Paper

# My City Dashboard: Real-time Data Processing Platform for Smart Cities

Catalin-Constantin Usurelu[1] and Florin Pop[1,2]

[1] *Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Bucharest, Romania*
[2] *National Institute for Research and Development in Informatics (ICI), Bucharest, Romania*

**Abstract—In recent years, with the increasing popularity of IoT, the rapid growth of smartphone usage enabled by the increase adoption of Internet services and the continuously decreasing costs of these devices and services has led to a huge increase in the volume of data that governments can use in the context of smart city initiatives. The need for analytics is becoming a requirement for smart city projects such as city dashboards to provide citizens with an easy to understand overview of the city. As such, data should be analyzed, reduced and presented in such a way that citizens can easily understand various aspects of the city and use this information to increase quality of life. In this paper, we firstly present the context and the start of the design and implementation of proposed solution for real-time data processing in smart cities, mainly an analytics processing pipeline and a dashboard prototype for this solution, named My City Dashboard. We focus on high scalability and modularity of this platform.**

***Keywords—big data, data analytics, real-time processing, smart cities, statistics.***

## 1. Introduction

Because the use of sensors is not always feasible given the inaccessibility of locations, lack of a complete understanding of where to best gather data from and costs, an avenue worth exploring is that of crowdsources initiatives. These entail citizen participations resulting in no costs (citizens don't directly benefit), the advantage of human decision making related to what data to collect and from where and sometimes better accuracy and fault detection compared to sensors [1]. One such approach is presented in [2] where the authors propose a crowdsourcing framework that lets user combine data collection, selection and assessment activities to allow local government to achieve complex goals [3]. The authors present a system where users submit queries that get transformed in a set of tasks that are further submitted to other users. Through the completion of tasks by the other users, such as collecting and assessing images of damaged roads, the query can be answered.

To evaluate city services, Motta *et al*. [4] propose a four-stage model in the design of City Feed – a crowd-sourced governance system. These stages are: publishing (provides government data), interacting (by social media and online service tools), transacting (service integrations) and evaluating. With the growing maturity, Quality of Service (QoS) increases. Briefly, the four-stage model is a roadmap of service evolution, that includes information display, online processing, online interaction, and holistic analysis. The system is composed of two parts: a transactional system (City Feed manager) responsible for processing citizen generated events, creating issues and de-duplicating them, and an analytic system (City Feed analyzer) that simply extract the data from the manager and uploads it to a data warehouse. The data are shown in different forms such as bar charts, and structured along different dimensions (e.g. location, time, event class, etc.).

ArcGIS, currently one of the most capable geographic information systems is another example that allows the processing of streaming events and generation of analytics with the help of the GeoEvent Server extension [5]. By defining both input and output connectors it can receive real-time event streams and push the analytics results to other systems (e.g. to a message queue). Nevertheless, it is limited to its analytics offering leading to extra work for users to use other analytics algorithms and it is also a commercial solution, not a free, open-source one.

Search-the-City [6] is a dashboard primarily concerned with processing large amounts of heterogeneous data (from sensors, cameras, social streams, user generated contented and data produced by city authorities) and displaying it in an easy to consume form. The dashboard's architecture is composed of two parts: a search layer and a visualization framework. The search layer is based on a Storm topology and the Terrier search engine. It is responsible for receiving data collected by edge servers (in the form of XML, RDF and Linked Data) and indexing. The visualization framework takes the concept of mash-up to a new level – the visual components themselves can communicate with each other. This is done by implementing the widgets as Java portlets thus giving them the ability to pass events between each other.

The Bandung Smart City dashboard [7] is a prototype project designed to help solve some of Bandung's – one of Indonesia's cities problems, caused by its fast-growing population. The authors propose architecture composed of sensor nodes that transmit data to processing servers over a classical TCP/IP Internet connection. The sensor nodes sample data using a specialized protocol to reduce energy

consumption. The servers themselves have a database used to store sensor data (although currently the platform only displays the last received value) and Geographical Information System (GIS) software. The result is a single dashboard that gives a summary of the current city-state.
In this context, the paper has the following contributions. First, an analytics architecture designed for city dashboards is presented. The existing solutions are analyzed and requirements for such architecture are provided. Then the analytics pipeline architecture together with statistics computation and clustering algorithms are described. The proposed architecture is evaluated with simulated date on Bucharest as a city example. Finally, the results of proposed algorithms on the city dashboard are presented.
The paper is structured as follows. Section 2 presents the existing solution and a lesson learned from all available approaches. Then, Section 3 presents the architecture and design consideration. Section 4 described the main used algorithms while Section 5 presents My City Dashboard architecture prototype. Section 6 introduces the methodology and experimental results. The paper ends with conclusions and future work in Section 7.

## 2. Related Work and Existing Solutions

In this section several existing solutions for city dashboard are analyzed.

### *2.1. Amsterdam City Dashboard*

The Amsterdam City Dashboard [8] was briefly launched as a prototype in 2014 and is currently a work in progress. The dashboard has two main modes of displaying data:

- a map view capable of displaying both points representing discrete information types and paths representing statistics along that path, for example the average speed along a road;
- a partition view, where each partition displays a certain category on which city elements are projected. The categories on which the city elements are projected to are: transport, environment, statistics, economy, community, culture, and security. Each category presents a citywide statistics based on blocks of 24 hours with data refreshed every 10 s. Similar approaches were presented in [9] and [10].

The project is based on the City SDK project [11], more specifically the Linked Data API. The API aims to help government agencies open up data and provides the ability to collect data from different sources, annotate, link and make the information available and searchable. Also by using Linked Data, datasets can be easily linked or enriched with user provided information, for example reporting road blockages or alternative routes. The project also provides a developer page [12]. Main characteristics of the SDK:

- authentication – simple session creation (through the use of username and password over HTTPS) and deletion;
- formats – the SDK supports JSON-LD for Linked Data and GeoJSON for representing geographical information;
- endpoints – RESTful endpoints;
- resource types:
  - layers represent data sets,
  - objects can be contained on one on or more layers,
  - owners of layers;
- technologies used – Ruby as a programming language, Grape as a REST framework and PostgreSQL with the addition of the PostGIS add-on in order to add support for geographic objects.

Although this solution is perfect for collecting data and making it available, it doesn't address a few potential necessities:

- handling massive amounts of real-time streaming data;
- the linked data must be generated by the application implementing the SDK and more research has to be done in order to properly use the linked data concept. Not only that, but for a city dashboard use case where we are mostly interested in displaying statistics or applying different machine learning algorithms that would work better on the raw data, the concept of linked data would only add more complexity;
- no way of integrating existing applications such as city service apps or social apps like Twitter, Foursquare, Instagram, etc.;
- user registration;
- dashboard personalization per user.

### *2.2. Dublin Dashboard*

The Dublin Dashboard [13] is part of The Programmable City project [14] and led by Professor R. Kitchin. A few of the most important motivational research questions R. Kitchin and his team [15] try to answer are: how city dashboards can change and influence the performance of a city, how can we display, structure, analyze and select information and is all this reproducible to compare a smart city – in this case Dublin – with other cities that would implement this platform, like benchmarking?
The Dublin Dashboard is an analytical dashboard pulls together data from data sources such as: Dublin City Council, Dublinked [16] – this platform provides most of the real-time data and static datasets, Central Statistics Office, Eurostat, government departments and several existing smart

Table 1

Analysis of existing solutions for city dashboards

| Dashboard name | City data sources | Social media networks integration | Data source formats | Provides contextual user information | User log-in | User preferences support | Persis-tence of data | Analytics support | Future work and observed needs |
|---|---|---|---|---|---|---|---|---|---|
| Amsterdam City Dashboard | Real-time: city wide statistics on transport, environment, statis-tics economy, community, culture and security | No | JSON-LD, GeoJson | No | No | No | Yes | No | • Linked data is created manually<br>• Not scalable (simple REST framework and an SQL database)<br>• User registration support |
| Dublin Dashboard | Static: Dublin City Council, existing static datasets, Dublinked platform<br>Real-time: Dublinked – APIs to existing apps | in progress – Facebook, Twitter, Flickr and Instagram | JSON, SML, CSV, XLS, etc. | No | No | No | No | No | • Opening new datasets<br>• Cleaning and processing those data<br>• Developing the site beyond data visualizations to include a roader set of data analytics, including modeling tools |
| London Dashboard | Static: government institutions<br>Real-time: RSS channels, Twitter, CASA University's wether station, OpenStreetMap updates, Yahoo stocks etc. | Yes – Twitter (trends and special accounts) | HTML – web scrapping, XML, JSON, CSV | No | No | No | No | No | The integration of future data-sources: energy network, sensor data, crowdsourced data – FitBit, atmospheric data |
| Dubai Personal Dashboard | Real-time: building information and status, weather, prayer timings, video streams etc. | Yes – Twitter and Facebook (requires authentication) | N/A | Yes – also integrates a "My Family" module | Yes | Yes – dashboard customizability | No | No | N/A |
| Bandung Smart City Dashboard | Real-time: city sensors | Binary | No | No | No | Yes | No | | • Analytical tools – precise evaluation and prediction<br>• Decision system support tools |
| CityEye | Real-time: sensor data (environmental, city service providers), GPS data, citizen | Yes – sentiments analysis of feedback from Facebook and Twitter | N/A | Yes – provides graphs predictions in the user's vicinity | No | No | Yes | Yes – service and indirect sentiment analysis | Include additional data sources: real-time traffic data, pedestrian activity, local wind sensors, water quality sensors |
| Search-the-City Dashboard | Streaming data oriented – sensors, cameras and social media streams | Yes – Facebook, Twitter, Foursquare | XML, RDF, Linked Data | Yes – provides data from all data sources near chosen location | Yes – split for citizens and municipal administrators | Yes – dashboard customizability and indirect infor-mation extracted from social net-works accounts and associated social graphs | Yes | Yes – query indexing and query person-alization with data steaming from social accounts, event detection, sentiment analysis, contextual statistics | Find optimal combinations of environment generated content originated components from sensors and social networks, in order to provide new visual components |

city and social applications (e.g. Twitter, Facebook, or applications which publish links to Dublinked). By using existing resources and applications it minimized duplicate effort.
Types of information it provides – static information, real-time information, time-series indicator data, and interactive maps.
The dashboard contains hundreds of data representations grouped in different modules. Some notable examples are:

- Dublin at a glance module – displays both overall statistics from the city (e.g. number of thefts in the city, or overall air quality index of Dublin) and information from key points in the city (e.g. current parking spots at certain locations or sound level of Blessington Street Basin). It also provides current top news;

- Dublin reporting module – provides links to sites (intuitively in the form of the frontpages of those target apps) that provide services for reporting: FixMyStreet and FixMyArea used for city related problem reporting and CityWatch, which aggregates and display sensor data received from citizens and also municipals;

- Dublin Near Me module – integrates (both in the context of the dashboard as a separate view or with links to separate sites providing the service) of apps such as: Dublin Community Maps (used to find amenities in Dublin), Rate My Area (used to rate city areas) or Vacant Spaces (an app used by users to indicate spaces in the city that are currently unused);

- Dublin RealTime – this module displays the following information: Dublin Environment Maps (displays air quality, ambient sound levels and water levels at certain key locations), City Traffic and Travel (displays available spaces at certain car parks, available bike stations – dynamically clustered on zooming in/out and travel times on certain routes), Maritime Traffic (link to an external site) and a Flight Radar (also a link to an external site);

- Dublin Apps module – is a list of smart city mobile apps that can be used in Dublin with links for easy installation;

- Dublin Social module – a work in progress modules, which suggests that it will integrate information from Facebook, Twitter, Flickr and Instagram.

- Modules providing maps with census, crime (displayed as clustered datasets), companies, housing, and planning information.

Overall, the Dublin Dashboard is the most advanced city dashboard that we could find, with a vast number of data sources, representations and overall smart city integrations. This is mostly the result of Dublin's smart city initiatives that have made it possible to develop many services and applications and open-up the data they provide to be used in new applications, in this case – the Dublin Dashboard.
In the future, the project aims to accomplish the following tasks [17]: opening new datasets, cleaning and processing those data, developing new applications and developing the site beyond data visualizations to include a broader set of data analytics, including modeling tools.
The current drawbacks we identified in this dashboard are: lack of user personalization (users cannot login and control what they want to see or receive more personalized information), no open-source initiative that could be used to allow public contribution to the dashboard, no way of integrating personal applications (e.g. personal Facebook account, or Smart Sports Watch applications that could be used to create key performance metrics related to city resident's health levels etc.) and as of yet, no way to create key performance indicators based on the received data sources as proposed in [17].

### *2.3. The London Dashboard*

The London Dashboard [18] is another UK initiative, this time developed by the Centre for Advanced Spatial Analysis research center of the University College London. It was mainly developed in the first half of 2012 and has been in maintenance since.
The design of the dashboard is simple [19]: a service that collects data from various websites (web scrapping) and APIs, a website composed of three views (a map data is retrieved from a CSV based API, a module-based view and a grid view data is retrieved from a HTML based API).
The data is obtained mainly in 2 ways [20]: web scrapping in the case of sites that do not provide other means of accessing it such as ScotRails tube style line running and APIs returning data in XML/JSON/CSV such as BBC RSS (for local news), OpenStreeMap, RSS updates, Twitter (tweets from a list of accounts related to general news and university news and also top Twitter trends), Mappines (an app that aims to capture the mood of the population across UK), CASA's radiation detector, DEFRA's air pollution data, etc.
The dashboard currently obtains its data from the following sources [21]:

- Department for Environment Food and Rural Affairs,
- National Oceanic and Atmospheric Administration,
- OpenStreetMap (and Pawel's Static Maps API),
- British Broadcasting Corporation,
- London School of Economics,
- Yahoo! Developer Network,
- Port of London Authority,
- Transport for London,

- Yahoo! Finance,
- UCL CASA,
- MapTube,
- ScotRail,
- Twitter.

Each module that displays information from a certain data-source also has a counter notifying the user of the next update. This is mostly because the server caches responses from APIs in order to improve performance and go around rate limits (such as from Twitter).
The project also provides developers with a set of APIs [22] that expose the aggregated data of the input data-sources and that are used in the creation of the dashboard so that they can be consumed by external services (e.g. PigeonSim, the London Periodic Table, the London Data Table, Prism, etc.). The APIs are very simple and come in two MIME-type flavors: CSV and HTML.
A few of the main challenges in the future development of the project are as follows [20]:

- finding real-time and reliable data,
- integration of various heterogeneous services,
- issues in filtering and representing social networking data,
- obtaining environment related sensor data,
- news, events and community data sources are not easily available.

The project also expects the integration of future data-sources [19] such as from the energy network or sensor data sources that have not yet opened-up their data and also from crowd initiatives (e.g. Air Pollution Egg for atmospheric data and the FitBit One for personal mobility).
Also, an initial design proposal for the city dashboard [20] was that of being customizable to user preferences so that is another direction in which the dashboards development might head towards. The lack of database storage is also noted, but currently the dashboard's functionality does require analyzing stored data.
One thing that can be improved: provide users with the ability to understand the cause of certain indicators by visualizing data over recent time intervals.

### *2.4. Dubai Personal Dashboard*

The Dubai Personal Dashboard [22] was launched by the Dubai Civil Defence (DCD) in October 2015 with the aim of allowing residents and visitors to visualize data, keep them informed and connected, make their daily city interactions more seamless, enable real-time public engagement and generally contribute to enabling Dubai to be the smartest city in the world.
The dashboard collects and displays data from public and private sources, and from social media networks (Facebook – personal feeds, Tweeter – government tweets). In order to enable seamless login, the dashboard uses Dubai's MyID service for single sign-on. The dashboard is module based, being formed from a mash-up of modules, each displaying a different concept. The modules are of 3 types:

- **General modules**: nearby fire stations, DCD newsletter, Gulf News, tweets from the government, weather, prayer timings;
- **Personalized modules**: building related information and alerts modules – requires the user to map their building ID; Facebook feeds – requires the user to sign-in with his account in the module; video streaming – required camera source and login information.
- **Reporting modules**: emergency button – for fire, police, ambulance etc; safety violation reporting – ability to create and send a safety violation report with detailed information, pictures and video.

The Dubai Personal Dashboard is one of the only dashboards that support personalized information. The dashboard does not currently have a lot of data sources and no complex city analytics so that is a direction it may take in the future.

### *2.5. Bandung Smart City Dashboard*

The Bandung Smart City Dashboard [24] is a prototype project designed to help solve some of Bandung's – one of Indonesia's cities – problems, caused by its fast-growing population. By monitoring what happens in the city, it can help the government detect problems and find solutions as soon as possible. The dashboard is design to monitor indicators such as: temperature, air pollution, water pollution, traffic situation, economic indicators, energy supplies, number of citizens, number of vehicles, number of houses, etc.
The authors of [24] propose architecture composed of sensor nodes that transmit data to processing servers over a classical TCP/IP internet connection. The sensor nodes sample data using a specialized protocol in order to reduce energy consumption. The servers themselves have a database used to store sensor data (although currently the platform only displays the last received value) and Geographical Information system (GIS) software. The result is a single dashboard that gives a summary of the current city-state.
In the future, the authors propose adding more decision support system and analytical and add functionalities such as prediction.

### *2.6. CityEye*

CityEye [25] is an urban visualization and management dashboard designed as a solution to the problem of urban

infrastructure and service transparency. The aim of the platform is to foster richer dialogue between users and urban service providers and give citizens an overall view on the state of the city.
The context of two cities was used to find current existing smart city platforms and problems associated with them: Barcelona (heterogeneity of data resulted from each urban service company resulted in integration problems leading to the need of data standardization) and Santander, where applications have failed to generate interest from citizens and companies.
The data sources included in CityEye can be divided into three categories: sensor data, data generated during the provision of services, and data generated by citizens (in the form of feedback and reports from applications or by email and sentiment analysis of Facebook and Twitter posts). Also, it has been proposed that some of these sensors can be embedded in urban maintenance vehicles, which combined with GPS capabilities can provide monitoring capabilities across the city.
There are still some problems identified in the development of the platform: data availability, existing sensor infrastructure, relationships between service providers and government, real-time traffic information, capture of pedestrian activity, water quality testing.

### *2.7. Search-the-City Dashboard*

Search-the-City dashboard [6] is primarily concerned with processing large amounts of heterogeneous data from sensors, cameras, social streams, user generated contented and data produced by city authorities, and displaying it in an easy to consume form.
The dashboard's architecture is composed of two parts: a search layer and a visualization framework. The dashboard itself is implemented using an off-the-shelf solution for building portals Liferay. It allows building sites with role-based access control, single sign-on support, CMS functionalities and the ability to integrate Java portlets.
The search layer is based on a Storm topology and the Terrier search engine and has 3 sub-layers:

- a layer of edge servers that are responsible for acquiring sensor data and passing it to the storm topology in the form of standard formats (XML, RDF, Linked Data),
- a search layer based on the Terrier search engine responsible for indexing data retrieved from edge servers and answering queries,
- end-user applications layer, which submit queries to the search engine.

The search layer also treats social media information (from Facebook, Twitter and Foursquare) as normal sensor data allowing for the use of algorithms such as: sentiment analysis, event detection etc. Another interesting aspect is that the search layer can also return personalized results based on the user's social applications.
The visualization framework takes the concept of mashup present in the previously presented dashboards to new level – the visual components themselves can communicate with each other. This is done by implementing the widgets as Java portlets thus giving them the ability to pass events between each other. This gives the possibility of doing a workflow in which a user enters a search query in a widget and upon receiving event data from the search layer. It distributes it to another widget that lists a search result set. Upon clicking on an item on the result set, events are transmitted to the other widgets resulting in them updating. The updated widgets are: a video widget with camera feeds near the location, a social media widget with posts geo-located near that location, a sensor widget with nearby sensor data and a maps widget that centers to that location.
The result of a query for a specific location is an event that contains all the data related to that location such as that presented in the workflow above.

## 3. Architecture and Design Considerations

For the design of the analytics platform several criteria have been chosen:

- *high scalability*– resulting from the great quantity of data generated by sensor streams, crowdsourcing produced streams and social streams;
- *modularity* – the system would be required to use a large amount of sub-systems. These systems must be allowed to evolve independently with no or reduced impact on the rest of architecture;
- *pluggability* – new processing components, event streams, data sinks etc. can be attached or detached without affecting the rest of the system;
- the main technologies used must be able to interact with each other without requiring the development of custom communication modules or modifying the open-source projects;
- data must be presented in an easy to understand way, mainly using key performance indicators and aggregated data [26].

These requirements have led to the decision to use the following technologies for each layer of the architecture. The overall architecture is presented in Fig. 1. As possible use cases for the system we have considered the following stream types: temperature and noise. We currently provide identical functionality for these data types so only the results of noise analytics [27], [28] will be presented.

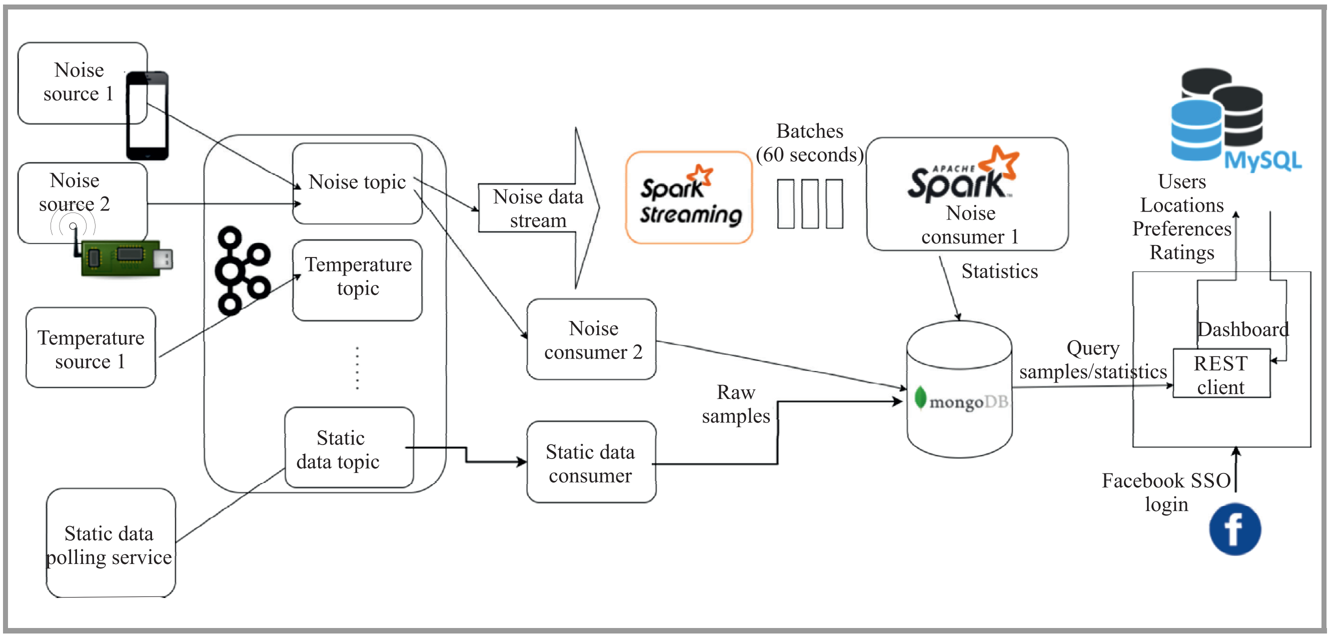



*Fig. 1.* Analytics pipeline architecture.

### *3.1. Acquisition Layer*

For the ingestion layer, we have chosen Apache Kafka, which is a high-throughput, low-latency and massively scalable publish/subscribe message queue for handling real-time data feeds [29]–[31]. The other technology we have considered is RabbitMQ, one of the most known and used message queues. The decision was made because RabbitMQ does not support multiple consumers per queue as such resulting in low pluggability (we require that developers should be able to write whatever analytics modules they want without being impacted by other existing modules) and its performance is about 2 times lower than that of Kafka [32]. Also Kafka provides a Spark Streaming modules that allows us to easily integrate its output in Spark as we will see later.

### *3.2. Processing Layer*

For the processing layer Apache Spark and Apache Flink have been considered. While Flink is more suited for streaming data, providing better streaming semantics support, the project is still in its infancy (the project is currently incubating). This lack of current support has impacted our decision mainly because there was no MongoDB connector that could easily stream data to our database thus requiring us to write files on disk and use the Hadoop Connector, which is not what we want. Also, while presented system is designed for real-time analytics we do not require pure stream processing because most of operations are done in small batches (e.g. 60 s batches).

### *3.3. Persistance Layer*

Because we receive a high amount of data a highly scalable database. Also, considering that most of the data has a geographic source that is directly used in processing pipeline and in our visualization layer we required a database that supports GIS operations such as retrieving all the information in a defined area on the map. The main two technologies that have been considered are MongoDB and the PostGIS module for PostgreSQL. While both solutions provide the required operators only MongoDB is designed to scale for a high amount of read and particularly write operations.

### *3.4. Dashboard Layer*

The dashboard layer is composed of a separate dashboard project in which we have plugged in the analytics functionality. The dashboard itself is built on top of a Service-Oriented Architecture (SOA) architecture composed of RESTful services. This layer, besides providing the UI also gives a proof of concept example of using the provided APIs for accessing and creating event, location and user resources. The location APIs are especially useful because they will be used in further research for providing a way to import and display location and event related data in the dashboard.

## 4. Algorithms

Two main algorithms have been implemented in the platform: an overall statistics algorithm and a clusterization algorithm used for aggregating the huge amount of data that would need to be display on the map and only compute statistics on data partitioned in these clusters.
The first algorithm is simple and composed of only a small list of steps as all the functionality is already provided by Spark (see Algorithm 1).

The clusterization algorithm is more complex and requires some description. The complexity does not derive from the algorithm itself but from mapping the algorithm in Spark (it is still a lot easier than manually implementing it).

**Algorithm 1** Overall statistics computation

```
MC ← MongoConnector;
samples ← Kafka.readStream("noise");
parsed ← samples.map(deserializeFunc);
samples.foreachRDD(
procedure function(RDD) {
    if !rdd.isEmpty() then
        convRDD ← RDD.mapToDouble(mapFunc);
        min = convRDD.min();
        max = convRDD.max();
        avg = convRDD.average();
        count = convRDD.count();
        rddRow = NewRDDRow(min,max,avg,count);
        MC.save(rddRow);
});
```

**Algorithm 2** Tile clustering algorithm.

```
samples ← Kafka.readStream("noise");
MC ← MongoConnector;
partitionedSamples ← samples.mapToKeyValuePair(
procedure function(sample){
    tileIdX ← convertToTileCoordinates(sample).X;
    tileIdY ← convertToTileCoordinates(sample).Y;
    key = tileIdX:tileIdY;
    value = sample.value;
    return (key, value);}
);
intermKeyValuePairs ← partitionedSamples.mapValue(
procedure function(sample){
    return (sample.value, 1); }
);
reduceResults ← intermediaryKeyValuePairs.reduce(
procedure function((key1, value1), (key2, value2)){
    return (key1 + key2, value1 + value2); }
);
reduceResults.foreachRDD(
procedure function(p){
    if !pairRDD.isEmpty() then
        Map< key,value > map = p.collectAsMap();
        for (key, value) ∈ map do
            tileIdX = parse(key).X;
            tileIdY = parse(key).Y;
            average = value.getComputedAverage();
            count = value.getComputedCount();
            rddRow = NewRDDRow(key, tileIdX,
                tileIdY, average, count,
                computeTialCoordinates());
            MC.setKey(key).save(rddRow);
});
```

The algorithm is based on the Cluster Griddy clustering type [33]. It works as follows: a reference point is chosen in geographic coordinates on the map. It is in the lower left of the map. A tiles size is chosen. Each data point is converted from geographic coordinates to tile-based coordinates, that is each point is assigned to a tile. Each tile is mapped to a unique key of the form $tileIdX : tileIdY$ in order to parallelize computation on Spark. For each tile, we compute the necessary statistics. On information retrieval, the tile coordinates are transformed in 4 geographical coordinates that represent that tile (see Algorithm 2).

# 5. *My City Dashboard* Architecture Prototype

In its current form, the city dashboard is based on a SOA architecture implemented using RESTful APIS in the Java Spring Boot Actuator framework.

The accessible endpoints are as follows:

- `/locations` – API used for creating and accessing locations,
- `/users` – API used for creating and accessing users,
- `/noise/samples` – API used for retrieving the total number of processed sensor samples,
- `/noise/latestAggrates` – API used for retrieving the current computed overall city aggregate data,
- `/noise/tiles` – API used for retrieving overlay tiles that create a heatmap representing sensor data on the map.

The user and location information are stored in a MySQL database and are accessed through the Hibernate ORM. The database tables used for storing user, location and relation entities are presented in Fig. 2.

The main interactions available currently in the dashboard are presented in the navigation diagram in Fig. 3.

# 6. Methodology and Experimental Results

The data was simulated using Bucharest as a city example. The geographic coordinates where generated using the coordinates of Bucharest's city center and a radius encompassing the city. The architecture was run on a virtual machine with 2 cores processor, 4 GB RAM and 20 GB SSD. The underlying hardware is a MacBook Pro 15 with Broadwell i7 2.5 GHz, 16 GB, 512 GB SSD. We present the results of our algorithms on the city dashboard.

In Fig. 4 we can see the results of running presented application. The tiles are 500 × 1000 meters in size and repre-

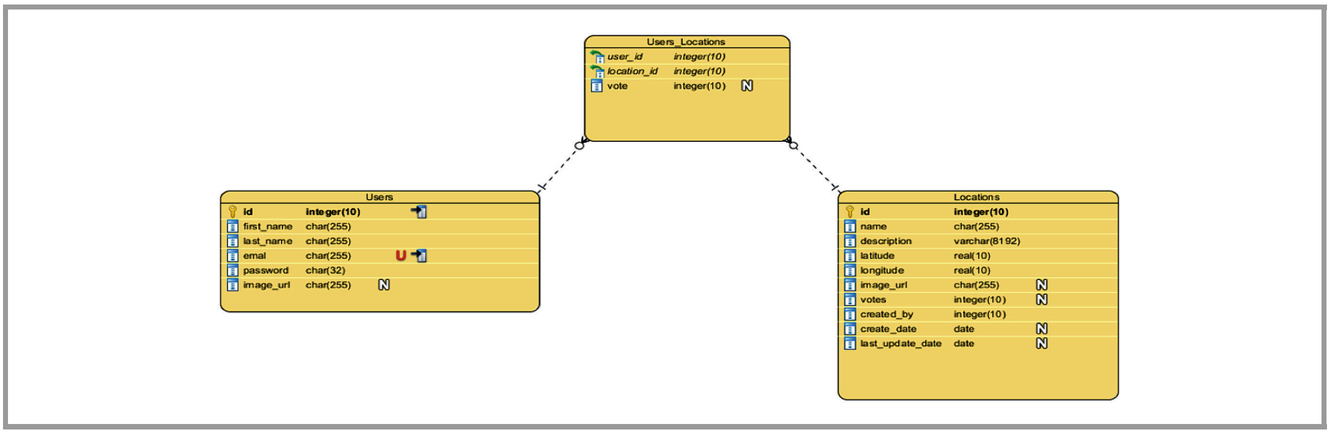



*Fig. 2.* Analytics pipeline architecture.

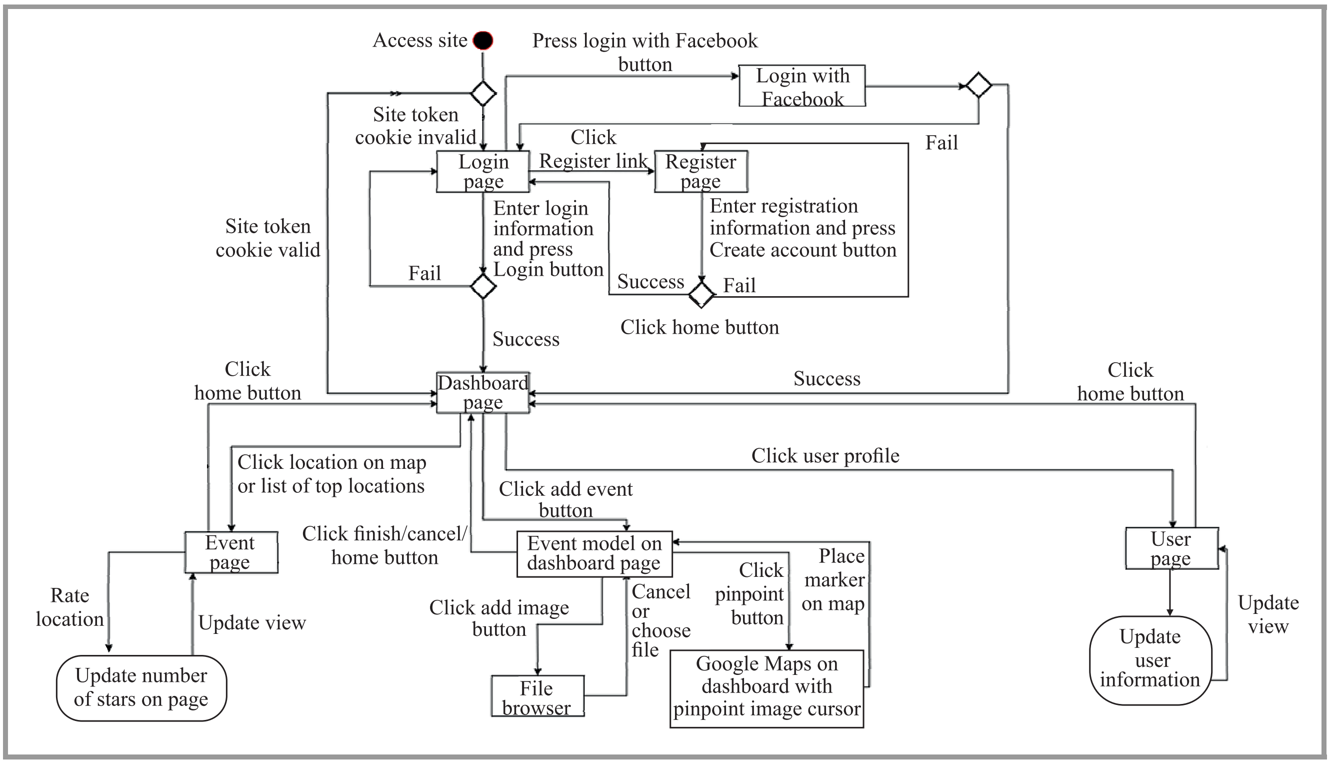



*Fig. 3.* Navigation diagram.

sent the average noise value over that region. Also, we can see the average noise statistics for the whole city. The overall processing speed of the platform results in a processing rate of approximately 523 sample/s.

While that is not a large value we have to keep in mind that 3 distributed systems and the dashboard were running on the same PC and in a virtual machine. In Fig. 5 we can see a more overall view of the city. In this running instance, only part of the city tiles received information.

In Fig. 6 we can see the complete tile rendering for a portion of the city. Also, to be noticed, one region in the upper left of the image is more green conveying the fact that that region is quiet compare to the rest of the city (at least in the time frame it was analyzed – we simulated the fact that the airport was shut down).

# 7. Conclusions

Our implementation and experiments were run with three test cases – city noise, temperature and pollution use cases. The results of our experiments are shown from two points of view: performance and visual representation. In terms of performance, despite the constricted running environment the results are promising. The visual representation is shown from 3 perspectives – a small map portion, a large map portion with partial data sources and a larger map portion completed covered with data sources where some zones stand out from the other.

Also, we have presented a first prototype of the dashboard with user login support, including SSO through an OAuth2 provider – Facebook to enable seamless login.

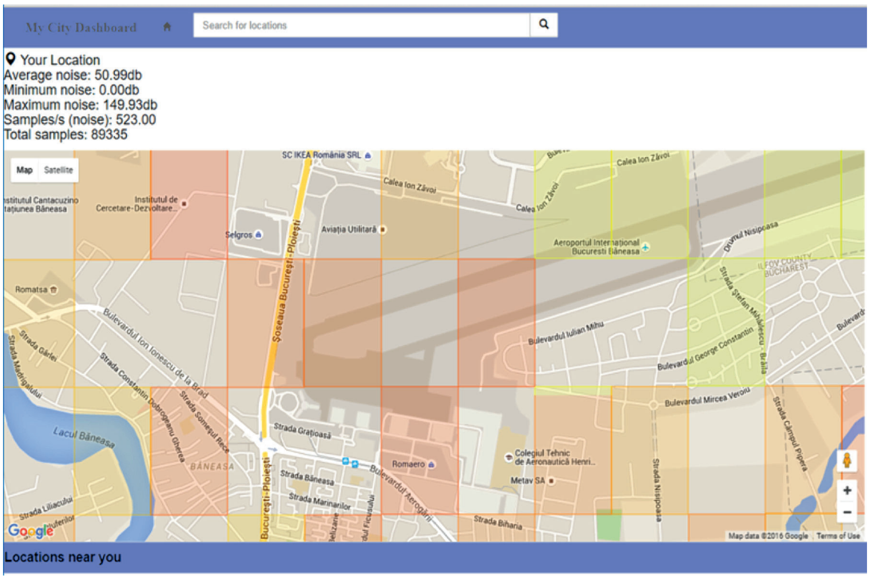



*Fig. 4.* City dashboard – analytics example 1.

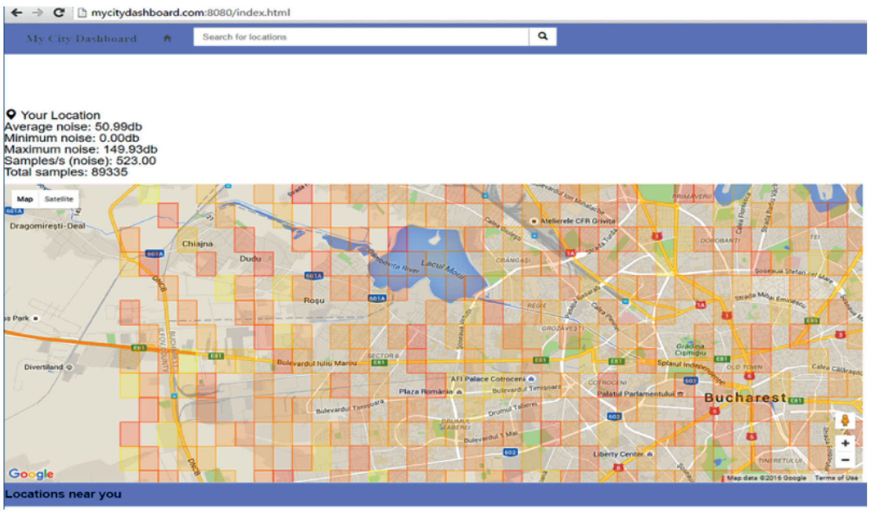



*Fig. 5.* City dashboard – analytics example 2.

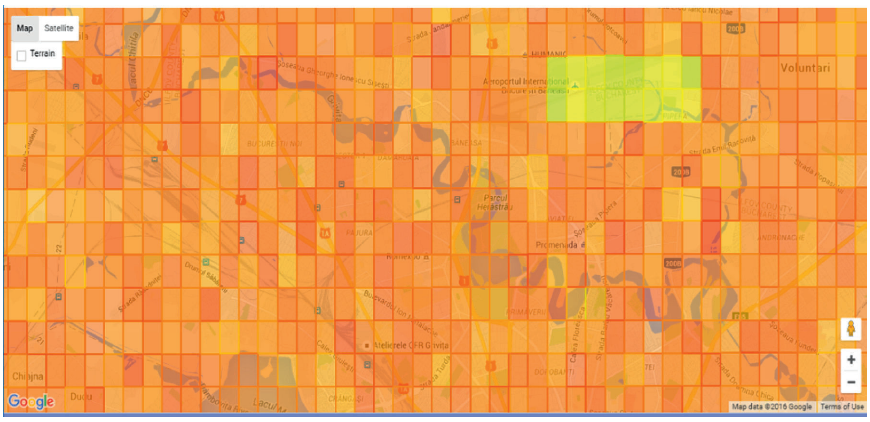



*Fig. 6.* City dashboard – analytics example 3.

Also, a navigation diagram was presented for various operations the user can currently do in the dashboard, such as creating interesting locations/events, find locations/events or event rating them. This functionality only serves as a proof of concept of the underlying APIs, which will be used to import and display data from external sources.
In the future, we will consider more data sources such as social streams or open data and provide additional processing capabilities to integrate these new sources.

# Acknowledgements

The research presented in this paper is supported by the following 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.

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

# References

[1] M. D'Arienzo, M. Iacono, S. Marrone, and R. Nardone, "Estimation of the energy consumption of mobile sensors in WSN environmental monitoring application", in *Proc. 27th Int. Conf. on Adv. Inform. Netw. & Appl. Worksh. WAINA 2013*, Barcelona, Spain, 2013, pp. 1588–1593.

[2] K. Benouaret, R. Valliyur-Ramalingam, and F. Charoy, "CrowdSC: Building smart cities with large-scale citizen participation", *IEEE Internet Comput.*, vol. 17, no. 6, pp. 57–63, 2013.

[3] D. Merezeanu, C. Vasilescu, and R. Dobrescu, "Context-aware Control Platform for Sensor Network Integration in IoT and Cloud, *Studies in Informatics and Control*, vol. 25, no. 4, pp. 489–498, 2016.

[4] G. Motta, L. You, D. Sacco, and T. Ma, "CITY FEED: A crowdsourcing system for city governance", in *Proc. IEEE 8th Int. Symp. on Service Oriented System Engin. SOSE 2014*, Oxford, UK, 2014, pp. 439–445.

[5] ArcGIS GeoEvent Server, "Real-Time Mapping and Analytics" [Online]. Available: http://www.esri.com/software/arcgis/arcgisserver/extensions/geoevent-extension (accessed on Jan. 2017).

[6] A. Moralis, G. Perreas, A. Glaros, and D. Dres, "Search-the-City – A versatile dashboard for searching and displaying Environment and User Generated Content in the context of the future Smart City", in *Proc. Information Access in Smart Cities i-ASC 2014*, Amsterdam, The Netherlands, 2014 [Online]. Available: http://dcs.gla.ac.uk/workshops/iASC2014/papers/iasc2014_moralis.pdf

[7] S. Suakanto, S. H. Supangkat, and R. Saragih, "Smart city dashboard for integrating various data of sensor networks", in *Proc. Int. Conf. on ICT for Smart Society ICISS 2013*, Jakarta, Indonesia, 2013.

[8] Amsterdam City Dashboard [Online]. Available: http://citydashboard.waag.org (accessed on Jan. 2017).

[9] C. Fratila, C. Dobre, F. Pop, and V. Cristea, "A transportation control system for urban environments", in *Proc. 3rd In. Conf. on Emerg. Intell. Data and Web Technol. EIDWT 2012*, Bucharest, Romania, 2012, pp. 117–124.

[10] C. Gosman, T. Cornea, C. Dobre, F. Pop, and A. Castiglione, "Controlling and filtering users data in Intelligent Transportation System", *Future Gener. Comp. Syst.*, in press, 2015 (doi: 10.1016/j.future.2016.12.014).

[11] CitySDK Linked Data API [Online]. Available: http://dev.citysdk.waag.org (accessed on January 2017).

[12] CitySDK Linked Data API Source Code [Online]. Available: https://github.com/waagsociety/citysdk-ld (accessed on Jan. 2017).

[13] Dublin Dashboard [Online]. Available: http://www.dublindashboard.ie (accessed on Jan. 2017).

[14] The Programable City project [Online]. Available: http://www.maynoothuniversity.ie/ (accessed on Jan. 2017).

[15] R. Kitchin, Rob, T. P. Lauriault, and G. McArdle, "Knowing and governing cities through urban indicators, city benchmarking and real-time dashboards", *Regional Studies, Regional Science*, vol. 2, no. 1, pp. 6–28, 2015 (doi: 10.1080/21681376.2014.983149).

[16] Dublin data sharing platform [Online]. Available: http://dublinked.ie/ (accessed on Jan. 2017).

[17] Dublin Dashboard challenges [Online]. Available: http://smartdublin.ie/ smartstories/dublin-dashboard (accessed on Jan. 2017).

[18] London Dashboard [Online]. Available: http://citydashboard.org (accessed on Jan. 2017).

[19] F. Roumpani, O. O'Brien, and A. Hudson-Smith, "Creating, visualizing and modelling the realtime city" [Online]. Available: http://casa.oobrien.com/ misc/presentations/roumpani2012a.pdf (accessed on Jan. 2017).

[20] Smart Cities – City DashBoards DashBoards Lecture [Online]. Available: http://www.spatialcomplexity.info/files/2013/06/Session-5-Lecture-2.pdf (accessed on Jan. 2017).

[21] London Dashboard API [Online]. Available: http://oobrien.com/2012/06/citydashboard-the-api/ (accessed on Jan. 2017).

[22] Dubai Personal Dashboard [Online]. Available: https://pdb.24x7dcd.ae/portal/dashboards/personal-dashboard (accessed on Jan. 2017).

[23] Dubai Civil Defence presentation [Online]. Available: http://24x7dcd.ae/pdf/DCD-Personal-Dashboard-23-Oct-2015-English.pdf (accessed on Jan. 2017).

[24] S. Suakanto, S. H. Supangkat, Suhardi, and R. Saragih, "Smart city dashboard for integrating various data of sensor networks", in *Proc. Int. Conf. on ICT for Smart Society ICISS 2013*, Jakarta, Indonesia, 2013.

[25] D. Lee *et al.*, "CityEye: Real-time visual dashboard for managing urban services and citizen feedback loops", in *Proc. 14th Int. Conf. on Comp. in Urban Plann. & Urban Manag. CUPUM 2015*, Cambridge, MA USA, 2015.

[26] V. Serbanescu, F. Pop, V. Cristea, and G. Antoniu, "Architecture of distributed data aggregation service", in *Proc. IEEE 28th Int. Conf. on Adv. Inform. Netw. & Appl. AINA 2014*, Victoria, Canada, 2014, pp. 727–73.

[27] E. Barbierato, M. Iacono, and S. Marrone, "PerfBPEL: A graph-based approach for the performance analysis of BPEL SOA applications", in *Proc. 6th Int. Conf. on Perform. Eval. Methodol. & Tools VALUETOOLS 2012*, Cargèse, France, 2012, pp. 64–73.

[28] E. Barbierato, M. Gribaudo, M. Iacono, and S. Marrone, "Performability modeling of exceptions-aware systems in multiformalism tools", in *Proc. Int. Conf. on Anal. & Stoch. Model. Techniq. & Appl. ASMTA 2011*, Venice, Italy, 2011, pp. 257–272.

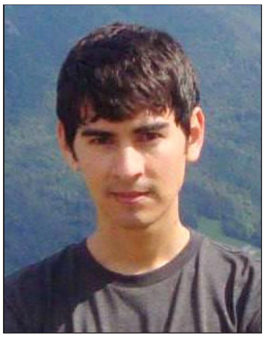
[29] C. Esposito, A. Castiglione, F. Palmieri, M. Ficco, and K. K. R. Choo, "A publish/subscribe protocol for event-driven communications in the Internet of Things", in *Proc. IEEE 14th Int. Conf. on Depend., Autonom. & Secure Computing DASC 2016, IEEE 14th Int. Con. on Perv. Intelligence & Computing PICom 2016, IEEE 2nd Int. Conf. on Big Data Intelligence & Computing DataCom 2016, IEEE Cyber Sci. & Technol. Congr. CyberSciTech 2016 (DASC-PICom-DataCom-CyberSciTec 2016)*, Auckland, New Zealand, 2016, pp. 376–383.

[30] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "A knowledge-based platform for Big Data analytics based on publish/subscribe services and stream processing", *J. of Knowledge-Based Syst.*, vol. 79, pp. 3–17, 2015 (doi: 10.1016/j.knosys.2014.05.003).

[31] C. Esposito, A. Castiglione, and K. K. R. Choo, "Challenges in Delivering Software in the Cloud as Microservices", *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 10–14, 2016.

[32] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing", in *Proc. 6th Int. Worksh. on Netw. Meets Databs. NetDB 2011*, Athens, Greece, 2011.

[33] Cluster Griddy [Online]. Available: http://maplarge.com/visual/clustering (accessed on Jan. 2017).



**Cătălin-Constantin Uşurelu** is a master in Computer Science student within Computer Science Department, University Politehnica of Bucharest. His research interests are oriented on big data processing for smart cities application, data cleaning and aggregation, real time processing. He participated in Google Summer of Code 2013 with a Debian project. He has an excellent knowledge on NoSQL databases (e.g. MongoDB, DynamoDB), RESTful services, AngularJS, JMeter.

E-mail: catalin.usurelu@cti.pub.ro
University Politehnica of Bucharest
Faculty of Automatic Control and Computers
Computer Science Department
Splaiul Independentei 313, Sector 6
Bucharest 060042, Romania

**Florin Pop** is Associate Professor at the Department of Computer Science and Engineering at the University Politehnica of Bucharest. His main research interests are in the field of large scale distributed systems concerning scheduling and resource management (decentralized techniques, re-scheduling), adaptive and autonomous methods, multi-criteria optimization methods, grid middleware tools and applications development (satellite image processing an environmental data analysis), prediction methods, self-organizing systems, data retrieval and ranking techniques, contextualized services in distributes systems, evaluation using modeling and simulation (MTS2). He was awarded with two Prizes for Excellence from IBM and Oracle, three Best Paper Awards (in 2013, 2012, and 2010), and one IBM Faculty Award.

E-mail: florin.pop@cs.pub.ro
University Politehnica of Bucharest
Faculty of Automatic Control and Computers
Computer Science Department
Splaiul Independentei 313, Sector 6
Bucharest 060042, Romania

National Institute for Research and Development
in Informatics (ICI)
8-10, Mareşal Averescu
011455 Bucharest, Romania