ORADIEX: A Big Data driven smart framework for real-time surveillance and analysis of individual exposure to radioactive pollution

Hadi Fadlallah
Lebanese University
Beirut, Lebanon
Hadi.Fadlullah@gmail.com

Yehia Taher
Université de Versailles – Paris-Saclay
Versailles, France
yehia.taher@uvsq.fr

Rafiqul Haque
Intelligencia R&D
Paris, France
Rafiqul.Haque@intelligencia.fr

Ali Jaber
Lebanese University
Beirut, Lebanon
ali.jaber@ul.edu.lb

***Abstract*— Radiation pollution has been always a critical concern, since it can cause a huge damage to humans and for nature. To minimize the damage, governments are collecting and monitoring radiation level using advanced systems. In the past years, Big data technologies such as distributed file systems, NoSQL databases and stream processing technologies was implemented in the radiation monitoring systems to improve their abilities to handle huge volume of data coming from different sources in a high speed. As Big data technologies are being improved frequently to handles the fast growth of the data, these systems need to be updated and improved periodically to adopt new technologies and to guarantee a higher control over radiation exposure. In this paper, we proposed a system called ORADIEX which is an improvement of our previous published work RaDEn [2]. It has the ability to (1) reading data from sensors and different sources, (2) processing data in real-time, (3) stores raw radiation data as it comes from sources, (4) clean data and stores it in a time-series database, (5) visualize and monitor data in real-time, (6) send alert when a high radiation level is detected and (7) allow performing advanced data retrieval operations over raw and processed data. In addition, this system was implemented and tested using a real dataset provided by the Lebanese Atomic Energy Commission (LAEC-CNRS).**

Keywords—Radiation, data engineering, Big Data, radiation monitoring, real-time processing

I. INTRODUCTION

Preventing and controlling radioactive exposures still one of the most critical duties of governments and researchers, since it has a catastrophic effect on every living beings [1]. Prevention activities can be classified into three main categories: (1) physical protection, (2) radiation monitoring and (3) handling exposures.

Radiation monitoring is considered as the most challenging part, since it requires building intelligent systems that are able to collect, analyze, visualize and raise alert when an exposure is detected.

Due to the fast technology growth, collecting radiation data can be done from a wider variety of data sources such as small wireless sensors, mobile phones, smart watches. In addition, the data management technologies are improved in frequently to be able for handling the data sources growth. To be able to handle data coming from multiple data sources in real-time, radiation monitoring systems must adopt the new data technologies and need to be improved periodically.

Several data engineering systems that relies on new data technologies such as NoSQL databases, distributed file system were proposed in literature such as [3][4][5][6][7] and other solutions. These solutions have two main limitations that (1) they cannot handle a huge volume of data in real-time, (2) fault-tolerance and scalability are not always guaranteed. As earlier, we proposed a radiation engineering system called RaDEn [2] which is built using Big Data technologies such as Hadoop [1] distributed file system, and real-time data ingestion tools, this system solved the problem related to reading and storing huge radiation data but it still have many limitations as it cannot (1) visualize data from different sources, (2) notification system was not implemented,(3) historical data cannot be visualized since it is saved in raw format,(4) historical alert information are not stored, (5) real-time graph is very basic and shows only last 30 measurements,(6) cleaned and processed data was visualized without being saved and (7) it doesn't have a user friendly interface.

In this paper, we are proposing a radiation monitoring system called ORADIEX were we improved the old system RaDEn [2] by (1) adding a distributed time-series NoSQL database (InfluxDB[2]) to store data after being cleaned and processed, (2) adopting a powerful real-time monitoring framework (Grafana[3]) that has a user friendly interface and allows drawing real-time graphs from different sources, designing dashboards, visualizing data already stored, saving historical information about exposures and sending email alerts when a radiation exposure detected.

[1] http://hadoop.apache.org
[2] http://influxdata.com
[3] http://grafana.com

The rest of this paper is organized as follows. In Section 2, we briefly introduce our solution called ORADIEX. The data processing is described in section 3. The development of ORADIEX will be detailed in Section 4. Section 5 demonstrates ORADIEX. We conclude our work in Section 9.

II. An Overview of ORADIEX

ORADIEX is a data engineering platform that has the ability to read huge amounts of data from multiple sources with different formats using a scalable and distributed message broker, clean and process the collected data, store data within a scalable storage, visualize data in real-time graphs and raise alert when a high radiation level is detected.

ORADIEX stores data in its raw format within a scalable and fault-tolerant data lake to insure data governance, and stores processed and cleansed data in JSON format within a scalable NoSQL time-series database that allows user to perform data retrieval operations.

ORADIEX can handles data caught by sensors in real-time, also it can handle data from other sources such as databases or flat files.

ORADIEX architecture is composed of 6 layers as shown in Figure-1:

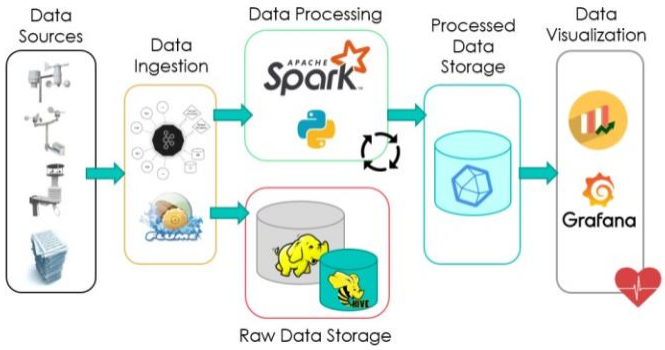

Figure 1 - ORADIEX architecture

- **Data Sources:** data sources consist of data generated from sensors, or data stored within flat files or relational databases
- **Data Ingestion:** This layers consist of a scalable message broker and other ingestion tools that allows reading data generated from the different sources and send it at the same time to the data processing and raw data storage layers.
- **Raw Data Storage:** This layer consists of a scalable data lake built on the top of a distributed file system where data is stored as it comes from the sources without editing. Beside of the data lake, metadata is stored in a metadata repository that allows users to perform data retrieval operations.
- **Data Processing:** This layer relies mainly on a distributed data processing framework that allows processing huge volume of data. In this layer, data are cleansed and transformed into JSON format to be stored within the processed data storage layer.
- **Processed Data Storage:** This layer consists of a distributed and scalable data warehouse that has the ability to store a time series data having different attributes.
- **Data Visualization:** This layer allows user to create dashboards that can visualize newly inserted data into the processed data storage layer in real-time. Also, it gives the ability to perform data retrieval operations and to send email notifications when a radiation exposure is detected.

III. Data Processing

Since data comes from different data sources, the incoming data quality must be assessed and improved. In the data processing layer, we implemented a simple data cleansing and quality assurance process using the following steps:

1. The measurement date is validated; if it is not a valid date time value the data is rejected, else it will be converted to a universal date time format *(yyyy-MM-ddTHH:mm:ss)*.
2. The other measurements are validated; if any measurement cannot be parsed to a numeric value it will be removed.
3. All empty strings are replaced with NULLs.
4. Data is converted to standard format (JSON[4]), to be stored in the processed data storage layer.

IV. Development of ORADIEX

Each layer of ORADIEX was deployed on a separate virtual machine where Ubuntu[5] 16.04 LTS was used as operating system.

For data ingestion, we have used one virtual machine where we installed and configured an Apache Kafka[6] broker to read data from different sources. Beside of the message broker, we installed an Apache Flume[7] agent to send data to the data lake directly when it is received by the message broker. In addition, we installed Apache Sqoop[8] on the ingestion layer, to give the user the ability to import archival data from relational databases directly into the data lake.

We have chosen these technologies since they all guarantee a high scalability and fault-tolerance.

For the raw data storage, we have built a 4-node Hadoop 3.1.0 cluster where we configured one virtual machine (node) to act as a master node and the others as slaves (data nodes). We have set the replication factor to 3, so when data is sent to the Hadoop master node it will be replicated in all 3 data nodes. The Hadoop Distributed File System (HDFS) guarantee a high level of scalability and fault-tolerance.

In addition, we have installed Apache Hive[9] to store the metadata so it can be used to perform data retrieval operations using SQL-like language over the raw data.

For the data processing, we have deployed a single node Apache Spark cluster on a separate virtual machine. Apache Spark is a distributed, scalable and fault-tolerant processing

[4] http://json.org
[5] http://ubuntu.com
[6] http://kafka.apache.org
[7] http://flume.apache.org
[8] http://sqoop.apache.org
[9] http:/ /hive.apache.org

framework, that has the ability to process data at rest and in real-time.

To implement the processing logic, we coded a Python[10] script that uses PySpark library which is a wrapper of Spark. The script listens to the message broker and read newly added data, then it filters the bad rows as described in the data cleansing section. After ensuring the data quality, the data is transformed to a JSON format to be stored in the processed data storage layer.

To store the processed data, we used a time-series NoSQL database called InfluxDB which guarantee a high scalability.The reason for using a time-series database is that the main key in the data we are working on is the date and time of measurement.

The data is stored in JSON format within InfluxDB. Each JSON value is composed of 4 parts as shown Figure-2:

```
[{
    "measurement": "radiationdata",
    "tags":
        {
            "station": "Beirut"
        },
    "time": "2018-01-01 00:00:00",
    "fields":
        {
            "rain_level": 0.00,
            "external_battery": 0.00,
            "internal_battery": 0.00,
            "temperature": 37.00,
            "dose_rate": 54.00
        }
}]
```

Figure 2 – Processed data JSON structure

- **Measurement**: The name of the table where data is stored
- **Time**: The date and time of the measurement
- **Fields**: A list of values that can be visualized (rain level, radiation level, …)
- **Tags**: A list of values that can be used to filter data (station name)

For visualization, we have installed a tool called Grafana used for real-time monitoring. It allows designing dashboards to visualize data, and querying the data stored within the InfluxDB. In addition to this , it allows defining a radiation level limit for each graph (we can define one for each station since the radiation level is affected by the weather and temperature factors which differs between locations) and to send email alert when this limit is reached. Grafana was installed on same machine of InfluxDB to guarantee a real-time visualization.

V. Demonstration of ORADIEX

In this section, we demonstrate ORADIEX. For our demonstration we used a radiation dataset supplied by the department of environmental radiation control at the Lebanese Atomic Energy Commission (LAEC-CNRS).

[10] http://python.org

A. Dataset

Accessing the sensors or the web server (relational database) was not made due to confidentiality issues. The dataset was provided as flat files with data collected from 2015-08-01 to 2016-08-01 from a testing sensor that was installed in Beirut. The data set structure is described in Table-1.

Column name	Data Type	Unit	Description
Measurement_time	Datetime		Measurement date and time
dose_rate	Numeric	nSv/h	The radiation dose rate
Temperature	Numeric	C	Temperature
Rain_Level	Numeric	mm/h	The rain level
Sensor_battery_power	Numeric	mV	The sensor internal battery power
External_battery_power	Numeric	mV	The sensor external battery power
Station_Name	Text		The sensor station name

Table 1 - Data set structure

B. Starting and Configuring ORADIEX

First of all, we started all virtual machines (Ingestion, Processing, Raw Storage, Processed data storage). We started the following services:

- Apache Kafka, Apache Flume services on the Ingestion machine.
- Hadoop Cluster (Name node and data nodes) on the Raw data storage machines.
- Apache Spark Cluster on the data processing machine.
- The python script on the data processing machine.
- InfluxDB and Grafana Services on the monitoring machine.

To simulate data ingestion from sensors, we have created a directory where we must copy data set provided, and we created a terminal script that creates a listener on this folder. When any file is inserted, the script loops over the lines and send them one by one to the Apache Kafka producer.

Using Grafana, we created a dashboard that contains one graph that visualize the radiation dose rate, the rain level and the temperature data received from only Beirut Station and we set the radiation limit to 45 as shown in Figure-3.

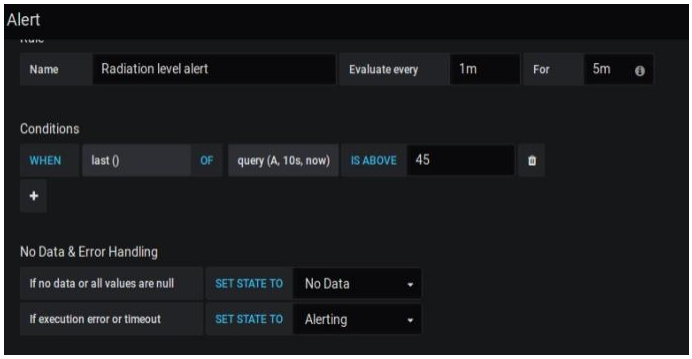

Figure 3 - Configuring Alert

Moreover, we have configured the email notification settings where you can add many recipients and write the custom message you want as shown in the Figure-4.

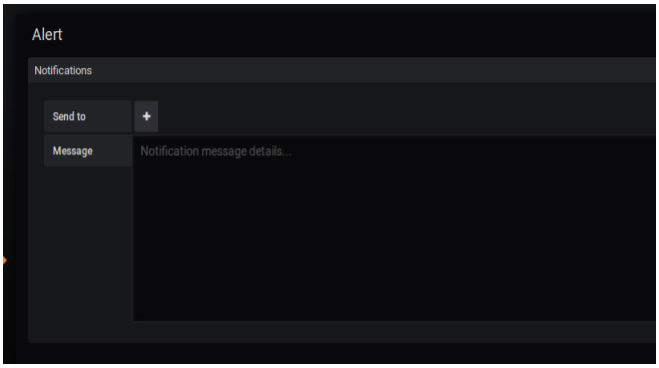

Figure 4 - Configring Email notification

C. Radiation Monitoring

After starting and configuring ORADIEX, we copied the data set to the ingestion directory. The data was visualized in real-time on the dashboard we created (Figure-5).

Figure 5 - Monitoring Dashboard

In addition, notification email was received when radiation level as exceeded at the same time all alert was recorded in the dashboard alert list as shown in Figure-6.

Figure 6 - Alert list

D. Retrieving Data

As shown in Figure-7, we can perform data retrieval operations from the InfluxDB database using Grafana interface, and the result is visualized as a graph.

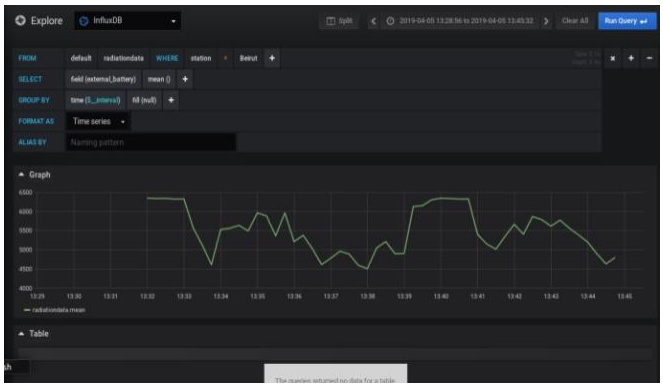

Figure 7 - Data Retrieval from InfluxDB using Grafana

VI. CONCLUSION

In this paper, we designed a solution called ORADIEX which is an improved version of our previous work RaDEn [2]. In this version, we added a NoSQL database that stores processed data as a time-series, and we replaced the old visualization tool (Matplotlib python library) by a powerful real-time monitoring tool called Grafana that has a user friendly interface and allows real-time monitoring, data retrieval and sending notification when a radiation exposure occurs.

We tried to cover all the limitations we identified in RaDEn at the beginning of our work but unfortunately, the implementation we have made have some limitations due to the following issues: (1) We did not get permissions to access the sensors or the databases. (2) The research time limit.

A list of works is lined up to be done in future. We can enrich the data by integrating the free weather data offered by online API's. Also, we can benefit from search engines such as Solr[11], Elastic Search[12] to perform data retrieval operations from raw data.

REFERENCES

[1] "Ionising Radiation and Human Health," Australian government - department of health, 07 December 2012. [Online]. Available: http://www.health.gov.au/internet/publications/publishing.nsf/Content/ohp-radiological-toc~ohp-radiological-05-ionising. [Accessed 17 September 2018].

[2] Fadlallah H., Taher Y., Jaber A., "RaDEn: A Scalable and Efficient Radiation Data Engineering", in International conference of Big Data and Cyber Security Intelligence, 2018.

[3] Avram C., Folea S., Dan Radu & Astilean A., "WIRELESS RADIATION MONITORING SYSTEM," in European Conference on Modelling and Simulation, Romania.

[4] Baker, C., Davidson, G., Evans, T. M., Hamilton, S., Jarrell, J., & Joubert, W., "High performance radiation transport simulations: preparing for Titan," in International Conference on High Performance Computing, Networking, Storage and Analysis, 2012.

[5] Jeong, M. H., Sullivan, C. J., & Wang, S., "Complex radiation sensor network analysis with big data analytics," in In Nuclear Science Symposium and Medical Imaging Conference, 2015.

[11] http://lucene.apache.org/solr
[12] http://elastic.io

[6] Liao, T. S., Wu, C. C., Chou, C. C., Hwang, C. H., Tang, Y. W., Tsai, D. P., & Chen, T. Y., "Simplified algorithm of ionizing radiation detecting based on image sensor," in Instrumentation and Measurement Technology Conference Proceedings, 2016.

[7] Kim, K. S., Kojima, I., Suzuki, R., Naito, W., & Ogawa, H., "RALFIE: a life-logging system for reducing potential radiation exposures," in the 1st ACM SIGSPATIAL International Workshop on the Use of GIS in Emergency Management, 2015.