *(n)*, according to

$$\text{avg} = \overline{x} = \frac{sum(x)}{n} \tag{2}$$

, where $\overline{x}$ is the average value (mean) of all the entries of the same measurement and x is the sum of the total values. Average values of water level entries are calculated in an effort to avoid false positives, including the cases where the amount of water does not reach the full capacity of water tanks, especially in areas where minimum water consumption occurs due to a limited number of households connected to a specific water supply network. It, then, computes the standard deviation *(s)* according to the following equation (3).

$$s = \sqrt{\frac{\Sigma(\text{x}i - avg)\text{^}2}{n}} \tag{3}$$

With a **Map (P4)** operation, the tuples are transformed to key-value pairs including the sensor id, the average values of each sensor, the standard deviation from the mean and the value of the tuple that was transmitted last. Eventually, a **Filter (P5)** operation is applied to filter and forward only the data partitions that satisfy the following condition to a data store for further exploitation

```
if {(current_value – avg) / 2s > 1}
```

, where *current_value* is the current value and *avg* is the average value, which is equal to the mean of a specific variable. If the condition returns TRUE, then a warning will be forwarded to city authorities. To broaden the previous condition, if the following returns TRUE indicates that a major issue is detected in the water supply infrastructure converging to potential water contamination.

```
if {(current_value – avg) / 3s > 1}
```

In that case, a notification system is automatically triggered to inform stakeholders.

8.4.2.2 Water leakage detection

In addition to the aforementioned, we implement a simplistic scenario of water leakage detection in the water pumps of Heraklion city, despite the fact that there is no such an infrastructure available yet. However, it is very important for cities to be able to monitor potential water loss, not only for the leaks that are visible as they rise to the surface but also for those that are invisible and run directly to the ground. These can result a considerable volume of water loss that is often observed retrospectively. Leaks may occur due to the age of the network, freezing pipes during winter, traffic, excavations or even excessive water pressure.

In this context, we consider the water supply network of Heraklion city divided into zones and subdivided to district areas with a number of water flow and pressure sensors being installed in different distances. To simplify our approach, we consider that pipelines have no branches, which would mean that the water flow at the end of each branch should be added and compared to the water flow at the starting point of the pipe, namely the sensors that are located near the water tank. To check if there is leak, i.e., divergence of water flow in the network, we need to compare the water flow values at the starting point of a pipeline with the values at its end. Moreover, to estimate the approximate location of the malfunction in the network, we need to extract the water flow values from different distances in the pipe and check whether they differentiate with each other.

More specific, for every main pipeline that distributes water around the city we extract an id and an average value of the flow along the pipe (e.g. *{id, total_avg}*). Each sensor installed within a pipe emits tuples of key-value pairs in the form of

{id, value, 'timestamp'}

As illustrated in Figure 41, data is processed through a **Window (P6)** function that keeps data frames generated during the last 5 seconds. Afterwards, through a **ReduceByKey (P7)** operation we calculate the total average number from the values extracted from all the sensors placed in a specific pipeline within the aforementioned time frame.

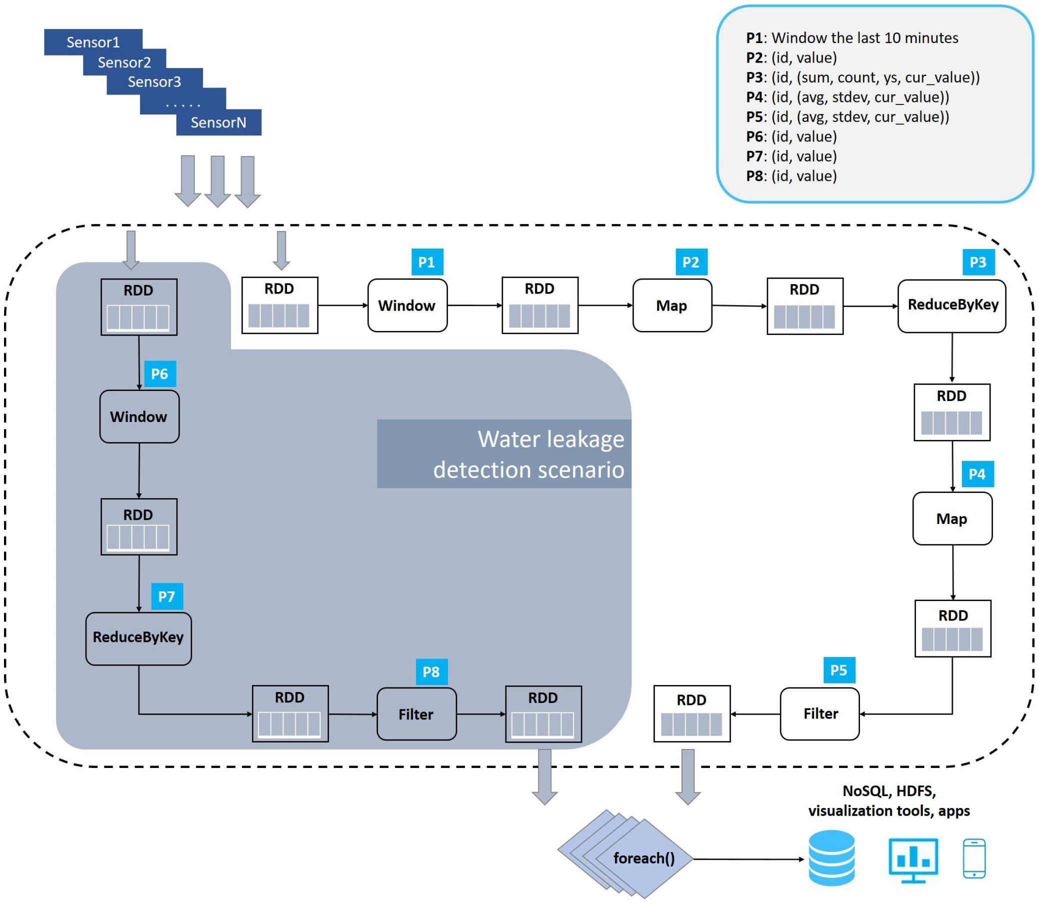
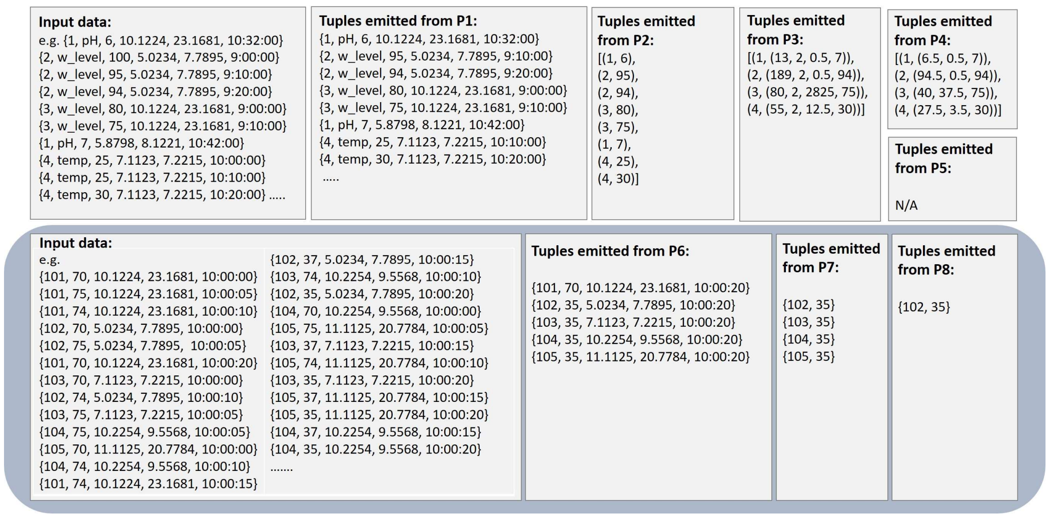

Sensor1
Sensor2
Sensor3
.....
SensorN
P1: Window the last 10 minutes
P2: (id, value)
P3: (id, (sum, count, ys, cur_value))
P4: (id, (avg, stdev, cur_value))
P5: (id, (avg, stdev, cur_value))
P6: (id, value)
P7: (id, value)
P8: (id, value)
RDD
Window
Map
ReduceByKey
Filter
P1
P2
P3
P4
P5
P6
P7
P8
Water leakage detection scenario
NoSQL, HDFS, visualization tools, apps
foreach()

Figure 41: A data pipeline for water monitoring in Heraklion city



Input data:
e.g. {1, pH, 6, 10.1224, 23.1681, 10:32:00}
{2, w_level, 100, 5.0234, 7.7895, 9:00:00}
{2, w_level, 95, 5.0234, 7.7895, 9:10:00}
{2, w_level, 94, 5.0234, 7.7895, 9:20:00}
{3, w_level, 80, 10.1224, 23.1681, 9:00:00}
{3, w_level, 75, 10.1224, 23.1681, 9:10:00}
{1, pH, 7, 5.8798, 8.1221, 10:42:00}
{4, temp, 25, 7.1123, 7.2215, 10:00:00}
{4, temp, 25, 7.1123, 7.2215, 10:10:00}
{4, temp, 30, 7.1123, 7.2215, 10:20:00} …..

Tuples emitted from P1:
{1, pH, 6, 10.1224, 23.1681, 10:32:00}
{2, w_level, 95, 5.0234, 7.7895, 9:10:00}
{2, w_level, 94, 5.0234, 7.7895, 9:20:00}
{3, w_level, 80, 10.1224, 23.1681, 9:00:00}
{3, w_level, 75, 10.1224, 23.1681, 9:10:00}
{1, pH, 7, 5.8798, 8.1221, 10:42:00}
{4, temp, 25, 7.1123, 7.2215, 10:10:00}
{4, temp, 30, 7.1123, 7.2215, 10:20:00}
…..

Tuples emitted from P2:
[(1, 6),
(2, 95),
(2, 94),
(3, 80),
(3, 75),
(1, 7),
(4, 25),
(4, 30)]

Tuples emitted from P3:
[(1, (13, 2, 0.5, 7)),
(2, (189, 2, 0.5, 94)),
(3, (80, 2, 2825, 75)),
(4, (55, 2, 12.5, 30))]

Tuples emitted from P4:
[(1, (6.5, 0.5, 7)),
(2, (94.5, 0.5, 94)),
(3, (40, 37.5, 75)),
(4, (27.5, 3.5, 30))]

Tuples emitted from P5:

N/A

Input data:
e.g.
{101, 70, 10.1224, 23.1681, 10:00:00}
{101, 75, 10.1224, 23.1681, 10:00:05}
{101, 74, 10.1224, 23.1681, 10:00:10}
{102, 70, 5.0234, 7.7895, 10:00:00}
{102, 75, 5.0234, 7.7895, 10:00:05}
{101, 70, 10.1224, 23.1681, 10:00:20}
{103, 70, 7.1123, 7.2215, 10:00:00}
{102, 74, 5.0234, 7.7895, 10:00:10}
{103, 75, 7.1123, 7.2215, 10:00:05}
{104, 75, 10.2254, 9.5568, 10:00:05}
{105, 70, 11.1125, 20.7784, 10:00:00}
{104, 74, 10.2254, 9.5568, 10:00:10}
{101, 74, 10.1224, 23.1681, 10:00:15}
{102, 37, 5.0234, 7.7895, 10:00:15}
{103, 74, 10.2254, 9.5568, 10:00:10}
{102, 35, 5.0234, 7.7895, 10:00:20}
{104, 70, 10.2254, 9.5568, 10:00:00}
{105, 75, 11.1125, 20.7784, 10:00:05}
{103, 37, 7.1123, 7.2215, 10:00:15}
{105, 74, 11.1125, 20.7784, 10:00:10}
{103, 35, 7.1123, 7.2215, 10:00:20}
{105, 37, 11.1125, 20.7784, 10:00:15}
{105, 35, 11.1125, 20.7784, 10:00:20}
{104, 37, 10.2254, 9.5568, 10:00:15}
{104, 35, 10.2254, 9.5568, 10:00:20}
…….

Tuples emitted from P6:

{101, 70, 10.1224, 23.1681, 10:00:20}
{102, 35, 5.0234, 7.7895, 10:00:20}
{103, 35, 7.1123, 7.2215, 10:00:20}
{104, 35, 10.2254, 9.5568, 10:00:20}
{105, 35, 11.1125, 20.7784, 10:00:20}

Tuples emitted from P7:

{102, 35}
{103, 35}
{104, 35}
{105, 35}

Tuples emitted from P8:

{102, 35}

Figure 42: Tuples emitted throughout the Spark topology

A user-defined function applied in a **Filter (P8)** operator checks if there is any water loss detected through the specific network, by computing if the following condition returns TRUE

if {abs(s_i – total_avg)} > threshold

s_i is the value of the water flow (liters per meter - lpm) recorded from the sensor *i* and s_i – *total_avg* calculates if each sensor's value deviates from the total average value of the pipeline, taking into consideration that normally, if no anomaly occurs in the network, all sensors should return the same value. According to this condition, we check if the absolute value of the abstraction is greater than a threshold (e.g. 2 lpm assuming that sensors have ±1 lpm level of accuracy). If the condition returns TRUE it indicates that water loss has been detected in the network.

To estimate the location of the damage we use a second **Filter (P8)** operator to check whether the condition below is applied.

if {(abs(s_i – s_{i+1}) > threshold) AND (abs(s_i – s_{i+2}) > threshold)}

That means that if the absolute difference of the value that a sensor (s_i) transmits is greater than a threshold for the next two sensors (s_{i+1}, s_{i+2}) then it will be concluded that the leakage takes place between sensors s_i *and* s_{i+1}. If this condition is TRUE then the operator transmits tuples consisted of the id and the value of the first sensor after the leakage point.

In that context, further investment in the IoT infrastructure of a water supply network of Heraklion city will allow a more efficient monitoring mechanism regarding the quality of water transferred through the water pipelines to households and the prevention of water loss due to malfunctions in the network, as well. The latter can be monitored through actuators installed near the water flow and pressure sensors. Thus, whenever a leakage event occurs, actuators can be automatically activated to stop the water flow to the point that the malfunction is spotted, until the technicians repair the damage.

8.4.3 Environmental monitoring using Apache Flink

Heraklion Smart City's infrastructure monitors environmental parameters such as temperature, humidity, wind and air pollutants, including gases (e.g. SO_2, NO, NO_2, CO_2, O_3), smoke concentration and dust particles in order to protect the public from areas condensed with air pollutants and extreme weather events. Such an infrastructure requires a big data platform able to handle a very large amount of data streams and able to process historic data with the aim of predicting upcoming non-normal events through event pattern recognition mechanisms. In this context, we use Apache Flink to process and analyze streaming data as well as to inform stakeholders (including susceptible groups of people) about threatening weather conditions and areas with hazardous air pollutant accumulation.

A high-level view of the computational logic of Apache Flink is introduced in Figure 43 represented as directed graph. According to the specific data flow program, data is received from a data source (i.e., sensors), applying a set of transformations (e.g. map, filter, window operations, etc.) and writes the results to data stores, such as NoSQL, HDFS, sinks, etc. In the presented diagram, the operators (nodes) represent computations, while edges represent data dependencies. Specifically, operators consume data from inputs, perform a computation and produce data to outputs for further processing.

A number of sensors emit weather data every 1 minute and air pollution data every 5 minutes (see Chapter 5.2). Stream data collected by sensors constitute a set of key-value pairs in the form of *{name$_1$: value$_1$, name$_2$: value$_2$,… name$_n$: value$_n$}*. For each set of key value pairs, the key refers to the identifier of the sensor and the value refers to (i) a measurement, (ii), the value of the parameter measured, (iii) the sensor's location, and (iv) the time of an event. More specific, data streams have the following form:

{id: 1, measurement: CO_2, value: 420, coordinates: 10.1224, 23.1681, timestamp: 10:45:00}

According to Figure 43, a time **Window (F1)** operator is used to collect events that occurred during the last 10 minutes so that the volume of data is decreased to a finite data set. Then, a **KeyBy (F2)** operator partitions the sensor readings with the same key (i.e., measurement) randomly to be processed across the tasks of the subsequent operator (e.g. *{id: 1, measurement: CO_2, value: 420, coordinates: 10.1224, 23.1681, timestamp: 10:45:00}, {id: 1, measurement: CO_2, value: 424, coordinates: 10.1224, 23.1681, timestamp: 10:50:00}, etc.*). Afterwards, stream data is **joined (F3)** together according to the sensors' location in correlation to existing geographic areas retrieved from a Kafka topic.

As illustrated in Figure 43, tuples of data is transmitted from the join operation through custom partitioning for three distinct computations: **min (F4)**, **max (F5)** and **average (F6)**. Custom partitioning indicates that tuples, which belong to the same geographic area, are mapped to the same task for processing. For each incoming event, the operators update the corresponding aggregate value and emit a new event with the updated value at runtime. As a next step, we calculate the average value of the total events for each sensor, in order to check whenever they exceed the given thresholds, as a way to avoid false positives. The tuples of data emitted from the average operator have the form *{'measurement', avg_value}* and are stored to a NoSQL or HDFS database.

According to Heraklion Smart City, any combination of the patterns regarding concentrations of the following parameters could be considered as an “air pollution” event when occurring for a period longer than 30 minutes.

Particulate matter: PM10 > 50 mg/m^3

Carbon dioxide: CO_2 > 380 ppm

Nitrogen dioxide: NO_2 > 19.7 ppb

Sulphur dioxide: SO_2 > 7.09 ppb

Ozone: O_3 > 47.3 ppb

Nitrogen oxide: NO > 25000 ppb

Volatile Organic Compound: VOC > 600 ppb

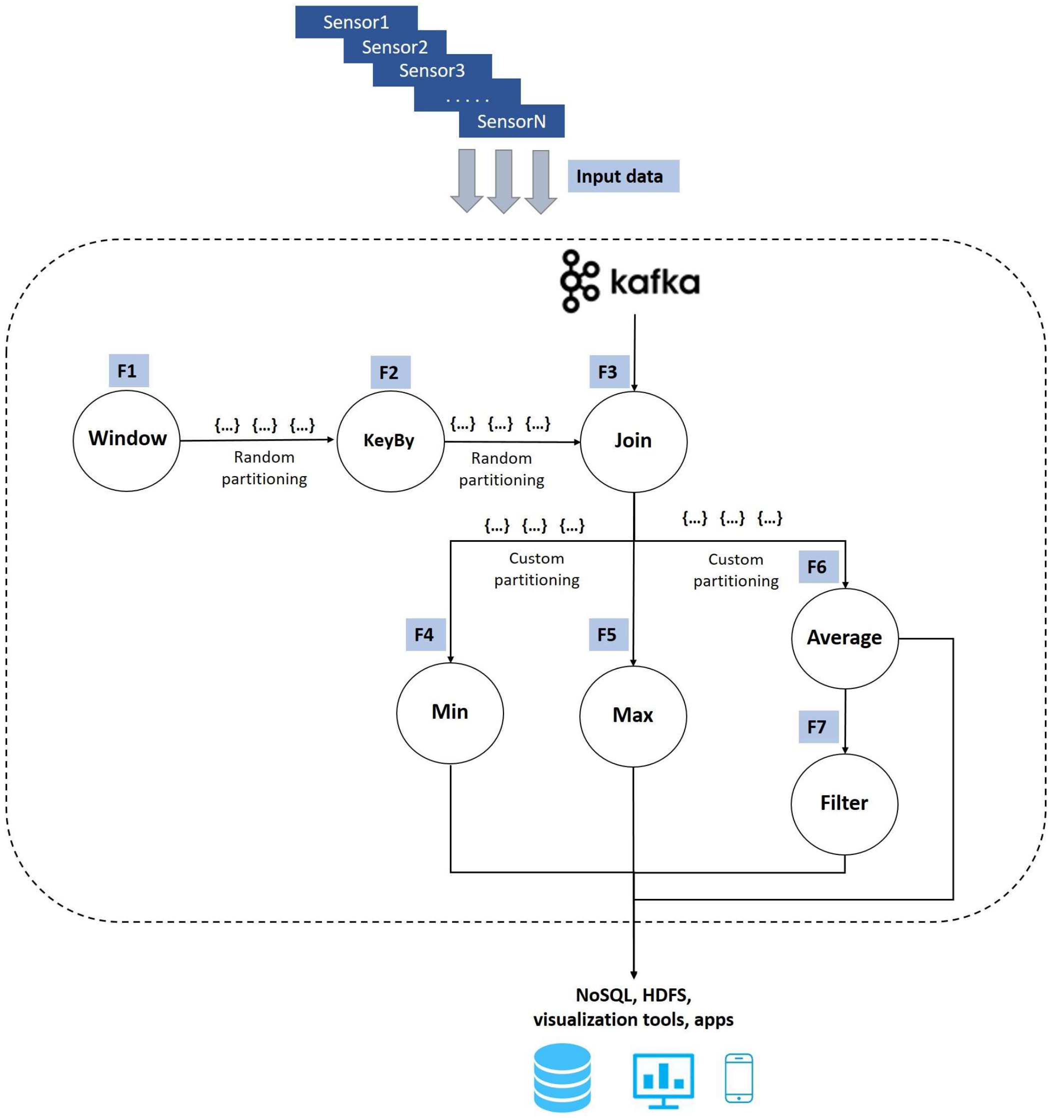
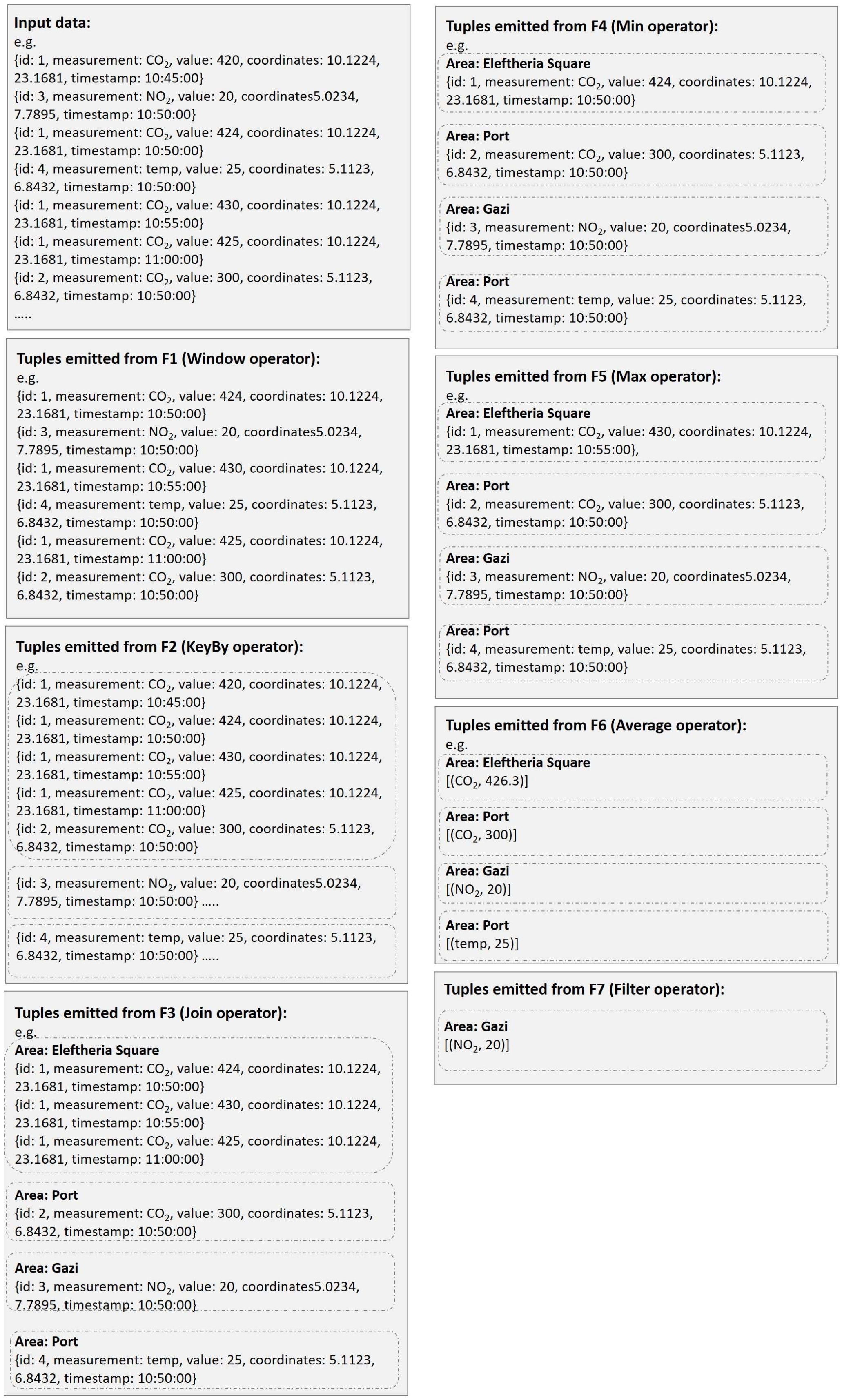


Figure 43: A physical dataflow plan for air quality monitoring



Input data:
e.g.
{id: 1, measurement: CO_2, value: 420, coordinates: 10.1224, 23.1681, timestamp: 10:45:00}
{id: 3, measurement: NO_2, value: 20, coordinates5.0234, 7.7895, timestamp: 10:50:00}
{id: 1, measurement: CO_2, value: 424, coordinates: 10.1224, 23.1681, timestamp: 10:50:00}
{id: 4, measurement: temp, value: 25, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}
{id: 1, measurement: CO_2, value: 430, coordinates: 10.1224, 23.1681, timestamp: 10:55:00}
{id: 1, measurement: CO_2, value: 425, coordinates: 10.1224, 23.1681, timestamp: 11:00:00}
{id: 2, measurement: CO_2, value: 300, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}
.....

Tuples emitted from F1 (Window operator):
e.g.
{id: 1, measurement: CO_2, value: 424, coordinates: 10.1224, 23.1681, timestamp: 10:50:00}
{id: 3, measurement: NO_2, value: 20, coordinates5.0234, 7.7895, timestamp: 10:50:00}
{id: 1, measurement: CO_2, value: 430, coordinates: 10.1224, 23.1681, timestamp: 10:55:00}
{id: 4, measurement: temp, value: 25, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}
{id: 1, measurement: CO_2, value: 425, coordinates: 10.1224, 23.1681, timestamp: 11:00:00}
{id: 2, measurement: CO_2, value: 300, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}

Tuples emitted from F2 (KeyBy operator):
e.g.
{id: 1, measurement: CO_2, value: 420, coordinates: 10.1224, 23.1681, timestamp: 10:45:00}
{id: 1, measurement: CO_2, value: 424, coordinates: 10.1224, 23.1681, timestamp: 10:50:00}
{id: 1, measurement: CO_2, value: 430, coordinates: 10.1224, 23.1681, timestamp: 10:55:00}
{id: 1, measurement: CO_2, value: 425, coordinates: 10.1224, 23.1681, timestamp: 11:00:00}
{id: 2, measurement: CO_2, value: 300, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}

{id: 3, measurement: NO_2, value: 20, coordinates5.0234, 7.7895, timestamp: 10:50:00} .....

{id: 4, measurement: temp, value: 25, coordinates: 5.1123, 6.8432, timestamp: 10:50:00} .....

Tuples emitted from F3 (Join operator):
e.g.
Area: Eleftheria Square
{id: 1, measurement: CO_2, value: 424, coordinates: 10.1224, 23.1681, timestamp: 10:50:00}
{id: 1, measurement: CO_2, value: 430, coordinates: 10.1224, 23.1681, timestamp: 10:55:00}
{id: 1, measurement: CO_2, value: 425, coordinates: 10.1224, 23.1681, timestamp: 11:00:00}

Area: Port
{id: 2, measurement: CO_2, value: 300, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}

Area: Gazi
{id: 3, measurement: NO_2, value: 20, coordinates5.0234, 7.7895, timestamp: 10:50:00}

Area: Port
{id: 4, measurement: temp, value: 25, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}

Tuples emitted from F4 (Min operator):
e.g.
Area: Eleftheria Square
{id: 1, measurement: CO_2, value: 424, coordinates: 10.1224, 23.1681, timestamp: 10:50:00}

Area: Port
{id: 2, measurement: CO_2, value: 300, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}

Area: Gazi
{id: 3, measurement: NO_2, value: 20, coordinates5.0234, 7.7895, timestamp: 10:50:00}

Area: Port
{id: 4, measurement: temp, value: 25, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}

Tuples emitted from F5 (Max operator):
e.g.
Area: Eleftheria Square
{id: 1, measurement: CO_2, value: 430, coordinates: 10.1224, 23.1681, timestamp: 10:55:00},

Area: Port
{id: 2, measurement: CO_2, value: 300, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}

Area: Gazi
{id: 3, measurement: NO_2, value: 20, coordinates5.0234, 7.7895, timestamp: 10:50:00}

Area: Port
{id: 4, measurement: temp, value: 25, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}

Tuples emitted from F6 (Average operator):
e.g.
Area: Eleftheria Square
[(CO_2, 426.3)]

Area: Port
[(CO_2, 300)]

Area: Gazi
[(NO_2, 20)]

Area: Port
[(temp, 25)]

Tuples emitted from F7 (Filter operator):

Area: Gazi
[(NO_2, 20)]

Figure 44: Tuples emitted throughout the Apache Flink topology

Moreover, as depicted in Figure 43, a **filter (F7)** operator is responsible to receive tuples from the average operator and group together the data streams that satisfy the following condition.

If { PM10 > 50 OR CO_2 > 430 OR NO_2 > 19.7 OR SO_2 > 7.09 OR O_3 > 47.3 OR NO > 25000 OR VOC > 600}

Similar to the previous logic, the thresholds for weather conditions applied in Heraklion Smart City are the following.

Temperature: < -10 °C OR > 50 °C

Rainfall > 8 mm

Wind speed > 50 km/h

Atmospheric pressure: < 750 hPa OR > 1050 hPa

The according **filter** operator groups together the data streams in refer to weather parameters that satisfy the condition below.

If { (temp < -10 OR temp > 50) OR rnfl > 8 OR wind_spd > 50 OR (atm_prs < 750 OR atm_prs > 1050)}

If the computed averages exceed the aforementioned thresholds, the corresponding alerting system referring to the specific area will be activated.

Finally, the tuples transmitted from the min, max and filter operators will be written to a NoSQL database (e.g. MongoDB, CouchDB, Apache HBase, Apache Cassandra, etc.). Data streams can be produced into message queues to make the events available to subsequent streaming applications, written to file systems / data stores for making the data available for further analysis. The stored data can be eventually visualized through analytics tools, allowing stakeholders to access the generated information in a user-friendly way.

Exiting the presented Flink topology, data records can be further used for building a machine-learning model aiming to predict intense concentrations of air pollutants in specific territories, such as schools, traffic lights and so on. Based on observed patterns a notification system can be triggered whenever outliers of the parameters measured are detected. For instance, an event-driven application can trigger actions such as sending an alert or an email or write events to an outgoing event stream to be consumed by another event-driven application. This information could be proven valuable to people with health issues or susceptible groups (e.g. children, pregnant women, etc.) that need to know in which territories intense gas accumulation is observed.

8.5 Using Apache Beam for cross platform data processing

In the previous chapters, we elaborated data processing pipelines in three distinct scenarios (i.e., transportation, water and environment) based on Heraklion's Smart City infrastructure. As a next step, we need to establish a global mechanism for data processing across different platforms able to interconnect all these scenarios in order to allow global queries and exploitation of the generated data to draw useful conclusions. To accomplish that, we apply Apache Beam, a cross-platform processing tool that runs on a number of runtimes (e.g., Apache Flink, Apache Spark) and simplifies the mechanics of large-scale batch and streaming data processing.

To that end, we examine how finding a parking place may affect carbon dioxide (CO_2) concentrations in the atmosphere. In general, Heraklion city suffers from excessive traffic in certain points and especially in the city center where the combination of commuters' traffic and shopping hours result in street congestion. That often hinders the smooth car flow on the streets. At the same time, having in mind that vehicles emit CO_2 gases we need to check whether potential correlation between the parking behavior and CO_2 accumulations in the air is observed. More specific, we assume that thousands of sensors are installed on every existing parking place, both on-road and in parking areas of Heraklion city. For the purposes of our scenario, we receive every 10 minutes (a) the parking status, namely the percentage of parking lots that are occupied, and (b) the CO_2 levels per area. If the available parking capacity in the city center reaches more than 60%, we hypothesize that this might be an indication that a considerable number of vehicles leaves the area. Such a mass behavior could imply that traffic congestion is expected to occur with high concentrations of carbon dioxide in the atmosphere.

More specific, the following Beam pipeline comprises Parallel Collections (PCollections), Aggregation Transforms and Transformations. PCollections represent distributed data sets that are both bounded, meaning they come from a fixed data source, like a file, and unbounded, meaning they come from continuously updated data sources. Aggregation Transforms, such as GrouByKey and Combine, work on a per-window basis, as the data set is generated, and they process each PCollection as a succession of these finite windows. Transforms represent a data processing operation in the pipeline. Every Transform takes one or more PCollections as input, it performs a processing function and it then produces zero or more output PCollection objects.

As illustrated in Figure 45, the Beam pipeline receives input data from both parking sensors and air quality sensors measuring CO_2 concentrations in the air. The tuples of data arriving from parking sensors are key-value pairs in the form of

{id, 'status', 'type', lon, lat, timestamp}

, with the key being the *id* of the sensor and the rest parameters being the value. The *status* parameter takes values of *'occupied'* or *'available'* and the *type 'PWD'* or *'PWND'* (see Chapter 8.4.1). Since parking data are unbounded data sets, we apply a non-global **Window** operation to group generated data to batches every 10 minutes (e.g. 10:30-10:40), according to the timestamp of each record. Through a Kafka topic, we extract information regarding the geographic areas, in which Heraklion city is bisected, and with a **ParDo** we correlate the records according to their coordinates and group together the ones within the same area. However, each sensor records data every 1 minute (see Chapter 5.2) that means hundreds of tuples are collected in batches grouped together every 10 minutes. To that end, a **Filter** transformation keeps only the last record for each sensor id and filters out the rest. Afterwards, we **Count** the total number of available and occupied parking sensors per area so that the emitted data of P5 have the form of

{count, 'status', 'geo_area'}

Then, a **ParDo** calculates the percentage value of occupied and available parking spots per area (e.g. *{'60%', 'occupied', 'Eleftheria Square'}, {'52%', 'available', 'Port'}).*

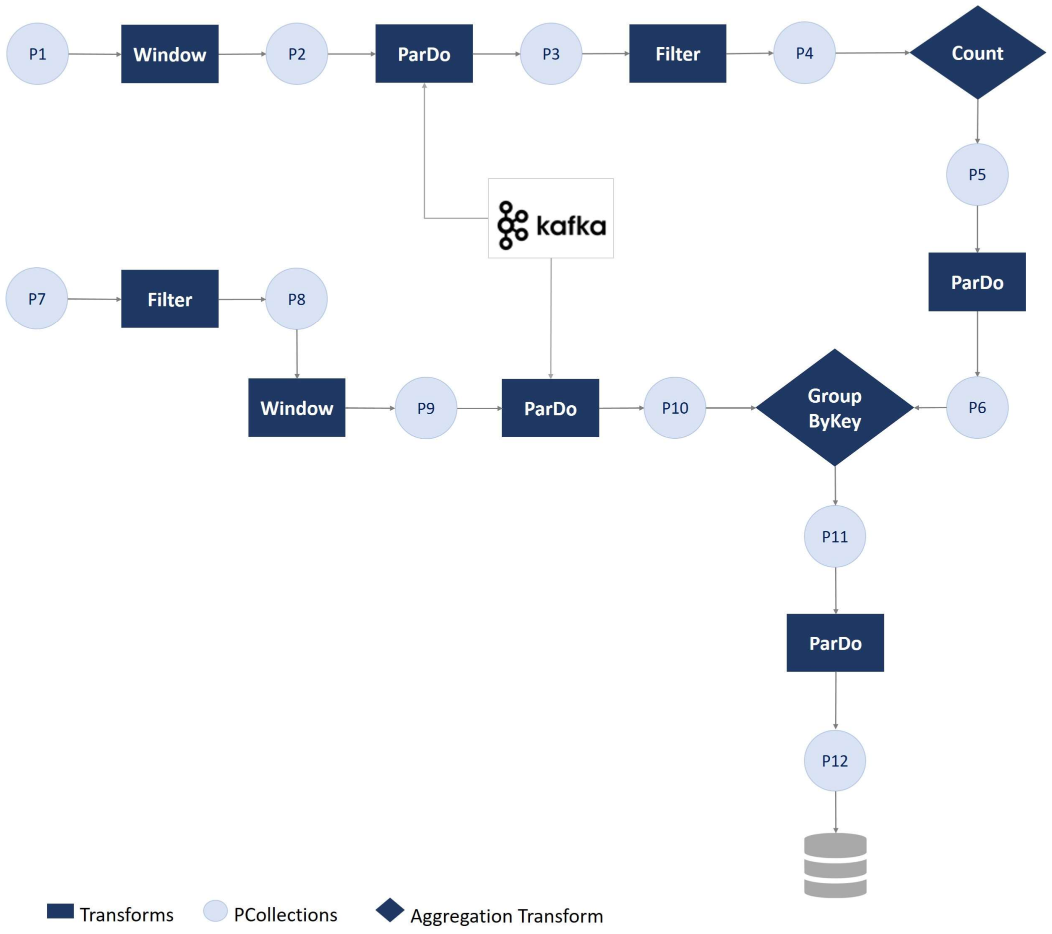


Figure 45: A Data processing pipeline with the Beam model

Input data P1:
e.g.
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:10:40}
{1, 'available', 'PWD', 10.3452, 9.0431, 10:20:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:30:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:31:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:32:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:37:40}
{2, 'occupied', 'PWD', 10.3582, 9.1215, 09:35:02}
{2, 'available', 'PWD', 10.3582, 9.1215 ,10:35:02}
{2, 'available', 'PWD', 10.3582, 9.1215 ,10:37:02}
{3, 'occupied', 'PWND', 6.8432, 10.4312, 10:34:42}
{4, 'available', 'PWND', 7.4834, 10.5254, 10:30:00}
….

PCollections of P2:

{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:30:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:31:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:32:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:37:40}
{2, 'available', 'PWD', 10.3582, 9.1215 ,10:35:02}
{2, 'available', 'PWD', 10.3582, 9.1215 ,10:37:02}
{3, 'occupied', 'PWND', 6.8432, 10.4312, 10:34:42}
{4, 'available', 'PWND', 7.4834, 10.5254, 10:30:00}

PCollections of P3:

Area: Eleftheria Square
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:30:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:31:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:32:40}
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:37:40}
{4, 'available', 'PWND', 7.4834, 10.5254, 10:30:00}

Area: Port
{2, 'available', 'PWD', 10.3582, 9.1215 ,10:35:02}
{2, 'available', 'PWD', 10.3582, 9.1215 ,10:37:02}
{3, 'occupied', 'PWND', 6.8432, 10.4312, 10:34:42}

PCollections of P4:

Area: Eleftheria Square
{1, 'occupied', 'PWD', 10.3452, 9.0431, 10:37:40}
{4, 'available', 'PWND', 7.4834, 10.5254, 10:30:00}

Area: Port
{2, 'available', 'PWD', 10.3582, 9.1215 ,10:37:02}
{3, 'occupied', 'PWND', 6.8432, 10.4312, 10:34:42}

PCollections of P5:
{1, 'occupied', 'Eleftheria Square'}
{1, 'available', 'Eleftheria Square'}
{1, 'available', 'Port'}
{1, 'occupied', 'Port'}

PCollections of P6:
{'40%', 'occupied', 'Eleftheria Square'}
{'60%', 'available', 'Eleftheria Square'}
{'52%', 'available', 'Port'}
{'48%', 'occupied', 'Port'}

Input data P7:
e.g.
{id: 1, measurement: CO_2, value: 420, coordinates: 10.1224, 23.1681, timestamp: 10:35:00}
{id: 3, measurement: NO_2, value: 20, coordinates5.0234, 7.7895, timestamp: 10:31:00}
{id: 1, measurement: CO_2, value: 424, coordinates: 10.1224, 23.1681, timestamp: 10:38:00}
{id: 4, measurement: temp, value: 25, coordinates: 5.1123, 6.8432, timestamp: 10:50:00}
{id: 1, measurement: CO_2, value: 430, coordinates: 10.1224, 23.1681, timestamp: 10:50:00}
{id: 1, measurement: CO_2, value: 425, coordinates: 10.1224, 23.1681, timestamp: 11:00:00}
{id: 2, measurement: CO_2, value: 300, coordinates: 5.1123, 6.8432, timestamp: 10:40:00}…..

PCollections of P8:
{id: 1, measurement: CO_2, value: 420, coordinates: 10.1224, 23.1681, timestamp: 10:35:00}
{id: 1, measurement: CO_2, value: 424, coordinates: 10.1224, 23.1681, timestamp: 10:38:00}
{id: 1, measurement: CO_2, value: 430, coordinates: 10.1224, 23.1681, timestamp: 10:50:00}
{id: 1, measurement: CO_2, value: 425, coordinates: 10.1224, 23.1681, timestamp: 11:00:00}
{id: 2, measurement: CO_2, value: 300, coordinates: 5.1123, 6.8432, timestamp: 10:40:00}

PCollections of P9:
{id: 1, measurement: CO_2, value: 420, coordinates: 10.1224, 23.1681, timestamp: 10:35:00}
{id: 1, measurement: CO_2, value: 424, coordinates: 10.1224, 23.1681, timestamp: 10:38:00}
{id: 2, measurement: CO_2, value: 300, coordinates: 5.1123, 6.8432, timestamp: 10:40:00}

PCollections of P10:
{422, 'Eleftheria Square'}
{300, 'Port'}

PCollections of P11:
{422, '60%', 'Eleftheria Square'}
{300, '52%', 'Port'}

PCollections of P12:
{422, '60%', 'Eleftheria Square'}

Figure 46: PCollections emitted through the Beam pipeline

At the same time, air quality sensors emit data measuring CO_2, NO_2, SO_2, O_3, etc. in key-value pairs **(P7)** in the form of

{id, 'measurement', value, 'coordinates', timestamp}

Then, a **Filter** transformation keeps only the records related to CO_2 emissions and discards the rest. Through a non-global **Window** we group finite sets of data in 10-minute batches and similarly to the parking sensors, a **ParDo** transform groups data to distinct areas in reference to a Kafka topic. Then, we calculate the average values of CO_2 per area in order to avoid false positive indications and we transform the tuples **(P10)** in the form of *{avg, 'geo_area'}*.

Furthermore, a **GoupByKey** aggregating transform acts as a reduction operation that collects and groups together the parking and CO_2 records that belong to the same geographic area and emits the following form of tuples **(P11)**

{avg, 'avail_pntg', 'geo_area'}

, where *avail_pntg* the percentage of the available parking lots.

Finally, a **ParDo** transformation filters the data set by checking whether the elements of the previous PCollection satisfy the following condition and groups them together.

*If {avail_pntg ≥ 0.6 * total_park AND CO_2 ≥ 400}*

The *total_park* parameter represents the total number of parking lots. It should be clarified that the threshold regarding the amount of available parking lots is an indicative example and it could be specified through a number of observations.

Moreover, a notification mechanism triggered every time high concentration of CO_2 in specific areas is observed, could inform vulnerable groups of people (e.g., elderly, people with health issues, children etc.). In addition, the aforementioned scenario can be further exploited combining records of data regarding the air temperature and water quality in seas and rivers in order to evaluate how rising CO_2 can affect in the long run the climate change and in what level it contributes to the ocean acidification. Applying the appropriate mechanisms (i.e., machine-learning tools) for pattern recognition based on these data can be proven very useful in order to predict extreme upcoming events. Having all these in mind, the Municipal of Heraklion may use these data to extract valuable conclusions based on both real time and historic data, which may prove important for decision making in matters related to the wider region.

9 Conclusion and Future Work

This thesis focuses on leveraging the knowledge stemming from a city's core services by utilizing the necessary IoT infrastructure in order to facilitate the decision-making process in smart cities' setups for the benefit of the society and its people, as well. To this end, we identify the needs of the city's core systems to facilitate its operational activities and coordinate its services including road traffic, parking, waste, water, air and weather monitoring. In this context, we underlined weaknesses and threats, they might entail in refer to city sustainability and resources consumption. We analyzed and translated usual problems into business objectives and underlined the technological challenges of utilizing large amounts of data and interpreted them in technical requirements. We provided an in –depth analysis of existing technologies for big data processing and reviewed a number of algorithms and algorithmic suites that can be implemented within such platforms. The algorithms presented can be deployed in various parts of the proposed architecture, so that they can be utilized while performing analytics tasks. Eventually, we incorporated all the above in a full-fledged architectural design proposal tailored for smart cities.

Followed by a comprehensive report about future benefits for the stakeholders, we exemplify the potential advantages in terms of business's logic efficiency of adopting mainstream data processing topologies including Apache Storm, Apache Spark and Apache Flink into a real smart city setup, namely Heraklion's Smart City. In this direction, we presented the processing pipeline of big data on various Heraklion city's domains including parking, water and air quality monitoring. Eventually, we interconnected two vertical sectors through a cross-platform technology, Apache Beam, to indicate data processing across different platforms.

Two main directions of further work are anticipated in a path towards supporting the utilization of big data processing pipelines for the benefit of smart city's stakeholders. The first direction concerns the improvement of the proposed infrastructure by minimizing the processing workload of sensor data in the data processing topologies presented, while the second concerns the potential benefits for stakeholders arising from big data processing as regards further city's domains such as smart governance, smart grid and smart health.

REFERENCES

1. 11th USENIX Symposium on Operating Systems Design and Implementation, Scaling Distributed Machine Learning with the Parameter Server, October 2014, https://www.usenix.org/node/186215
2. 6sqft, NYC water towers: History, use, and infrastructure, July 2017, https://www.6sqft.com/nyc-water-towers-history-use-and-infrastructure/
3. A vision of smarter cities, IBM Global Business Services, IBM Institute for Business Value (https://www-03.ibm.com/press/attachments/IBV_Smarter_Cities_-_Final.pdf)
4. Agrawal, D., Chawla, S., Contreras-Rojas, B., Elmagarmid, A., Idris, Y., Kaoudi, Z., ... & Papotti, P. (2018). RHEEM: enabling cross-platform data processing: may the Big Data be with you!. Proceedings of the VLDB Endowment, 11(11), 1414-1427.
5. Air pollution to be monitored at twenty-two schools across Newcastle. Newcastle University, 20 Nov 2019 (https://www.ncl.ac.uk/press/articles/latest/2019/11/airpollutionandschools/)
6. Al Nuaimi, E., Al Neyadi, H., Mohamed, N., & Al-Jaroodi, J. (2015). Applications of Big Data to smart cities. Journal of Internet Services and Applications, 6(1), 25.
7. Anagnostopoulos, T., Zaslavsky, A., Kolomvatsos, K., Medvedev, A., Amirian, P., Morley, J., & Hadjieftymiades, S. (2017). Challenges and opportunities of waste management in IoT-enabled smart cities: a survey. IEEE Transactions on Sustainable Computing, 2(3), 275-289.
8. Ang, L. M., & Seng, K. P. (2016). Big sensor data applications in urban environments. Big Data Research, 4, 1-12.
9. Apache Cassandra, http://cassandra.apache.org/
10. Apache Data Sketches, Sketch Capability Matrix, https://datasketches.apache.org/docs/MajorSketchFamilies.html
11. Apache DataSketches, https://datasketches.github.io/.
12. Apache Flink, FlinkML – Machine Learning for Flink, https://ci.apache.org/projects/flink/flink-docs-release-1.8/dev/libs/ml/#supervised-learning
13. Apache Flink, https://ci.apache.org/projects/flink/flink-docs-master/zh/internals/stream_checkpointing.html
14. Apache Flume, http://flume.apache.org/
15. Apache Hadoop, http://hadoop.apache.org/
16. Apache HBase, https://mapr.com/products/apache-hbase/
17. Apache Kafka, A distributed streaming platform, https://kafka.apache.org/intro
18. Apache MapReduce. IBM(https://www.ibm.com/analytics/hadoop/mapreduce)
19. Apache MESOS, Nvidia GPU Support, http://mesos.apache.org/documentation/latest/gpu-support/
20. Apache SAMOA, Scalable Advanced Massive Online Analysis, http://samoa.incubator.apache.org/
21. Apache Samza, A distributed stream processing framework, http://samza.apache.org/
22. Apache Spark MLib, https://spark.apache.org/mllib/
23. Apache Zeppelin, https://zeppelin.apache.org/

24. Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer networks, 54(15), 2787-2805.
25. AWS Amazon, Benefits of Pub/Sub Messaging, https://aws.amazon.com/pub-sub-messaging/benefits/
26. Barcelona: smart city revolution in progress. The Spanish metropolis is rethinking the way it uses the internet of things. Financial Times (https://www.ft.com/content/6d2fe2a8-722c-11e7-93ff-99f383b09ff9)
27. Baron, C. A. (2016). NoSQL key-value DBs riak and redis. Database Systems Journal, 4, 3-10.
28. Baru, C., Bhandarkar, M., Nambiar, R., Poess, M., & Rabl, T. (2012, August). Setting the direction for Big Data benchmark standards. In Technology Conference on Performance Evaluation and Benchmarking (pp. 197-208). Springer, Berlin, Heidelberg.
29. Beam, Documentation, https://beam.apache.org/documentation/runners/spark/
30. Bibri, S. E. (2015). The Human Face of Ambient Intelligence, cognitive, emotional, affective, behavioral, and conversational aspects. Atlantis Press, Springer-Verlag, Berlin, Heidelberg (2015)
31. Bibri, S. E. (2018). The IoT for smart sustainable cities of the future: An analytical framework for sensor-based Big Data applications for environmental sustainability. Sustainable Cities and Society, 38, 230-253.
32. Bibri, S. E., & Krogstie, J. (2017). Smart sustainable cities of the future: An extensive interdisciplinary literature review. Sustainable Cities and Society, 31, 183-212.
33. Bibri, S. E., & Krogstie, J. (2017). The core enabling technologies of big data analytics and context-aware computing for smart sustainable cities: a review and synthesis. Journal of Big Data, 4(1), 38.
34. Big Data Zone, Data Visualization Using Apache Zeppelin, October 2017, https://dzone.com/articles/data-visualization-using-apache-zeppelin
35. Bijnens, Nathan. "A real-time architecture using Hadoop and Storm". 11 December 2013, lambda-architecture.net
36. Biswas, K., & Muthukkumarasamy, V. (2016, December). Securing smart cities using blockchain technology. In 2016 IEEE 18th international conference on high performance computing and communications; IEEE 14th international conference on smart city; IEEE 2nd international conference on data science and systems (HPCC/SmartCity/DSS) (pp. 1392-1393). IEEE.
37. Bitbucket, FERARI, https://bitbucket.org/sbothe-iais/ferari/src/master/
38. Boulos, M. N. K., & Al-Shorbaji, N. M. (2014). On the Internet of Things, smart cities and the WHO Healthy Cities.
39. Brown, L. C., & Mac Berthouex, P. (2002). Statistics for environmental engineers. CRC press.
40. Chen, M., & Chang, T. (2011, June). A parking guidance and information system based on wireless sensor network. In Information and Automation (ICIA), 2011 IEEE International Conference on (pp. 601-605). IEEE.
41. Chen, Y., & Han, D. (2018). Water quality monitoring in smart city: A pilot project. Automation in Construction, 89, 307-316.

42. Cheng, B., Longo, S., Cirillo, F., Bauer, M., & Kovacs, E. (2015, June). Building a Big Data platform for smart cities: Experience and lessons from santander. In Big Data (BigData Congress), 2015 IEEE International Congress on (pp. 592-599). IEEE.

43. Cheung, S. Y., & Varaiya, P. P. (2006). Traffic surveillance by wireless sensor networks (Doctoral dissertation, University of California, Berkeley).

44. Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J. R., Mellouli, S., Nahon, K., ... & Scholl, H. J. (2012, January). Understanding smart cities: An integrative framework. In System Science (HICSS), 2012 45th Hawaii International Conference on (pp. 2289-2297). IEEE.

45. Clouder, Kafka in Truckin IoT, https://hortonworks.com/tutorial/kafka-in-trucking-iot-on-hdf/section/1/

46. Compose, Redis Resource and Scaling, https://help.compose.com/docs/redis-resources-and-scaling

47. Cormode, G., Garofalakis, M., Haas, P. J., & Jermaine, C. (2011). Synopses for massive data: Samples, histograms, wavelets, sketches. Foundations and Trends® in Databases, 4(1–3), 1-294.

48. Data Science Central, Apache Beam – Create Data Processing Pipelines, Michael Walker, May 2016, https://www.datasciencecentral.com/profiles/blog/show?id=6448529:BlogPost:427157&utm_content=bufferbf154&utm_medium=social&utm_source=linkedin.com&utm_campaign=buffer

49. Databricks, Apache Spark, https://databricks.com/glossary/what-is-spark-streaming

50. Databricks, Apache Spark, https://databricks.com/spark/about

51. Data-Smart City Solutions, How Cities Are Using the Internet of Things to Map Air Quality, April 2017, https://datasmart.ash.harvard.edu/news/article/how-cities-are-using-the-internet-of-things-to-map-air-quality-1025

52. Deloitte insights, Forces of change: Smart cities, 2018 (https://www2.deloitte.com/insights/us/en/focus/smart-city/overview.html)

53. Deshpande, T. (2017). Learning Apache Flink. Packt Publishing Ltd.

54. Devaraju, J. T., Suhas, K. R., Mohana, H. K., & Patil, V. A. (2015). Wireless portable microcontroller based weather monitoring station. Measurement, 76, 189-200.

55. Devops, PySpark: Operations Overview, May 2016, http://trimc-devops.blogspot.com/2016/05/r-eslient-d-istributed-d-ataset-dataset.html

56. Druid, http://druid.io/

57. Emaldi, M., Peña, O., Lázaro, J., & López-de-Ipiña, D. (2015). Linked open data as the fuel for smarter cities. In Modeling and Processing for Next-Generation Big-Data Technologies (pp. 443-472). Springer, Cham.

58. Ertico, Pisa and Deutsche Telekom launch 6-month smart city pilot project to optimize city parking, September 2017, https://erticonetwork.com/pisa-and-deutsche-telekom-launch-6-month-smart-city-pilot-project-to-optimize-city-parking/

59. Eyecademy, Data Visualisation for IoT: Barcelona's Smart Bins, https://www.eyecademy.com/blog/data-visualisation-for-iot/

60. Fisher, D., DeLine, R., Czerwinski, M., & Drucker, S. (2012). Interactions with Big Data analytics. interactions, 19(3), 50-59.

61. Flouris, I., Giatrakos, N., Deligiannakis, A., & Garofalakis, M. (2020). Network-wide complex event processing over geographically distributed data sources. Information Systems, 88, 101442.
62. Flouris, I., Giatrakos, N., Garofalakis, M., & Deligiannakis, A. (2015, August). Issues in complex event processing systems. In 2015 IEEE Trustcom/BigDataSE/ISPA (Vol. 2, pp. 241-246). IEEE.
63. Flouris, I., Manikaki, V., Giatrakos, N., Deligiannakis, A., Garofalakis, M., Mock, M., ... & Krizan, T. (2016, June). Ferari: A prototype for complex event processing over streaming multi-cloud platforms. In Proceedings of the 2016 International Conference on Management of Data (pp. 2093-2096). ACM.
64. Folianto, F., Low, Y. S., & Yeow, W. L. (2015, April). Smartbin: Smart waste management system. In Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on (pp. 1-2). IEEE.
65. Fujdiak, R., Masek, P., Mlynek, P., Misurec, J., & Olshannikova, E. (2016, July). Using genetic algorithm for advanced municipal waste collection in Smart City. In 2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP) (pp. 1-6). IEEE.
66. Galán-García, J. L., Aguilera-Venegas, G., & Rodríguez-Cielos, P. (2014). An accelerated-time simulation for traffic flow in a smart city. Journal of Computational and Applied Mathematics, 270, 557-563.
67. Garofalakis, M., Gehrke, J., & Rastogi, R. (2016). Data stream management: A brave new world. In Data Stream Management (pp. 1-9). Springer, Berlin, Heidelberg.
68. Gartner, Information Technology (https://www.gartner.com/en/information-technology/glossary/big-data)
69. Genuer, R., Poggi, J. M., Tuleau-Malot, C., & Villa-Vialaneix, N. (2017). Random forests for big data. Big Data Research, 9, 28-46.
70. Giatrakos, N., Alevizos, E., Artikis, A., Deligiannakis, A., & Garofalakis, M. (2019). Complex event recognition in the Big Data era: a survey. The VLDB Journal, 1-40.
71. Giatrakos, N., Artikis, A., Deligiannakis, A., & Garofalakis, M. (2017). Complex event recognition in the big data era. Proceedings of the VLDB Endowment, 10(12), 1996-1999.
72. Giatrakos, N., Deligiannakis, A., Garofalakis, M., & Kotidis, Y. (2018). Omnibus outlier detection in sensor networks using windowed locality sensitive hashing. Future Generation Computer Systems.
73. Giatrakos, N., Deligiannakis, A., Garofalakis, M., Keren, D., & Samoladas, V. (2018). Scalable approximate query tracking over highly distributed data streams with tunable accuracy guarantees. Information Systems, 76, 59-87.
74. Giatrakos, N., Deligiannakis, A., Garofalakis, M., Sharfman, I., & Schuster, A. (2014). Distributed geometric query monitoring using prediction models. ACM Transactions on Database Systems (TODS), 39(2), 16.
75. Giatrakos, N., Kotidis, Y., & Deligiannakis, A. (2010, September). PAO: power-efficient attribution of outliers in wireless sensor networks. In Proceedings of the Seventh International Workshop on Data Management for Sensor Networks (pp. 33-38).
76. Giatrakos, N., Kotidis, Y., Deligiannakis, A., Vassalos, V., & Theodoridis, Y. (2010, June). TACO: tunable approximate computation of outliers in wireless sensor

networks. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (pp. 279-290). ACM.
77. Github, Apache Flink streaming engine, https://github.com/FlinkML/flink-jpmml
78. Github, Datasketch, https://github.com/ekzhu/datasketch
79. Github, GPU based Dynamic k-Nearest Neighbors, https://github.com/artifabrian/dynamic-knn-gpu
80. Github, Parameter Server implementation in Apache Flink, https://github.com/FlinkML/flink-parameter-server
81. Github, Proteus incremental analytics, https://github.com/proteus-h2020/proteus-incremental-analytics
82. Github, Proteus Solma, https://github.com/proteus-h2020/proteus-solma
83. Github, Stream summarizer and cardinality estimator, https://github.com/addthis/stream-lib
84. Github, Streaminer: a collection of algorithms for mining data streams, https://github.com/mayconbordin/streaminer
85. Giuffrѐ, T., Siniscalchi, S. M., & Tesoriere, G. (2012). A novel architecture of parking management for smart cities. Procedia-Social and Behavioral Sciences, 53, 16-28.
86. Google Cloud, Dataflow, https://cloud.google.com/dataflow/
87. Grafana Labs, https://grafana.com/grafana/
88. Grafana vs. Kibana: The Key Differences to Know, February 2020, https://logz.io/blog/grafana-vs-kibana/
89. Hadoop Distributed File System (HDFS). TechTarget, Search Data Management (https://searchdatamanagement.techtarget.com/definition/Hadoop-Distributed-File-System-HDFS)
90. Harrison, C., Eckman, B., Hamilton, R., Hartswick, P., Kalagnanam, J., Paraszczak, J., & Williams, P. (2010). Foundations for smarter cities. IBM Journal of Research and Development, 54(4), 1-16.
91. Hemsoth, Nicole. "Druid Summons Strength in Real-Time", Datanami, 08 November 2012
92. Heraklion Smart City, https://smartcity.heraklion.gr/en/our-vision/
93. High Performance Computing (HPC). Amazon. 31 Jan 2019 (https://aws.amazon.com/hpc/)
94. Hofmann, M., & Klinkenberg, R. (Eds.). (2013). RapidMiner: Data mining use cases and business analytics applications. CRC Press.
95. Hӧjer, M., & Wangel, J. (2015). Smart sustainable cities: definition and challenges. In ICT innovations for sustainability (pp. 333-349). Springer, Cham.
96. Hollands, R. G. (2008). Will the real smart city please stand up? Intelligent, progressive or entrepreneurial. City, 12(3), 303-320.
97. Hu, H., Wen, Y., Chua, T. S., & Li, X. (2014). Toward scalable systems for Big Data analytics: A technology tutorial. IEEE access, 2, 652-687.
98. ITU-T Focus Group on Smart Sustainable Cities: An Overview of Smart Sustainable Cities and the Role of Information and Communication Technologies (ICTs)
99. ITU-T Focus Group on Smart Sustainable Cities: Key performance indicators related to the sustainability impacts of information and communication technology in smart sustainable cities.

100. Jain, A. (2017). Mastering apache storm: Real-time big data streaming using kafka, hbase and redis. Packt Publishing Ltd.
101. Javatpoint, Support Vector Machine Algorithm, https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm
102. Kakadia, D. (2015). Apache Mesos Essentials. Packt Publishing Ltd.
103. K alaitzakis, M., Bouloukakis, M., Charalampidis, P., Dimitrakis, M., Drossis, G., Fragkiadakis, A., Fundulaki, I., Karagiannaki, K., Makrogiannakis, A., Margetis, G., Panousopoulou, A., Papadakis, S., Papakonstantinou, V., Partarakis, N., Roubakis, S., Tragos, E., Ymeralli, E., Tsakalides, P., Plexousakis, D., & Stephanidis, C. (2018). Building a Smart City Ecosystem for Third Party Innovation in the City of Heraklion. In A. Stratigea & D. Kavroudakis (Eds.), Mediterranean Cities and Island Communities: Smart, Sustainable, Inclusive and Resilient (pp. 19-56). Switzerland: Springer International Publishing AG.
104. Kanellou, E., Chrysos, N., Mavridis, S., Sfakianakis, Y., & Bilas, A. (2018, August). GPU Provisioning: The 80-20 Rule. In European Conference on Parallel Processing (pp. 352-364). Springer, Cham.
105. Karim, S., Zhang, Y., Yin, S., & Asif, M. R. (2018, July). An Efficient Region Proposal Method for Optical Remote Sensing Imagery. In IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium (pp. 2455-2458). IEEE.
106. Kelly, F. J., Fuller, G. W., Walton, H. A., & Fussell, J. C. (2012). Monitoring air pollution: Use of early warning systems for public health. Respirology, 17(1), 7-19.
107. Khanna, A., & Anand, R. (2016, January). IoT based smart parking system. In Internet of Things and Applications (IOTA), International Conference on (pp. 266-270). IEEE.
108. Kostakos, V., Ojala, T., & Juntunen, T. (2013). Traffic in the smart city: Exploring the potential of city-wide sensing to augment a traffic control center. IEEE Internet Computing, 1.
109. Kramers, A., Höjer, M., Lövehagen, N., & Wangel, J. (2014). Smart sustainable cities– Exploring ICT solutions for reduced energy use in cities. Environmental modelling & software, 56, 52-62.
110. Kreps, J., Narkhede, N., & Rao, J. (2011, June). Kafka: A distributed messaging system for log processing. In Proceedings of the NetDB (pp. 1-7).
111. Kulkarni, S., Bhagat, N., Fu, M., Kedigehalli, V., Kellogg, C., Mittal, S., ... & Taneja, S. (2015, May). Twitter heron: Stream processing at scale. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (pp. 239-250). ACM.
112. Kumar, A., & Prakash, A. (2014). The role of Big Data and analytics in smart cities. Int J Sci Res (IJSR), 6(14), 12-23.
113. Labrinidis, A., & Jagadish, H. V. (2012). Challenges and opportunities with Big Data. Proceedings of the VLDB Endowment, 5(12), 2032-2033.
114. Lamda Architecture, http://lambda-architecture.net/
115. Lanza, J., Sotres, P., Sánchez, L., Galache, J. A., Santana, J. R., Gutiérrez, V., & Muñoz, L. (2016). Managing large amounts of data generated by a smart city internet of things deployment. International Journal on Semantic Web and Information Systems (IJSWIS), 12(4), 22-42.
116. Lin, T., Rivano, H., & Le Mouël, F. (2017). A survey of smart parking solutions. IEEE Transactions on Intelligent Transportation Systems, 18(12), 3229-3253.

117. Living Asean, https://livingasean.com/explore/3-apps-check-air-pollution-levels-pm2-5-pm10/

118. Lucas, J., Idris, Y., Contreras-Rojas, B., Quiane-Ruiz, J. A., & Chawla, S. (2018, April). RheemStudio: Cross-Platform Data Analytics Made Easy. In 2018 IEEE 34th International Conference on Data Engineering (ICDE) (pp. 1573-1576). IEEE.

119. Mahajan, K., & Chitode, J. S. (2014). Waste bin monitoring system using integrated technologies. International Journal of Innovative Research in Science, Engineering and Technology, 3(7).

120. Medvedev, A., Fedchenkov, P., Zaslavsky, A., Anagnostopoulos, T., & Khoruzhnikov, S. (2015, August). Waste management as an IoT-enabled service in smart cities. In Conference on smart spaces (pp. 104-115). Springer, Cham.

121. MentalFloss, How much land in Los Angeles is dedicated to parking spaces?, March 2016, http://mentalfloss.com/article/77143/how-much-land-los-angeles-dedicated-parking-spaces

122. Mobile Environmental Sensor Systems Across a Grid Environment - the MESSAGE Project (https://ercim-news.ercim.eu/en68/special/mobile-environmental-sensor-systems-across-a-grid-environment-the-message-project)

123. Mobilesyrup, UVLens helps protect against the harshness of the sun, August 2017, https://mobilesyrup.com/2017/08/20/uvlens-app-of-the-week/

124. Mora, L., Bolici, R., & Deakin, M. (2017). The first two decades of smart-city research: A bibliometric analysis. Journal of Urban Technology, 24(1), 3-27.

125. Nam, T., & Pardo, T. A. (2011, June). Conceptualizing smart city with dimensions of technology, people, and institutions. In Proceedings of the 12th annual international digital government research conference: digital government innovation in challenging times (pp. 282-291).

126. Nature, The big challenges of big data, June 2013, https://www.nature.com/articles/498255a

127. Nerantzaki, S., Moirogiorgou, K., Efstathiou, D., Giannakis, G., Voutsadaki, S., Zervakis, M., ... & Nikolaidis, N. P. (2015, April). Development of an Integrated Suspended Sediment Sampling System-Prototype Results. In EGU General Assembly Conference Abstracts (Vol. 17).

128. New York City’s agency (DOT). Traffic signals (https://www1.nyc.gov/html/dot/html/infrastructure/signals.shtml)

129. Ng, E. H., Tan, S. L., & Guzman, J. G. (2009, February). Road traffic monitoring using a wireless vehicle sensor network. In Intelligent Signal Processing and Communications Systems, 2008. ISPACS 2008. International Symposium on (pp. 1-4). IEEE.

130. Nguyen, T. T., Ngo, H. H., Guo, W., Wang, X. C., Ren, N., Li, G., ... & Liang, H. (2018). Implementation of a specific urban water management-Sponge City. Science of the Total Environment.

131. Nordsense, Smart Containers, https://nordsense.com/overview/

132. Ntoutsi, I., Mitsou, N., & Marketos, G. (2008). Traffic mining in a road-network: How does the traffic flow?. International Journal of Business Intelligence and Data Mining, 3(1), 82-98.

133. Odendaal, N. (2003). Information and communication technology and local governance: understanding the difference between cities in developed and emerging economies. Computers, Environment and Urban Systems, 27(6), 585-607.
134. OPENSENSE, Community-based sensing using wireless sensor network technology to monitor air pollution (https://gitlab.ethz.ch/tec/public/opensense/wikis/home)
135. Opensource, A guide to Apache's Spark Streaming, https://opensource.com/business/15/4/guide-to-apache-spark-streaming
136. Penza, M., Suriano, D., Villani, M. G., Spinelle, L., & Gerboles, M. (2014, November). Towards air quality indices in smart cities by calibrated low-cost sensors applied to networks. In SENSORS, 2014 IEEE (pp. 2012-2017). IEEE.
137. Perkins, L., Redmond, E., & Wilson, J. (2018). Seven databases in seven weeks: a guide to modern databases and the NoSQL movement. Pragmatic Bookshelf.
138. Polycarpou, E., Lambrinos, L., & Protopapadakis, E. (2013, June). Smart parking solutions for urban areas. In 2013 IEEE 14th International Symposium on (pp. 1-6). IEEE.
139. Prekopcsak, Z., Makrai, G., Henk, T., & Gaspar-Papanek, C. (2011, June). Radoop: Analyzing big data with rapidminer and hadoop. In Proceedings of the 2nd RapidMiner community meeting and conference (RCOMM 2011) (pp. 1-12).
140. Prima Consulting UK, Blog – Qlik View vs. Glik Sense, https://primaconsulting.co.uk/news/2018/3/26/qlikview-vs-qliksense
141. Qlik, Qlik Analytics Platform overview, https://help.qlik.com/en-US/sense-developer/February2019/Subsystems/Platform/Content/Sense_PlatformOverview/Architecture/qlik-analytic-platform.htm
142. Rabl, T., & Jacobsen, H. A. (2012). Big Data generation. In Specifying Big Data Benchmarks (pp. 20-27). Springer, Berlin, Heidelberg.
143. Redis, Introductio to Redis, https://redis.io/topics/introduction
144. Robles, T., Alcarria, R., de Andrés, D. M., Navarro, M., Calero, R., Iglesias, S., & López, M. (2015). An IoT based reference architecture for smart water management processes. JoWUA, 6(1), 4-23.
145. Science Soft, IoT for Smart Cities: Use Cases and Implementation Strategies, https://www.scnsoft.com/blog/iot-for-smart-city-use-cases-approaches-outcomes
146. Sensoneo, Waste Management Solution for City: https://sensoneo.com/smart-waste-management-for-cities/
147. Shah, J., & Mishra, B. (2016, January). IoT enabled environmental monitoring system for smart cities. In 2016 International Conference on Internet of Things and Applications (IOTA) (pp. 383-388). IEEE.
148. Shahanas, K. M., & Sivakumar, P. B. (2016). Framework for a smart water management system in the context of smart city initiatives in India. Procedia Computer Science, 92, 142-147.
149. Sifuentes, E., Casas, O., & Pallas-Areny, R. (2011). Wireless magnetic sensor node for vehicle detection with optical wake-up. IEEE Sensors Journal, 11(8), 1669-1676.
150. Sinaeepourfard, A., Garcia, J., Masip-Bruin, X., Marín-Tordera, E., Cirera, J., Grau, G., & Casaus, F. (2016, June). Estimating Smart City sensors data generation. In 2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net) (pp. 1-8). IEEE.

151. Smart cities and the weather. Meeting of the Minds, IBM (https://meetingoftheminds.org/smart-cities-weather-22100)
152. Smart Dublin, https://data.smartdublin.ie/
153. Spark – A modern data processing framework for cross platform analytics. Deploying Spark on HPE Elastic Platform for Big Data Analytics. Hewlett Packard Enterprise. (https://h20195.www2.hpe.com/V2/getpdf.aspx/4AA6-8143ENW.pdf)
154. Staricco, L. (2013). Smart Mobility: opportunità e condizioni. Tema. Journal of Land Use, Mobility and Environment, 6(3), 342-354.
155. Statista, https://www.statista.com/statistics/270860/urbanization-by-continent/
156. Stockholms stad, How the smart city develops, https://international.stockholm.se/governance/smart-and-connected-city/how-the-smart-city-develops/
157. Tableau, https://www.tableau.com/
158. TendersOnTime, Emerald Drinking Water Supply Networks, https://www.tendersontime.com/blogdetails/emerald-drinking-water-supply-networks-24272/
159. The Big-Data Ecosystem Table, http://bigdata.andreamostosi.name/
160. The Digitization of the World, From Edge to Core, David Reinsel, John Gantz, John Rydning, An IDC white paper, Data Age 2025, Nov 2018 (https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf)
161. The Local, Spain, Spain's Santander hailed as global pioneering 'smart city', https://www.thelocal.es/20160409/spains-santander-becomes-global-pioneering-smart-city
162. The New Stack, All the Apache Streaming Projects: An Exploratory Guide, https://thenewstack.io/apache-streaming-projects-exploratory-guide/
163. The New York Times, To Fight Gridlock, Los Angeles Synchronizes Every Red Light, https://www.nytimes.com/2013/04/02/us/to-fight-gridlock-los-angeles-synchronizes-every-red-light.html
164. Tibco Community, https://community.tibco.com/questions/apache-kafka-messaging-seems-be-having-interesting-feature
165. Tibco community, Random Forest Template for TIBCO Spotfire, https://community.tibco.com/wiki/random-forest-template-tibco-spotfirer-wiki-page
166. Tibco ComputeDB, https://tibco-computedb.readthedocs.io/en/docv1.1.0/
167. Tibco, https://www.snappydata.io/product
168. Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., ... & Bhagat, N. (2014, June). Storm@ twitter. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data (pp. 147-156). ACM.
169. Towards Data Science, Hands on Apache Beam, building data pipelines in Python, November 2018, https://towardsdatascience.com/hands-on-apache-beam-building-data-pipelines-in-python-6548898b66a5
170. Treboux, J., Jara, A. J., Dufour, L., & Genoud, D. (2015, March). A predictive data-driven model for traffic-jams forecasting in smart santader city-scale testbed. In 2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW) (pp. 64-68). IEEE.

171. Tutorialspoint, Redis – Publish Subscribe, https://www.tutorialspoint.com/redis/redis_pub_sub.htm
172. Ubidots, 4 Big IoT Wins for Smart City Pioneer, Santander, Spain, April 2017https://ubidots.com/blog/smart-city-pioneer-santander-spain/
173. Uddin, M. F., & Gupta, N. (2014, April). Seven V's of Big Data understanding Big Data to extract value. In American Society for Engineering Education (ASEE Zone 1), 2014 Zone 1 Conference of the (pp. 1-5). IEEE.
174. Vasudewa, K., & Gupta, P. (2016, February). A survey on elastic resource allocation algorithm for cloud infrastructure. In 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH) (pp. 199-203). IEEE.
175. What makes Apache Flink the best choice for streaming applications, April 2018, https://hackernoon.com/what-makes-apache-flink-the-best-choice-for-streaming-applications-fc377858a53
176. Wikipedia, Apache Beam, https://en.wikipedia.org/wiki/Apache_Beam
177. Wikipedia, Cumulant, https://en.wikipedia.org/wiki/Cumulant
178. World Health Organization, Climate change and human health, Urbanization and health (https://www.who.int/globalchange/ecosystems/urbanization/en/)
179. World Health Organization: Ambient air pollution – a major threat to health and climate (https://www.who.int/airpollution/ambient/en/)
180. World population prospects 2019, United Nations, Department of Economic and Social Affairs.
181. Xu, J., Chen, Z., Tang, J., & Su, S. (2014, June). T-storm: Traffic-aware online scheduling in storm. In 2014 IEEE 34th International Conference on Distributed Computing Systems (pp. 535-544). IEEE.
182. Yang, F., Merlino, G., Ray, N., Léauté, X., Gupta, H., & Tschetter, E. (2017). The RADStack: Open source lambda architecture for interactive analytics.
183. Yi, W., Lo, K., Mak, T., Leung, K., Leung, Y., & Meng, M. (2015). A survey of wireless sensor network based air pollution monitoring systems. Sensors, 15(12), 31392-31427.
184. Zhang, Q., Wang, G., Gong, K., Li, B., Meng, M. Q. H., & Song, S. (2013, August). Wireless magnetic sensor node for vehicle detection using finite element simulation. In Information and Automation (ICIA), 2013 IEEE International Conference on (pp. 248-251). IEEE.
185. Zoomdata, Modern BI and Data Visualization Platform, https://www.zoomdata.com/product/