JOURNAL OF TECHNIQUES

Journal homepage: http://journal.mtu.edu.iq

RESEARCH ARTICLE - ENGINEERING

Performance Evaluation for Related Techniques of the Proposed Environment Pollution Monitoring System for an Iraqi Case Study

Taha E Al-jarakh[1*], Osama A Hussein[1], Alaa K Al-azzawi[2] and Mahmood F Mosleh[1]

[1] Department of Computer Engineering Techniques, Electrical Engineering Technical College, Middle Technical University, Baghdad, Iraq.

[2] Department of Mechatronics Engineering, Technical Engineering College, Middle Technical University, Baghdad, Iraq.

*Corresponding author E-mail: tahaessam87@gmail.com

Article Info.

Article history:

Received
07 March 2021

Accepted
10 April 2021

Publishing
30 June 2021

Abstract

The main challenge of this research is to scale the IoT platform aspects related to exchanging, processing, and archiving messages at the lowest cost compute-wise, through evaluating and selecting the most appropriate techniques that can be used in the design of the environment pollution monitoring system for a case study of Iraq. The entirety of the optimization process aims to provide a nation-wide community-oriented service via the scalable platform. The platform provides an intake for a huge number of sensing nodes. Compute-operations following the form of data analysis, aggregation, sensors' monitoring for the five air pollutants (SO_2, CO, O_3, NO_2, and PM), in addition to radioactive contamination. Thus system-level performance evaluation takes place on the major compute-intensive operations. Thus, proposals are made to optimize the performance in terms of reducing the scripts execution time and the size of data and messages transmitted and stored in the system.

Keywords: Internet of Things (IoT); Distributed Systems; Compute Resources; Air Pollution; Radioactive Contamination; Air Quality Index (AQI); JavaScript Object Notation (JSON); Constrained Application Protocol (CoAP); Mongo Database (MongoDB)

1. Introduction

The population of Iraq currently suffers from a health crisis due to the increasing use of local generators, cars, buses, motorcycles, trucks, and construction equipment, all of which have been in use since 2003. This only adds to other health concerns resulting from the remnants of wars that in and of themselves have been a major cause of the increase in the environmental pollution in Iraq [1 - 7].

The threats to human health, due to these environmental issues that exist in Iraq, have led some researchers to propose and design a system for monitoring environmental pollution in Iraq. This system is supported by a web application, as a service to society, showing real-time air quality information on a planimetric map [8].

This monitoring system has the ability to calculate and display the concentrations of air pollutants adopted by the Environmental Protection Agency (US EPA). It also monitors air pollutants including (Sulfur Dioxide (SO_2), Ground-Level Ozone (O_3), Particulate Matter (PM10&2.5µg), Nitrogen Dioxide (NO_2), and Carbon Monoxide (CO)) which are measured as part of an air quality index, in addition to analyzing and displaying the rates of radioactive contamination depending on their risk levels adopted by the Centers for Disease Control & Protection (US CDC) [8].

Given the importance of this topic and its direct impact on human health. The system is expected to handle a huge number of requests thus a system level performance evaluation is a must. This will not only ensure the continuity of the service but it reduce the compute cost required to run the system as it spans the entirety of Iraq whose population will access the system via web application. The idea emerged to prepare this paper as a means to support this service in a way that makes it effective in accomplishing the critical human health-related tasks entrusted to it to the fullest and best manner possible, as well as, to give a guide for choosing the best techniques that can be used to design and implement such systems on a wide-scale.

Nomenclature		MongoDB	Mongo Database
PM	Particulate Matter	NoSQL	Non-Structured Query Language
SO_2	Sulfur Dioxide	CoAP	Constrained Application Protocol
O_3	Ground-Level Ozone	MQTT	Message Queuing Telemetry Transport
CO	Carbon Monoxide	IoT	Internet of Things
NO_2	Nitrogen Dioxide	MySQL	My Structured Query Language
RD	Radiation	GPRS	General Packet Radio Service
AQI	Air Quality Index	XMPP	Extensible Messaging and Presence Protocol
US EPA	US Environmental Protection Agency	AMQP	Advanced Message Queuing Protocol
US CDC	US Centers for Disease Control and Prevention	HTTP	HyperText Transfer Protocol
UDP	User Datagram Protocol	RDBMS	Relational Database Management System
HTML	Hyper Text Markup Language	TCP	Transmission Control Protocol
CSS	Cascading Style Sheets	API	Application Programming Interface
PHP	Personal Homepage (Hypertext Preprocessor)	**Symbols**	
JSON	JavaScript Object Notation	μg	Microgram

Overall, the paper is organized as follows: the aim of this paper given in the second section, the third section describes an overview of the evaluated system design, the fourth section shows the evaluated system techniques and results. While we discuss the results in section fifth. Finally, we conclude the paper in section sixth.

2. The Aim of This Paper

This paper is focused on evaluating and selecting the most suitable techniques that can be used in designing the proposed system in the research paper "Design and implementation of IoT environment pollution monitoring system: A case study of Iraq" that is mentioned in [8]. Thus, this system becomes very suitable for performing its nation-wide duty as a community service with the lowest possible scripts execution time and the least storage space for the collected data.

3. Overview of the System Under Evaluation

The system has been designed with the ability to detect and analyze the concentration of the five air pollutants (NO_2, O_3, PM, SO_2, and CO) and show the Air Quality Index (AQI) on a web application. It also detects and displays the levels of ionizing radiation. The system follows the Internet of Things platform architecture to exchange the distributed sensors' data with the system application. A proposal was made to distribute the system sensors in the sites of the cellphone networks to take advantage of their spread in creating an efficient wide-scale IoT platform to cover all regions of Iraq. A cloud-based system design has been proposed to utilize the following servers (MQTT server, MongoDB server, MySQL server, and Web server) thus exchange, archive, process, analyse, and display the collected sensors data as a real-time service to the society. The system backend web-application has been designed and programmed using (HTML, CSS, JavaScript, PHP, Mongo Database, MySQL Database, and Google Maps API). The system uses the MQTT messaging protocol as a means of transmitting sensor messages to the system database. The JSON message format is used by this system to exchange sensor data and display the concentrations of environmental pollutants on the system map. Fig. 1 shows an overview of the system structure [8]:

4. Techniques and Results of the Evaluated System

The evaluated system follows the modular monolith architecture as shown in Fig. 2:

4.1. Messages format

The evaluated system depends on two kinds of JSON message formats carrying sensors' data that are sent by transmitter modules (e.g. T-call ESP32 Wi-Fi/GPRS modules) to the MQTT server, as shown below:

4.1.1. Type-1 message format

This message carries the latitude and longitude of the sensor location in addition to three pollutants concentrations (Sulfur Dioxide (SO_2), Nitrogen Dioxide (NO_2), and ionizing radiation (RD)) with an approximate size of 75 bytes. An example of what this message might look like is {"Lat":33.235597,"Lng":44.339928,"NO2": 5.85493,"SO2":112.45158,"RD":2.0588}. The evaluated system has been loaded with random values equal to one-year readings data of 1000 sensing units sending their messages every 7.5 minutes. For such messages the insert stats were as follows:

- The execution time of insert 1000 messages in Mongo database is 1.134 sec.
- Two indexes for each message stored in Mongo DB.
- The total index size is 1.3 GB to save (70,080,000 messages) as one-year estimated data in Mongo DB.
- The total document size is 8 GB to save (70,080,000 messages) as one-year estimated data in Mongo DB.

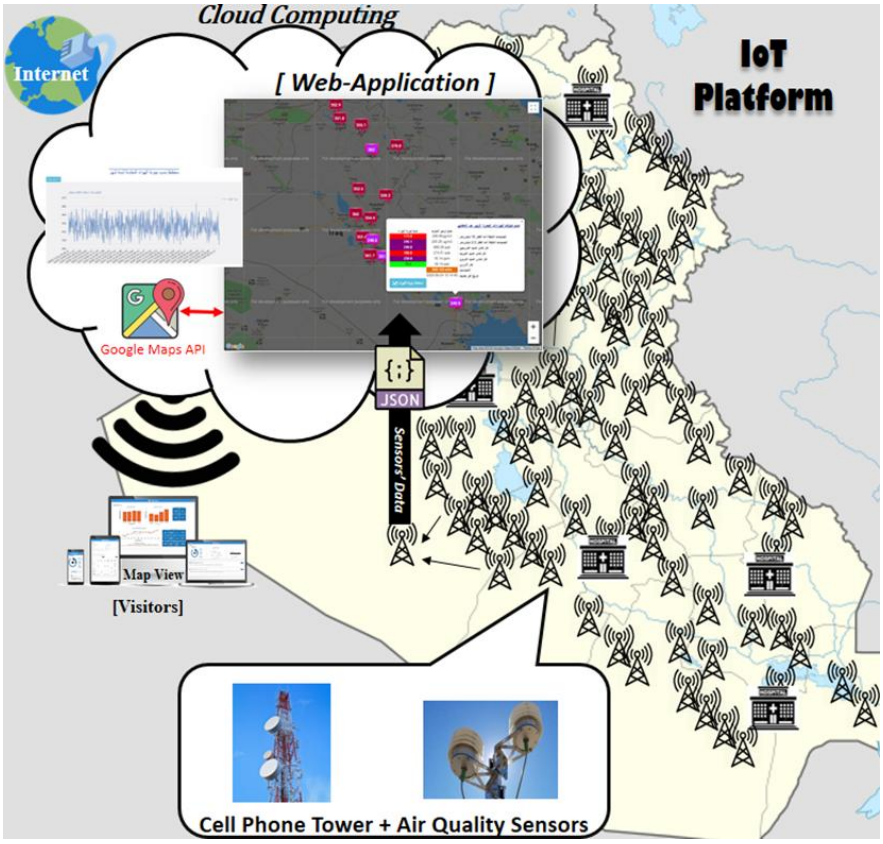

Fig. 1 system overview [8]

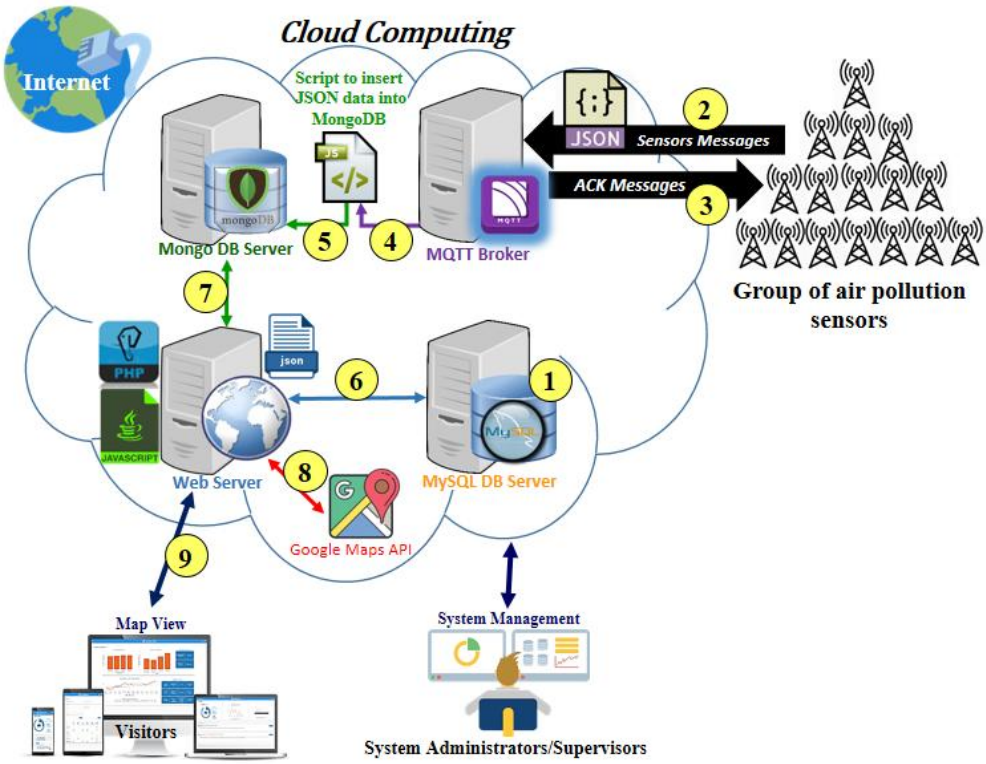

Fig. 2 architecture of the evaluated system [8]

4.1.2. Type-2 Message Format

This message carries the latitude and longitude of the sensor location in addition to four pollutants concentrations (Carbon Monoxide (CO), Ground-Level Ozone (O_3), Particulate Matter ($PM_{2.5\mu g}$), and Particulate Matter ($PM_{10\mu g}$)) with an approximate size of 84 bytes. An example of what this message might look like is {"Lat":33.114547,"Lng":44.344789,"PM10":620.8764,"PM2":749.8257, "CO":4.21,"O3":8.45}. The evaluated system has been loaded with random values equal to one-year readings data of 1000 sensing units sending their messages every 7.5 minutes. For such messages the insert stats were as follows:

- The execution time of insert 1000 messages in Mongo database is 1.228 sec.
- Two indexes for each message stored in Mongo DB.
- The total index size is 113.5 MB to save (8,760,000 messages) as one-year estimated data in Mongo DB.
- The total document size is 1.1 GB to save (8,760,000 messages as one-year estimated data in Mongo DB.

4.2. Messages format (Optimization)

We propose a look-ahead approach where we eliminate sending some of the lengthy identifiers in favor for sending a shortcut identifier. Thus, a test was conducted to reduce the execution time, internet bandwidth, and storage size required to load the evaluated system messages in the database. The best way to obtain these goals was found using a simple table stored in the database containing information for the distributed sensors (ID-Primary Key, Latitude and Longitude). This table fills out manually and needs to be brought before every query that requires fetching sensor messages from the Mongo database. The advantage of this table is that it eliminates the need of sending the longitude and latitude with each message (note that the table ID represents the sensor ID) which eliminates redundancy and provides the following:

4.2.1. Type-1 message (Improved format)

The new message format is {"ID":597,"NO2":5.85493,"SO2":112.45158,"RD":2.0588} with an approximate size of 52 bytes, and it requires:

- The execution time of insert 1000 messages in Mongo database is 0.946 sec.
- One index for each message stored in Mongo DB.
- The total index size is 681.9 MB to save (70,080,000 messages) as one-year estimated data in Mongo DB.
- The total document size is 6.9 GB to save (70,080,000 messages) as one-year estimated data in Mongo DB.

4.2.2. Type-2 message (Improved format)

The new message format is {"ID":106,"PM10":620.8764,"PM2":749.8257,"CO":4.21,"O3":8.45} with an approximate size of 61 bytes, and it requires:

- The execution time of insert 1000 messages in Mongo database is 0.979 sec.
- One index for each message stored in Mongo DB.
- The total index size is 84.8 MB to save (8,760,000 messages) as one-year estimated data in Mongo DB.
- The total document size is 952.4 MB to save (8,760,000 messages) as one-year estimated data in Mongo DB.

4.3. Messaging protocol

The evaluated system depends on the MQTT protocol to exchange sensor data with the system's web application, and this process requires the following to achieve the receiving system messages:

4.3.1. Requirements of Type-1 message

- The total internet download bandwidth required for the evaluated system to receive (1000 messages) of this type via the MQTT server is 610.35 KB/s including MQTT overhead, TCP overhead, IP overhead, and messages topic, while each message requires 625 bytes download bandwidth.
- The total internet upload bandwidth required for the evaluated system to receive (1000 messages) of this type via the MQTT server is 279.29 KB/s including MQTT, TCP, and IP overhead, while each message requires 286 bytes upload bandwidth.

4.3.2. Requirements of Type-2 message

- The total internet download bandwidth required for the evaluated system to receive (1000 messages) of this type via the MQTT server is 619.14 KB/s including MQTT overhead, TCP overhead, IP overhead, and messages topic, while each message requires 634 bytes download bandwidth.

- The total internet upload bandwidth required for the evaluated system to receive (1000 messages) of this type via the MQTT server is 279.29 KB/s including MQTT, TCP, and IP overhead, while each message requires 286 bytes upload bandwidth.

4.4. Messaging Protocol (Optimization)

In fact, there are many protocols used in IoT messaging applications such as CoAP, MQTT, DDS, XMPP, AMQP, and HTTP, but it is very necessary to know what the highest priority requirements for the evaluated system platform are, in order to choose the most suitable protocol to work with. After reviewing the advantages and disadvantages of these protocols as shown in [9][10], which generally revolve around aspects (security, interoperability, service provision, scalability, reliability, protocol & communication overhead, latency, resource requirement, bandwidth consumption, and power consumption).

The proposed sensors distribution of the evaluated system across cellphone networks, specifically in the tower sites, gave the greatest attention to how to reduce the bandwidth consumption including messaging protocol and communications overhead to reduce the cost and complexity of the evaluated system back-end design, while the other remaining required aspects are a foregone conclusion as they are considered within the duties cellphone network.

So, after testing the most popular lowest overhead and bandwidth messaging protocols, namely MQTT and CoAP, it is concluded that the CoAP protocol is well suited to perform tasks of the evaluated system due to the following results compared to the MQTT protocol results shown above.

4.4.1. Requirements of Type-1 message using CoAP protocol

- The total internet download bandwidth required for the evaluated system to receive (1000 messages) is 147.46 KB/s including CoAP overhead, UDP overhead, IP overhead, and messages URL, while each message requires 151 bytes download bandwidth.
- The total internet upload bandwidth required for the evaluated system to Acknowledge receive (1000 messages) is 44.92 KB/s, while each message requires 46 bytes upload bandwidth.

4.4.2. Requirements of Type-2 message using CoAP protocol

- The total internet download bandwidth required for the evaluated system to receive (1000 messages) is 154.29 KB/s including CoAP overhead, UDP overhead, IP overhead, and messages URL, while each message requires 158 bytes download bandwidth.
- The total internet upload bandwidth required for the evaluated system to Acknowledge receive (1000 messages) is 44.92 KB/s, while each message requires 46 bytes upload bandwidth.

4.5. Databases

In the evaluated system, the MySQL database is responsible for storing supervisors, departments, sensors, and notifications information (it stores all information requiring relationships between each other), while the Mongo database is responsible for storing all messages sent by sensors to the system.

The Mongo database handles a large number of queries related to reading and writing sensor messages that greatly and directly affect the efficiency of the evaluated system, so the possibility of using another database that reduces the time spent in executing the main scripts of the evaluated system was considered in this paper by re-programming the evaluated system with three main databases (MySQL RDBMS, Cassandra NoSQL Column DB, and Mongo NoSQL Document DB) to find out what is the most appropriate database that improves efficiency of the evaluated system. Table 1 shows the results of the tested databases. Note: All tested databases were indexed equally and loaded with one-year estimated data for fairness during the comparison.

Table 1 Databases Performance

No.	Script	Description	MySQL DB Execution time	Cassandra DB Execution time	Mongo DB Execution time
1	Insert type-1 message	Insert (1000 messages) of JSON message format that send every 7.5 min to MongoDB. * *Use JSON insert query to save this kind of messages plus timestamp for each message received.*	4.266 sec	1.924 sec	0.946 sec
2	Insert type-2 message	Insert (1000 messages) of JSON message format that send hourly to MongoDB. * *Use JSON insert query to save this kind of messages plus timestamp for each message received.*	4.437 sec	2.269 sec	0.979 sec

3	Hourly Auto Running	Run the system Auto Running Script that is responsible for: 1) Calculating the pollutants average concentration levels and AQI for all sensors location. 2) Archive the results to Mongo DB. 3) Update the system file (JSON format) used to display sensors info and locations on Google map. This script has been tested for 18 sites (each site consists of 7 sensors) with one-year of estimated data. * *Use select query to fetch and extract a number of the last keys' values of the two received JSON messages from all sites.* * *Use insert query to archive AQI and Avg. of pollutants concentrations for all sites.*	1.5253 sec	1.4305 sec	0.8867 sec
4	Sensors inspection	Run the sensors inspection script (This script has been tested for 19 sites with one-year of estimated data). * *Use select query to fetch and extract the last keys' values of the two received JSON messages from all sites.*	280.223 sec	0.6593 sec	0.3036 sec
5	Chart	Run the chart script for showing last month's reading of AQI data. * *Use count query to count AQI column/key that includes 487787 values for a single site.* * *Use select query to fetch the last 720 values of AQI column/key using the count result as an offset.*	6.5444 sec	23.2162 sec	2.8110 sec
6	Search	Fetch 26288 docs/rows (Each doc/row contains all pollutant concentrations and AQI for one site.) * *Use select query to fetch multiple-column/key values, some of the selected columns/keys restricted by a range of values.*	4.9438 sec	14.6232sec	1.3532 sec

What was observed from the execution of the six main scripts shown above of the evaluated system, especially after testing the system with more than one database, as shown in Table 1, that there is a significant impact of some queries used in reading and writing data through which the MongoDB database outperformed other tested databases as shown in the following detail:

- Scripts 1, 2: These scripts show that MongoDB has better performance for importing JSON messages into the evaluated system database because MongoDB does not need to split the received messages into structured separate storage units like tables and columns, it drops the message as a single piece due to its native JSON data store.
- Scripts 3, 4: The select queries used in these scripts are very complex due to their duties related to searching inside JSON messages which have variable formats from site to site as needed, so these types of queries require certain features to fetch the last stored messages that carry a specific sensor ID and pollutant key which means that every query needs (searching inside all stored messages to obtain values of the required keys, and sorting all documents in descending order to get the last messages for the specified keys, in addition to that these sorted messages need to be limited to a number of messages). These queries got the best results with MongoDB as shown in Table 1 because MongoDB works with simpler data structures that make data retrieval tend to be faster in it especially for large volumes of unstructured data such as JSON messages that the evaluated system handles.
- Scripts 5: This script requires two queries one of which is to count all archived AQI values of a specific sensor ID that are stored in the collection or table depending on the database used, then use the counted result in another query as an offset to fetch the last 720 documents or rows to be displayed on a chart. MongoDB gave a faster execution time to show this chart as shown in Table 1 because MongoDB has a simple document-oriented data structure that helps to do the duty of this operation in a very high-speed manner.
- Script 6: This script relies on a flexible select query to fetch a set of unorganized values as needed for multiple keys or columns (including timestamps, averages of pollutants, and AQIs), and this query has been greatly affected by the database indexing, so we indexed the tested databases fairly before testing them. As can be seen from Table 1, MongoDB achieved a great result in performing this process compared to other databases due to its flexibility and ease of navigation between the multiple indexed objects in the same document, which showed better results than moving between multiple indexed columns bound by a set of values that required more complex tasks.

5. Results and Discussion

The environmental pollution monitoring system discussed in this paper requires performance evaluation, in addition to increasing its efficiency in some important points in terms of reducing message size, internet bandwidth, scripts execution time, and storage size. This is required in order to get the best performance at the lowest cost. Thus, this paper focused on the main aspects of this system that have a direct impact on its efficiency and through which the following conclusions were reached:

5.1. Messages format

The enhancement made to the evaluated system message formats using the table shown in the results section gives the following improvements:

5.1.1. Type-1 message format

- Reducing the size of this message by about (23 bytes).
- Reducing the number of indexes required to save this message in the database to only one.
- Reducing the required internet bandwidth to receive messages of (1000 sites) by about (22.46 KB).
- Reducing the required execution time to insert (1000 messages) to the database by about (0.2756 sec).
- Reducing the total required index size of the received messages for a one-year period by about (649.3 MB).
- Reducing the total required document size of the received messages for a one-year period by about (1.1 GB).

5.1.2. Type-2 message format

- Reducing the size of this message by about (23 bytes).
- Reducing the number of indexes required to save this message in the database to only one.
- Reducing the required internet bandwidth to receive messages of (1000 sites) by about (22.46 KB).
- Reducing the required execution time to insert (1000 messages) to the database by about (0.2515 sec).
- Reducing the total required index size of the received messages for a one-year period by about (28.7 MB).
- Reducing the total required document size of the received messages for a one-year period by about (174 MB).

5.2. Messaging protocol

Through this research, it was concluded that using the CoAP protocol instead of the MQTT protocol to transmit sensors messages of the evaluated system reduces communication overhead since it relies on the UDP transport layer protocol instead of the TCP protocol used in MQTT, as well as that it gives the following improvements:

5.2.1. Type-1 message format

- Reducing the required internet download bandwidth for the evaluated system to receive (1000 messages) by about (462.89 KB/s).
- Reducing the required internet upload bandwidth for the evaluated system to Acknowledge receiving (1000 messages) by about (234.37 KB/s).

5.2.2. Type-2 message format

- Reducing the required internet download bandwidth for the evaluated system to receive (1000 messages) by about (464.85 KB/s).
- Reducing the required internet upload bandwidth for the evaluated system to Acknowledge receiving (1000 messages) by about (234.37 KB/s).

5.3. Databases

The comparison of databases listed in the results section shows that the Mongo database is the best choice for the evaluated system architecture due to the excellent performance in writing and reading JSON messages especially through the robust response to inserting, sorting, and retrieving queries for the multi-key indexed data that the most system scripts handle which faced a slow response with other tested databases, the MongoDB gives a powerful schema-free architecture that makes it well-suited for very high-speed logging & caching especially to the changeable message formats that the evaluated system needs to deal with, as it sometimes needs to delete or add pollutants carried over the message according to the area's requirement. In addition, the MySQL database has been selected to store the relational data of the evaluated system that needs security due to its effective ability to manage such data.

6. Conclusion

Evaluating the most influential aspects that have been used to build the proposed environmental pollution monitoring system for the Iraq case study, concluded the following:

- Using standard web-technology instead of using custom proprietary applications hugely saves on the system's cost i.e., using (web-sockets, HTML5, HTTP-servers, MySQL Database, and Mongo Database).
- The JSON message format is to be used for carrying sensors data of the evaluated system due to its ease of native-database support, low parsing cost, and generation in comparison to the other formats, in addition to its format being highly compatible for such systems that have variable messages format from site to site, but there is a need to use a simple table in the database of the evaluated system to reduce the message size, thus reducing the importing and retrieving execution time, internet bandwidth, and storage size required to load the system messages as shown in the result section.
- CoAP messaging protocol show superior results in comparison to MQTT protocol due to its minimum overhead and bandwidth required for managing communications of the exchanged system messages in comparison to the MQTT protocol. Thus, using a CoAP server instead of an MQTT server (industry norm) reduced the download bandwidth by a factor of 76% and the upload bandwidth by a factor of 84% when receiving and sending system messages.
- Mongo database is much more compatible for managing queries of the evaluated system compared to other tested databases due to its simple data structures. Thus, making it suitable for importing and retrieving a large number of JSON messages that the system handles. While the MySQL database is best suited for managing the small-size relational data that the evaluated system deals with.

Reference

[1] M. T. Chaichan, H. A. Kazem, and T. A. Abed, "Traffic and outdoor air pollution levels near highways in Baghdad, Iraq," Environment, Development and Sustainability, 2018.

[2] S. H. Shehabalden, and N. M. Azeez, "AIR QUALITY INDEX OVER BASRA PROVINCE, SOUTH OF IRAQ," International Journal of Technical Research and Applications, 2017.

[3] D. Khoshnevisan, et al., "Environmental pollution in the common borders between Iran and Iraq and the international governing documents," EurAsian Journal of BioSciences, 2019.

[4] N. A. Al-Ansari, S. Knutsson, and R. Pusch, "Environmental Implications of Depleted Uranium in IRAQ and Principles of Isolating It," Waste Management, 2014.

[5] S. A. Menkhi, F. H. Shanoon, and B. A. Almayahi, "Radiation pollution and cancer risks in Sulaimaniyah and Ninawa Cities, Iraq," Annual Research & Review in Biology, 2017.

[6] I. T. Al-Alawy, and O. A. Mzher, "Radiological characterization of the irt-5000 (14-Tammuz) research nuclear reactor at Al-Tuwaitha nuclear center in Iraq," Environmental Earth Sciences, 2019.

[7] F. H. Shanoon, et al., "Spatial and Temporal Variability of Environmental Radioactivity in Basra and Baghdad Cities, Iraq," Annual Research & Review in Biology, 2017.

[8] T. E. Al-jarakh, O. A. Hussein, A. K. Al-azzawi, and M. F. Mosleh, "Design and implementation of IoT based environment pollution monitoring system: A case study of Iraq," IOP Conference Series: Materials Science and Engineering, Iraq, 2020.

[9] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," IEEE international systems engineering symposium (ISSE), Austria, 2017.

[10] E. Al-Masri, et al., "Investigating messaging protocols for the Internet of Things (IoT)," IEEE Access, 2020.