An Integrated Architecture for Future Studies in Data Processing for Smart Cities

Cristian Chilipirea[a], Andreea-Cristina Petre[a], Loredana-Marsilia Groza[a], Ciprian Dobre[a], Florin Pop*[a]

[a] *Computer Science Department, Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Romania*

Abstract

Data processing for Smart Cities become more challenging, facing with different handling steps: data collection from different heterogeneous sources, processing sometimes in real-time and then delivered to high level services or applications used in Smart Cities. Applications used for intelligent transportation systems, crowd management, water resources management, noise and air pollution management, require different data processing techniques. The main subject of this paper is to propose an architecture for data processing in Smart Cities. The architecture is oriented on the flow of data from the source to the end user. We describe seven steps of data processing: collection of data from heterogeneous sources, data normalization, data brokering, data storage, data analysis, data visualization and decision support systems. We consider two case studies on crowd management in smart cities and on Intelligent Transportation Systems (ITS) and we provide experimental highlights.

Keywords: architecture; big data; data processing; crowd sensing; crowd dynamics; intelligent transportation systems

1. Introduction

More and more applications today use, generate and handle very large volumes of data. In particular, this is true for Smart City applications, which attract a rapidly increasing interest from government, companies, citizens, developers, scientists, etc. They cover a large spectrum of needs in public safety, water and energy management, smart buildings, government and agency administration, social programs, transportation, health, education. They are fed with huge amounts of input data, in various formats, from a continuously increasing number of sources (sensors, governmental, regional, and municipal sources, citizens, public open data sources, etc.), and are described by a complex workflow

*Corresponding author, Tel.: +40-723-243-958; Fax: +40-318-145-309; *Email address*: `florin.pop@cs.pub.ro`.

and in many cases impose real-time processing capabilities, useful in decision taking.

The large volume of data coming from a variety of sources and in various formats, with different storage, transformation, delivery or archiving requirements, complicates the task of context data management. At the same time, fast responses are needed for real-time applications. Despite the potential improvements of the Smart City infrastructure, the number of concurrent applications needing quick data access will remain very high. With the emergence of the recent cloud infrastructures, achieving highly scalable data management in such contexts is a critical challenge, as the overall application performance is highly dependent on the properties of the data management service.

Extracting valuable information from raw data is especially difficult considering the velocity of growing data from year to year and the fact that 80% of data is unstructured. In addition, data sources are heterogeneous (various sensors, users with different profiles, etc.) and are located in different situations or contexts. This is why the Smart City infrastructure runs reliably and permanently to provide the context as a "public utility" to different services. Context-aware applications exploit the context to adapt accordingly the timing, quality and functionality of their services. The value of these applications and their supporting infrastructure lies in end-users always operating in a context: their role, intentions, locations and working environment constantly change.

As the scale, complexity and dynamism of distributed systems is dramatically growing, their configuration and data management have started to become a limiting factor of their development. This is particularly true in the case of Cloud is used for data storage and also for data processing, where the task of managing hundreds or thousands of nodes while delivering highly reliable services entails an intrinsic complexity. Furthermore, Cloud computing introduces another challenge which impacts on the resource management decisions. In these contexts, self-management mechanisms have to take into account the cost-effectiveness of the adopted decisions.

Considering all of these aspects, the main subject of this paper is to propose an architecture for Big Data processing in Smart Cities. The architecture is oriented on the flow of data from the source to the end user. We describe seven steps of data processing: collection of data from heterogeneous sources, data normalization, data brokering, data storage, data analysis, data visualization and decision support systems. We describe two case studies on crowds' management in smart cities and on Intelligent Transportation Systems (ITS).

The paper is structured as follows. Section 2 presents the related work on crowd data smart cities and on ITS. The proposed architecture is presented in Section 3. Two use cases are described in Section 4. Then, the experiments obtained for these use cases are presented in Section 5. The paper ends with conclusions and future work presented in Section 6.

2. Related Work

Smart Cities [1] represent an important goal which can dramatically improve the life of citizens. There is a lot of research aiming to get us closer and closer to this goal. The idea of a smart city is in accordance to other movements in research such as Internet of Things [2, 3, 4] and Big Data [5]. New York times actually declared this period the "Age of Big Data" [1].

In order to enable Smart Cities technologies such as Internet of Things, Wireless Sensor Networks [6] and Crowd Sensing [7] are the catalysts providing data about our cities. The need for sensing in Smart Cities is explored in [8]. This data needs to be processed often using Big Data techniques in order to extract the information required to make decisions about the cities. This information and the decisions are then used in order to inform the citizens to take certain actions or to activate actuators for enabling automatic processes. A good example where actuators can improve Smart Cities is given by the management of green spaces [9].

Probably the most important issues addressed in order to build Smart Cities are the ones of Crowd Dynamics [10]. In order to understand Crowd Dynamics, we need data on the movements of as many people as possible. These movements need to be recorded for both pedestrians [11] and for vehicles [12]. The problem of tracking is not solved in any of the two scenarios. This is surprising, considering the problem of tracking a particular individual is usually solved by the use of GPS [13]. However, GPS requires user participation which is difficult to obtain, in contrast WiFi [14] or cellular methods [15] can be used to gather data on large crowds. These systems also do not work indoors and require the cooperation of the individual being tracked in order to generate a position estimate.

It is important to treat both indoor and outdoor cases when considering human mobility. This is because modern vital facilities, such as hospitals, which are part of the backbone of many cities consist of large areas with multiple buildings. An example of how the dynamics inside these facilities can be used in order to improve the layout of the facility is given by Ruiz et al [16]. Similarly, Universities campuses, another type of large facilities at the core of cities, are analyzed [17], [18] in order to better understand the dynamics inside them.

Crowd tracking experiments are taking place in a wider variety of places like mass events [19] or festivals [20]. They are also used in order to measure queues using only WiFi signals [21]. This queue can represent waiting time at a counter, which directly affects customer experience or the movement through security lines at an airport [22].

Crowd Sensing can be used in order to extract all types of data for smart cities. A powerful example is given by the authors of [23] where students are asked to take pictures of plants around the campus. The pictures are then analyzed by scientists in order to better understand the status of flora. Projects

[1] "The age of big data" - Steve Lohr, New York Times, 11, 2012

like this could potentially be used at the scale of a city in order to measure a large variety of features. It is not always necessary for people to be active in their participation of data gathering. Passive systems require only their presence in the monitored location, which can even be obtained in an opportunistic manner. Whenever any citizens carrying the scanner walks or drives on a specific street data about the street can be gathered. In this way maps can be enhanced with features [24] such as roundabouts or pot-holes. Diverse uses include even earthquake detection [25] and soon maybe even the detection of effects produced by these large natural disasters.

There are many projects and platforms targeted directly at crowd sensing: Medusa [26], Matador [27], Mosden [28] and mCrowd [29]. And these platforms already implement important features for Smart Cities such as crowd sources new reporting [30] but they do not yet combine the data sets or offer a method to analyze the data in order to extract information hidden inside it. This type of information represents answers to questions that we don't yet have and they can currently only be obtained by using Big Data techniques.

The data gathered from all these systems is usually analyzed by experts or scientist manually. This is the case for [16], where categorization of individuals into different groups such as patients or staff is done by using rules built by experts. More information can be extracted from these data sets if they are combined and Big Data systems are used to process them.

Real-time processing is used to designate a category where the job outcome is needed as fast as possible, and usually the task itself is not something taking a long time to process. These systems can be categorized as hard or soft. A Hard real-time system is an OS for a nuclear plant or a plane. Tasks must be scheduled and completed fast because otherwise a catastrophe could happen. These systems are usually governed by hard deadlines and the scheduler must insure they are achieved. Soft real-time systems are the ones like hotel booking or video streaming sites [31]. The answers must be delivered fast to the customers, but a delayed frame now and then cannot lead to disastrous results. One article which explores this type of hard real-time scheduling is [32]. In the paper the authors try to improve the scheduling capabilities of a system by also adding security checks to the incoming jobs. The added module can detect threats brought by snooping, alteration of spoofing and can be easily added to any real-time scheduler. Their security module name SAREC (security-aware real-time heuristic strategy for clusters) integrates with the popular Earliest Deadline First algorithm to create a security aware scheduler named SAEDF. Although the matter of securing the interactions between the users and the cluster infrastructure is important, in our case a large portion of these measures could be implemented in an intermediate cluster proxy module if needed, with little overhead to the job itself. By using a proxy to mediate all user-cluster interactions we can alleviate a large number of security risks. If a user has a malicious intent and manages to submit a job that poses a security risk, runing all jobs in virtual machines on the cluster infrastructure will limit the damages to only the users task.

Another example of real-time processing and scheduling [33]. The authors talk about the problem of soft real-time scheduling in rendering 3D images inside the Google Earth software. The Google Earth software allows one to navigate anywhere in the world and has multiple viewing modes from virtual 3D renderings to satellite imagery. A frame is a static 2D representation, rendered on the screen at a given time. To ensure a smooth navigation experience, at least 60 of these frames must be rendered on the users' screen in a second. When a scheduling deadline is not met, the previous frame is redisplayed causing the application to "stutter". In order to alleviate the problems, the authors have devised a new algorithm that better estimates rendering time on multiple devices, in order to improve scheduler accuracy. We are in particular interested in their scheduling model and discovered they also abstracted some of the events into "single-active sporadic tasks" (triggered by a specific rendering phase) and "soft real-time aperiodic tasks (triggered by receiving new imagery through network)". We will use similar terms to define the submit patterns and properties of different types of jobs.

Talking about the arrival patterns of the jobs, the authors from [34] build a common approach to schedule static and dynamic tasks, in a system which also has to deal with hard real-time deadlines. They divide their tasks in 3 categories, based on their arrival pattern and number of instances they require for running. Aperiodic tasks need only one instance to run, and can enter the system at any time. Both periodic and sporadic tasks require multiple instances to run, but while the former come at a specific interval of time, the latter can be submitted like the aperiodic tasks, at any time, but no sooner than a specified interval. The authors have extended a previous static time-based scheduling algorithm into a dynamic version which constantly changes the expected start and end time of jobs while still keeping the end time in the necessary deadline. They have thus provided two versions of their scheduler, one accepting aperiodic tasks without affecting the existing task instances deadline, and another, with the same properties, accepting periodic and sporadic tasks. Before accepting any task, a formal schedulable test is run, to see if the system can handle the tasks deadline. If not, it is rejected. The scheduled tasks are considered to be preemptive, and a list of static tasks known beforehand is expected to be provided at system startup. To account for dependencies between tasks, start and end times are parameterized instead of being given a fixed value.

We also investigated solutions related to intelligent transport systems, since this is the type of workload we are going to test our scheduler on. The [35] project tries to act as a hub for such endeavors in order to help each of the individual current ITS system grow and communicate through a common point of contact. These systems are increasingly important since optimizing traffic can also reduce CO_2 emissions along with the benefits brought to all the inhabitants of a city. Current implemented solutions are mostly proprietary and involve infrastructure changes. There are a number of existing solutions trying to estimate the state of the traffic, ranging from sensors in the road, to GPS systems on cars, to cameras interpreting images. Indifferent of the chosen solution, all of these systems will generate a large amount of data which has to be interpreted city-wide. Although our solution uses a small part of this data, it could grow and adapt to provide the necessary analysis needed to drive an intelligent city of today.

3. Data Flow based Architecture

We propose an architecture which contains several steps represented by the flow of data from the source to the end user. The data represents the input of our system, it is used to create valuable good for the users, usually in the form of information or automatic actions.

We created a seven step architecture to accomplish our goal to make from data a value (see Figure 1): first we need to aggregate the data sources, then we need to perform data normalization, but before doing that we need to anonymize the data which comes from personal devices. The next step is to create a context for the gathered data and after that we should send it to be stored and processed in a parallel and distributed way. The result of the processing will provide the starting point for data analysis which will generate the patterns and discover the insights we need. In the end all the findings need to be visualized in an advanced style to empower the decision makers.

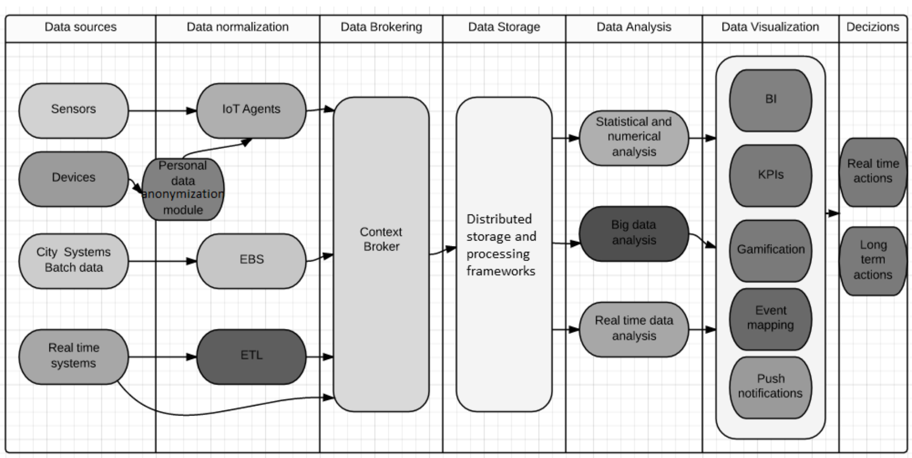

Figure 1: Proposed Architecture.

Data comes from different sources and we need to collect it from everywhere: smart sensors, personal devices, batch data from city systems, real-time systems in order to be able to extract as much knowledge as we can from it. When we combine different data sources related to the same context then we can get more insights from it and this empowers decision makers to minimize the risks.

We have plenty of law constraints and each country has its own regulations regarding data privacy so we need to address this important issue because when a user decided to contribute to a system he needs to be sure he remains anonymous and other users cannot trace him back starting from the pieces of data provided by him. We need to make sure he cannot be identified from a group of users which contribute to the system with their data by using techniques and algorithms for data anonymization. Data collection needs to be done while considering data privacy. This can be achieved by using a data anonymization module. Anonymization needs to be done as close to the source as possible to avoid any data leaks, that could identify an individual as a data provider.

Real-time systems data needs to be normalized by using ETL - Extract, Transform and Load - representing 3 database functions joined in one tool in order to get the data out from a database and introduce it into another one.

City systems batch data needs to be normalized by EBS - Electronic Batchload Service –which is an "Online Computer Library Center" service permitting to the batch load participants to send data to it over the Internet [36, 37].

In the next step the data reaches the context broker which takes individual pieces of data and puts them into a relevant context. A context broker is represented by a service which needs to gather context data from different types, sources and velocity, then it needs to create the conditions, integrate the data, create the rules to be able to provide prepared context data. A certain piece of data is meaningful only in appropriate relation to other pieces of data, which happens only in a given context.

After we created the links between the data and the context information we send the data to the distributed storage and processing frameworks. In order to create a powerful platform for big data processing we need to combine the patterns extracted from the batch data processing with the speed from real-time data processing. The main idea is to bring together real-time data processing and batch processing when dealing with large data sets.

We proposed two well-known frameworks to be used in this step of the data flow: Hadoop, which is focused on batch data processing and Storm, which handles real-time data processing. Hadoop, architected around batch processing, is the most popular open-source software framework for distributed storage and distributed processing of big data on clusters. The main advantages are given by being designed to be fault-tolerant, it is highly scalable and cost effective. The main components of Hadoop are the storage called Hadoop Distributed File System (HDFS), and the processing part called MapReduce. Real-time data processing involves a continuous input, process and output of data. Data must be processed in a small time period (or near-real-time) so we recommend to use Storm because it is a free, open source, distributed real-time system that can compute over a million tuples per second on each node. Other big advantages are given by the scalability, fault-tolerance and by guaranteeing the data processing. Also it is simple to set up, utilize, and integrate with other queueing and database technologies, which is a big plus especially when you need to create a big data platform for smart cities.

When processing the data, we can perform big data analytics, statistical and numerical analysis or real-time data analysis to gain valuable assets. In the end, we need advanced data visualizations to enable the user to take the right decisions, or to make long term actions based on historical data.

Real-time data processing and analytics allows decision makers the opportunity to take immediate action when it is required and batch data processing makes the results to be more accurate due to the patterns discovered and then applied in real-time to get more relevant data.

We need to combine the data from multiple sources to be able to predict future events in order to respond in an efficient way to make a difference. It is important to engage the users and to achieve this we need to empower and motivate them. A way of smart user engagement and advanced visualization is gamification. For example, users of a mobile application can share status about how much do they recycle different things, or how much CO_2 they produced based on how many km they were driving in a day, and enter in competitions with others on social media.

4. Architecture Use Cases

4.1. Intelligent Transportation Systems

Large cities present many problems with their systems, but only transportation system entertains the dynamics of this environment. Currently, it cannot cope anymore with the enormous number of cars driven on its streets using classical traffic systems. Any problems like congestion, accidents, high fuel consumption, pollution, etc. which affect us daily in a city can have as root causes the bad usage of current infrastructure or not enough streets for current traffic flow.

Trying to solve the second cause is a temporary solution due to the continuously increasing cars' number, because any new street added will move the problems from one street to another or in short time if the city area which presents these issues will bring more traffic to it once new streets will be added. Also, adding new streets for vehicular traffic is very hard to be done in cities' centers. The majority of the problems encountered by citizens of a metropolis in traffic are especially determined by the bad traffic planning or by the lack of traffic control systems. Before to try to extend current streets infrastructure of a city, it has to be checked if the largest part of its roads are used at their maximum traffic flow as much as possible and then to try another expensive solution.

The congestion type presents its particularities for each city not having a predefined pattern, but using different information for city infrastructure layout, drivers' behavior and habits etc. together with proper traffic prediction systems, it can be realized a generic traffic system which diminishes the congestion. The majority of navigators guides the user during its ride based on the decisions taken locally not having the global perspective about traffic from the areas that are crossed by the vehicle or about the other participants' decisions. All routing applications from cars see the same traffic events in the above scenario and all from the same area choose locally the same optimum alternative road. For instance, if there is a congestion event in same area of a city, all cars being around see it and compute their routes to the same alternative roads moving indirectly the congestion to the routes' roads.

Intelligent Traffic Control Systems (ITCS) are designed to reduce the global level of congestion in a city, by sensing the city environment through streets infrastructure and the traffic participants counting on Inter-Vehicle-Communication (IVC) and Road-to-Vehicle Communication (RVC) in order to exchange data about roads congestion level, cars' speed, cars' direction, etc. ITCS is able to collect complete information about traffic in a large city, because it exchanges data with various entities from road infrastructure and traffic participants. In order to perform traffic optimization, this system is realized to support three phases (traffic monitoring and data collecting; driving conditions perspective built using the traffic model; traffic controlling by offering to participants' feedback/new routes and controlling the World Transportation Laws (WTLs) to improve the traffic flow.

The ITCS' key entities involved directly in the traffic are cars which are the only component from the traffic flow which behaves according to the driver's decisions. Their main target is to collect data from the environment and then to exchange it with the other traffic participants and infrastructure. They can collect data using the sensors from incorporated navigators or using smartphones (e.g. GPS, accelerometer, barometer, etc.). Offering data to the system, they obtain feedback about traffic in real-time and also new routes suggestions. The local decision capability is used only when they do not have possibility to communicate to the other system entities in order to receive a new route in exchange of the provided data, instead the global routing decisions are coordinated by servers.

4.2. Smart Cities and Crowds

As to our knowledge, there are no complete architectures for crowd sensing or crowd tracking taking into account processing of the data information extraction using Big Data techniques.

The architecture we presented in the previous section is well suited for crowd applications. In order to show this, we detail each of the major parts of the architecture and show how they can be mapped for a simple crowd tracking system using WiFi scanners.

Crowd tracking using WiFi scanners is based on the ubiquitousness of smartphones. These devices now have powerful processing, a large variety of sensor and communication capabilities. Most importantly for our application they almost always have a WiFi module. The WiFi module sends 802.11 packets in order to perform communication or auxiliary functions such as searching for networks. Because most of these packets contain a device identifier in the form of the MAC address, this means a device can be tracked by deploying WiFi scanners which record packets [19].

By looking at the architecture the WiFi scanners represent the sensors which gather data about the movements of crowds. This data needs to first be cleaned and filtered [20] as not all packets can be considered useful detections of a device. This initial cleaning and filtering procedures take place both at the scanners themselves in order to minimize bandwidth usage and at the central server that gathers data from all the scanners. This represents the second step in the architecture.

After the data from the WiFi scanners is cleaned, normalized and standardized form it can be directly correlated with context data. There are numerous sources of context data freely available on the Internet. The simplest examples of context data sources are schedules or news posts. Both schedules and news posts offer a clear reasoning behind certain movements, for instance they can explain why a shop area has a lot of movement during work days and almost none in the weekend or during an important event.

Having multiple data sources and a continuous flow of information which can be correlated with historical events imposes the need for long term storage. Both context and sensor data is stored as well as any correlations between them. This data can be then analyzed in real-time or at a set time. The storage and data analysis steps match the next steps in our architecture.

Finally, after the data is analyzed visualization tools need to be used in order to create an accessible way of making sense of the data for the individuals that need it. In the case of crowd tracking data visualization can take many forms. Usually it takes the form of a map where the density of people is shown by varying color or intensity. More information can be displayed in the form of a city map such as flows of people or events that happen at particular locations. Some decisions can skip the visualization step and directly announce the user. For instance, if a traffic jam is detected people can be automatically informed in order for them to avoid the affected area.

5. Experiments

The first use case considers the Intelligent Transportation systems. The application model is as follow. The cabs are viewed as clients, which generate data with a sporadic schedule in a variety of sizes. The car GPS position is recorded every 15 seconds, and by default, the cluster client on the car sends the last 4 known positions every minute. However, if a car experiences a loss of connectivity, it may exhibit a pause in generating jobs and submit a larger data task when connectivity is reestablished. These tasks are considered as real-time ones, and the aggregated data is computed as soon as possible. More experimental results have been presented in [38].

A step by step workflow of the implemented application respect the model presented in Figure 1, as follows:

- A client sends position information to Cluster Proxy;
- The Cluster Proxy writes cab data to distributed file system;
- The Cluster Proxy encapsulates client data and puts a job in appropriate scheduler queue;
- The scheduler finds available cluster resources and creates a job container on a node;

- The map process reads data from the distributed file system and processes it;
- The map process aggregates new data with old data from distributed database or creates new DB entry if client is at its first report;
- The map process writes data back on the distributed database environment;
- The job is finished and resources are freed.

The flow of a request from inception until the end of its processing, from the technological point of view is as follows:

- Client thread reads positions from file and, depending on the profile it is assigned at startup, starts acting like a normal, mixed, or batch client;
- Proxy receives JSON through Camel;
- Proxy writes the data onto HDFS;
- Proxy triggers a new Hadoop job and submits it to the appropriate queue;
- Map process reads input HDFS data;
- Map reads existing data from HBase and aggregates it with the new data;
- Map writes end result back to HBase.

Our experimental setup consists of 4 Virtual Box machines on top of a single physical host. The host has a 4-core CPU with hyper threading at 2.4/max 3.4 GHz, SSD drive and 16 GB of memory. Out of the 4 virtual machines, 3 were kept purely for computation and storage needs (Data nodes in the case of HDFS) and one was considered a master machine, which ran all the master nodes in the Hadoop architecture and also ran the Cluster Proxy module. The virtual machines had 20 GB of storage assigned, 2 CPUs, and 3 GB of memory, out of which 2 GB were assigned for yarn containers in the case of the slaves.

The Hadoop Scheduler was configured with he following capacity parameters: The batch queue gets 30% of the capacity and may dynamically grow to no more than 60%, and the real-time queue has 70% of the capacity, but no more than 90% if the batch queue is underutilized.

The experimental results are presented in Figure 2. The experiments were run with 1 and 2 queues. We can see that the processing time for batch jobs became comparable with time for real-time jobs. In conclusion, we can combine these type of jobs, without any performance decreasing. Moreover, by interpreting the average processing time, it is clearly that the performance of the cluster is best when the pattern of the input is similar to the one it was designed for. Although its resource limitations are flexible, they do not cater for extreme situations when the load is clearly not balanced. This problem could

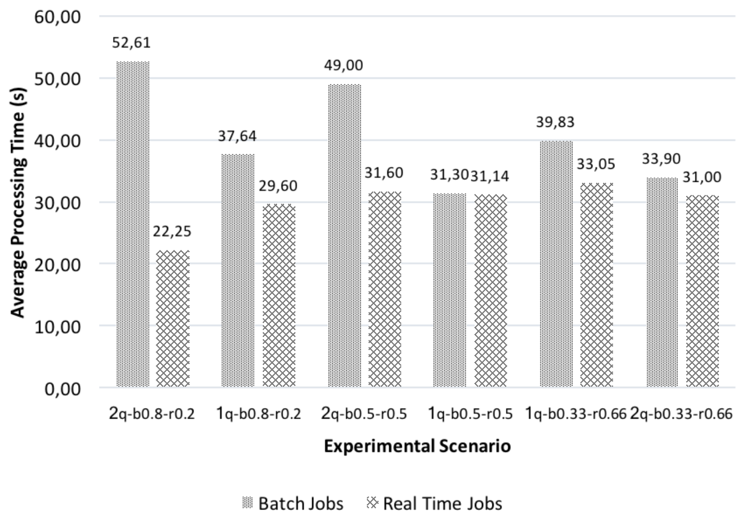

Figure 2: Comparison of average time for real-time and batch processing for different scenarios. These are total times, including data transfers time, time of data writing in HDFS and processing time, which require access to large data-sets collected from cabs.

be solved with greater flexibility in resource limitations, as we imposed a rather fixed margin of resource distribution in configuring the scheduler.

Secondly we looked at crowd sensing data. We were interested to see what information the architecture can provide given an extensive data set. The data set we used was the roma-taxi data set available on Crowdad. This data set consists of timestamped GPS data from multiple taxis that travel around the city of Rome following their normal routines.

We imagine a future on which any car and in this case any taxi is equipped not only with the necessities of every day transport but with sensors which are able to provide all types of data. In order to understand how the data spans across the city we measured how popular each individual part of the city is for taxis. The data source for our architecture is given by the taxi GPS sensors. This data is cleaned and normalized in the second part of the architecture. For example, all positions outside the city limits are removed. After the data is stored we move to the data analysis. We split the city in a grid of 100x100 and count the number of items with GPS coordinates in each of the grids. In Figure 3 we displayed the results. This is equivalent to the data visualization part of our network. Red marks areas with high density and yellow the ones with low density.

Another visualization is available in Figure 4. Here we visualize the same data but we set the maximum values as the maximal ones in the data set. This permits us to accurately identify the center of the city, the most popular area.

Using these visualizations an individual can then start the decision processes.

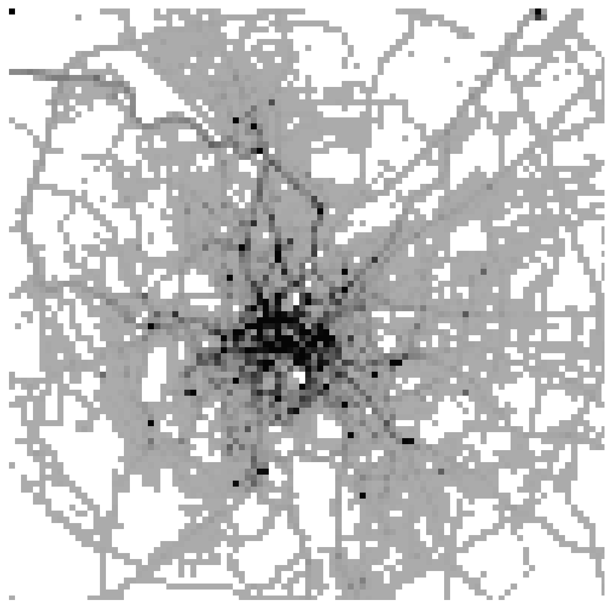

Figure 3: Rome - regions visited by taxis.

Automatic systems can monitor the flow of people or cars and can decide which areas are over-crowded and need assistance. They can also be used to identify expected behavior when a large event such as a concert takes place in town.

6. Conclusions and Future Work

In this paper we proposed a generic architecture for data flow handling specific for Smart Cities. We describe the functions and components for each step and identify specific technologies. Then we provide two use cases on crowd management and intelligent transportation systems. We highlight experimental results from applications developed using the model proposed in our architecture. As further work we will analyze self-adaptive optimization methods used in this architecture, focusing on data reduction and data cleaning, patter extraction and data aggregation.

Acknowledgment

The research presented in this paper is supported by projects: *DataWay*: Real-time Data Processing Platform for Smart Cities: Making sense of Big Data - PN-II-RU-TE-2014-4-2731; *MobiWay*: Mobility Beyond Individualism: an Integrated Platform for Intelligent Transportation Systems of Tomorrow - PN-II-PT-PCCA-2013-4-0321; *CyberWater* grant of the Romanian National Authority for Scientific Research, CNDI-UEFISCDI, project number 47/2012; *clueFarm*:

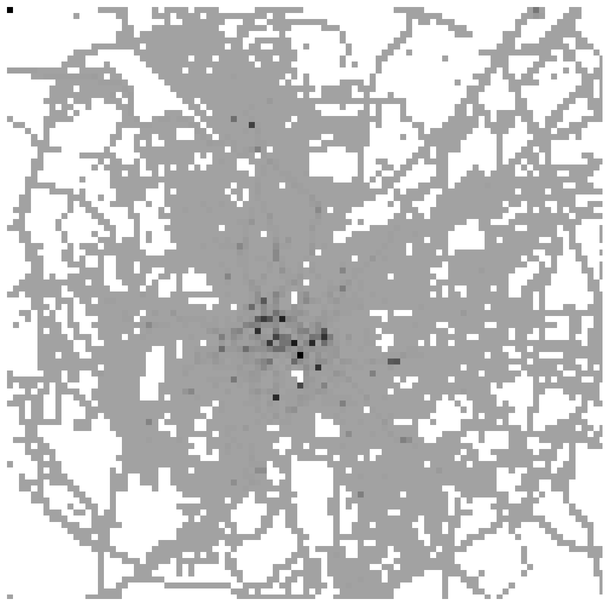

Figure 4: Rome - most important regions.

Information system based on cloud services accessible through mobile devices, to increase product quality and business development farms - PN-II-PT-PCCA-2013-4-0870.

We would like to thank the reviewers for their time and expertise, constructive comments and valuable insight.

References

[1] V. Albino, U. Berardi, R. M. Dangelico, Smart cities: Definitions, dimensions, performance, and initiatives, Journal of Urban Technology 22 (1) (2015) 3–21.

[2] A. Whitmore, A. Agarwal, L. Da Xu, The internet of things—a survey of topics and trends, Information Systems Frontiers 17 (2) (2015) 261–274.

[3] J. M. Batalla, P. Krawiec, Conception of id layer performance at the network level for internet of things, Personal and Ubiquitous Computing 18 (2) (2014) 465–480.

[4] J. M. Batalla, M. Gajewski, W. Latoszek, P. Krawiec, C. X. Mavromoustakis, G. Mastorakis, Id-based service-oriented communications for unified access to iot, Computers & Electrical Engineering 52 (2016) 98–113.

[5] A. Gandomi, M. Haider, Beyond the hype: Big data concepts, methods, and analytics, International Journal of Information Management 35 (2) (2015) 137–144.

[6] C. S. Raghavendra, K. M. Sivalingam, T. Znati, Wireless sensor networks, Springer, 2006.

[7] H. Ma, D. Zhao, P. Yuan, Opportunities in mobile crowd sensing, Communications Magazine, IEEE 52 (8) (2014) 29–35.

[8] G. P. Hancke, G. P. Hancke Jr, et al., The role of advanced sensing in smart cities, Sensors 13 (1) (2012) 393–425.

[9] K. Su, J. Li, H. Fu, Smart city and the applications, in: Electronics, Communications and Control (ICECC), 2011 International Conference on, IEEE, 2011, pp. 1028–1031.

[10] G. K. Still, Crowd dynamics, Ph.D. thesis, University of Warwick (2000).

[11] H. Du, Z. Yu, F. Yi, Z. Wang, Q. Han, B. Guo, Group mobility classification and structure recognition using mobile devices, 2016 IEEE International Conference on Pervasive Computing and Communications, PerCom 2016.

[12] D. Kumar, H. Wu, Y. Lu, S. Krishnaswamy, M. Palaniswami, Understanding Urban Mobility via Taxi Trip Clustering, 2016 17th IEEE International Conference on Mobile Data Management (MDM) (2016) 318–324.

[13] M. S. Grewal, L. R. Weill, A. P. Andrews, Global positioning systems, inertial navigation, and integration, John Wiley & Sons, 2007.

[14] Y. Chon, S. Kim, S. Lee, D. Kim, Y. Kim, H. Cha, Sensing WiFi packets in the air, Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '14 Adjunct (2014) 189–200.

[15] M. Dash, K. K. Koo, S. P. Krishnaswamy, Y. Jin, A. Shi-Nash, Visualize people's mobility - Both individually and collectively - Using mobile phone cellular data, Proceedings - IEEE International Conference on Mobile Data Management 2016-July (2016) 341–344.

[16] A. J. Ruiz-Ruiz, H. Blunck, T. S. Prentow, A. Stisen, M. B. Kjaergaard, Analysis methods for extracting knowledge from large-scale wifi monitoring to inform building facility planning, in: Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on, IEEE, 2014, pp. 130–138.

[17] L. Vu, K. Nahrstedt, S. Retika, I. Gupta, Joint bluetooth/wifi scanning framework for characterizing and leveraging people movement in university campus, in: Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems, ACM, 2010, pp. 257–265.

[18] M. Zhou, Z. Tian, K. Xu, X. Yu, X. Hong, H. Wu, Scanme: location tracking system in large-scale campus wi-fi environment using unlabeled mobility map, Expert systems with applications 41 (7) (2014) 3429–3443.

[19] B. Bonne, A. Barzan, P. Quax, W. Lamotte, Wifipi: Involuntary tracking of visitors at mass events, in: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a, IEEE, 2013, pp. 1–6.

[20] C. D. C. Chilipirea, A.-C. Petre, M. v. Steen, Filters for wi-fi generated crowd movement data, in: 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, IEEE, 2015, pp. 285–290.

[21] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, R. P. Martin, Measuring human queues using wifi signals, in: Proceedings of the 19th annual international conference on Mobile computing & networking, ACM, 2013, pp. 235–238.

[22] L. Schauer, M. Werner, P. Marcus, Estimating crowd densities and pedestrian flows using wi-fi and bluetooth, in: Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 171–177.

[23] K. Han, E. Graham, D. Vassallo, D. Estrin, et al., Enhancing motivation in a mobile participatory sensing project through gaming, in: Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, IEEE, 2011, pp. 1443–1448.

[24] H. Aly, A. Basalamah, M. Youssef, Map++: A crowd-sensing system for automatic map semantics identification, in: Sensing, Communication, and Networking (SECON), 2014 Eleventh Annual IEEE International Conference on, IEEE, 2014, pp. 546–554.

[25] M. Faulkner, M. Olson, R. Chandy, J. Krause, K. M. Chandy, A. Krause, The next big one: Detecting earthquakes and other rare events from community-based sensors, in: Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on, IEEE, 2011, pp. 13–24.

[26] M.-R. Ra, B. Liu, T. F. La Porta, R. Govindan, Medusa: A programming framework for crowd-sensing applications, in: Proceedings of the 10th international conference on Mobile systems, applications, and services, ACM, 2012, pp. 337–350.

[27] I. Carreras, D. Miorandi, A. Tamilin, E. R. Ssebaggala, N. Conci, Matador: Mobile task detector for context-aware crowd-sensing campaigns, in: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on, IEEE, 2013, pp. 212–217.

[28] P. P. Jayaraman, C. Perera, D. Georgakopoulos, A. Zaslavsky, Efficient opportunistic sensing using mobile collaborative platform mosden, in: Collaborative Computing: Networking, Applications and Worksharing (Collabo-

ratecom), 2013 9th International Conference Conference on, IEEE, 2013, pp. 77–86.

[29] T. Yan, M. Marzilli, R. Holmes, D. Ganesan, M. Corner, mcrowd: a platform for mobile crowdsourcing, in: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, ACM, 2009, pp. 347–348.

[30] H. Väätäjä, T. Vainio, E. Sirkkunen, K. Salo, Crowdsourced news reporting: supporting news content creation with mobile phones, in: Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, ACM, 2011, pp. 435–444.

[31] J. M. Batalla, Advanced multimedia service provisioning based on efficient interoperability of adaptive streaming protocol and high efficient video coding, Journal of Real-Time Image Processing 12 (2) (2016) 443–454.

[32] T. Xie, X. Qin, Scheduling security-critical real-time applications on clusters, Computers, IEEE Transactions on 55 (7) (2006) 864–879.

[33] J. P. Erickson, G. Coombe, J. H. Anderson, Soft real-time scheduling in google earth, in: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th, IEEE, 2012, pp. 141–150.

[34] B. Sprunt, L. Sha, J. Lehoczky, Aperiodic task scheduling for hard-real-time systems, Real-Time Systems 1 (1) (1989) 27–60.

[35] C. Dobre, G. Suciu, C. Chilipirea, C. Gosman, Mobility beyond individualism: an integrated platform for intelligent transportation systems of tomorrow, in: ITS Romania Congress, 2014, pp. 31–35.

[36] Y. Kryftis, G. Mastorakis, C. X. Mavromoustakis, J. M. Batalla, E. Pallis, G. Kormentzas, Efficient entertainment services provision over a novel network architecture, IEEE Wireless Communications 23 (1) (2016) 14–21.

[37] J. M. Batalla, M. Kantor, C. X. Mavromoustakis, G. Skourletopoulos, G. Mastorakis, A novel methodology for efficient throughput evaluation in virtualized routers, in: Communications (ICC), 2015 IEEE International Conference on, IEEE, 2015, pp. 6899–6905.

[38] C. Barbieru, F. Pop, Soft real-time hadoop scheduler for big data processing in smart cities, in: Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on, IEEE, 2016, pp. 863–870.