A Big Data Framework for Urban Noise Analysis and Management in Smart Cities

Article *in* Acta Acustica united with Acustica · July 2017

DOI: 10.3813/AAA.919084

3 authors, including:

Juan Miguel Navarro
Saint Anthony Catholic University
53 PUBLICATIONS **682** CITATIONS
SEE PROFILE

José Escolano
Hudson Data, NY, United States
79 PUBLICATIONS **728** CITATIONS
SEE PROFILE

Acta Acustica united with Acustica
Vol. 103 (2017) 1 – 1

A Big Data Framework for Urban Noise Analysis and Management in Smart Cities

Juan M. Navarro[1)], J. B. Tomas-Gabarron[2)], Jose Escolano[3)]
[1)] Polytechnic School, UCAM Catholic University of San Antonio, 30107 Murcia, Spain. jmnavarro@ucam.edu
[2)] Peranoid SL, 30800 Lorca, Murcia, Spain
[3)] Sophandrey Research, 11224 Brooklyn NY, USA

Summary
Environmental pollution monitoring is a major concern in the development of smart cities. Nowadays, urban noise is one of the most relevant pollutants, so many networks of acoustic sensors have been deployed to measure sound pressure levels at various locations. These acoustic sensors collect huge amounts of data, which can be helpful to manage noise events in urban planning. In this paper, a big data framework is proposed to properly analyse the considerably large amounts of noise monitoring data and obtain useful information for urban planning. A map and reduce approach is proposed to process the massive data captured from acoustic sensor networks, mobile phones and open data platforms. Using the map and reduce model, several statistical environmental acoustic parameters, including both temporal and spatial indices, can be calculated. As an example application, two algorithms are implemented to evaluate both day-evening-night equivalent levels (L_{den}) and percentile levels (L_n). An experimental case with data obtained from the Dublin open data platform shows the benefits of this framework for urban noise analysis and management.

PACS no. 43.50.q, 43.50.Yw, 43.58.Ta

1. Introduction

The concept of a Smart City, as well as the application of the so-called Internet of Things (IoT) in it, has emerged during recent years as a new technology revolution. A Smart City is a holistic vision of a city that applies information and communication technologies to improve the life quality of its inhabitants and ensure an economically, socially and environmentally sustainable development based on continuous improvement.

Collecting data and efficiently making decisions about the environment are mandatory in a Smart City. Therefore, for the development of the Internet of Things [1, 2], and Smart Cities [3], wireless sensor networks [4] are used. These networks allow fast deployment of a large number of devices and enable real-time monitoring of many parameters, e.g., those for traffic control systems, irrigation of parks and gardens and noise pollution evaluation, that can be used to facilitate sustainable life styles, save costs and improve the quality of life of people.

This vast amount of information is gathered by sensors, and merged with data acquired from other sources, such as mobile applications, social networking and administration management software. However, all the data must be properly stored and processed in order to obtain useful information. Analysing large volumes of data is a significant challenge, so big data techniques [5, 6] should be applied to collect, store and analyse these big data sets efficiently and accurately.

Today's citizens are exposed to excessive noise levels, which cause subjective annoyance [7]. This noise disturbance is the result of repeated and prolonged exposure to these high levels of noise, leading to a decrease in residents' quality of life by interfering with their daily work and rest, and leading disease [8]. Wireless sensor networks have been used to perform acoustic measurements and obtain basic parameters such as sound pressure levels [9, 10] and subjective annoyance [11] to design and update noise maps in cities. Citizens have also collaborated in research by using their mobile phones as mobile sensors for both general parameters [12] and noise [13, 14, 15], showing that the latter greatly depends on the type of device used [16].

In the acoustics field, big data techniques have been applied to analyse sound in captured audio files due the files' large sizes and complexities, as an acoustic sensor can generate several gigabytes of audio data per day. Zhang et al. [17] presented different techniques to store and process environmental audio data obtained from monitoring. In [18], Mugagga et al. used big data to improve sound source localization. Big data techniques have been also applied to audio data for speech analysis [19].

In this paper, a big data framework for noise monitoring in Smart Cities is proposed. There are several potential uses for the huge amounts of noise data acquired

Received 13 September 2016,
accepted 7 February 2017.

from Smart Cities. For example, the analysis of large volumes of long-term noise data could be used to predict future noise levels in areas and improve traffic noise prediction models [20]. Dynamic updating of strategic noise maps [21] with interpolation methods, temporal variability [22] and spatial interpolation in environmental noise monitoring [23] is another example that can also be improved with big data techniques. In this work, two algorithms are implemented, applying a map and reduce strategy to calculate statistical environmental acoustic parameters defined in the European Directive of environmental noise [24], such as day-evening-night equivalent levels (L_{den}) and percentile levels (L_n) from collected massive data.

This paper presents a big data framework to process big data captured by different collection approaches and describes its advantages and possibilities. The main contribution of this paper is to propose a scalable solution that can be used when the data cannot fit into a commodity computer. The manuscript is divided into the following sections: after this introduction, Section 2 outlines big data basics and enunciates different phases of the regular procedure. Section 3 presents the implementation of the map and reduce algorithms. Next, an experimental application of this map and reduce strategy for a data set captured from the Dublin open data platform is described and results are discussed in Section 4. Finally, conclusions are enunciated.

2. Big data basics

The term ”big data” is used to define data sets that are so large or complex that traditional data processing applications are inadequate. Big data requires several techniques and technologies to capture, store, curate and process data and present insights from data sets that are diverse and complex. In this section, some fundamental concepts of big data stages and techniques are described, together with the required frameworks and tools for smart data analysis using big data.

2.1. Obtaining and storing data

The first phase in performing big data analysis is to capture and store valuable information from different data sources. In general, noise pollution data can be massively collected using three different approaches: acoustic sensor networks, mobile phones, and open data platforms.

An acoustic sensor network is composed of nodes that capture sound in near real time and process it to send data (in this case, sound pressure levels) to a gateway or concentrator. These gateways collect data from nodes and transmit them to be stored in the central server, i.e., the Smart City platform. In Smart Cities, wireless communication protocols, such us 3G, WiFi or Zigbee [25, 11], are mostly deployed to obtain data from these kinds of nodes. An acoustic sensor node mostly consists of a hardware board that interfaces with an electret external microphone, a pre-amplifier and an external power supply. It usually has solar panels and batteries for backup, and comes with a communication module. The system should be autonomous and should operate continuously for months with minimal maintenance. In recent years, there has been a huge uptick in noise monitoring network deployment. For instance, a pilot test has been conducted across India’s seven major cities during the last three years [26] enabling the cities to detect the moderate increase of noise in most locations. Moreover, Santander, a Spanish city, has been transformed into a Smart City testbed [27] with the deployment of a vast wireless sensor network, including noise sensors.

Instead of monitoring a fixed location, mobile phones allow both fixed and moving measurements. It is important to note that, an external microphone is recommended to increase precision and accuracy [16]. However, the high battery consumption of mobile phones could be a restriction to a long period of monitoring.

Several already-working Smart Cities have deployed a variety of sensor networks that publish open data sets through their cloud platform. Most of them are based in a public-private cloud platform whose goal is to develop and implement a network of applications and services for the future Internet. In an open cloud platform, several public and private entities share large data traces that they gather with their architecture of sensors.

These platforms should implement a context broker, a so-called generic enabler that enables the input of data to the cloud or the extraction of data captured by sensors. A context broker typically uses a REST API interface, which is a set of functions to perform requests and receive responses via HTTP protocols such as GET and POST. The context broker manages the entire life cycle regarding the update, query, registration and subscription of data from sensors. Likewise, all data are usually stored in a non-relational database. In this regard, no-SQL storage engines are characterized by their ease of use with respect to SQL approaches, as well as their greater horizontal scalability and resilience. This is the reason why no-SQL databases are becoming popular nowadays, especially in the emerging fields of IoT, and are increasingly supported by giants such as Amazon, Facebook and Google. One of the most important open data systems, initially supported by the European Commission, is called FiWare [28]. Other open source initiatives are OpenIoT [29] and the Kaa Project [30]. There also exist commercial solutions as main competitors, such as Microsoft Azure and Amazon Web Services.

2.2. Analysing data

A noise sensor network provides environmental professionals with the capability to massively scale acoustical observations, both temporally and spatially. Due to the nature of the data and their analysis, batch processing seems to be a natural solution: data are stored as they are streamed, but the analysis is performed at a certain point, such as on a daily basis. Therefore, in big data analysis, a

framework is used for distributed processing. One of the top frameworks in use today is Hadoop with its processing model MapReduce [31]. MapReduce is a programming model, written in Java©, and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster [32]. The main advantage of MapReduce is to provide a simplified framework where the developer does not need to solve a distributed/batch problem and instead can focus on the use case itself. One may consider batch techniques in distributed parallel clusters as an alternative approach to solve this problem, but in this case, the developer would need to focus on both the use case and distributed/parallel computing programming, making the application development phase extremely difficult and increasing the development time.

The major advantage of MapReduce is that it is easy to scale data processing over multiple distributed computing nodes. The scalability achieved using MapReduce to implement data processing with low implementation costs across a large volume of CPUs, whether on a single server or multiple machines, is an attractive feature. While MapReduce is not perfect for every use case, it does provide a framework and programming discipline that can be successfully applied to the problem of scaling software applications through multi-core processors and multiple machine cluster infrastructures. The framework abstracts the complex parallelization, synchronization and communication mechanisms and protects the programmer from having to consider these aspects of scalability in each system implementation. Under the MapReduce model, the data primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers can be difficult, but once an application is written in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what attracts many programmers to the MapReduce model.

A MapReduce-based infrastructure orchestrates the processing by arranging the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance. The key contributions of the MapReduce model are the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once. As such, a single-threaded implementation of MapReduce will usually not be faster than a non-MapReduce implementation; any gains are usually only observed with multi-threaded implementations. The use of this model is only beneficial when the optimized distributed shuffle operation, which reduces the network communication cost, and fault tolerance features of the MapReduce model come into play. Therefore, optimizing the communication cost is essential for a good MapReduce algorithm.

A MapReduce algorithm is divided into two important stages, namely Map and Reduce. They perform filtering and a summary operation or roll-up logic, respectively. First, the Map stage takes a set of data and converts it into another set of data, where individual elements are broken down into tuples or key/value pairs. Second, the Reduce task takes the output from a map as an input and combines those data tuples into a smaller set of tuples. Each Reduce stage will process those tuples that have the same key; therefore, a developer controls the degree of parallelism based on the number of different keys. As the name "MapReduce implies", the reduce task is always performed after the map job. Once the reduce task has finished, each reducer generates a single record that contains the output information: sensor id, date, time period (day/evening/night) and equivalent sound pressure level. Moreover, different analytic calculations can be performed to describe the results classified by node, different time intervals (period, day, week, month, and year) and different areas (zones and the whole city).

3. Map and reduce algorithms

Due to the massive amount of data available, it can be challenging to compute analytics in an efficient way. For that reason, a MapReduce-based framework (see Section 2.2) has been chosen to perform the environmental acoustic parameter calculation. The following implementation will perform the calculations in a distributed platform in a simple way, without spending too much effort focusing on how to develop the algorithm in a distributed manner. The importance of using big data appears when dealing with data coming from many sensors, with high refresh rates and a long period of measurement time.

One of the most essential steps relies on how the map phase generates one or more key/value pairs for each record. Every key/value pair that contains the same key will be processed by the same reduce process, independent of and in parallel with other similar reducer processes. As an example, when using the MapReduce paradigm for acoustic purposes, a map stage may be used to read all the records and creates a unique key for each different date. For each sound pressure level value recorded on that particular date, it would create a single key/value pair such as $< 2016/08/25, L_1 >, < 2016/08/25, L_2 >, \ldots$ Therefore, a single Java Virtual Machine instance will run the reduce stage and only would process those key/pair values with the same key. In this way, it can control the degree of parallelism.

For more advanced problems, MapReduce enables control of some intermediate steps between the map and reduce stages. One of the most important intermediate stages is known as the Shuffle-and-Sort stage. By default, all key/value pairs with keys defined as a simple standard type, such as an integer or float, are sorted by key. In some cases, this stage of the framework can provide the developer with control over the order in which the values for a given key arrive at the reduce stage. This is controlled by a technique called secondary sorting. As Parsian, M. enunciated in Ref [33], a secondary sort problem relates to

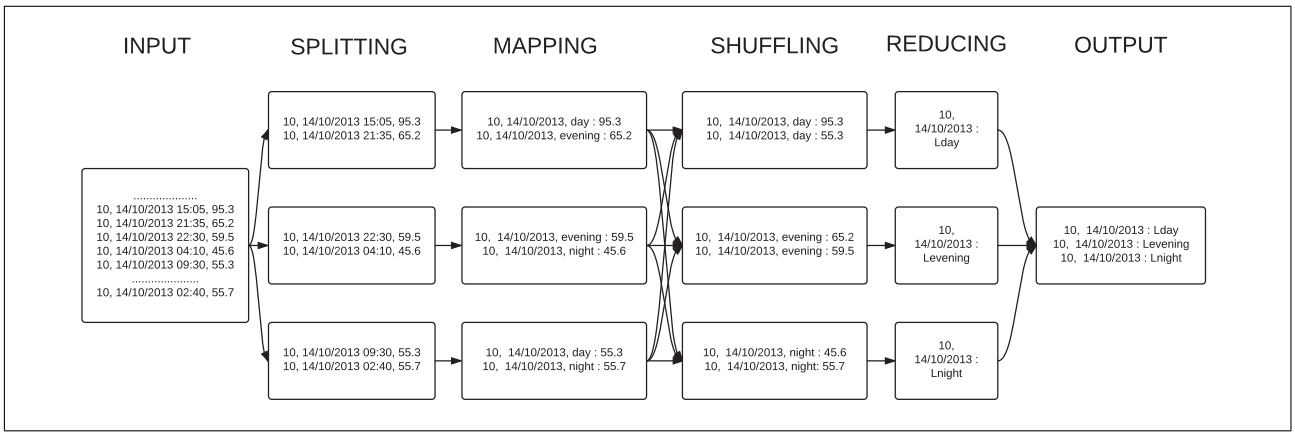

Figure 1. Diagram of a MapReduce process to calculate the equivalent sound pressure level for every period.

sorting values associated with a key in the reduce phase. Sometimes, it is called value-to-key conversion. The secondary sorting technique is used in these algorithms to sort the values in ascending or descending order and pass them to each reducer. It allows the developer to take the value into account during the sorting phase with a slight manipulation of the format of the key object. Following the previous example for sound pressure levels, this provides an additional advantage of receiving the key/values sorted by value, so the percentile distribution, for example, can be calculated easily.

In this paper, as an initial example of the possibilities of using MapReduce in environmental acoustics, every collected data from noise nodes are processed in order to calculate the Community Noise Equivalent Level (CNEL) [34], or day-evening-night equivalent level (L_{den}). This parameter, L_{den}, is an A-weighted equivalent noise level measured over a 24 hour period with a penalty in the evening and the night periods, as defined in

$$L_{\text{den}} = 10\log\left[\frac{12}{24}\cdot 10^{L_{\text{day}}/10} + \frac{4}{24}\cdot 10^{(L_{\text{evening}}+5)/10} + \frac{8}{24}\cdot 10^{(L_{\text{night}}+10)/10}\right]. \quad (1)$$

L_{day} is the A-weight equivalent noise level measured between 7 and 19 hours, $L_{evening}$ is the A-weight equivalent noise level measured between 19 and 23 hours, and L_{night} is the A-weight equivalent noise level measured between 23 and 7 hours. Therefore, the equivalent noise level in the different periods, day, evening, and night, must be calculated for each node, and then L_{den} can be computed for nodes, zones and the whole city. L_{den} is defined as a long-term measurement because it must be determined for all day-evening-night periods of one year.

Figure 1 is a diagram of the MapReduce process used to obtain the equivalent sound pressure level in each period. In this particular case, node number 10 is shown. The input data are collected from different sensors, formatted as sensor identification number, date, time and sound pressure level and split into several mappers where the tuples are created. The key for each tuple corresponds to the information about sensor identification number, date and calculated period of time (day, evening or night), and its corresponding value is the sound pressure level. Next, in the shuffling stage, pairs with the same key are grouped and passed to a single machine, which will then run the reduce script over them. The reduce script takes a collection of key/value pairs and processes them to calculate the equivalent sound pressure level of each period, i.e., L_{day}, $L_{evening}$ and L_{night} in Figure 1.

Going into detail for this first problem, L_{den} evaluation, the map phase has two main purposes:

- To clean and filter records that do not contain substantial information by eliminating sound pressure levels that are value is missing or zero.
- To create a proper key to submit each record to its corresponding reducer. Since the purpose is to parallelise the analysis of each sensor's contribution for a given time period, the key is defined as a 3-tuple that contains the sensor identifier, date and period of time that it belongs to.

Regarding the sort and shuffle phase, it proceeds in the following way:

- Knowing that by default, for single value keys, the MapReduce-based framework sorts the keys before sending the key/value pairs to their corresponding reducers, it is necessary to customize how keys are sorted when *n*-tuples are used as keys. For complex keys (*n*-tuples), it is necessary to indicate the sorting hierarchy, using a secondary sorting technique [33].
- In this particular case, each reducer will receive all the keys with the same sensor identifier, period and date and sort them by the time of the given date.

The pseudo-code representing this map function is outlined in Algorithm 1.

Algorithm 1: Map function pseudo-code for L_{den} evaluation.

```
map(String key, String value)
// key: document name
// value: single record

for each value
 if (value.isValid())
```

```
  newKey.setSensorID(value.getSensorID)
  newKey.setDate(value.getDate())
  newKey.setPeriod(value.getPeriod())
  EmitIntermediate(newKey, value)
```

Algorithm 2: Reduce function pseudo-code for L_{den} evaluation.

```
reduce(String key, Iterator values):
// key: contains Id, Date and Period
// values: a list of records

for each v in values:
 dT = getDeltaTime(v.getDate(), prev)
 prev=v.getDate()
 totalTime+=dT
 result += dT*pow(10, v.getSPL()/10)
Emit(key, AsString(result))
```

Once the two previous stages are finished, the reducer function will proceed with a set of aggregations to calculate the desired average sound pressure levels (see Algorithm 2). Because every record is sorted by time, it is possible to calculate the time difference and perform some of the analytics that this paper aims at.

Moreover, a second application of MapReduce is presented to calculate the percentile distribution and evaluate the percentile sound pressure levels, L_n, for the overall nodes each day. L_n, where n can range from 1 to 99, is the noise level exceeded for n% of the measurement time [34]. Therefore, an advanced MapReduce algorithm is designed using the secondary sorting technique described above.

Algorithm 3: Map function pseudo-code for L_n evaluation.

```
map(String key, String value)
// key: document name
// value: single record

for each value
 if (value.isValid())
  newKey.setSensorID(value.getSensorID)
  newKey.setDate(value.getDate())
  EmitIntermediate(newKey, value)
```

Algorithm 4: Reduce function pseudo-code for L_n evaluation.

```
reduce(String key, Iterator values):
// key: contains Id and Date
// values: a list of records

for each v in values:
 dT = getDeltaTime(v.getDate(),prev)
 prev=v.getDate()
 totalTime+=dT
 result += dT*pow(10, v.getSPL()/10)
Emit(key, AsString(result))
<percentils>=Percentils(sort(values))
Emit(key, AsString(<percentils>))
```

Regarding this second job, L_n evaluation, some readjustments of the previous codes are necessary (see Algorithms 3 and 4 for details). The map phase now creates a key/value pair in which the key contains the sensor identifier and only the given date. The shuffle and sorting phase will time sort the records for a given sensor ID. Then, the reducer algorithm will calculate both average value and percentile distributions. Moreover, each reducer instance will only apply to a given sensor and for a given day in this case.

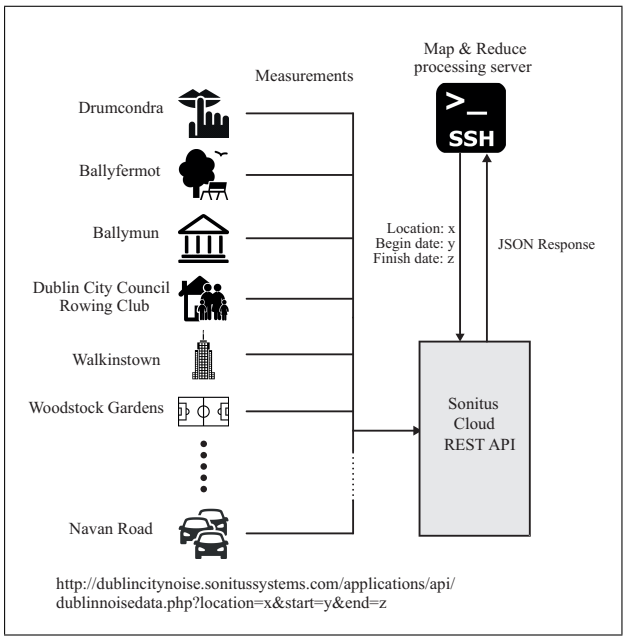

Figure 2. Architecture of the system that shows some of the measuring locations (on the left side).

4. Experiment with real dataset from a Smart City

In this section, an experiment is presented that test the developed big data algorithms, see Section 3, using a real data set from an open data platform.

4.1. Infrastructure for the experiment

Thanks to the Sonitus Systems Platform [35], a large data set of environmental noise levels in the city of Dublin, Ireland, has been shared as open data. This platform provides a REST API that provides access to noise level recordings from the Dublin City Council Sound-Level Monitoring Network (see Figure 2). A group of environmental noise monitors, EM2010 Sound Level Monitoring Stations, are located at strategic positions around the city (illustrated vertically in Figure 2, see [36] for an online map). Each acoustic monitoring station operates on a 24/7 basis and reports noise statistics at user-programmed intervals via a commercial mobile communication network. Each unit is equipped with a Class 2 environmental microphone, ensuring that noise measurements are compliant with IEC61672 [37], and the dynamic range of the system is 33 dBA to 121 dBA. By sending REST requests to the system via the Uniform Resource Locator at the bottom of Figure 2, it is possible to receive noise levels in the JSON format according to the specific station (x parameter), beginning monitoring date (y parameter), and ending monitoring date (z parameter).

In this experiment, an analysis of a data set covering a time interval of 3 years, from 2/8/2013 to 2/8/2016, has been conducted. The monitoring stations capture several noise levels at regular intervals. In particular, the equivalent sound pressure levels are gathered in dBA, L_{eq}, and

calculated for 5 minute periods from 13 sensors. This leads to 288 measurements per sensor every day, which totals 105,120 measurements a year per sensor, 1,366,560 measurements for all 13 sensors, and 4,099,680 registrations. In this specific case, it is possible to obtain the same results without using the big data framework but instead using a standard data structure such as a hashmap. However, this standard solution can quickly run out of memory and will be limited to only applications where the number of records is small enough to fit into memory. Therefore, that standard solution will limit either the number of active sensors or sampling frequency. The proposed framework is completely scalable, and the number of sensors can reach hundreds of thousands, increasing the sampling frequency without affecting the application.

In terms of computing storage, each measurement is around 100 bytes, totalling 409.9 MB for all the data captured during the three year period . Thus, each node has an individual storage of 10.5 MB per year. To achieve a big data use case, we imagine the hypothetical storage that would be needed for 200 sensors capturing data for a 1 minute period for 5 years. This amount of data reaches a total of 10.5 GB for a 1 year period and thus 52.5 GB for 5 years. Given the exponential growth of acoustic sensor deployment in cities, such as in Dublin, more than 500 additional units could be installed. Therefore, the amount of streamed data implies a big data situation.

To extract and gather the data from the Dublin open data platform, a context broker using the REST API was implemented. This broker, as explained in Section 2.1, stores data in a no-SQL persistent database. For this experiment, the broker feeds an Elastic MapReduce infrastructure into Amazon Web Services [38], where all the corresponding calculations were carried out by applying the algorithms presented in Section 3. Finally, L_{den} and L_n results are obtained together with several analytics. An Amazon Web Services cluster with 4 virtual CPUs and 16 GB of memory was used to perform the calculations. The total runtime for both MapReduce jobs was less than 6 minutes.

It is important to note that the proposed framework will scale directly if either the amount of data or the number of sensors increases. A MapReduce infrastructure is able to manage any of these circumstances directly. In case the runtime is a constraint, simply increasing the number of virtual machines in the cluster environment will reduce the runtime.

4.2. Results and discussion

This section presents some examples of results that can be obtained by applying the big data algorithms described in Section 3 to the experimental infrastructure presented in the above section.

To evaluate the resulting noise parameters, the guideline values recommended by the World Health Organization [7] have been used. A L_{den} of 70 dBA was selected because the guidelines indicate potential hearing impairment above this level. The level of 65 L_{eq} dBA, when important annoyance starts, was chosen for both day and evening time periods. Moreover, the level of 55 L_{eq} dBA was selected for the nighttime period.

Table I. L_{den}, L_{day}, $L_{evening}$ and L_{night} levels in dBA for the 13 sites.

Node	L_{den}	L_{day}	$L_{evening}$	L_{night}
1	65.0	64.4	61.2	55.1
2	76.0	75.7	72.1	65.7
3	66.7	66.1	62.7	56.8
4	72.5	71.9	68.5	56.8
5	65.9	65.4	61.8	56.2
6	64.5	64.0	60.6	54.5
7	64.5	59.1	55.8	50.2
8	65.3	64.8	61.3	55.4
9	64.3	63.9	60.2	54.5
10	69.9	69.8	65.4	59.8
11	70.9	70.5	66.8	60.9
12	63.5	62.7	59.9	53.8
13	65.2	64.8	61.0	55.3

L_{den} and L_{night} were obtained as output values of the first map and reduce algorithm using the three years' data from the 13 nodes in Dublin. They are presented in Table I. Three sites exceed an L_{den} of 70 dBA: nodes 2, 4 and 11. They experienced undesirably high sound levels, but only nodes 2 and 11 exceed an L_{night} of 60 dBA, corresponding to two sites located close to major arterial roads. Moreover, a group of nodes, numbers 1, 3, 5, 6, 7, 8, 9, 12 and 13, can be highlighted that have L_{den} values around 65 dBA and L_{night} values around 55 dBA, which are considered desirable sound levels [7]. The lowest L_{den} was obtained at node 12, 63.5 dBA, but the lowest L_{night} was calculated at node 7, 50.2 dBA.

A statistical analysis of the complete data set was performed to examine the variation between sensor locations and periods at each location. In Figure 3, a boxplot is shown using the median and quartile results, obtained with the map and reduce algorithm, of the 3 years of L_{eq} dBA measurements. In this graph, each device is designated with the number of the node and a notation for the period: D, E and N correspond to day, evening and nighttime respectively. It is observed that the interquartile range is small for most locations, except for node 2. Moreover, this range is always bigger for daytime, followed by evening, and nighttime has the smallest range. In general, all locations present a symmetrical distribution of higher and lower values around the box. The median value follows the same trend as the interquartile range by period, with the higher levels corresponding to daytime and the lower ones to nighttime. Nodes 2, 4 and 11 show that the highest values overall are locations near roads.

To explore the behaviour of one of these highlighted nodes, a boxplot for monthly measurements at node 4 during 2014 is presented in Figure 4. With this type of analysis, the sound environment of the location can be mapped. Node 4 presents interquartile ranges lower than 10 dBA, which is relatively small for all months. Moreover, all of the months, except for January, show higher values in the upper whiskers, which indicates that the site can be ex-

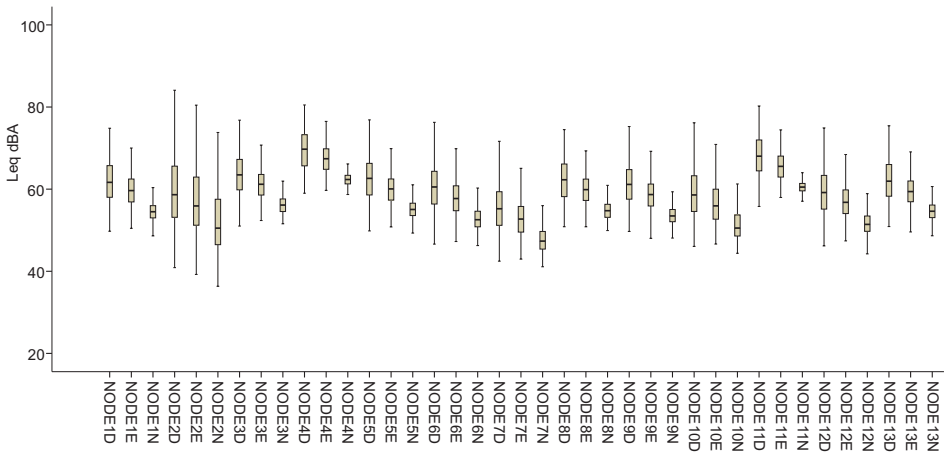

Figure 3. Box plot of daytime, evening and nighttime L_{eq} dBA for every node of the three year interval.

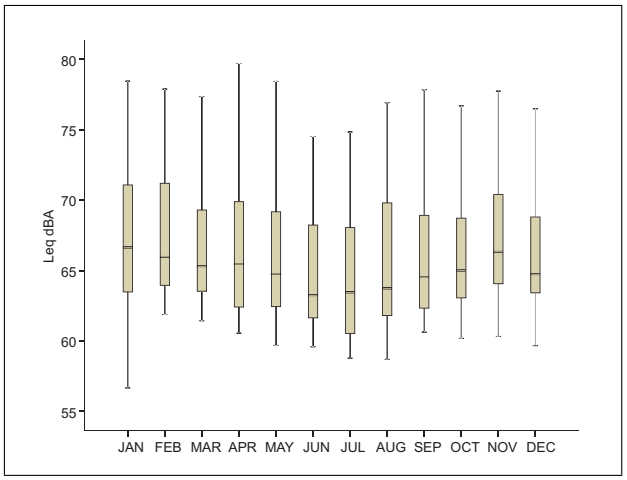

Figure 4. Boxplot of L_{eq} dBA per month at node 4 during the year 2014.

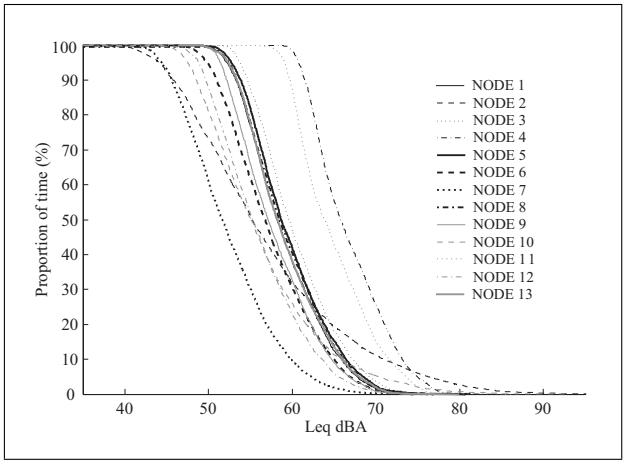

Figure 5. Percentage of time of exceeded L_{eq} dBA for every node of the three year interval.

posed to high noise levels at times. The summer months June, July and August have reduced levels of noise, which may be due to lower traffic during the holiday months.

The second implemented map and reduce algorithm was applied to calculate the experimental cumulative distribution function of noise levels for each node over all measurements. Then, using the obtained percentile levels, an analysis of the L_{eq} dBA levels was performed. The percentage of time that noise levels are exceeded are presented in Figure 5. In general, there are noticeable differences between the locations. However, nodes 1, 5, 8 and 13 show similar trends. The lowest level that is exceeded 100% of the time is 40 dBA at node 2. However, most sites have levels over 50 dBA 100% of the time. Moreover, Figure 5 can also be used to determine what proportion of the time some reference noise levels are exceeded. For example, node 7 shows levels above 60 dBA for only 9% of the time, compared to node 4, wich exceeds it 98% of the time. The remainder of the nodes exceed this level from 22% to 78% of the time.

5. Conclusions

The large increase in the amount of noise level data collected by both fixed and mobile acoustic sensor networks in Smart Cities requires new analysis approaches. Big data has techniques to analyse these data and extract useful information to help understand the data and make better decisions. In this paper, a big data framework for statistical analysis of noise level monitoring in smart cities has been proposed. Big data basics have been described with special emphasis on several approaches to obtain and store data, sensor networks, mobile phones and open data platforms. Moreover, we have discussed the advantages of Hadoop and its processing model MapReduce in analysing large data sets. A map and reduce model was then used to implement two different algorithms, which calculate several environmental acoustic parameters, e.g., the L_{den} and L_n noise levels. Finally, these algorithms were tested in an experiment with real data obtained from the Dublin open data platform. These implemented map and reduce algorithms are an example of efficient procedures to extract environmental acoustic parameters from an acoustic wire-

less sensor network in a distributed and scalable manner. In future works, larger and more diverse audio and acoustic parameters data sets will be retrieved from several city databases to obtain conclusions about the temporal and spatial variability of noise with the help of big data techniques. Moreover, the application of near-real time analytics to gain insights from acoustic data could be interesting for noise control and management.

Acknowledgement

This research is jointly supported by the Spanish MINECO under Project ref. TIN2016-78799-P and by the UCAM Catholic University of San Antonio under Project ref. PMAFI-02-14.

References

[1] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami: Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems **29(7)** (2013) 1645–1660.

[2] L. Atzori, A. Iera, G. Morabito: The internet of things: A survey. Computer networks **54(15)** (2010) 2787–2805.

[3] R. Kitchin: The real-time city? Big data and smart urbanism. GeoJournal **79(1)** (2014) 1–14.

[4] I. F. Akyildiz, M. Can Vuran: Wireless sensor networks. John Wiley & Sons, 2010.

[5] C. Lynch: Big data: How do your data grow?. Nature **455(7209)** (2008) 28–29.

[6] V. Marx: The big challenge of big data. Nature **498(7453)** (2013) 255–260.

[7] B. Berglund, T. Lindvall, H. Dietrich: Guidelines for community noise. World Health Organization, 1999.

[8] W. Passchier-Vermeer, W. F. Passchier: Noise exposure and public health. Environmental health perspectives **108(1)** (2000) 123.

[9] A. Prabahar: Development of high performance wireless sensor node for acoustic applications. Proc. IEEE International Conference on Green High Performance Computing (ICGHPC), 2013.

[10] Y. You, Y. Jaehyun, C. Hojung: Event region for effective distributed acoustic source localization in wireless sensor networks. Proc. IEEE Wireless Communications and Networking Conference (WCNC), Mar. 2007.

[11] J. Segura, S. Felici, J. Perez-Solano, M. Cobos, J. M. Navarro: Low-cost alternatives for urban noise nuisance monitoring using wireless sensor networks. IEEE Sensors Journal **15(2)** (2015) 836–844.

[12] L. Jaimes, J. Idalides, A. Raij: A Survey of Incentive Techniques for Mobile Crowd Sensing. IEEE Internet of Things Journal **2(5)** (2015) 370–380.

[13] N. Maisonneuve, M. Stevens, M. Niessen, L. Steels: NoiseTube: Measuring and mapping noise pollution with mobile phones. In Information Technologies in Environmental Engineering (pp. 215–228), Springer Berlin Heidelberg, 2009.

[14] R. Rana, C. T. Chou, N. Bulusu, S. Kanhere, W. Hu: EarPhone: A context-aware noise mapping using smart phones. Pervasive and Mobile Computing **17** (2015) 1–22.

[15] E. Kanjo: Noisespy: A real-time mobile phone platform for urban noise monitoring and mapping. Mobile Networks and Applications 15.4 (2010): 562-574.

[16] C. A. Kardous, P. B. Shaw: Evaluation of smartphone sound measurement applications. J. Acoust. Soc. Am. **135(4)** (2014) 186–192.

[17] J. Zhang, K. Huang, M. Cottman-Fields, A. Truskinger, P. Roe, S. Duan, X. Dong, M. Towsey, J. Wimme: Managing and analysing big audio data for environmental monitoring. Proc. IEEE 16th International Conference on Computational Science and Engineering (CSE), Dec. 2013.

[18] P. Mugagga, K. Basajjabaka, S. Winberg: Sound source localisation on Android smartphones: A first step to using smartphones as auditory sensors for training AI systems with Big Data. Proc. IEEE AFRICON, Sep. 2015.

[19] B. W. Schuller: Speech Analysis in the Big Data Era. Text, Speech, and Dialogue **14** (2015) 3–11.

[20] C. Steele: A critical review of some traffic noise prediction models. Applied acoustics **62(3)** (2001): 271–287.

[21] W. Weigang, T. Van Renterghem, B. De Coensel, D. Botteldooren: Dynamic noise mapping: A map-based interpolation between noise measurements with high temporal resolution. Applied Acoustics **101** (2016) 127–140.

[22] D. Geraghty, M. O'Mahony: Investigating the temporal variability of noise in an urban environment. International Journal of Sustainable Built Environment **5(1)** (2016) 34–45.

[23] A. Can, L. Dekoninck, D. Botteldooren: Measurement network for urban noise assessment: Comparison of mobile measurements and spatial interpolation approaches. Applied Acoustics **83** (2014) 32–39.

[24] Directive 2002/49/EC of the European parliament and the Council of 25 June 2002 relating to the assessment and management of environmental noise. Official Journal of the European Communities **189** (2002).

[25] P. Barontib, P. Pillaia, V. Chooka, S. Chessab, A. Gottab, F. Hua: Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards. Computer communications **30(7)** (2007) 1655–1695.

[26] N. Garg, A. K. Sinha, V. Gandhi, R.M. Bhardwaj, A.B. Akolkar: A pilot study on the establishment of national ambient noise monitoring network across the major cities of India, Applied Acoustics **103** (2016) 20–29.

[27] L. Sanchez, L. Munoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, D. Pfisterer: SmartSantander: IoT experimentation over a smart city testbed. Computer Networks **61(14)** (2014) 217–238.

[28] F. Ramparany, F. G. Marquez, J. Soriano, T. Elsaleh: Handling smart environment devices, data and services at the semantic level with the FI-WARE core platform. IEEE International Conference on Big Data (Big Data), Oct. 2014.

[29] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J. P. Calbimonte, M. Riahi, L. Skorin-Kapov: Openiot: Open source internet-of-things in the cloud. In Interoperability and Open-Source Solutions for the Internet of Things (pp. 13-25), Springer International Publishing, 2015.

[30] Kaa Project, www.kaaproject.org, September 2016.

[31] T. White: Hadoop: The definitive guide. O'Reilly Media Inc., 2012.

[32] J. Dean, S. Ghemawat: MapReduce: simplified data processing on large clusters. Communications of the ACM **51(1)** (2008) 107–113.

[33] Parsian, M.: Data Algorithms: Recipes for Scaling Up with Hadoop and Spark. O'Reilly Media Inc., 2015.

[34] ISO 1996– 2: Acoustics – Description, measurement and assessment of environmental noise – Part 2: Determination of environmental noise levels, 2007.

[35] Dublin City Environmental Noise. Sonitus Systems. Noise monitoring web-based interface, dublincitynoise.sonitussystems.com, September 2016.

[36] Dublin City Environmental Noise. Sonitus Systems. Noise monitoring locations, dublincitynoise.sonitussystems.com/locations.php, September 2016.

[37] IEC 61672: Electroacoustics – Sound Level Meters Tests. International Electrotechnical Commission, 2013.

[38] Amazon Elastic MapReduce web service, aws.amazon.com/emr, September 2016.